User Guide v25.2

See all our docs at docs.snorkel.ai

PDF generated on April 11, 2025

Welcome to Snorkel Flow

Welcome to this orientation to Snorkel Flow. Here you can:

- Learn more about Snorkel Flow and data development concepts, or
- Get started with a task

Learn about Snorkel Flow and data development

Snorkel Flow is a data development platform. What does that mean, and how can it help you create efficient and effective machine learning applications?

These articles introduce the product and key concepts:

- What is Snorkel Flow?
- What is data-centric AI?
- Glossary

Get started with a Snorkel Flow task

When you log in to Snorkel Flow, you'll see the home page, which you can also navigate to by selecting **Home** or the **Snorkel** logo.

Snorkel Flow is a data development platform

From the home page, you can access existing applications, datasets, and assigned batches for labeling. If you are a subject matter expert tasked with labeling data, you will see only the batches assigned to you.

Continue reading for more information about getting started with specific tasks.

Deploy Snorkel Flow

If you are an admin responsible for deploying and configuring Snorkel Flow, read Installation Overview and Conventions.

Label data

If you want to label or annotate data, read the Walkthrough for Annotators.

Develop prompts

If you want to do iterative and effective prompt engineering, read about Snorkel Flow's Prompt Builder.

Create a machine learning application

If you want to set up an end-to-end machine learning application for classifying or generating content, read Create application: A guided flow

Evaluate an AI application

If you want to evaluate the performance of an ML application, follow along with the example in the LLM fine-tuning and alignment tutorial.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

Next steps

Follow along with Getting Started to get hands-on with Snorkel Flow.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

What is Snorkel Flow?

Snorkel Flow is a data development platform. Snorkel Flow enables organizations, teams, and individuals to raise the quality and development velocity of machine learning (ML) applications through more effective data development.

This article introduces Snorkel Flow's key use cases and features.

Why is data development important?

Data development is the process of applying a variety of data science techniques—for example, labeling—to a raw dataset with the goal of developing the dataset into a curated dataset that performs well for a specific application.

Data is the raw material for building machine learning (ML) applications. Off-the-shelf ML models rely on vast, non-specialized troves of data for their training. While their response patterns are impressive, they often aren't suitable for use cases with proprietary data, bespoke objectives, or high accuracy requirements. These types of use cases require curated datasets to achieve acceptable response quality. A data-first approach to ML application development is data-centric AI development.

Levels of data development

Data development can occur on many levels:

- **Prompt engineering:** Prompt engineering adds relatively small amounts of curated data to a model's input as in-context learning.
- **Retrieval-augmented generation (RAG):** Like prompt engineering, RAG adds relatively small amounts of curated data to any single prompt, but draws on a potentially large database of curated data to include the right chunk(s) of data.
- **Model fine-tuning:** Model fine-tuning adjusts an existing ML model's parameters with a curated dataset.
- **Custom model training:** Model training trains an entirely new ML model with a curated dataset.

How does Snorkel Flow improve data development?

Snorkel Flow provides a suite of tools that enable users to unlock the value of their data. You can use that data to train and fine-tune models and to create embeddings databases and prompts. These curated datasets and resulting models and databases yield real enterprise value.

Snorkel Flow offers:

- A centralized tool to build and store your curated datasets, instead of spreadsheets
- Faster, automated labeling workflows that save time for subject matter experts (SMEs), without losing significant label relevance
- Versioned, cumulative data labeling so you don't have to start from scratch when you add to your dataset or update your label schema

These features leads to faster iterations and faster creation of high quality curated datasets.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

Data development workflow

Data scientists and subject matter experts can collaborate using Snorkel Flow to craft datasets and models in days instead of weeks or months.

These are the steps in the typical data development loop:

1. Build labeling functions (LFs).
2. Approximate the combined accuracy of the labeling functions by training a quick model.
3. Analyze the results of the quick model.
4. Add and refine labeling functions to improve accuracy.
5. Build more robust models using the probabilistic data labels to check the current model's likely deployment-ready accuracy.
6. Build a final, deployment-ready model in the MLFlow format.
7. Evaluate the model's real-world performance.

Snorkel Flow offers specific tools to enable the core data development workflow.

Snorkel Flow tools

Snorkel Flow's suite of tools enable the following data development tasks:

- Labeling functions for programmatic labeling
- Weak supervision
- Labeling function evaluation
- Data labeling and correction
- Rapid small-model training
- Final model training

Labeling functions

Labeling functions codify expert knowledge and intuition into scalable rules. Labeling functions take the same knowledge your SMEs would use to manually label data, such as looking for a certain phrase in a document, and apply it automatically.

Labeling functions (LFs) can take many forms, including:

- Simple string searches
- Legacy rule-based systems
- Prompts for cutting-edge large language models (LLMs)

Labeling functions aren't perfect:

- Some labeling functions apply to fragments of a dataset and others cover large portions.
- Labeling functions can disagree with each other.
- Labeling functions can produce wrong labels.

Snorkel Flow handles these imperfections with the next step of the data development workflow: weak supervision.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

Weak supervision

Snorkel Flow's weak supervision algorithm determines which labeling function to trust for any particular data point.

Weak supervision is a statistical approach that sorts through potentially noisy or conflicting sources of labels. By evaluating the likelihood of each labeling function being correct in each case, the algorithm applies a probabilistic label to each data point.

Labeling function evaluation

Snorkel Flow provides a friendly user interface for examining the labels produced by the current model. With Snorkel Flow, you can quickly identify the individual data points that have incorrect or low-confidence labels by:

- Comparing the labels they produce to existing ground truth labels
- Assessing the Snorkel Flow weak supervision algorithm's confidence for the label for each data point

This process helps your experts focus additional labeling efforts to redraw classification boundaries or shore up model confidence.

> INFO
>
> **Example:**
> A user creating a spam-identification application can write a labeling function to tag all emails that contain the words "wire transfer" as `spam`. When evaluating the results of this labeling function, the user discovers that the model labels legitimate messages from banks as `spam`. The user can create a second labeling function that cancels out the first when the "wire transfer" message originates from a bank email address.

Through repeated iteration, users add and adjust labeling functions to capture more nuance until the model reaches the required level of accuracy.

Data labeling and correction

In addition to labeling functions, Snorkel Flow allows users to view and label individual data points.

Subject matter experts can use this feature to:

1. Label some ground truth for the cases when this hasn't been done already.
2. Explain their reasoning with comments and slices. This helps SMEs communicate their reasoning to data scientists, who can use comments to inform labeling functions.
3. Correct ground truth labels. Sometimes a labeling function disagrees with a label because the label is wrong. Once users identify one of these points, Snorkel Flow makes it easy to relabel it.

Each of these options helps nudge the probabilistic data set closer to usability.

Rapid small-model training

Snorkel Flow has fast model training options that allow for rapid iteration and evaluation. Once you have a fast model trained, you can use it to quickly evaluate the performance of your labeling functions. Read more about how to use fast models to create good labeling functions.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

Final model training

Snorkel Flow also offers production-level model training for when your model is producing good results and ready to export. Explore the model training options in Configure and train models.

Get started with Snorkel Flow

Follow along with Getting Started to get hands-on with Snorkel Flow.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

Getting started with Snorkel Flow

This series of getting started guides is designed to help you quickly get up to speed with the core features and capabilities of Snorkel Flow. Whether you're new to Snorkel Flow or looking to deepen your understanding, these resources guide you through key workflows to make the most of your experience.

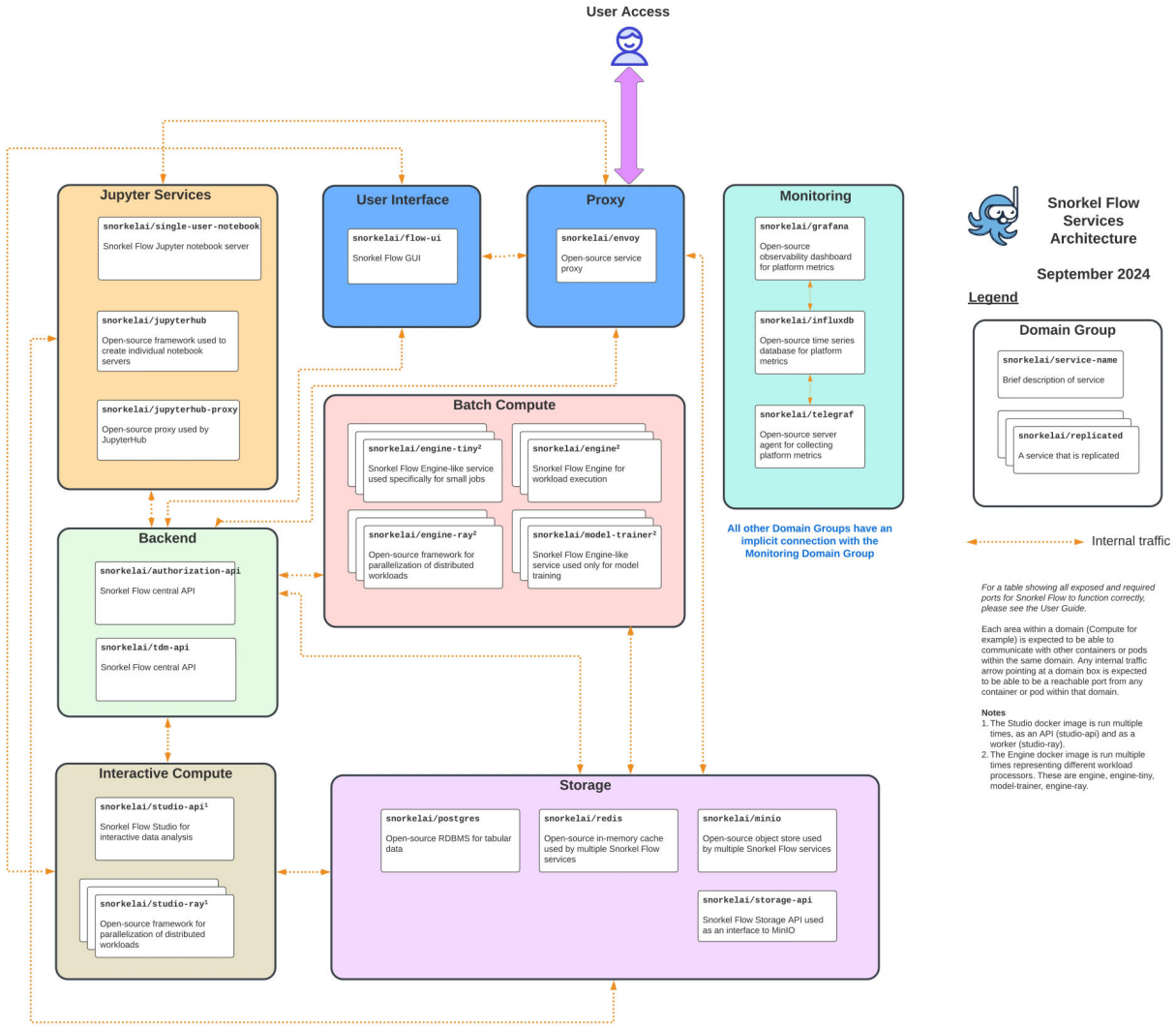
Each of these guides helps you quickly get up and running with the platform. Explore each topic and start building your AI solutions today.

Tutorial	Description
Document classification	Use Snorkel Flow to label your training data programmatically, train labeling models, and analyze the results. This tutorial guides you through an iterative development process and introduces key workflows for using Labeling Functions (LFs) to speed up labeling for document classification.
Information extraction	Snorkel Flow supports the extraction of structured information from semi-structured documents such as PDFs, HTML, and docx files. You can then prepare your data for analysis and model training by leveraging text, layout, and image modalities to write labeling functions (LFs).
GenAI evaluation tutorial	Snorkel's generative AI evaluation module enables enterprise AI teams to develop specialized, fine-grained, actionable GenAI evaluations in an iterative, programmatic way. Implement your genAI evaluation by defining a benchmark, creating and analyzing evaluation dashboards, annotating the datasets, and refining the benchmark or moving into data development. This guide is ideal for users interested in refining generative AI models.
SDK quickstart	If you're a developer, the SDK quickstart shows you how to programmatically classify documents. This guide includes creating your dataset, creating an application, writing labeling functions, training a model, and deploying your model.
Walkthrough for annotators	This walkthrough explains the process of manually annotating documents in Annotation Studio, which are used to generate high-quality labeled datasets, ensuring consistency and accuracy across your projects. This guide also includes an interactive demo showing how to annotate your data.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

Snorkel Flow architecture overview

Snorkel Flow's architecture is comprised of the components illustrated below. Below the diagram, read further about each component.



From a foundation level, the platform is an aggregation of the following services:

1. Jupyter Services

This is made up of the open source framework that is used to create individual notebook servers along with the open source proxy used by Jupyterhub.

2. User Interface

This is the Snorkel Flow Graphical User interface, where users can perform data labeling, data management, and annotation workflows within our on-platform UI.

3. Proxy

This is the open source service proxy that is the main entry point into the platform.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

4. Batch Compute

This section comprises of the engine services for small jobs, workload execution, parallelization of distributed workload and model training.

5. Backend Services

This is made up of the tdm-api and authorization-api, which forms the backbone of Snorkel Flow's central API services.

6. Storage Services

This layer is made up of Postgres that provides an open source RDBMS for tabular data, Redis open source in-memory cache, open source MinIO object storage and Storage API which acts a gateway to MinIO object storage.

7. Interactive Compute

This part of the platform is represented by studio-api that provides interactive data analysis and studio-ray which is the open source framework for the parallelization of distributed workloads.

8. Monitoring

This section provides telemetry and observability into the platform is is made up of Grafana, InfluxDB and Telegraf that provide an observability dashboard for platform metrics, a time series database and the collection of platform metrics respectively.

Implementation context

The Snorkel Flow platform is designed to be deployed on Kubernetes environments that are hosted on AWS (EKS), Azure (AKS) and GCP (GKE), as well as with platforms running on private cloud environments. For details on deployment options that are a good fit for you, please contact a Snorkel AI representative for discussing the choices that are available.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

What is data-centric AI?

This document introduces data-centric artificial intelligence (AI) and how it differs from model-centric AI. It also explains how the data-centric approach can benefit machine learning (ML) applications and how it can be applied in an organization.

Data-centric AI vs. model-centric AI

In data-centric AI, data is the key factor in how well an ML application performs. A data-centric approach means your effort is weighted towards making sure AI is learning what you want it to learn. Snorkel Flow makes this process of data development scalable with operations that accelerate labeling, managing, slicing, augmenting, and curating data efficiently. The model stays relatively fixed.

This contrasts with model-centric AI, where the choice of model is the differentiating factor. Model-centric AI uses training datasets as static collections of ground truth labels. The ML model is trained to fit that labeled training data as a static artifact.

Improvements in the data and improvements in the model are not mutually exclusive. Successful AI requires both well-conceived models and good data.

A brief history of data and models

Machine learning has always been about data. Historically, data science and machine learning teams would start with a dataset, and then achieve gains in ML application performance by iterating on static aspects of the data like feature engineering, or by working on algorithm design and bespoke model architecture.

Now that sophisticated models are available as off-the-shelf commodities, data science teams can provide value by shifting their focus towards continuously improving the data that makes their applications unique.

The power of iterating on data

Curated data shouldn't be treated as a static artifact.

Much like code forms the building blocks of traditional software, data forms the building blocks of ML applications. Just like you can iterate on code to improve traditional software, you can iterate on data to achieve higher performance in ML applications.

This holds true whether you're using your data in-context with prompt engineering and retrieval-augmented generation (RAG), or using data to fine-tune or train high-performance machine-learning (ML) models. If your focus is on fine-tuning and training models, today's models require significantly higher volumes of and higher quality training data.

Evaluating generative model responses is also a data task that benefits from iteration. For example, when you have a chatbot that can respond to user queries about your product, it may take several rounds of iteration to establish all of the criteria for an acceptable response, and then to label responses against those criteria.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

Data-centric AI and the data development lifecycle

The dataset is the interface for collaboration between subject matter experts (SMEs) and data scientists, who turn those experts' knowledge into software.

The process of creating and improving curated datasets is called *data development*.

Just like software development, data development can be done programmatically, saving many hours of SME time. Snorkel Flow is a data development platform that enables data development as a discipline.

Example: Identify risk exposure using AI

To illustrate the value of data-centric AI, here's an example from one of Snorkel Flow's customers.

Snorkel Flow helped a top-three US bank reduce the time it takes to identify and triage risk exposure in their loan portfolio from months to days.

Previously, this process required:

- 6 person-months of analyst time to manually label the bank's documents
- 1 day to develop a model for the specific project
- 3 months to perform error analysis, validate the model, and review and improve the datasets

That six months of manual data labeling was a severe bottleneck. The bank was able to perform this risk exposure analysis only on an ad-hoc basis, and their datasets still suffered from inconsistent and unreliable labels. The labeling approaches weren't auditable and were slow to adapt to existing issues.

Before adopting Snorkel Flow, the bank had these options for data labeling:

- Outsourcing to labeling vendors: Outside manual-labeling vendors brought significant privacy concerns regarding the confidentiality of the bank's data. Additionally, most annotators lacked domain expertise, so their work required additional review before use.
- Labeling with in-house experts: Using in-house subject matter experts (SMEs) solely for labeling was significantly time consuming and had a high opportunity cost, as SME time could be used on higher-value efforts.

When the bank decided to use Snorkel Flow for data labeling, they significantly reduced total labeling time while increasing label accuracy:

- Increased speed: After developing their initial model in Snorkel Flow, the bank could label ~250k new documents in less than 24 hours.
- Guided error analysis: Data scientists could rapidly iterate on model errors to improve results, collaborating with SMEs and codifying their input into the existing labeling approach.
- Minimal validation effort: Label validation and iteration could be accomplished in minutes with minor code modifications.

Key principles of data-centric AI

Here are some principles that can guide an organization or individual towards a data-centric approach to AI:

- Spend effort on data to efficiently improve ML applications: As models become more user-friendly and commoditized, gains in ML application quality increasingly rely on curated and updatable datasets.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

- **Iterate on datasets:** Rather than treating your datasets as static inputs to ML applications, treat datasets like code and keep developing them.
- **Adopt programmatic data labeling:** Data development should be programmatic. This allows developers to cope with the volume of training data that today's deep-learning models require. Manually labeling millions of data points is not practical. A programmatic process for labeling and iterating on data is crucial to make consistent progress.
- **Collaborate with subject matter experts (SME):** Data-centric AI should treat subject-matter experts (SMEs) as integral to the development process. Include SMEs who understand how to label and curate your data so data scientists can inject their domain expertise directly into the model. Once done, this expert knowledge can be codified and deployed for programmatic supervision.

Benefits of data-centric AI

Organizations developing their AI solutions using data-centric AI capture additional value from creating higher-quality data:

- **Faster development:** A Fortune 50 bank built a news analytics application 45x faster and with +25% higher accuracy than a previous system.
- **Cost savings:** A large biotech firm saved an estimated $10 million on unstructured data extraction, achieving 99% accuracy using Snorkel Flow.
- **Higher accuracy:** One of the largest custodian banks used Snorkel Flow to extract and add metadata to a dataset. The bank improved the accuracy of their AI agent answers from 25% to 95% by using RAG from the enhanced dataset.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

Active learning and weak supervision

In Snorkel Flow, programmatic labeling combines active learning and weak supervision.

Active learning

Active learning is a machine learning framework in which a model is initially trained on a small set of high-quality labeled data. The model then iteratively selects additional data points to be labeled by an expert, based on criteria such as uncertainty or informativeness relative to the current model's understanding.

This is the typical active learning process:

1. **Initial model training:** Start with a small, carefully labeled dataset.
2. **Uncertainty evaluation:** Assess which data points the model is least certain about.
3. **Expert querying:** Request an expert to label these uncertain data points.
4. **Model re-training:** Update the model with the newly acquired labels.
5. **Iterative improvement:** Repeat the cycle to enhance model performance continuously.

Weak supervision

Weak supervision is an approach that facilitates labeling large datasets using less accurate or noisier sources. It often involves multiple weak labelers whose outputs are integrated to enhance the overall label quality.

The weak supervision process may include these elements:

- **Heuristic rules:** Automated rules based on domain knowledge.
- **Crowdsourced data:** Labels gathered from non-experts or a large group of annotators.
- **Previously trained models:** Outputs from other models that can serve as provisional labels.

Snorkel Flow hybrid framework

By combining active learning and weak supervision, Snorkel Flow creates a powerful hybrid framework for data labeling and machine learning:

1. **Initial label generation:** Snorkel Flow starts with a small amount of annotated data, which defines the ground truth (GT) . Then Snorkel Flow applies the proprietary weak supervision algorithms to quickly generate a broad set of initial labels across a large dataset. This initial labeling uses heuristic rules, crowdsourced data, and outputs from other models to rapidly label data.
2. **Refined learning process:** Once the dataset is preliminarily labeled, Snorkel Flow enables users to train an initial model and evaluate the model outputs on the initial GT dataset. With the built-in Error Guided Analysis tools, apply active learning techniques to identify samples of data where the model's performance can be improved. In the Annotation Suite, assign these subsets of data to Subject Matter Experts (SMEs) for their review. This process ensures that resources are focused where they are most needed: SMEs are focused on labeling active learning batches while data scientists focus on programmatic labeling. Both team work in unison to improve data quality and end-model performance.
3. **Programmatic labeling integration:** After receiving SME feedback, modify the existing labeling functions or create new labeling functions. This iterative development continues with the creation of a

Copyright © 2025 Snorkel AI, Inc. All rights reserved

new higher quality dataset, which is then used to train a new model to be analyzed. In this way, Snorkel Flow uniquely integrates programmatic labeling with active learning, combining the scalability of weak supervision with the precision of active learning.

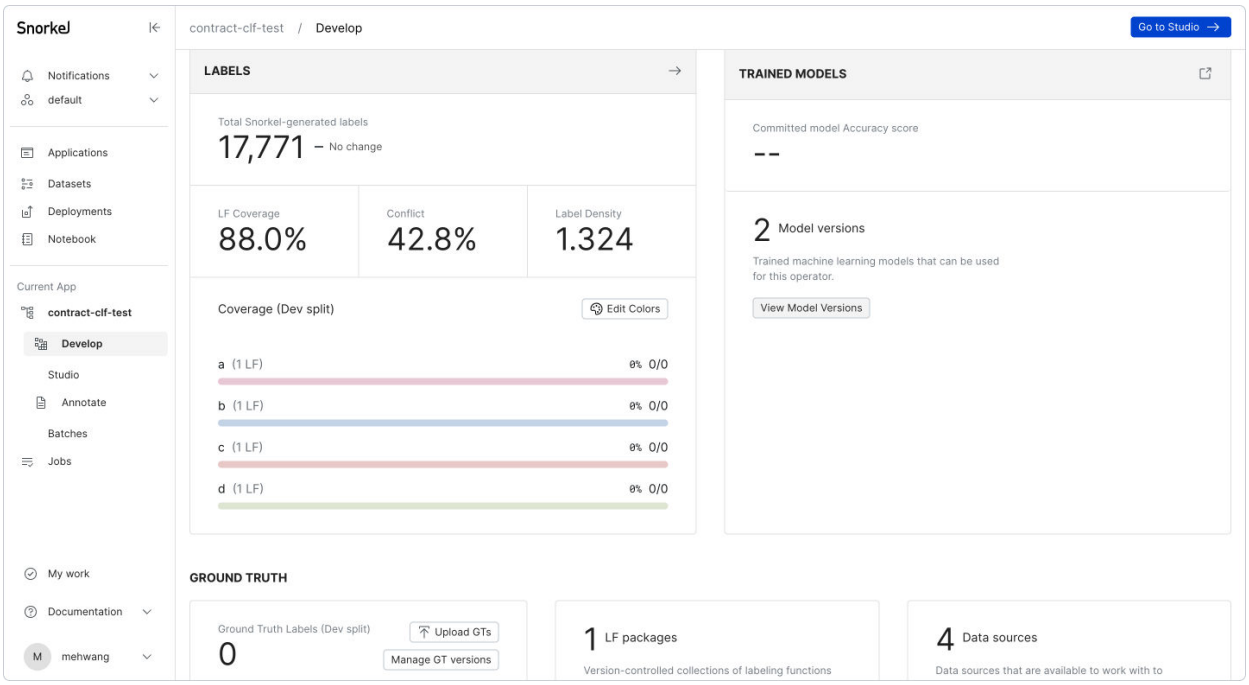
4. **Continuous enhancement:** The iterative nature of this integrated approach means that each cycle of active learning and re-training with new expert labels makes the model progressively smarter and more reliable. Additionally, this iterative cycle doesn't stop after the model is deployed: Use Snorkel Flow to update a model anytime you need to, including post deployment.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

Application

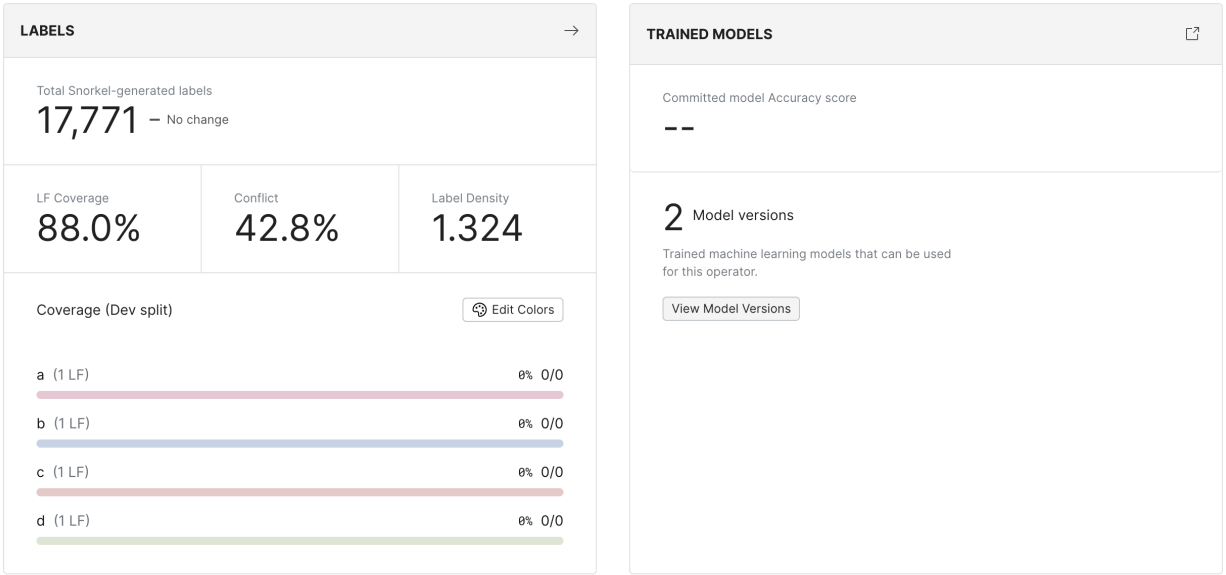
The **Overview** page lets you review your labels, models, metrics, and Snorkel artifacts.

The model overview shows you a short summary of each of the label, train, and analysis components of the Snorkel development loop. It also provides management access to a range of Snorkel artifacts and assets, like training sets and enabled data sources.



Label, train, and analyze

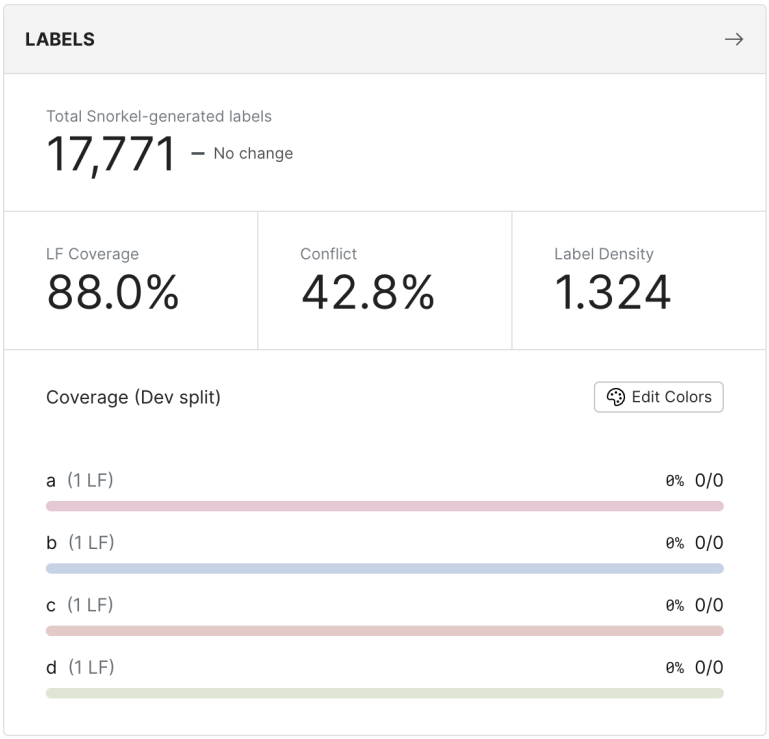
The top row of the model overview page includes a small summary for each of the primary components of the Snorkel model-development loop.



Copyright © 2025 Snorkel AI, Inc. All rights reserved

Labels

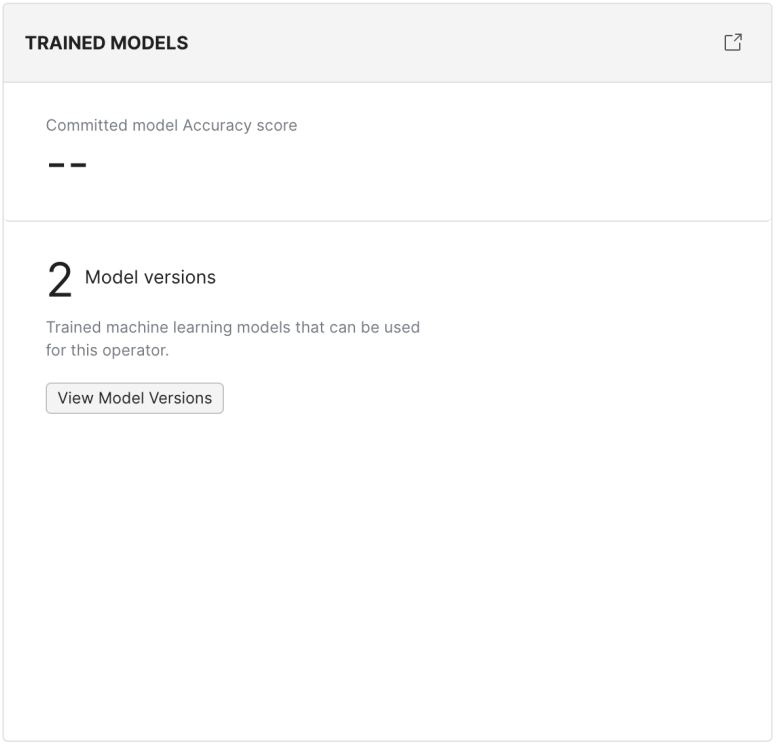
Shows an overview of the total Snorkel generated labels, i.e., labels that have been generated by your labeling functions (LFs) in Snorkel Flow. It also provides label statistics to help you review the LF coverage (% of data points that have at least 1 LF vote), conflict (data points where there is a conflict between 2 or more LF votes) and label density (average number of LF votes per data point) for the generated labels.



Trained models

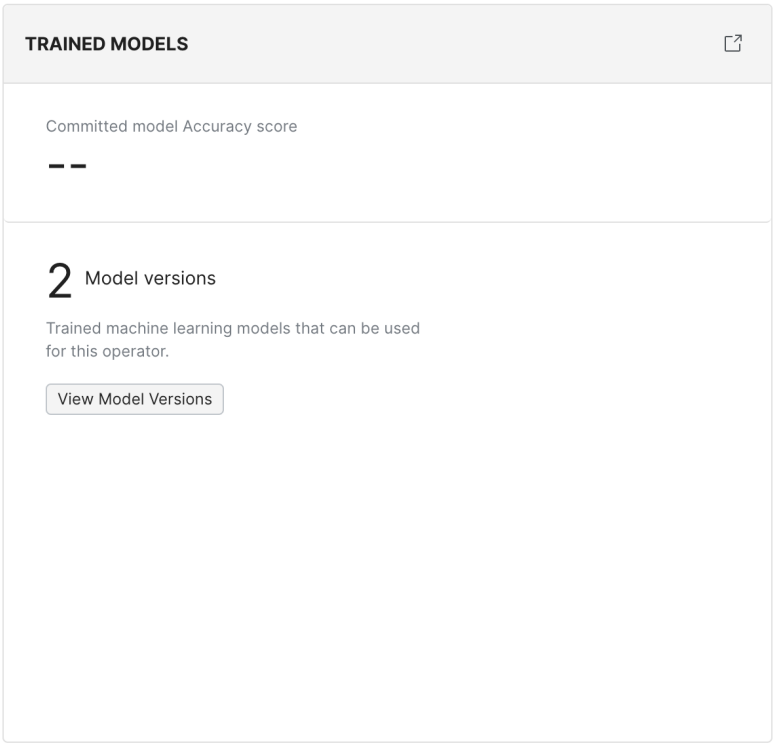
Provides access to the 'train' page with all your versioned models and relevant performance statistics for your best model.

Copyright © 2025 Snorkel AI, Inc. All rights reserved



Snorkel artifacts

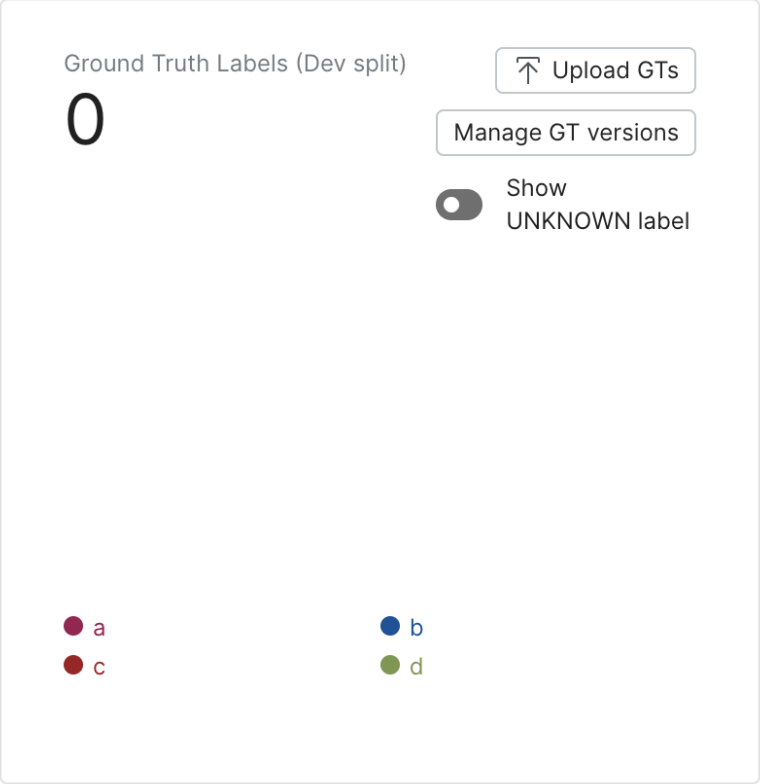
The bottom row of the Develop page summarizes some of your Snorkel-generated artifacts and provides access to management tools. Each of these panels is summarized below.



Copyright © 2025 Snorkel AI, Inc. All rights reserved

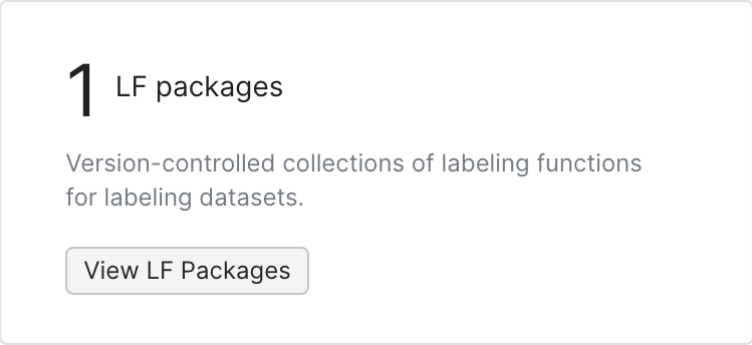
Ground truth labels

Displays the distribution of the ground truth labels associated with this model. You can upload ground truth from a file and/or manage ground truth versions directly from this panel.



Labeling function packages

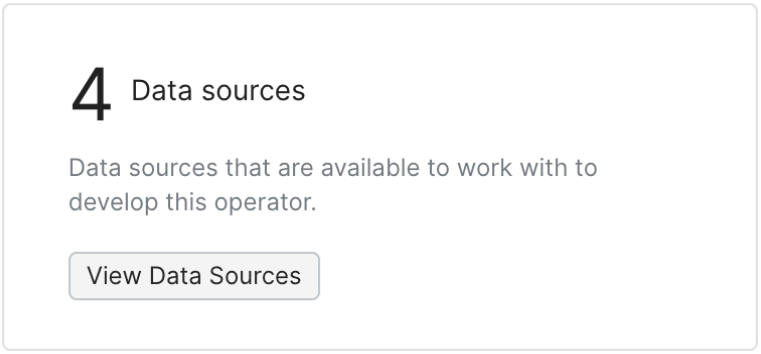
The *View LF Packages* button takes you to the LF Packages page, where you can manage your version-controlled set of existing LF packages for this model.



Data sources

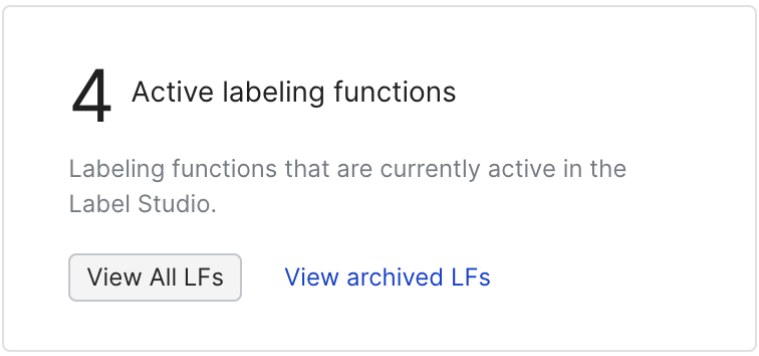
This panel lists the count of currently enabled data sources. The *View Data Sources* button provides access to the Manage the data sources for a model page, where you can manage the active data sources for the model.

Copyright © 2025 Snorkel AI, Inc. All rights reserved



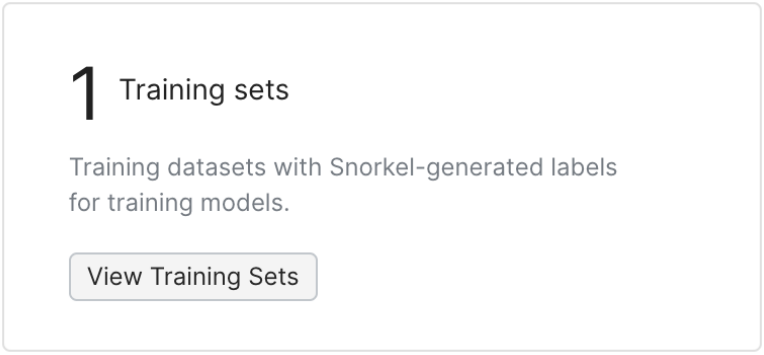
Active labeling functions (and archived labeling functions)

The *View All LFs* button links out to your current set of LFs, while the *View archived LFs* text link takes you to your archived LFs. You can click Restore to convert an archived LF back to active.



Training sets

The *View Training Sets* button takes you to the Training set overview: Review your training sets page, where you can manage your set of existing version controlled training sets for this model.



Copyright © 2025 Snorkel AI, Inc. All rights reserved

Data and application limit requirements

This page is a reference for data and application limit requirements for a standard Snorkel Flow installation. When creating new datasets or applications, please refer to the these class, dataset, and datapoint limits for the best performance.

	# Unique classes	Max total dataset size	Max datapoint size (i.e., single row)	GPU required?
Text classification (multi-class)	2 - 100	2 GB	10 KB	Recommended
Text classification (multi-label)	1 - 100	2 GB	10 KB	Recommended
Text extraction (candidate-based)	2 - 100	2 GB	10 KB	Recommended
Text extraction (sequence tagging)	1 - 25	250 MB	10 KB	Required
PDF extraction (candidate-based)	2 - 20	1.6 GB	1.6 MB	Recommended
Image classification	1 - 10	20 GB	500 KB	Required

Additional Limits

In addition to the application specific limits above, the following table outlines limits that apply per each application:

Limit Type	Maximum Value
Number of Model Nodes	20
Number of Data Sources	50
Max Datasource Size	100 MB

These limits are designed to ensure optimal performance and resource utilization within Snorkel Flow. They may be adjusted based on your specific installation and requirements.

Reach out to your Snorkel representative for more information about instance sizing.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

Dataviewer overview

The dataviewer is the primary canvas for data exploration and supervision. You will use the dataviewer throughout Snorkel Flow, from verifying your uploaded data to annotating it with labels and developing models. Use the dataviewer to:

- View your data in different formats, individual and aggregate.
- Filter and sort your data.
- Compare metadata, including ground truths, model predictions, LF votes, and more.
- Edit ground truths.
- Slice and comment on your data.

Access the dataviewer

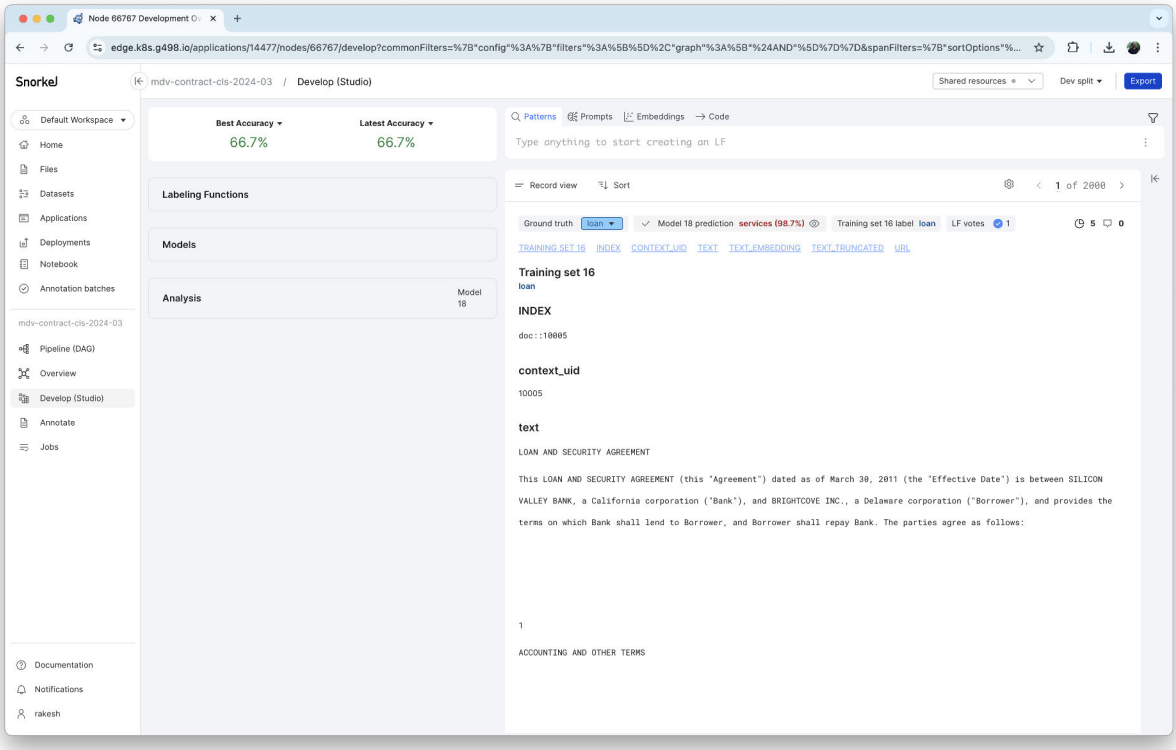
Because data is so central to everything you will do in SnorkelFlow, the dataviewer is available in several places throughout your application. Some of these places include:

1. On the right hand side of your screen when you are creating an application after you have uploaded your dataset. This is outlined in the Create Application section.
2. On the right hand side of your screen in the Application Studio after creating an application. This is located below the search bar for creating labeling functions (LFs) and shown in the screenshot below.
3. In the center of your screen when annotating documents.

Dataviewer feature reference

Different features will be available in each place the dataviewer is seen and based on the application type, but there is significant commonality between them. These features are outlined below.

Copyright © 2025 Snorkel AI, Inc. All rights reserved



Keyboard shortcuts

Note: The below shortcuts are configured for the **Develop (Studio)**. Some keyboard shortcuts on dataset-level annotation views are not supported.

Record view

Shortcut	Description
Right Arrow (→)	Navigate to next data point
Left Arrow (←)	Navigate to previous data point
Up Arrow (↑)	Navigate to previous candidate, only available in extraction tasks
Down Arrow (↓)	Navigate to next candidate, only available in extraction tasks

Document view

Shortcut	Description
Right Arrow (→)	Navigate to next data point
Left Arrow (←)	Navigate to previous data point

Copyright © 2025 Snorkel AI, Inc. All rights reserved

Shortcut	Description
Up Arrow (↑)	Navigate to previous candidate
Down Arrow (↓)	Navigate to next candidate

Snippet view

Shortcut	Description
Right Arrow (→)	Navigate to next page of data
Left Arrow (←)	Navigate to previous page of data

Table view

Shortcut	Description
Right Arrow (→)	Navigate to next page of data
Left Arrow (←)	Navigate to previous page of data

Next steps

Explore the features and functionality of the dataviewer in the following sections:

- Filter data: Learn how to apply filters to your dataset
- Display control pane: Learn how to switch between different data views, sort your data, change which columns are displayed, navigate through your dataset, and use other advanced controls.
- Data summary pane: Learn about the right hand side panel in Application Studio that can show you aggregate statistics for your dataset.
- Data content views: Classification applications: Learn about features that are specific to classification applications.
- Data content views: Candidate-based extraction applications: Learn about features that are specific to candidate-extraction applications.
- Data content views: Sequence tagging applications Learn about features that are specific to sequence tagging applications.

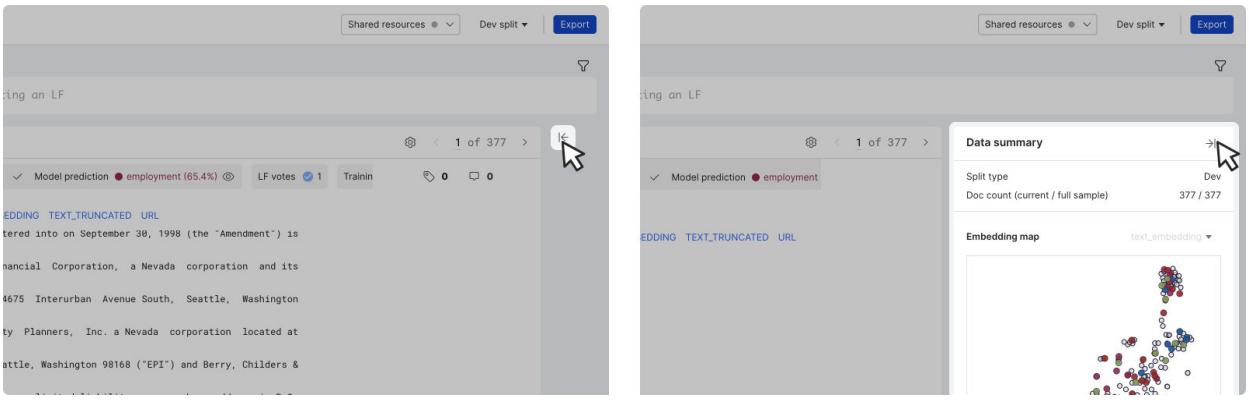
Copyright © 2025 Snorkel AI, Inc. All rights reserved

Dataviewer: Data summary pane

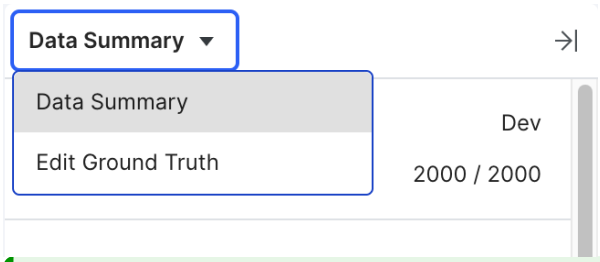
This page walks through the functionality and information that is provided the data summary pane.

The data summary section provides aggregated information of all data points currently in view in the data content section.

To open the data summary pane, click the **arrow** button to the right of the data content section. To close the pane, click the **arrow** button on the top right corner of the pane.



Ensure that the **Data Summary** is selected in the drop down menu.



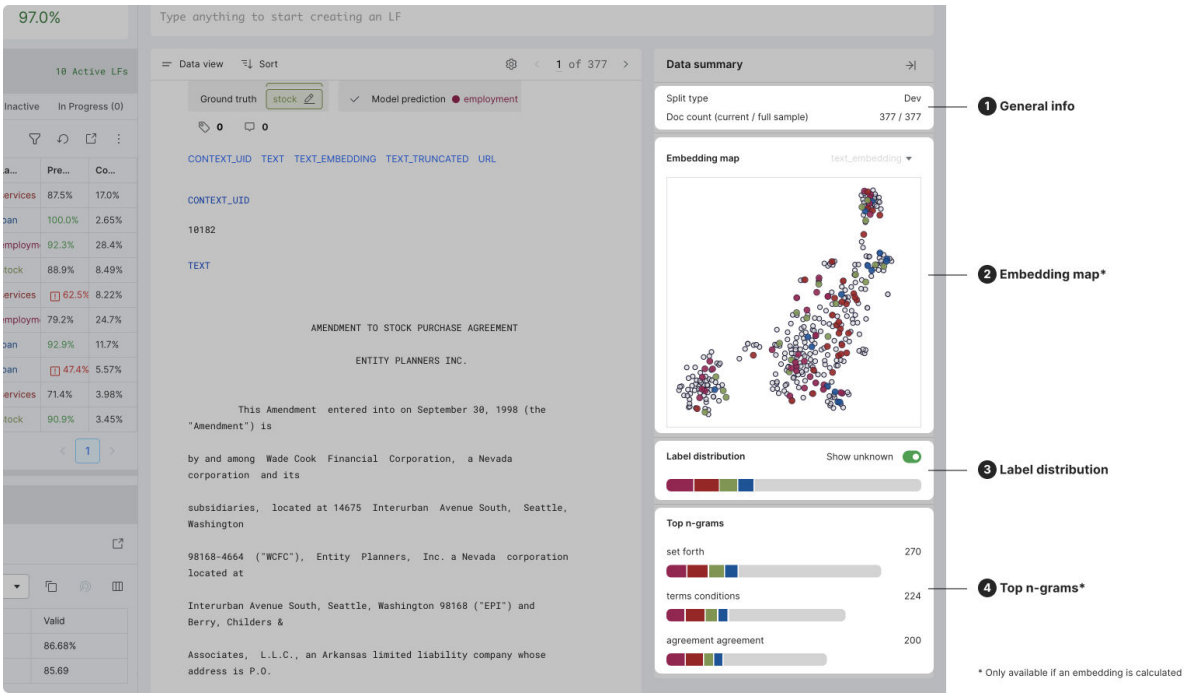
> **TIP**
>
> In this pane, you also have the ability to edit ground truth and update the label schema. See Ground truth annotations and Updating the label schema for more information.

The data summary includes the following information:

- **Split type**: The split of the data (i.e., Dev, Train, Valid, or Test).
- **Doc count**: The number of documents that are currently in view in the data content section versus the total number of documents in the sample. You can use filters to drill down into a subset of the total documents.
- **Embedding map**: If available, the map of the selected embedding cluster LF.
- **Label distribution**: The distribution of the labeled data points
- **Top n-grams**: If available, the top n-grams of the selected embedding cluster LF. N-grams are sequences of words or tokens that appear frequently in your documents.

> **NOTE**
>
> The Embedding map and Top n-grams are populated only when embeddings are computed for your data. See Utilize Embeddings for more information.

Copyright © 2025 Snorkel AI, Inc. All rights reserved



Copyright © 2025 Snorkel AI, Inc. All rights reserved

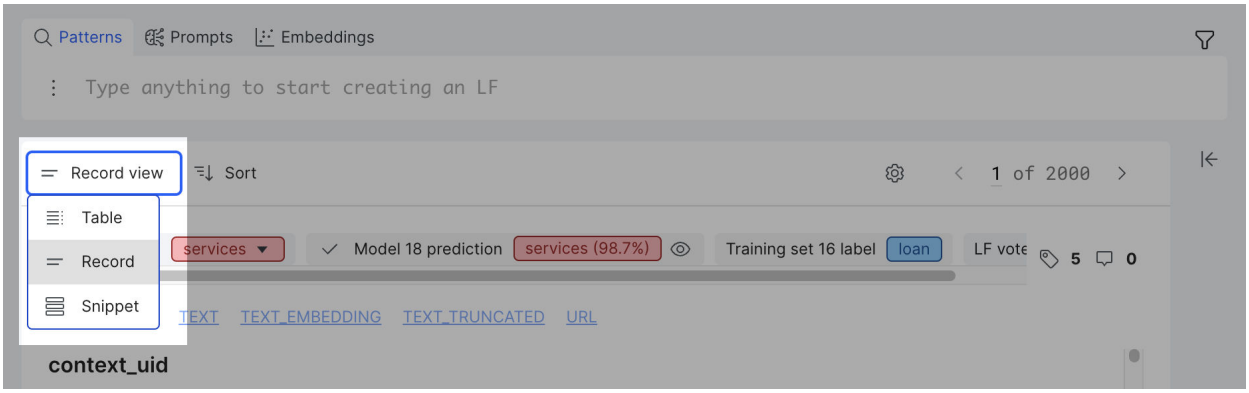
Dataviewer: Display control pane

This page walks through the functionality and settings of the display control pane.

The display control pane includes controls to the display format, list order, and navigation of data, as well as the entry point to advanced controls.

View mode

Click **Record view** (or whichever is the default view in your application) to see and select from all available view options. Your selected data view will persist through other actions in **Develop (Studio)** (e.g., filtering and LF composition) until you select another view.



The view options vary depending on your data and task type. For details on each view mode, see the following pages:

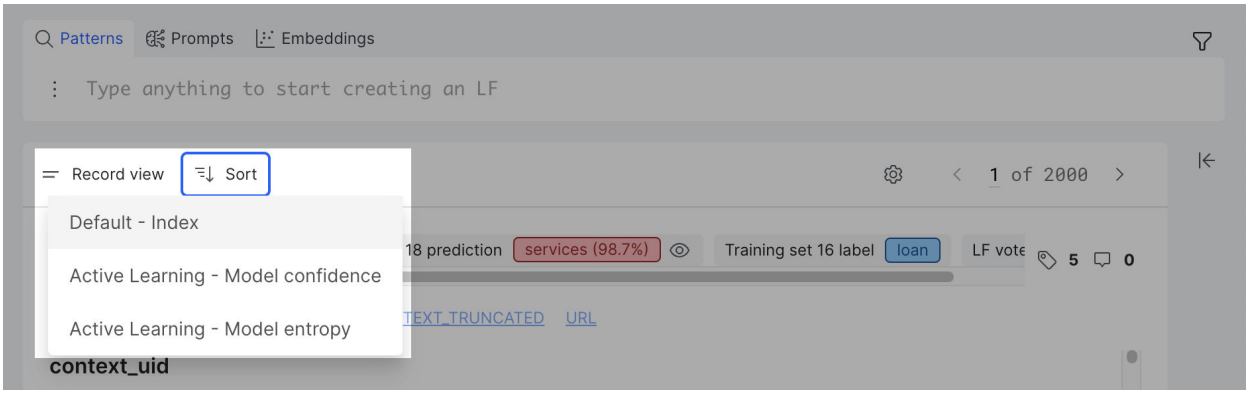
- Data content views: Classification applications
- Data content views: Candidate-based extraction applications
- Data content views: Sequence tagging applications
- Data content views: Generative AI annotation

Sort

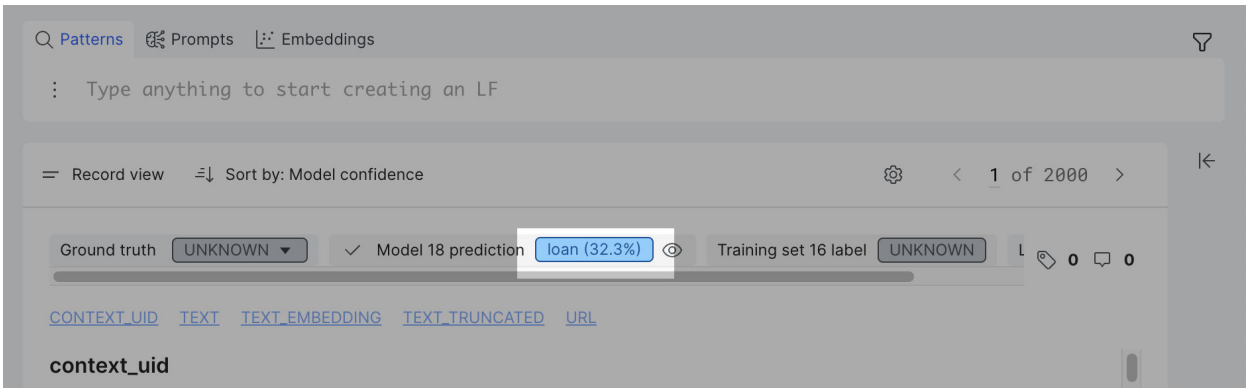
Click **Sort** to see and select from all available view options.

- **Default - Index**: the default sort method that sorts your data by index (UID) from low to high.
- **Active Learning - Model confidence**: Sorts your data by model confidence from low to high. The values are between 0% and 100%. This option is only available in classification applications.
- **Active Learning - Model entropy**: Sorts your data by model entropy from low to high. The values are between 0 and 100. This option is only available in classification applications.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

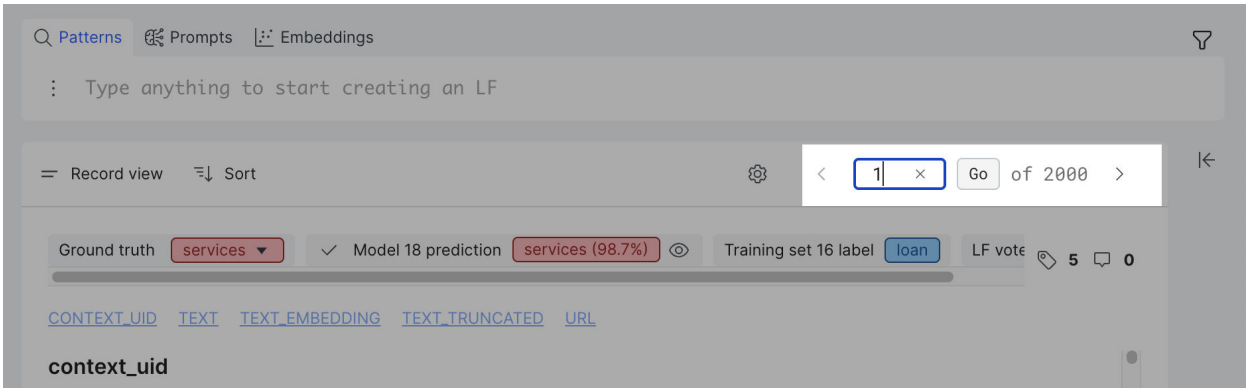


When you sort by **model confidence or model entropy**, you can see the model confidence or entropy number of each data point when you are in **Record view.**



Pagination

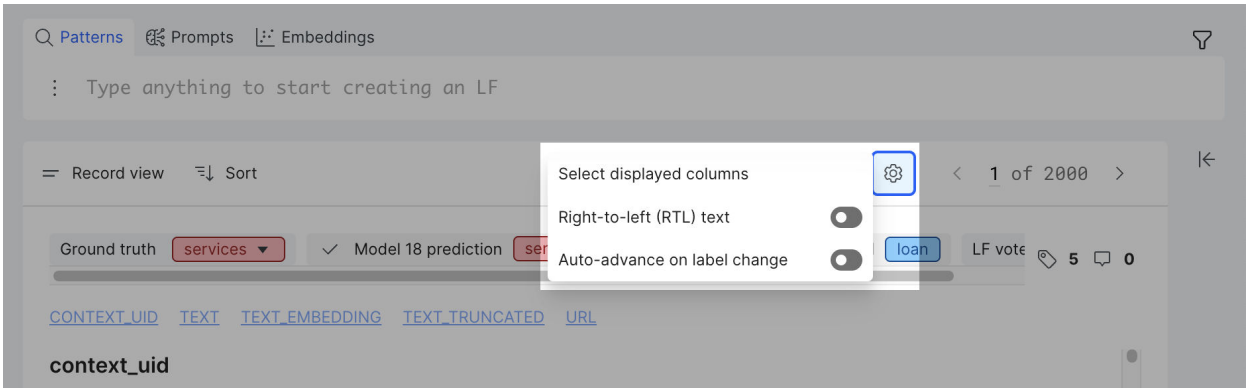
Click the **chevron icons (< or >)** to navigate between individual data points when you are in **Record view**, or groups of data when you are in **Table view**. To go to a specific data point or group, click on the number in the square and enter desired number.



Advanced controls

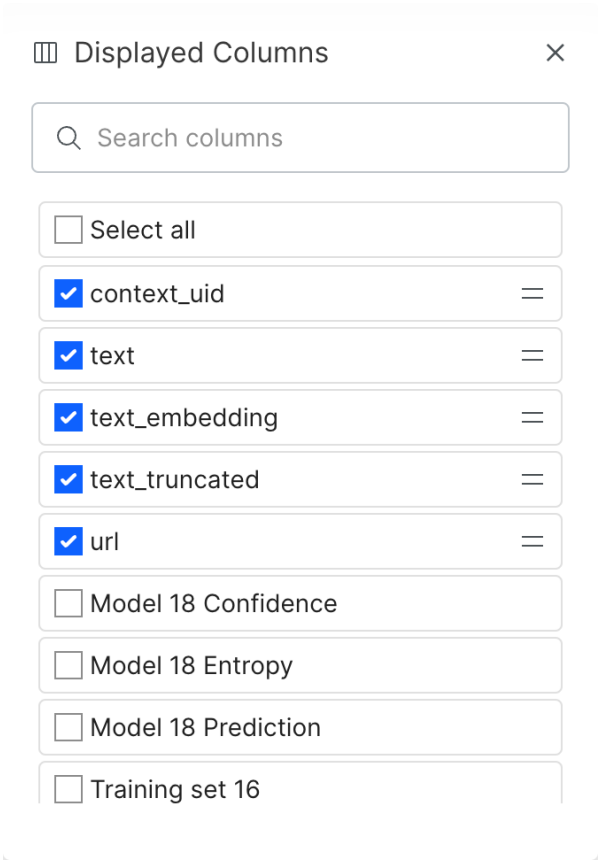
Click the **gear icon** to access various advanced control options. Depending on the view mode that you have selected, all options may not be available.

Copyright © 2025 Snorkel AI, Inc. All rights reserved



Select displayed columns

Click **Select displayed columns** to customize the data fields that are displayed in **Record view** and **Table view** accordingly.



Right-to-left (RTL) text

Toggle **Right-to-left (RTL) text** to right justify your document text when you are viewing documents in **Record view**. By default, the document text is left justified.

Auto-advance on label change

Toggle **Auto-advance on label change** to automatically advance to the next document after you change a ground truth label. This option is available when you are viewing documents in **Record view**. This saves you the time and effort of manually moving to the next document after each change.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

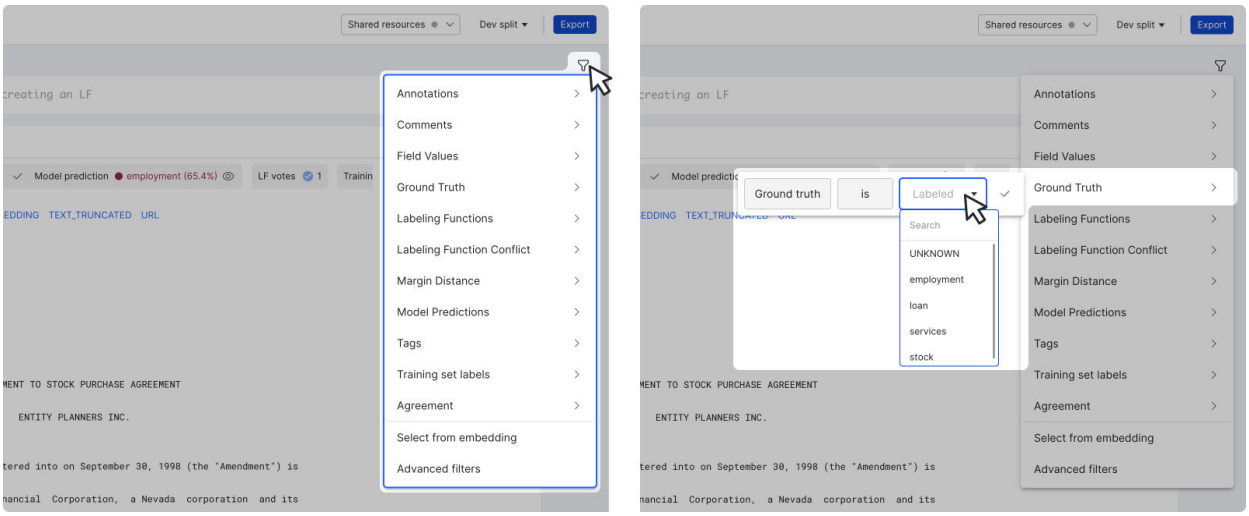
Dataviewer: Filter data

This page walks through how to filter the data points that you see in the dataviewer.

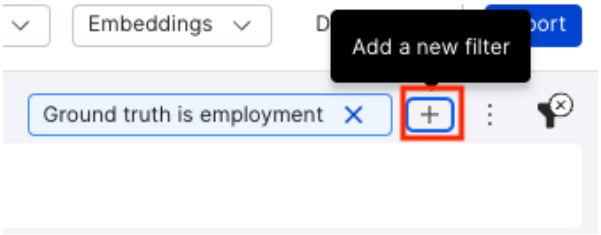
Add and delete filters

To add a filter:

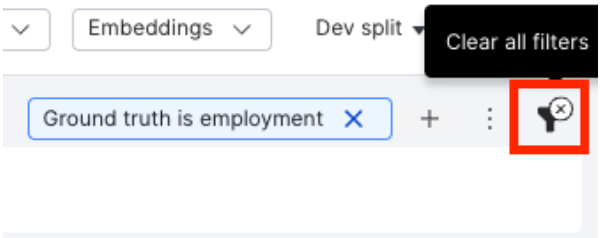
- Click the **filter** icon on the top right corner of the dataviewer to open the filter menu.
- Choose the desired filter type and input the parameters.
- Click the checkmark to save the filter.



To add more filters, click the **+** icon to open the filter menu.



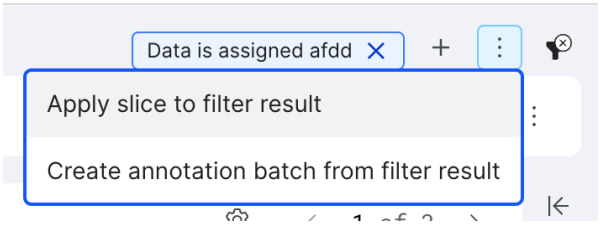
To clear all filters, click the **filter (x)** icon.



Create slices and batches from filters

To apply slices or create a batch from the filtered set, click the three dot overflow menu.

Copyright © 2025 Snorkel AI, Inc. All rights reserved



Click **Apply slices to filter result** to bulk add slices to the filtered set of data points. In the modal, you have the option to select an existing slice or create a new one.

Click **Create annotation batch from filtered set** to assign a new batch of data points to an annotator. In the modal, you can select which annotator to assign the new batch to as well as the percentage of documents from the filtered dataset to be sampled in the new batch. See Annotation Studio overview and Create batches for more information about annotating in Annotation Studio.

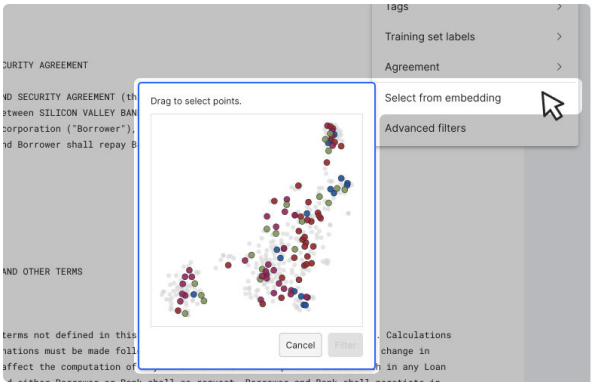
This option is particularly useful when your model is struggling to predict certain chunks of your data well. You can filter your data to just the problem documents, and then send a sample of those to annotators to get more ground truth for model training. This targeted approach decreases the amount of documents that need to be manually annotated to improve model performance, saving both time and money!

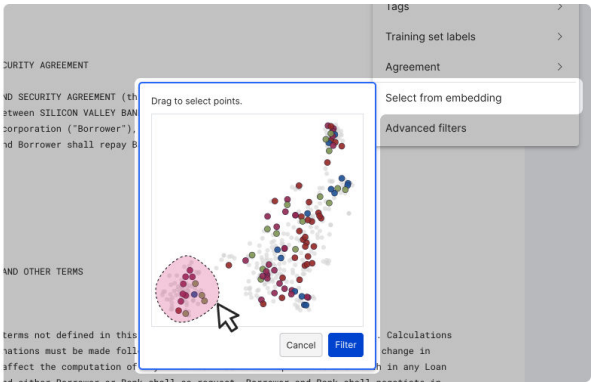
Available filters

The following filters are available in Snorkel Flow:

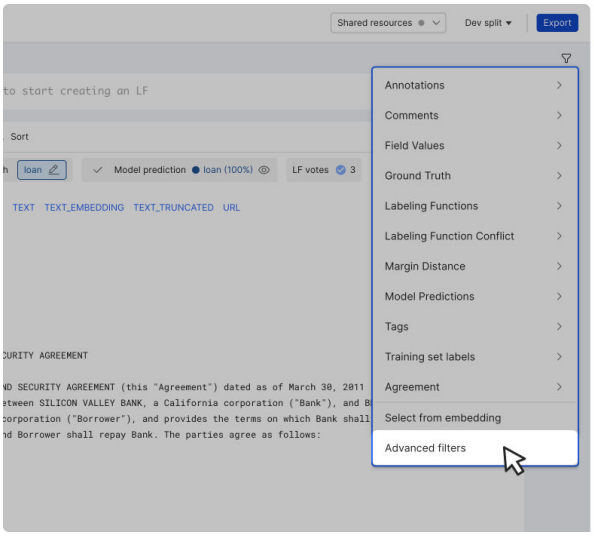
- **Annotations**: Filter by how a particular data point was labeled by an annotator. You can filter by a particular label or by whether or not the annotator labeled the data point correctly.
- **Comments**: Filter by comments from a user.
- **Field Values**: Filter by the value of one of the fields in your data. You can use a number of different operators (e.g., matches or contains).
- **Ground Truth**: Filter by a particular ground truth label.
- **Labeling Functions**: Filter by how a particular data point was labeled by a labeling function. You can filter by a particular label or by whether or not the labeling function labeled the data point correctly.
- **Labeling Function Conflict**: Filter on datapoints where the labeling functions voted incorrectly.
- **Margin Distance**: Select two labels, one being the most confident label and the other being the second most confident label, that the model predicted and specify the margin distance between them.
- **Model Predictions**: Filter by a particular label that a model predicted.
- **Slices**: Filter by data points that are or aren't included in slices.
- **Training set labels**: Filter by how a particular data point in the training set was labeled by the model. You can filter by a particular label or by whether or not the model labeled the data point correctly.
- **Agreement**: Filter by whether two annotators agreed or disagreed on a document label.
- **Select from embeddings**: Use a lasso tool to select clusters from an embedding map. This option is only available on applications with embeddings.

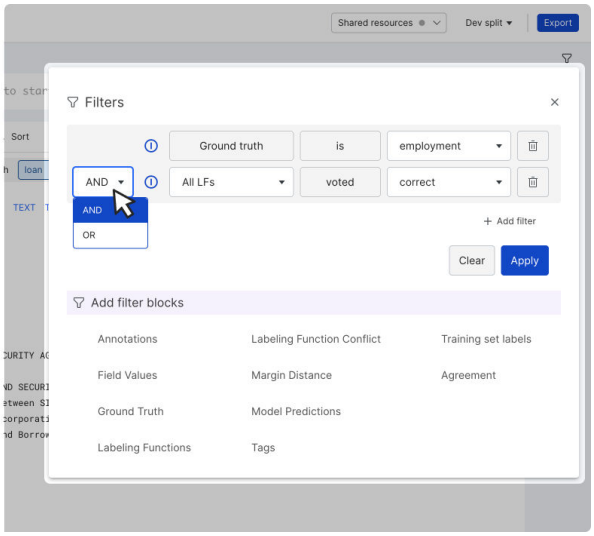
Copyright © 2025 Snorkel AI, Inc. All rights reserved





- **Advanced filters**: Further customize your filter and apply more complex logic, such as OR or NEGATE.





Copyright © 2025 Snorkel AI, Inc. All rights reserved

Data content views: Candidate-based extraction

This page walks through the various ways that you can view and interact with your data in dataviewer.

> NOTE
>
> This page covers the views and options for candidate-based extraction applications. For other application types, see Data content views: Classification or Data content views: Sequence tagging for more information.

The data content section is the main canvas to show case your data in individual or aggregated formats.

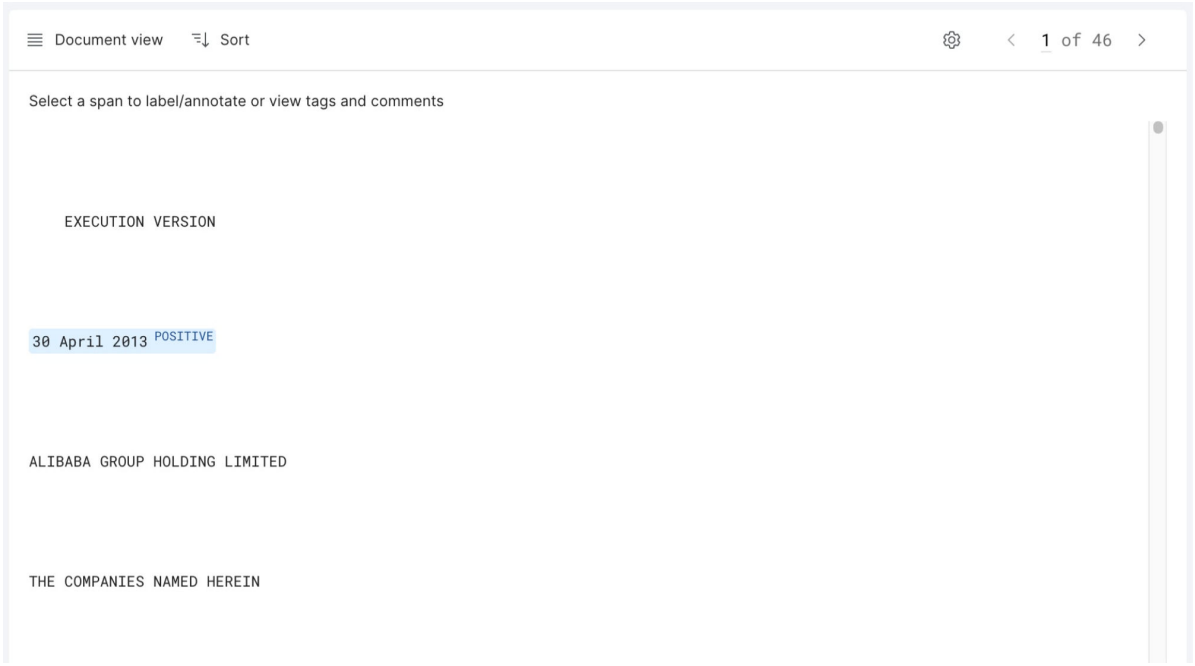
The following views are supported in candidate-based extraction applications:

- Document view
- Record view
- Snippet view
- Table view

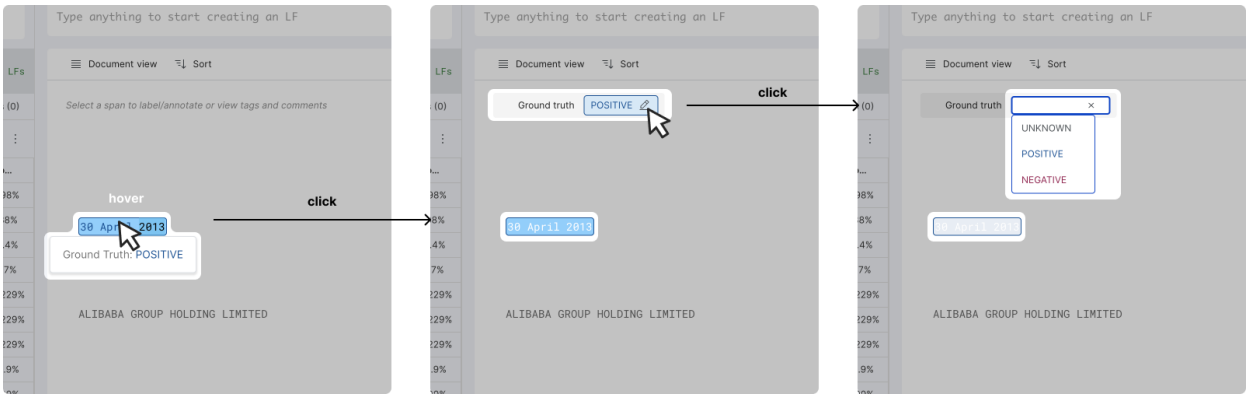
Document view

The **Document view** is the default view of data content in candidate-based extraction applications. It displays multiple spans associated with a document at once. **Document view** only displays content of the primary field, where spans are extracted.

Copyright © 2025 Snorkel AI, Inc. All rights reserved



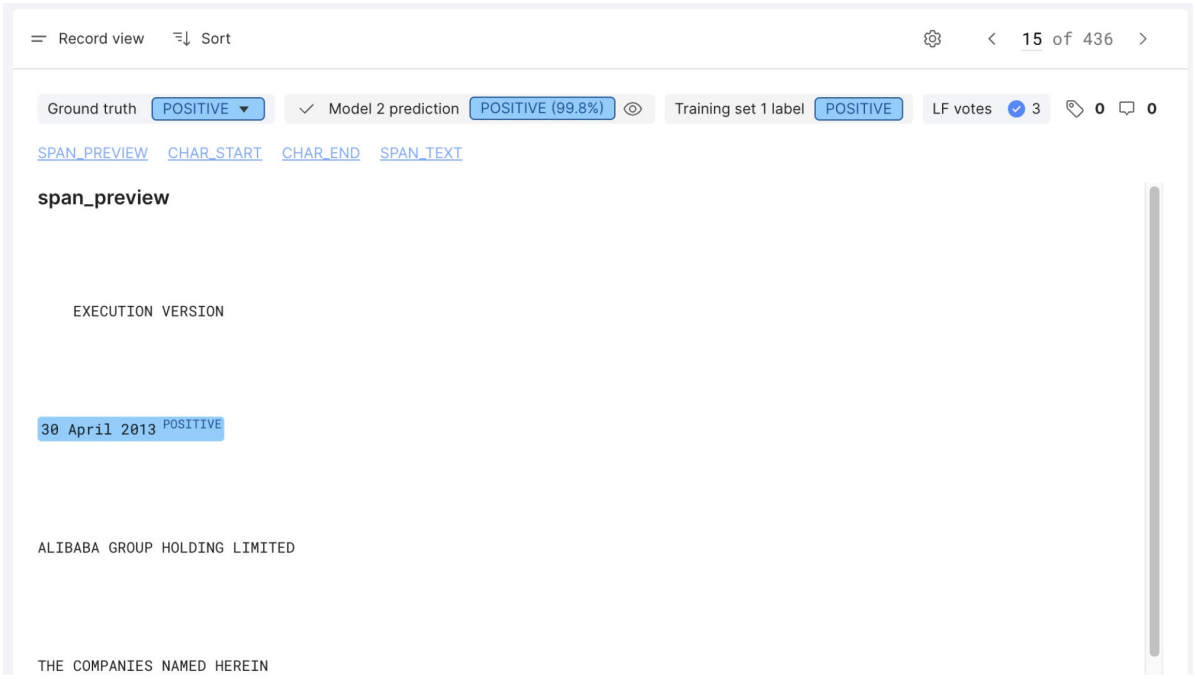
Click a span in the data content area to show the metadata row where the ground truth, training set label, slices, and comments are shown. See Record view for information about how to interact with these fields.



Record view

The **Record view** displays a singular span and its relevant context, including all fields of the document in which the span appears. The ground truth, model prediction, LF votes, slices, and comments are displayed in the metadata row is associated with the singular highlighted span.

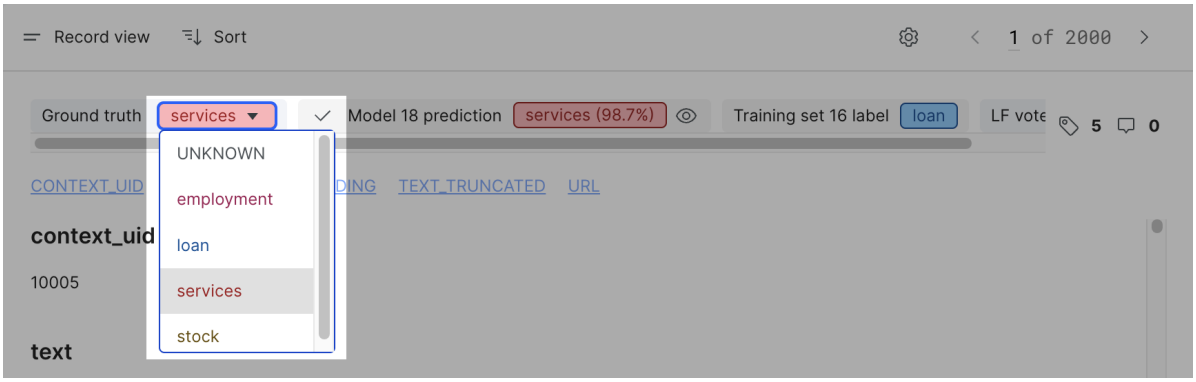
Copyright © 2025 Snorkel AI, Inc. All rights reserved



The following sections walk through the actions that you can take while in the **Record view**.

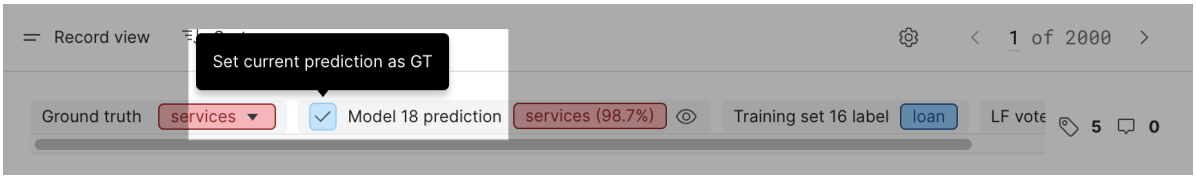
Change the ground truth of a document

In the **Ground truth** chip, click to open the dropdown list and change the ground truth of the document.



Apply model prediction as ground truth

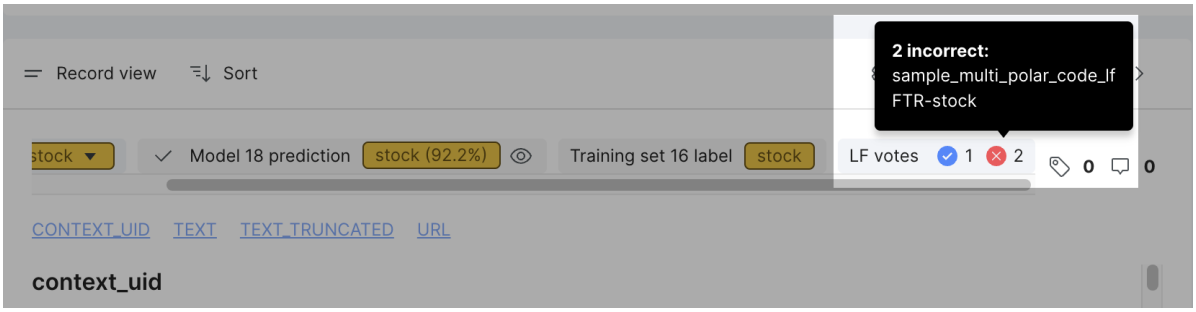
Click the **checkmark** icon to overwrite the data point's ground truth with the current model prediction. This is helpful in the error analysis stage when errors are caused by incorrect ground truth while the model prediction is correct.



View LF votes details

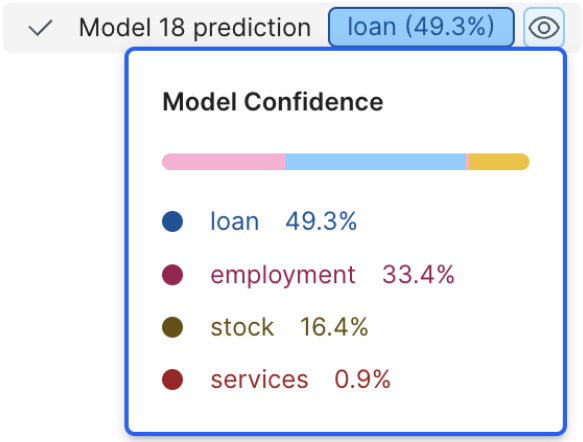
In the **LF votes** chip, hover over the **blue checkmark** icon to see all LFs that voted correctly. Hover over the **red X** icon to see all LFs that voted incorrectly.

Copyright © 2025 Snorkel AI, Inc. All rights reserved



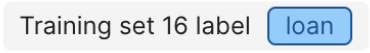
View the prediction distribution

Click the **eyeball** icon next to the **Model prediction** to view the **Prediction distribution**. The distribution represents the likelihood or probability associated with each class as predicted by the model.



View the training set label

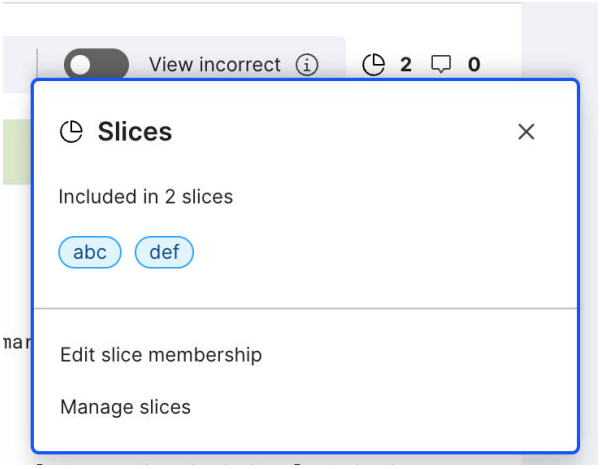
In the **Training set label** chip, the training set label is displayed.



Add and manage slices

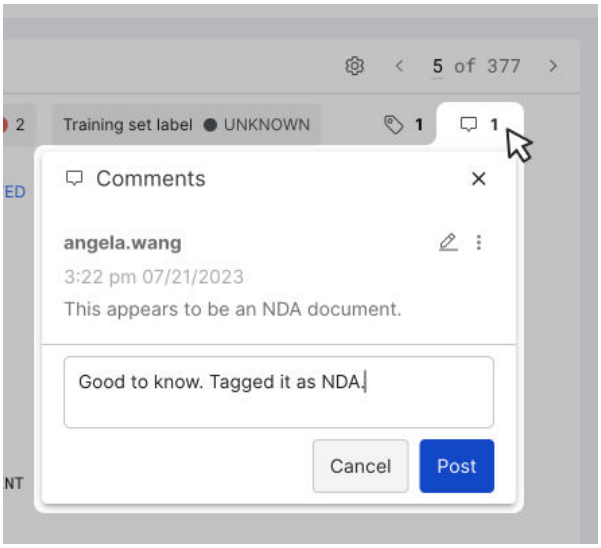
Click the slice icon to open and select from a list of slices. Slices are useful when you want to put data into specific categories to discover and perform error analysis. To manage all slices, click **Manage** to open the management pane.

Copyright © 2025 Snorkel AI, Inc. All rights reserved



View and add comments

Click the **speech bubble** icon to view all existing comments and add your comment. To find existing comments, you can filter by comments using the data filter.



Snippet view

The **Snippet view** provides a "snippet," or a short piece of data content of multiple data points. It is useful to display search results or series of data points voted by a LF, where applicable matching strings are highlighted in the snippets. Use this view to:

- Quickly confirm a desired string match and its immediate context.
- View ground truth, LF votes, etc.

The following actions are supported in **Snippet view**:

- Click on a span to navigate to the Document view for that span.
- To change the field that is displayed in **Snippet view**, use the **Display field** dropdown at the top right corner of the dataviewer.
- The **LF votes** (blue/red dots) represent how your LFs voted on the span. Hover over the blue/red dot to see the name of the LF.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

- When editing or creating an LF, any matching text pattern as described in LF is highlighted yellow in the snippets.

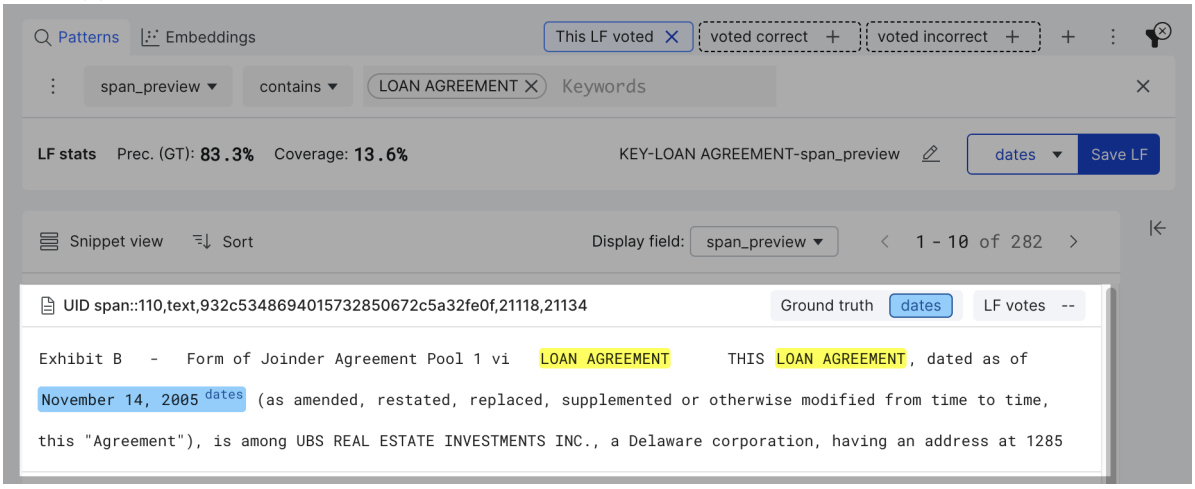
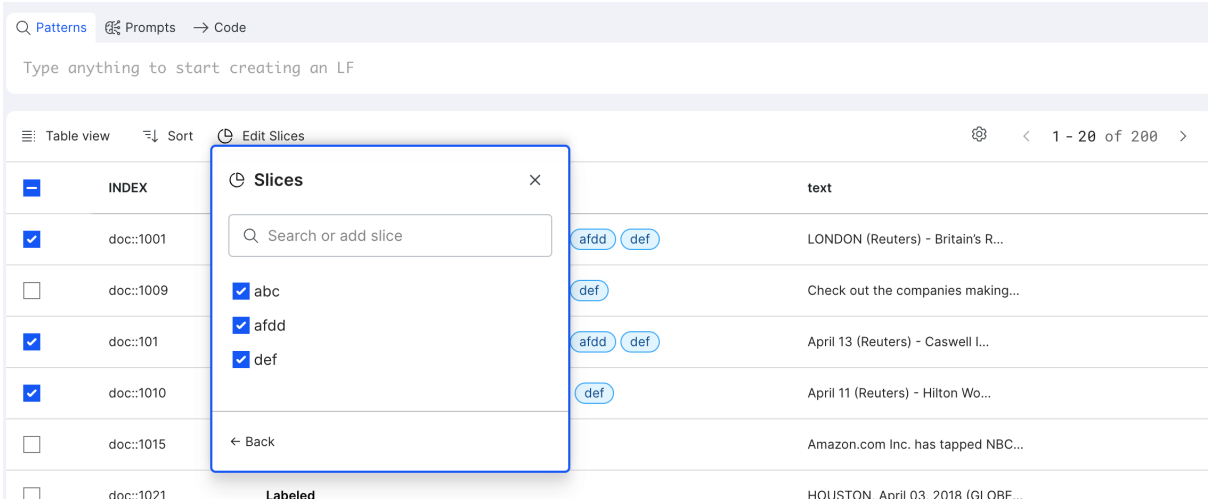


Table view

The **Table view** displays data points and their field values in a tabular format. Each row represents a singular data point (a span and its metadata). The **span_preview** column displays the span as well as the context around the span. This view is most useful to perform bulk actions to edit slices.

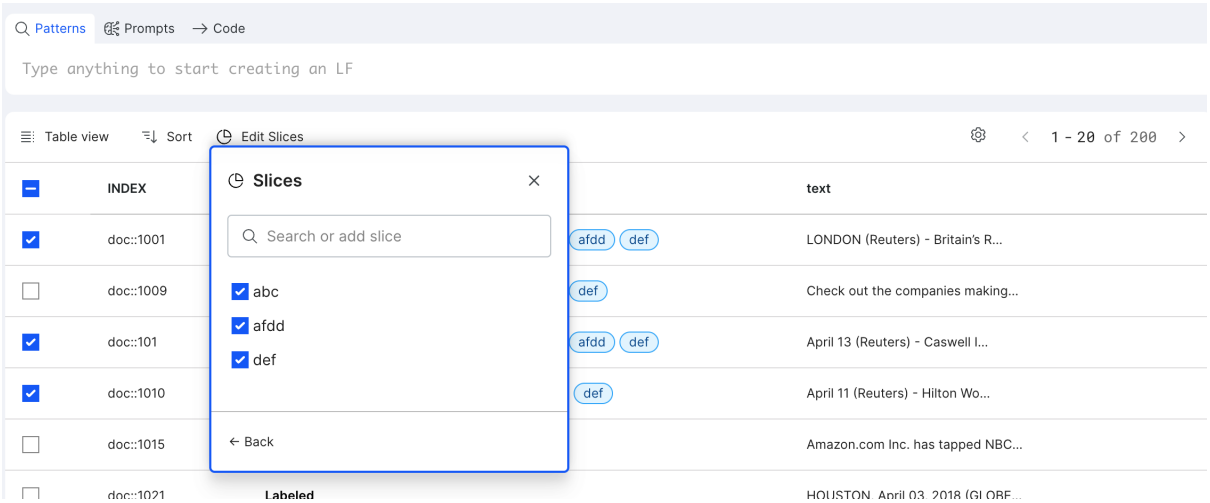
The following actions are supported in **Table view**:

- To change the fields that are shown in the table, in the advanced controls, click Select displayed columns.
- To change the sort order, click the header.



- Use the checkbox to select desired data points, then click **Edit Slices** to add, delete, or edit slices in bulk.

Copyright © 2025 Snorkel AI, Inc. All rights reserved



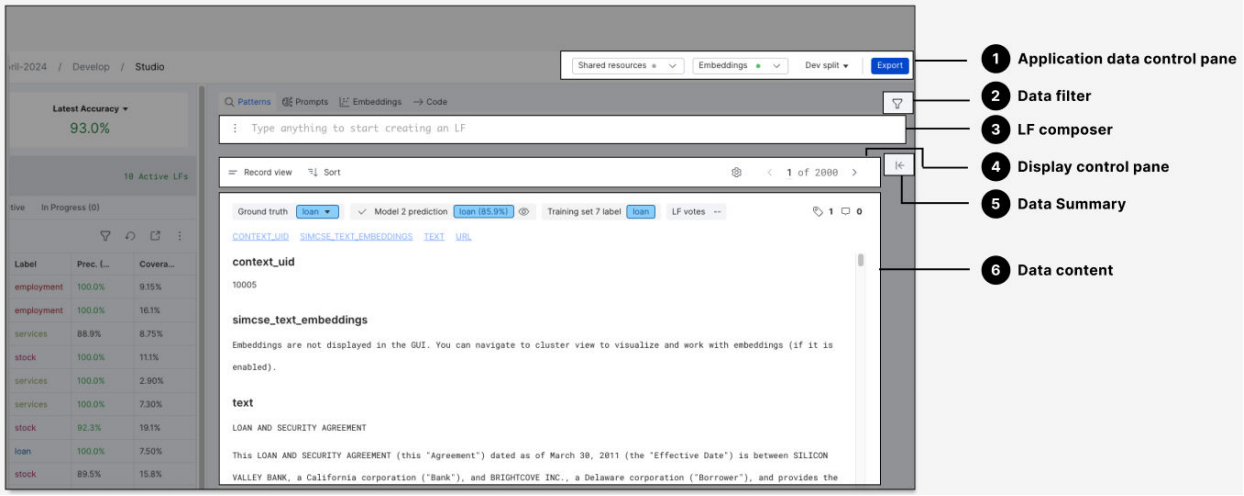
Copyright © 2025 Snorkel AI, Inc. All rights reserved

Data content views: Classification

This page walks through the various ways that you can view and interact with your data in dataviewer.

> NOTE
>
> This page covers the views and options for classification applications. For other application types, see Data content views: Candidate-based or Data content views: Sequence tagging for more information.

The data content section is the main canvas to show case your data in individual or aggregated formats.



The following views are supported in classification applications:

- Record view
- Snippet view
- Table view

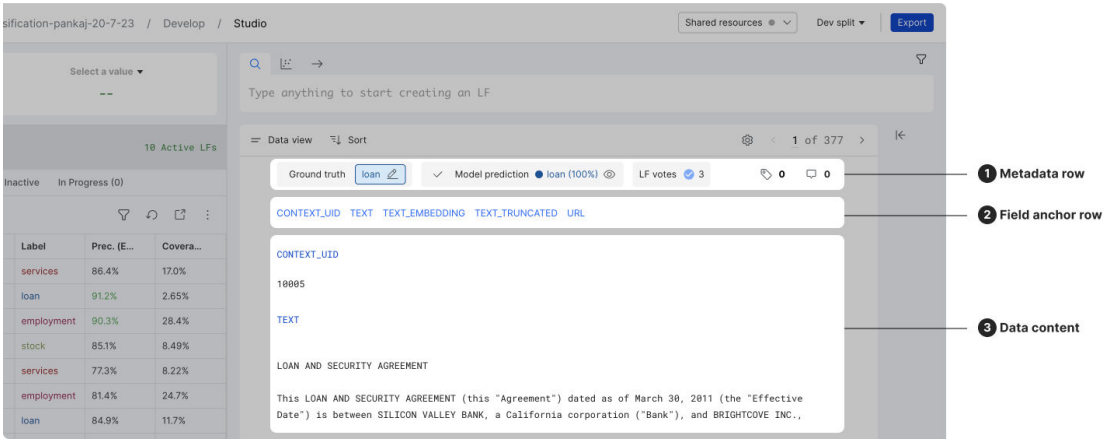
Record view: View and edit metadata

Record view is the default view of data content in classification applications. For a singular data point, it displays key metadata as well as the full content of each field in a list format. Use this view to:

- View metadata (e.g., ground truth, model prediction, labeling function (LF) votes, and training set label)
- Edit metadata
- View data content

The **Record view** in classification applications includes a metadata row where ground truth, model prediction, LF votes, slices, and comments are shown. Below the metadata row, the data content is displayed in a list format, with an anchor row to quickly navigate to specific filed content.

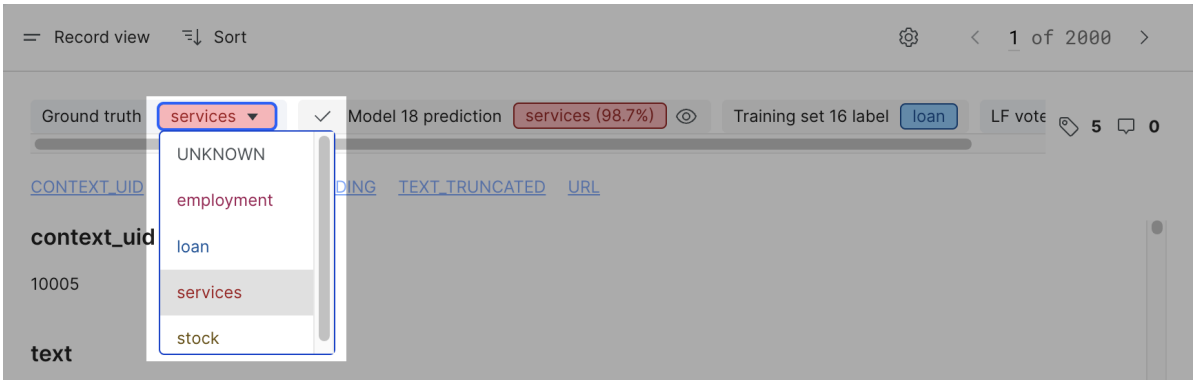
Copyright © 2025 Snorkel AI, Inc. All rights reserved



The following sections walk through the actions that you can take while in the **Record view**.

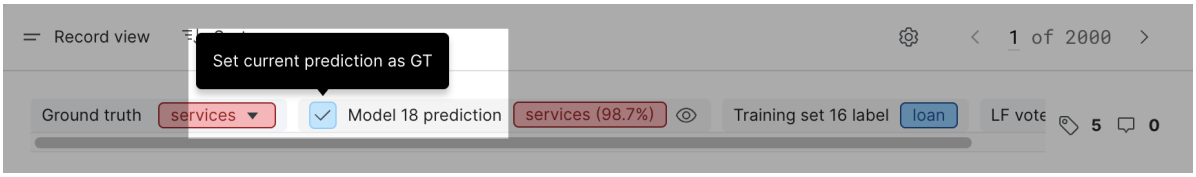
Change the ground truth of a document

In the **Ground truth** chip, click to open the dropdown list and change the ground truth of the document.



Apply model prediction as ground truth

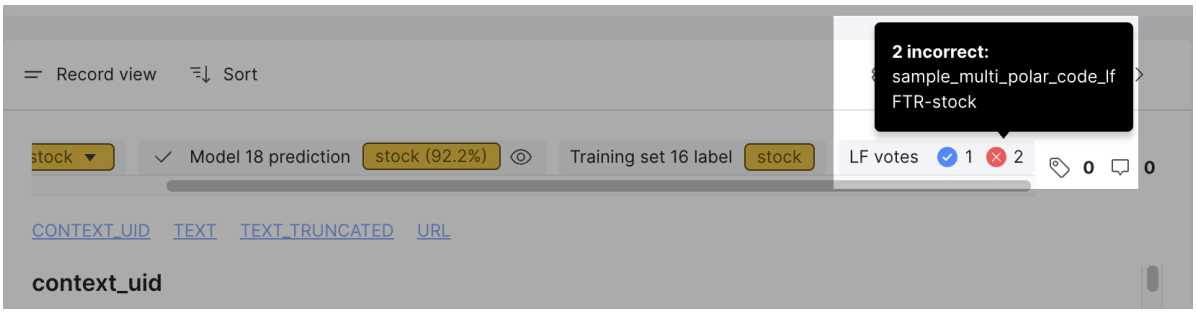
Click the **checkmark** icon to overwrite the data point's ground truth with the current model prediction. This is helpful in the error analysis stage when errors are caused by incorrect ground truth while the model prediction is correct.



View LF votes details

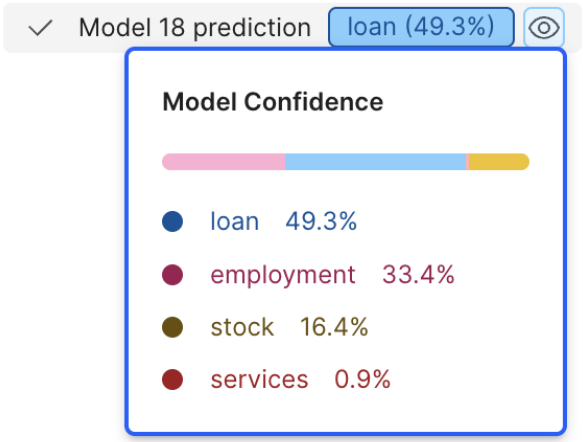
In the **LF votes** chip, hover over the **blue checkmark** icon to see all LFs that voted correctly. Hover over the **red X** icon to see all LFs that voted incorrectly.

Copyright © 2025 Snorkel AI, Inc. All rights reserved



View the prediction distribution

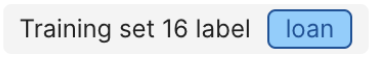
Click the **eyeball** icon next to the **Model prediction** to view the **Prediction distribution**. The distribution represents the likelihood or probability associated with each class as predicted by the model.



> **NOTE**
>
> If you train a model with `Tune threshold on valid = True`, the predicted label might not be the label with the highest probability. For this example, the tuned threshold is 70%. If the prediction probability for the positive label is 65% and the prediction probability for the negative label is 35%, the predicted label will be `negative`, even though it has a lower probability. This label is because the probability of the positive label is lower than the threshold.

View the training set label

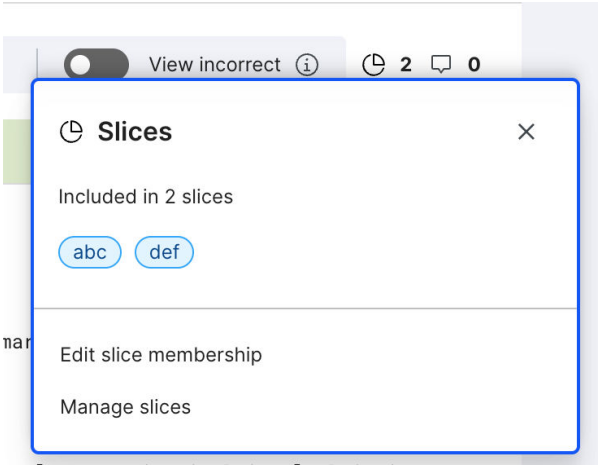
In the **Training set label** chip, the training set label is displayed.



Add and manage slices

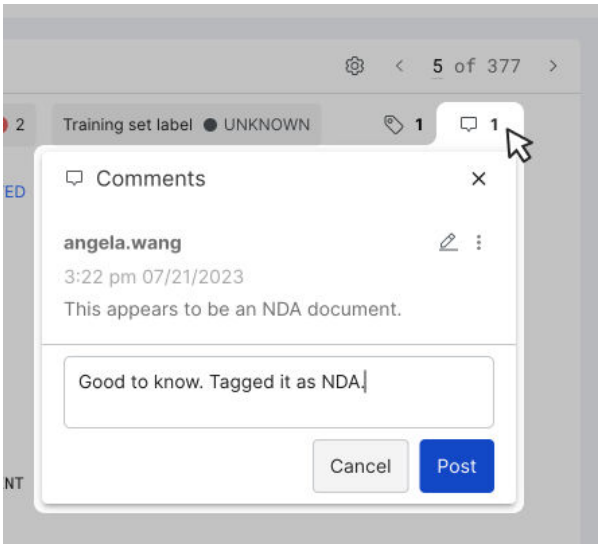
Click the slice icon to open and select from a list of slices. Slices are useful when you want to put data into specific categories to discover and perform error analysis. To manage all slices, click **Manage** to open the management pane.

Copyright © 2025 Snorkel AI, Inc. All rights reserved



View and add comments

Click the **speech bubble** icon to view all existing comments and add your comment. To find existing comments, you can filter by comments using the data filter.



Snippet view

The **Snippet view** provides a "snippet," or a short piece of data content of multiple data points. It is useful to display search results or a series of data points that were voted by a LF, where applicable matching strings are highlighted in the snippets. Use this view to:

- Quickly confirm a desired string match and its immediate context.
- View ground truth, LF votes, etc.

The following actions are supported in **Snippet view**:

- **Ground truth** (text) and **LF votes** (blue/red dots) are displayed at the document level. Each document is displayed once.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

- To change the field that is displayed in **Snippet view**, use the **Display field** dropdown at the top right corner of the dataviewer.

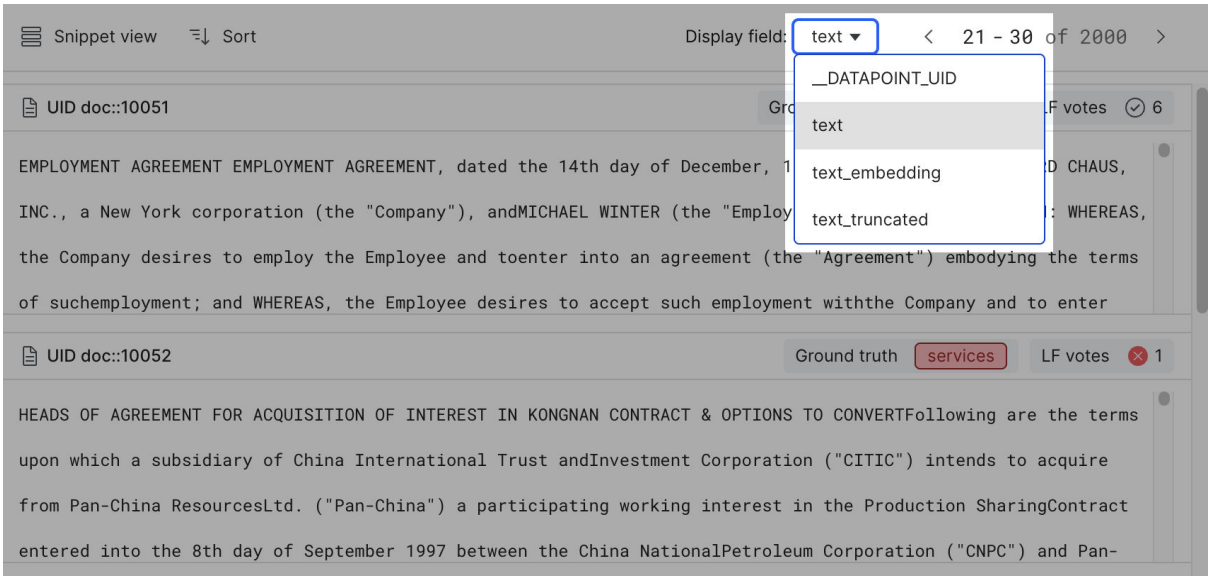


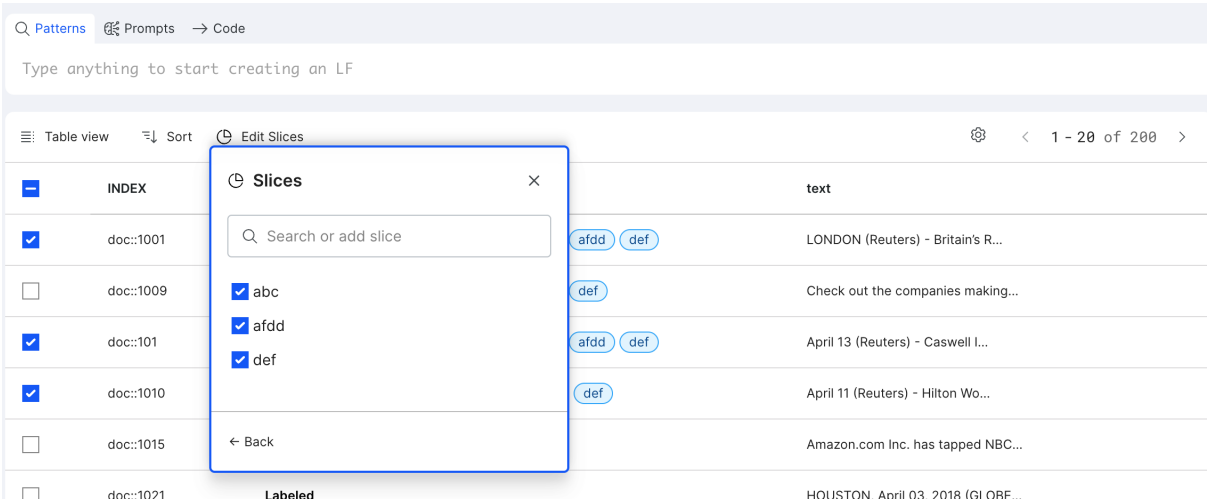
Table view

The **Table view** displays data points and their field values in a tabular format. Each row in the table represents a single data point. This view is most useful to perform bulk actions to edit slices.

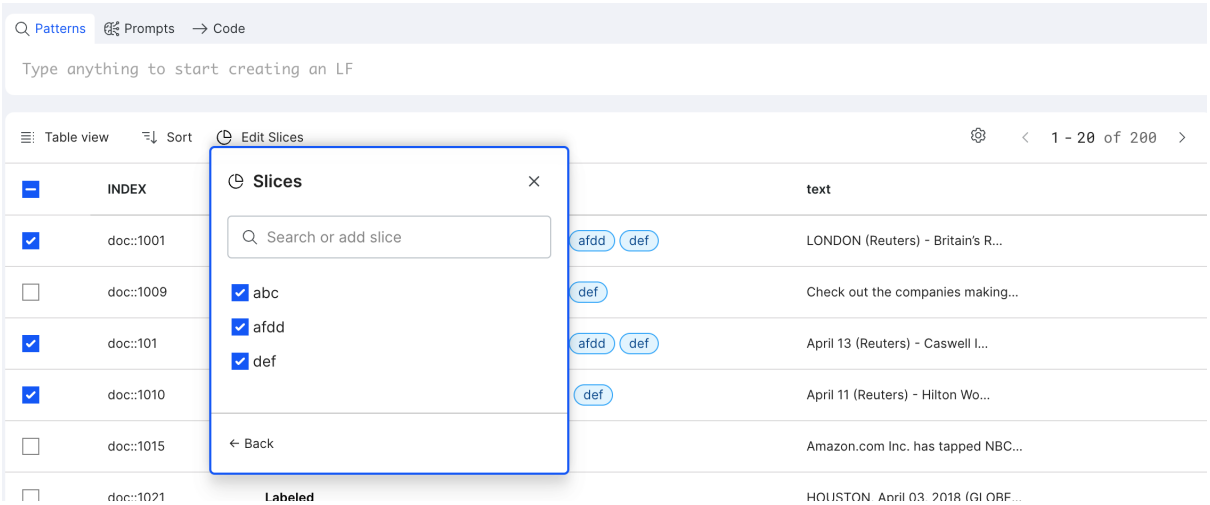
The following actions are supported in **Table view**:

- To change the fields that are shown in the table, in the advanced controls, click Select displayed columns.
- To change the sort order, click the header.

Copyright © 2025 Snorkel AI, Inc. All rights reserved



- Use the checkbox to select desired data points, then click **Edit Slices** to add, delete, or edit slices in bulk.

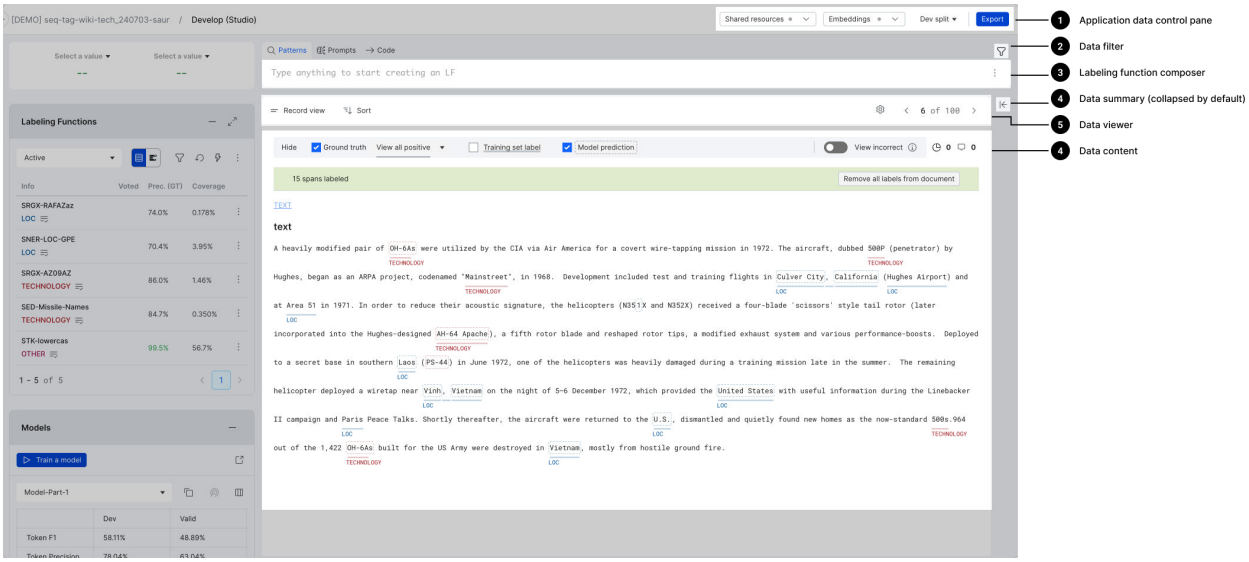


Copyright © 2025 Snorkel AI, Inc. All rights reserved

Data content views: Sequence tagging

This article explains the different ways you can view and interact with your data for sequence tagging applications. You'll learn what types of data views are available, and how you can use each view to support sequence tagging tasks.

> **NOTE**
>
> This page focuses specifically on sequence tagging applications. If you're working with other application types, check out our guides for Data content views: Classification or Data content views: Candidate-based extraction.



The data content section is the main canvas to showcase your data in individual or aggregated formats. The following views are supported in sequence tagging applications:

- Record view: Inspect individual datapoints
- Snippet view: Compare small sections across datapoints
- Table view: See aggregated datapoints in a tabular format

Record view: View and edit metadata

Record view is the default view of data content in sequence tagging applications. It displays each individual datapoint along with its associated fields and metadata. Use this view to:

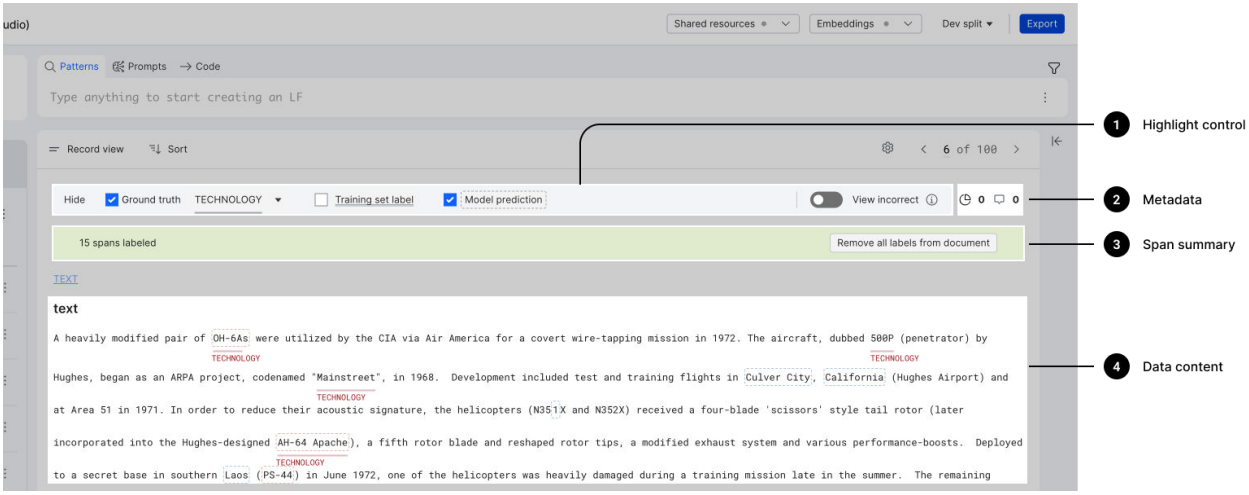
- View metadata (e.g., ground truth, model prediction, labeling function (LF) votes, and training set label)
- Edit metadata
- Add spans
- View data content

For sequence tagging applications, the **Record view** includes:

- A highlight control pane.
- Entry points to document-level slices and comments.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

- A data content area where you can view and edit metadata associated with individual spans in a document.

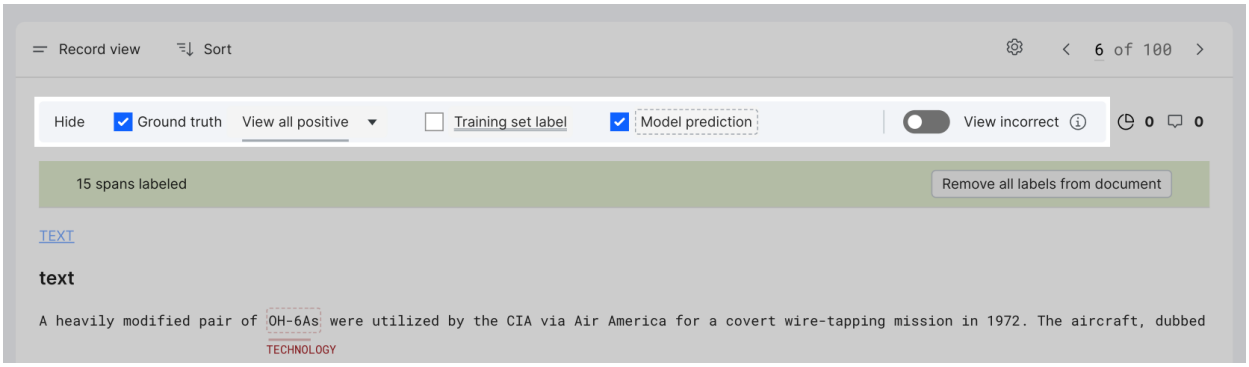


The following sections walk through the actions that you can take while in the **Record view**.

Use the highlight control pane

The **Highlight control** pane allows you to turn on/off highlights. Use the dropdown to select specific types of highlights, and scroll horizontally to access all available controls. The available controls include:

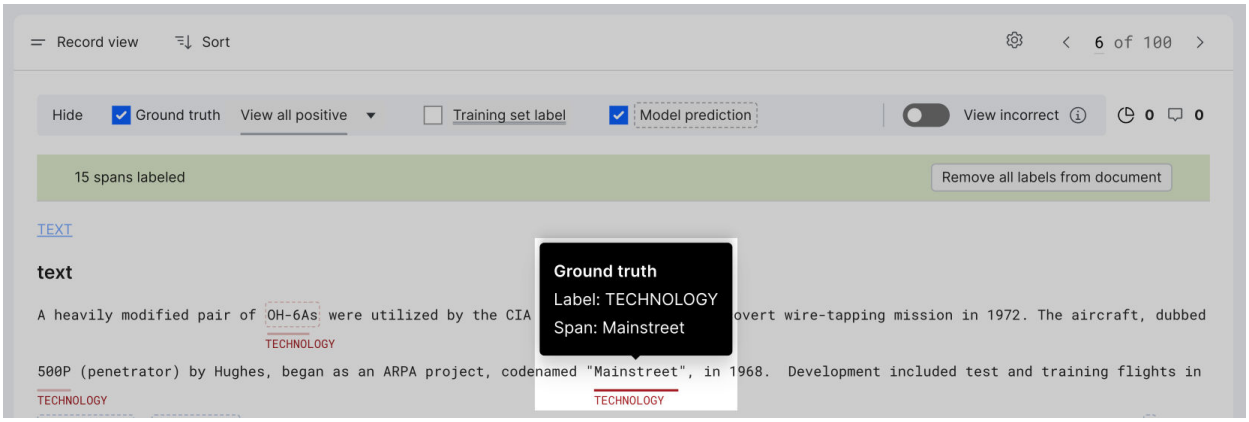
- **Ground truth:** View all labels, or view a specific label
- **LF vote:** See how a LF voted on spans
- **Training set label:** View the label applied during training
- **Model prediction:** Display the predictions of the current model
- **Spotlight mode:** Focuses on any instance where Training set (TS) and/or Model prediction (MP) disagree with Ground truth (GT).



View the ground truth of a span

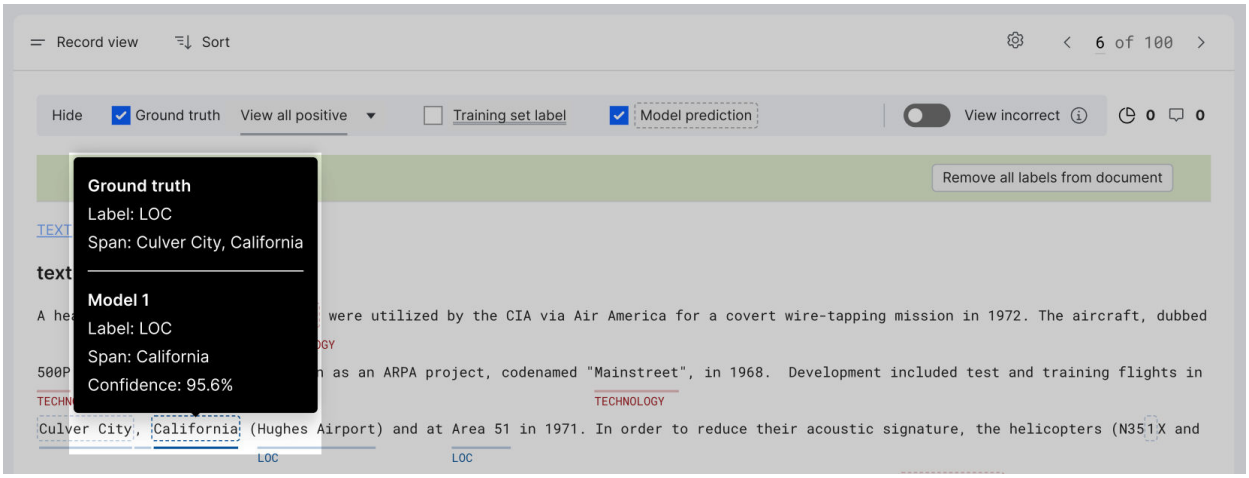
When the **Ground truth** highlight is turned on (in the highlight control pane), select a span highlight to see the ground truth label and span.

Copyright © 2025 Snorkel AI, Inc. All rights reserved



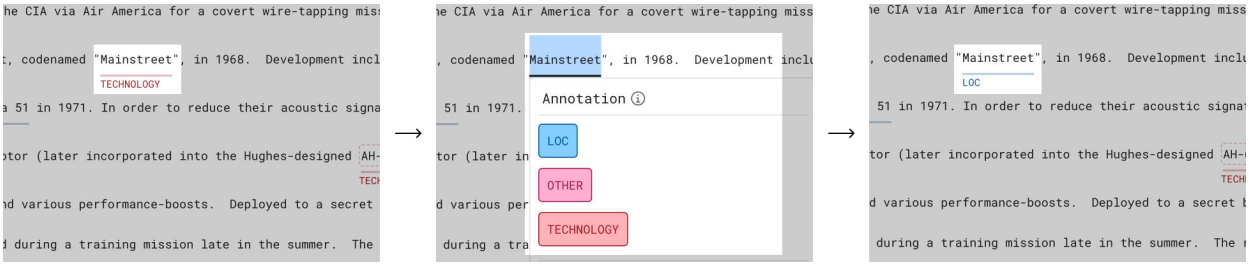
View information of overlapping spans

When multiple highlights are enabled (in the highlight control pane), you might encounter overlapping spans. Click anywhere in the overlapping span to show a tooltip containing full metadata information for the span.



Add or change the ground truth of a span

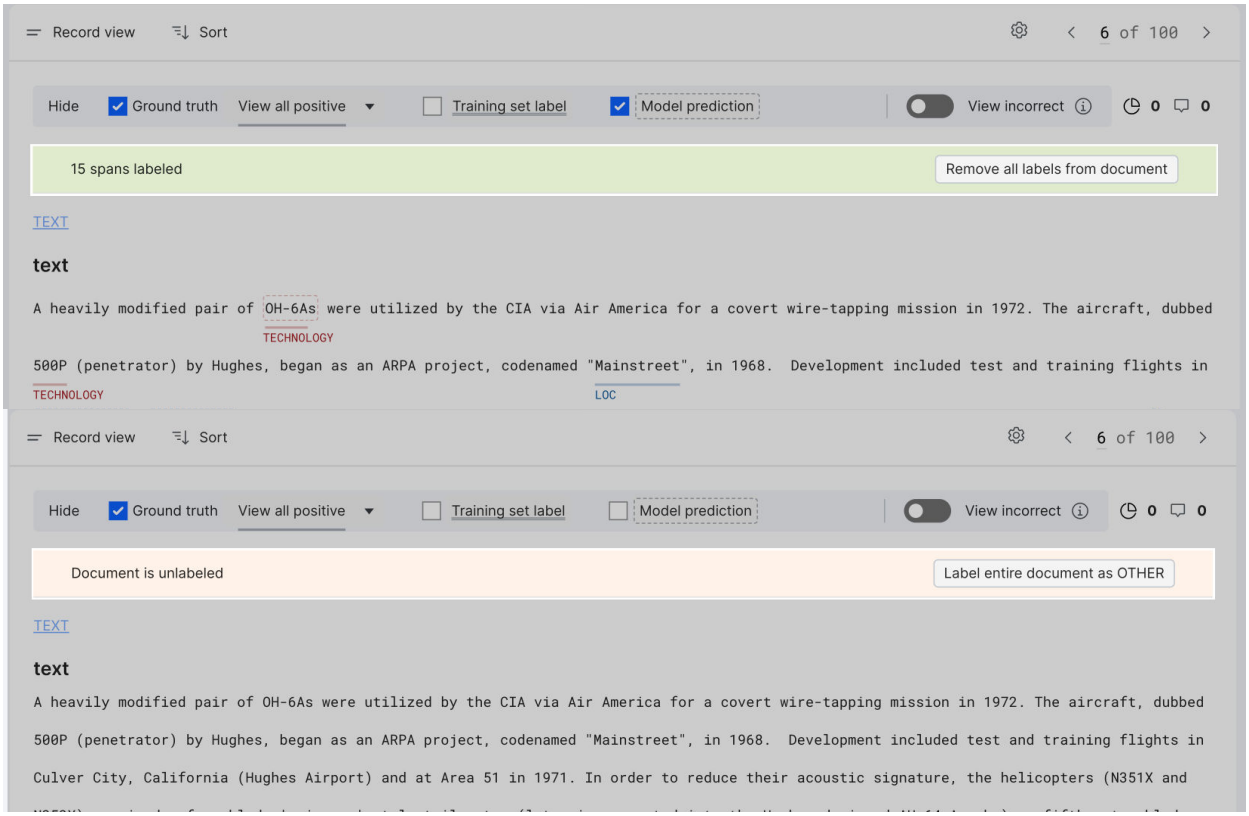
Use the cursor to highlight the desired span text, and then use the modal that appears to add or change the ground truth label for that span.



Remove all ground truth labels

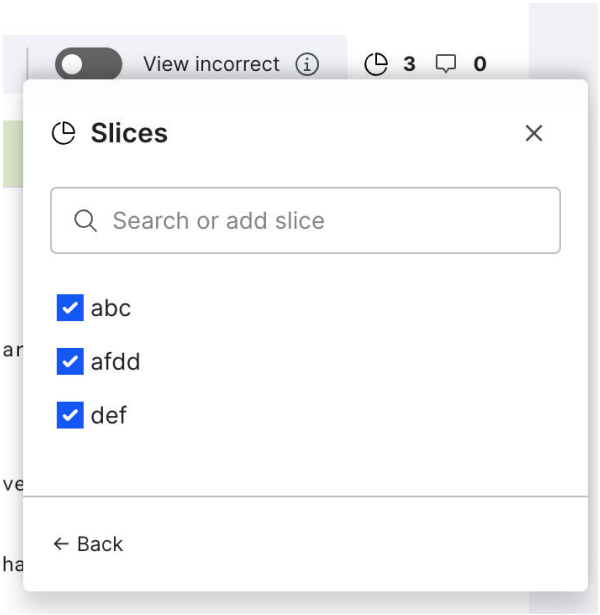
Click the **Remove all labels from document** button in the green bar below the highlight control row to remove all ground truth labels.

Copyright © 2025 Snorkel AI, Inc. All rights reserved



Add and manage slices

To categorize data for better discovery or error analysis, you can add or manage slices. Select the slice icon to open and select from a list of slices. To manage all slices, select **Manage** to open the management pane.

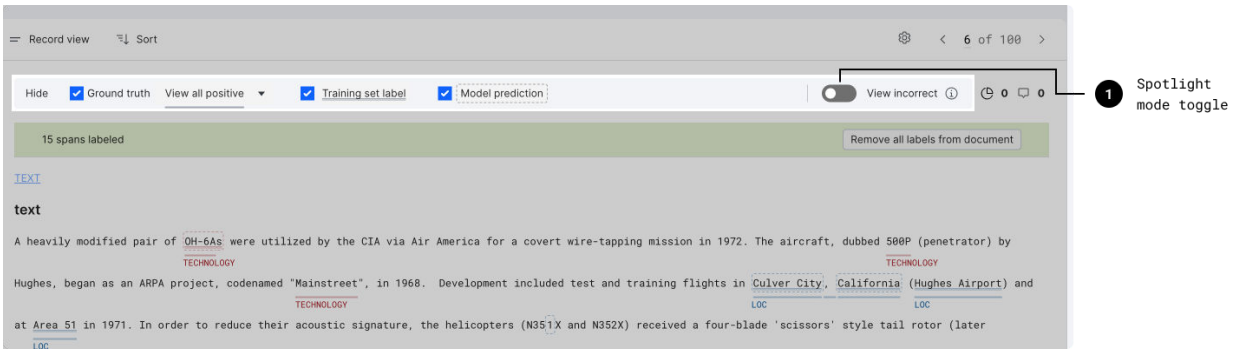


Find errors with spotlight mode

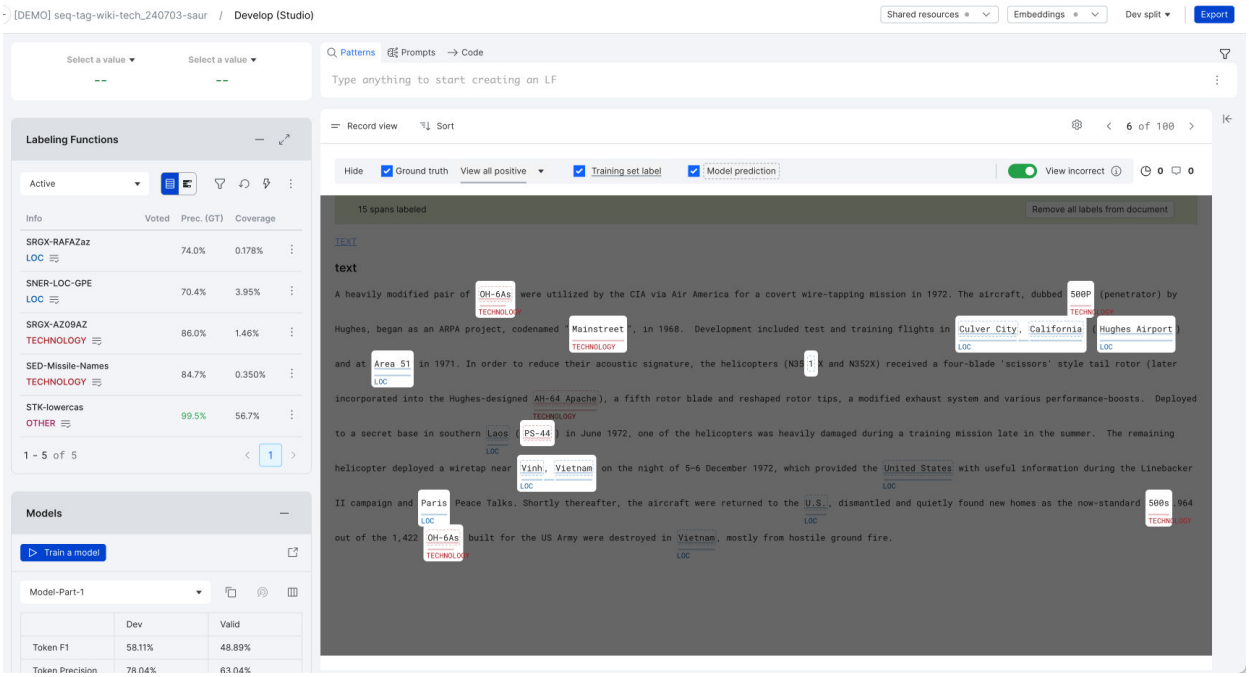
Spotlight mode focuses the record view on any instance where training set (TS) or model prediction (MP) disagree with the ground truth (GT). This mode does not filter the data.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

Turn spotlight mode on and off from the highlight panel within the data viewer.

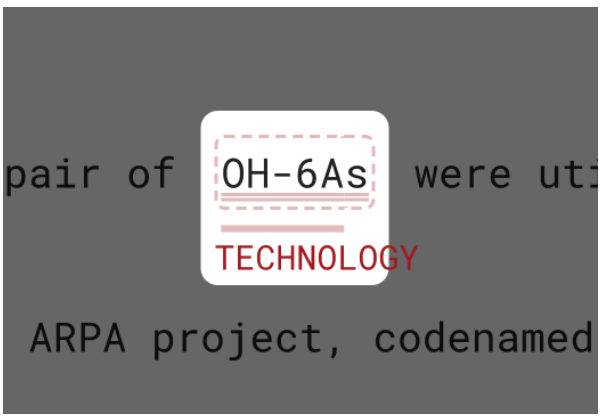


With spotlight mode toggled on, you can see where TS and MP disagree with GT.



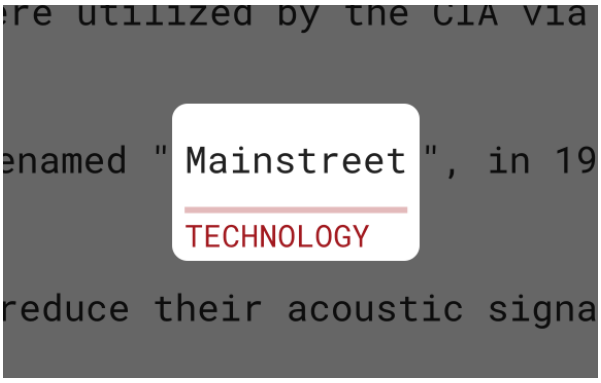
When you take a closer look at the first spotlight using the legend, you can see the GT is represented as a solid underline. The GT class of technology is colored red and has the accompanying label `TECHNOLOGY`. TS is represented as a double underline and MP as a dashed box.

The GT correctly labeled “OH-6A” as `TECHNOLOGY`. Both the TS and MP incorrectly labeled “OH-6As” as `TECHNOLOGY`, capturing an extra “s”.

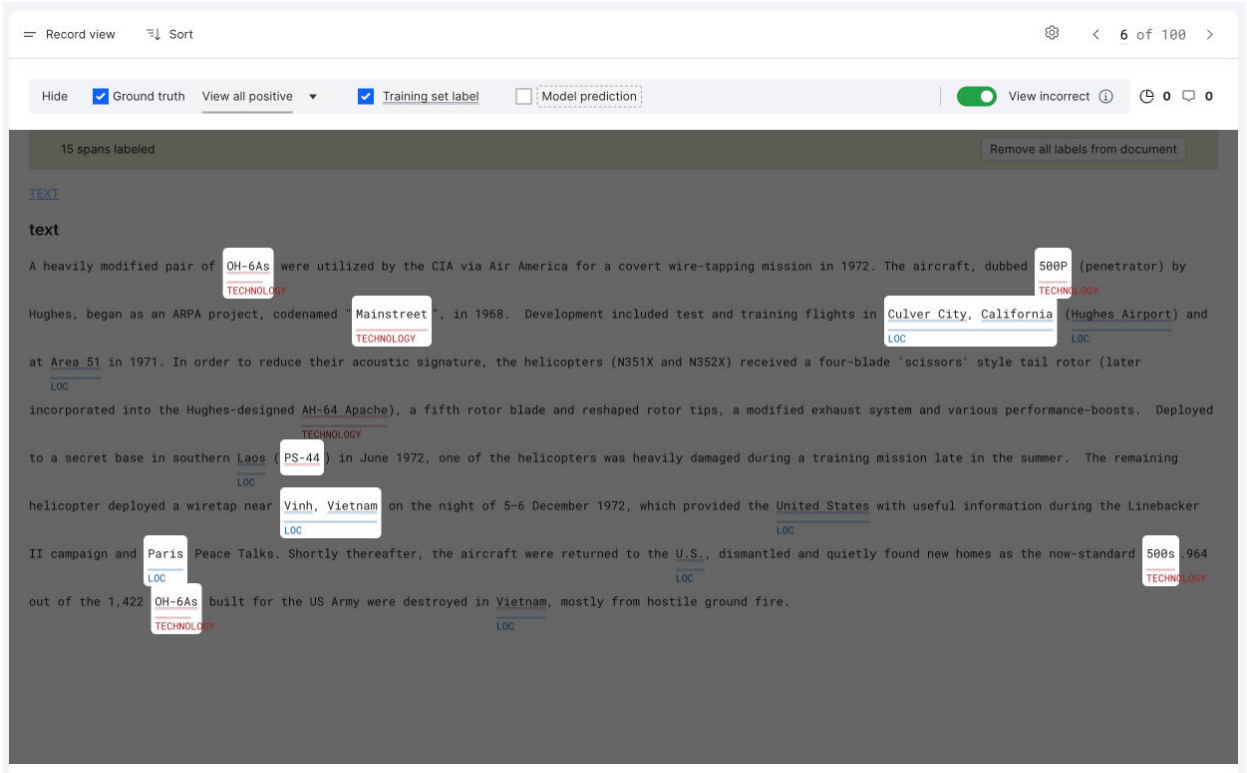


Copyright © 2025 Snorkel AI, Inc. All rights reserved

In another example, “Mainstreet” is labeled as GT `TECHNOLOGY`. Both the TS and MP for technology missed it.

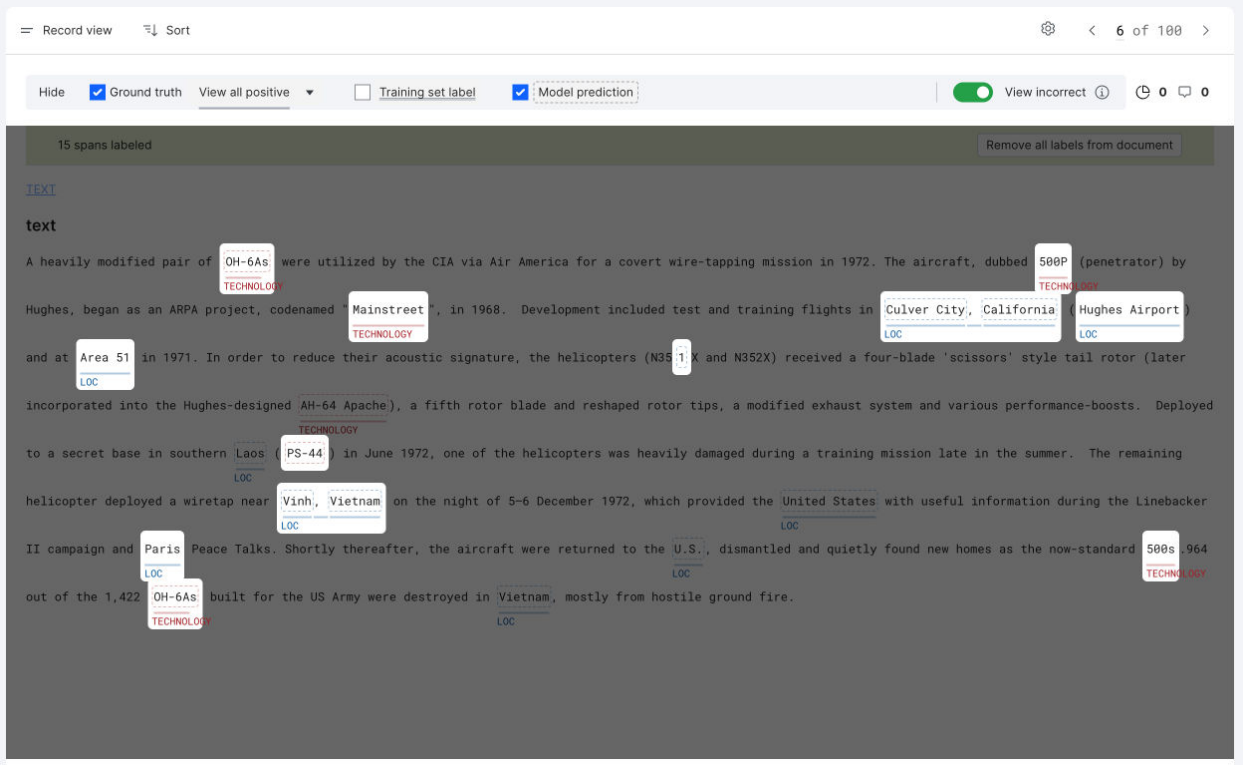


If you’d like to focus on only instances where TS conflicts with GT, then uncheck the box next to MP.



If you’d like to focus on only instances where MP conflicts with GT, then uncheck the box next to TS.

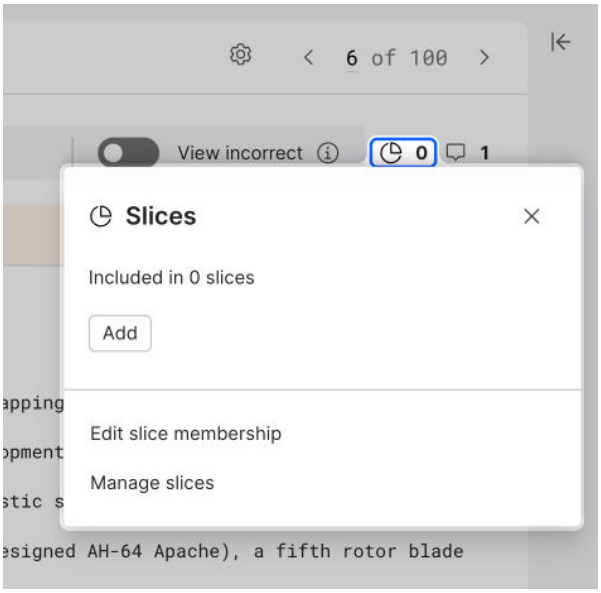
Copyright © 2025 Snorkel AI, Inc. All rights reserved



While spotlight mode doesn't filter data, it does work with filters applied.

Add and manage slices

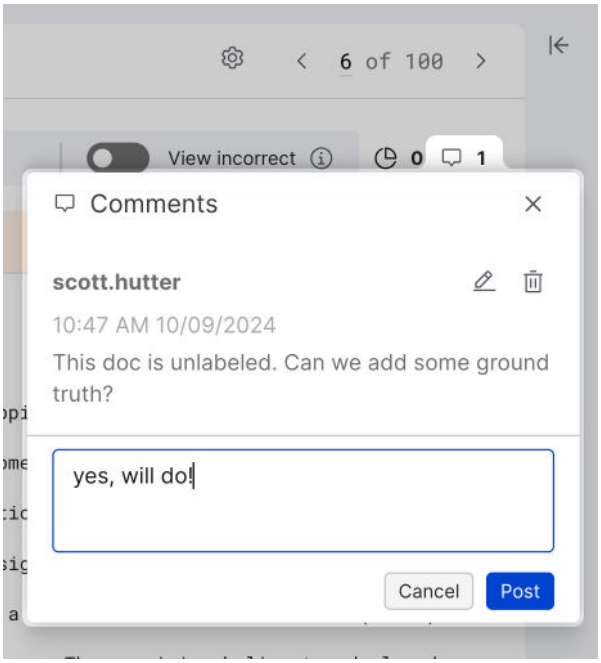
To categorize data for better discovery or error analysis, you can add or manage slices. Select the **slice** icon to open and select from a list of slices. To manage all slices, click **Manage** to open the management pane.



View and add comments

Click the **speech bubble** icon to view all existing comments and add your comment. To find existing comments, you can filter by comments using the data filter.

Copyright © 2025 Snorkel AI, Inc. All rights reserved



Snippet view

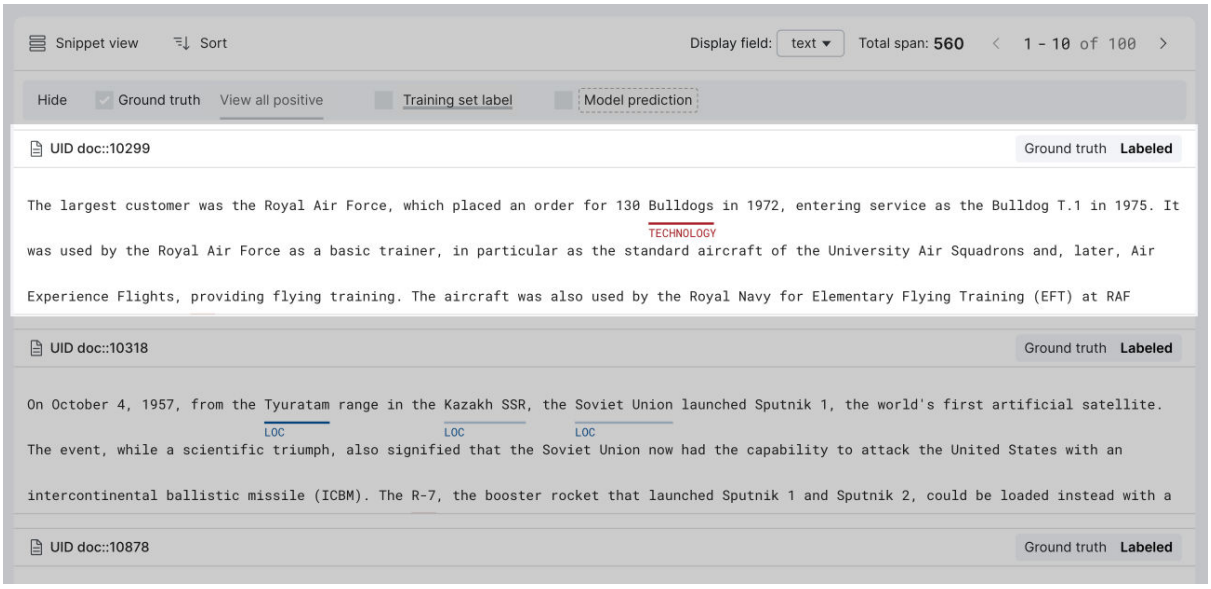
The **Snippet view** provides a "snippet", or a short piece of data content of multiple data points. It is useful to display search results or series of data points voted by a LF, where applicable matching strings are highlighted in the snippets. Use this view to:

- Quickly confirm string matches and its immediate context.
- View ground truth, LF votes, etc.

For sequence tagging applications, the **Snippet view** includes a highlight control pane.

The following actions are supported in **Snippet view**:

- **Scrolling within snippets** to display more spans within a document.
- Navigate to the Record view by clicking on a span.



- When editing or creating a LF, matching text patterns are highlighted in yellow.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

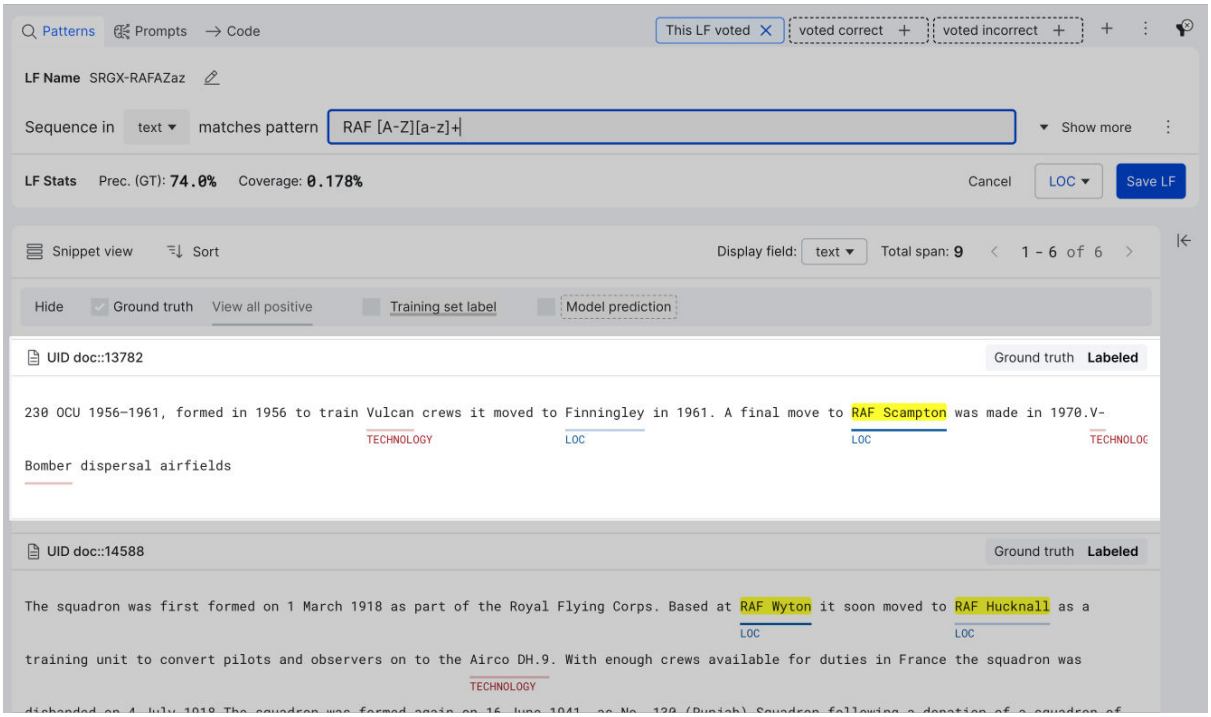
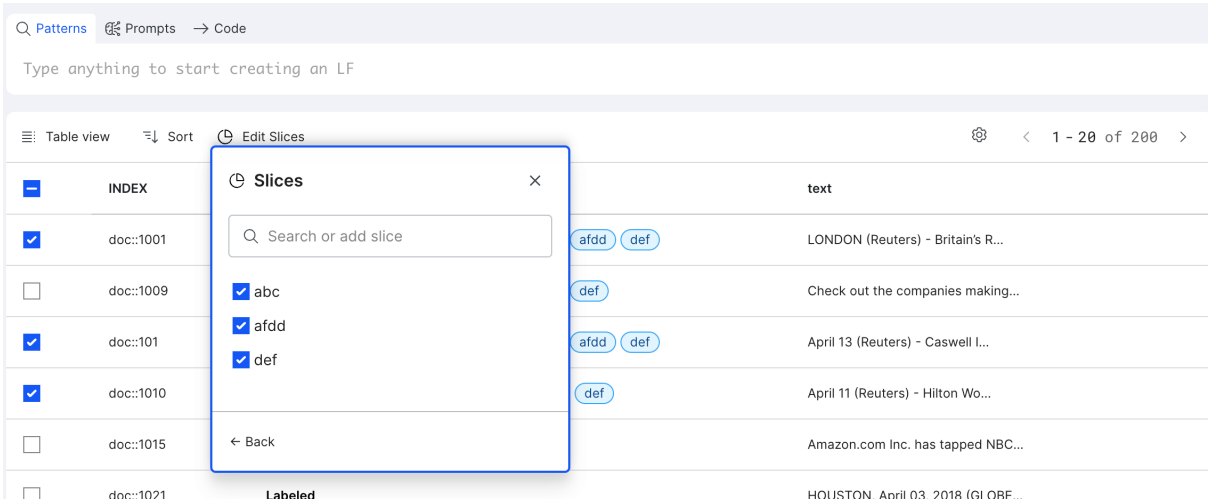


Table view

The **Table view** displays your data points in a tabular format, with each row representing a document. This view is most useful to perform bulk actions to edit slices.

The following actions are supported:

- **Customize columns** by using the advanced controls > click on Select displayed columns.
- **Sort** the data by clicking on column headers.
- Use the checkbox to select desired data points, then click **Edit Slices** to add, delete, or edit slices in bulk.



Copyright © 2025 Snorkel AI, Inc. All rights reserved

Tutorial: Document classification for SEC filings

In this tutorial, you'll use Snorkel Flow to train and build a model that classifies over 20,000 SEC filings into one of the following categories: employment, services, stock, and loan. With Snorkel Flow's programmatic labeling, you'll label your training data using Labeling Functions (LFs), train your models, and analyze the results, guiding you through an iterative development process.

This tutorial is designed for first-time users of Snorkel Flow who are new to data-centric AI, programmatic labeling, and machine learning models.

What you'll learn

- Uploading data into Snorkel Flow
- Creating a new application
- Programmatically labeling your data with Labeling Functions (LFs)
- Training your Machine Learning (ML) model
- Analyzing model errors and results

Concept: Data splits

In this tutorial, you'll upload four data splits for the example application, each with a specific purpose for model development:

- **Train:** The initial training dataset used to develop your machine learning model. This split does not need to have labels. However, this tutorial includes a small number of labels for development purposes, to help you understand the classification problem. This tutorial walks you through how to build consistent labels in the data in your `train` split.
- **Dev:** A randomly sampled dataset used for developing Labeling Functions. To limit the amount of data in memory, Snorkel Flow randomly samples the `train` split to create the `dev` split. By default, it samples 10% or up to 2,000 samples. The `dev` split dataset is viewable from your application's **Develop > Studio** page.
- **Test:** The ground truth dataset held out from the training data for evaluation purposes. While not strictly necessary, it's recommended to use the `test` split for a final evaluation of your model against a fresh set of expert-annotated labels.

> ⚠ CAUTION
> To avoid biasing your evaluation procedure, do not look at data points in the `test` split.

- **Valid:** Like `test`, the `valid` split contains some ground truth labels for evaluation purposes. This split can be used to tune your ML model.

For more information about uploading and splitting data, see the Uploading a dataset article.

Upload data

Before you create the example Snorkel Flow classification application, you need to upload the data you want to classify.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

You will upload this data to a new dataset in your Snorkel Flow instance. Data can be ingested into Snorkel Flow from a range of sources, including cloud storage, databases, and local files. For this tutorial, use the provided dataset from AWS S3 Cloud Storage. This is the set of SEC filings that you will classify with Snorkel Flow.

1. From the left menu, select **Datasets**.
2. Select **+ Upload new dataset**.
3. In the **Upload new dataset** dialogue, enter the following information:
   - **Dataset name:** Enter `contract-clf` as your dataset name.
   - **Select a data source:** Select **Cloud Storage** as your data source.
   - **File path** and **Split:** Add these four datasets:

File path	Split
`s3://snorkel-contracts-dataset/train-cleaned.parquet`	train
`s3://snorkel-contracts-dataset/dev-cleaned.parquet`	train
`s3://snorkel-contracts-dataset/valid-cleaned.parquet`	valid
`s3://snorkel-contracts-dataset/test-cleaned.parquet`	test

4. Select **Verify data source(s)**. Snorkel Flow runs data quality checks to ensure data is cleaned and ready to upload.
5. Scroll down to the **Define schema** section. From the **UID Column** menu, select `uid`. This is the unique entry ID column in the data sources.
6. Select **Add data source(s)**. When the data is done ingesting from all four data sources it shows the status **Data ingestion complete**.

Troubleshooting: If one or more of the example data splits does not upload, delete the dataset and try to upload the data again.

After your data is uploaded, you can create your Snorkel Flow classification application.

Create an application

Create a new Snorkel Flow application. Within this application, you can start using Snorkel Flow's powerful tools to classify the documents in your dataset. To create a new application:

1. From the left menu, select **Applications**.
2. Select the dropdown arrow from the **Create application** menu in the top right corner of the page, then select **Create from template**.
3. Choose the **Classification** template from the menu of preset application templates. This template creates an application that labels entire documents across the desired schema.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

4. Click **Create application**.
5. Fill in the **New Classification application** dialogue. Enter the following information:
   - **Application name:** `classification-example-app`
   - **Choose default dataset:** `contract-clf`
   - **Specify required input fields:** `text` (leave other options unchecked)
   - **Ground truth column (Optional):** `label`
6. After inputting this information, select **Autofill Labels** to populate the **Labels (optional)** field with your labeling schema, based on existing data in your label column.
7. Click **Save** to initiate application creation. This takes 2-5 minutes, as Snorkel Flow ingests your data into the application template and prepares the data for programmatic labeling.
8. Once the application has initialized, select the **Go to Studio** button to access the **Develop (Studio)** page.

Next, you will develop a machine learning model that will significantly speed up the contract classification task, starting with labeling functions (LFs).

Task overview: Programmatic labeling

The remainder of this guide will walk through a working classification example that demonstrates the three core tasks of programmatic labeling:

1. Develop labeling functions (LFs).

   Labeling functions capture the mental models that your subject matter experts (SMEs) use when they label data manually as Python functions that can be applied to larger amounts of data. The end result is a large amount of labeled training data.

2. Train a machine learning (ML) Model.

   Using the large amount of labeled training data, Snorkel Flow applies standard machine learning techniques to train an ML model customized to your dataset.

3. Analyze model results.

   Evaluate the model's performance, and adjust the labeling functions and training until the model meets your performance criteria.

Develop labeling functions (LFs)

Machine learning model training works better with larger amounts of labeled data to train on. **Labeling Functions (LFs)** allow you to programmatically label large amounts of data. LFs codify expert opinion and criteria for how to label your data. These labeled data points create more training data, so you can achieve better model results at the training stage.

Follow along with the steps in the example to create a precise LF for **employment** contracts.

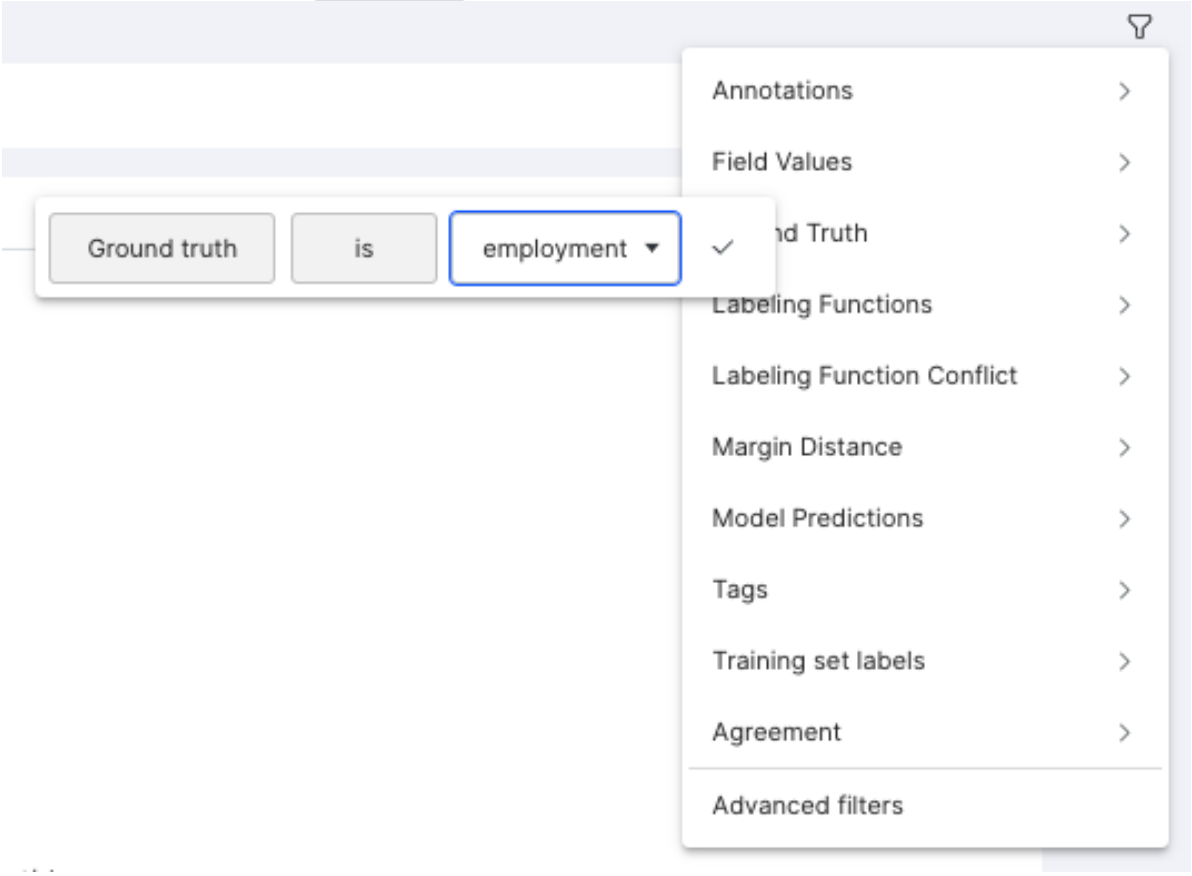
Filter data

The first stage in creating LFs is exploratory. Often, a data scientist and an SME work together to identify patterns in already-labeled data that are good candidates for an LF. Filter the uploaded data for only

Copyright © 2025 Snorkel AI, Inc. All rights reserved

documents that are already labeled `employment`. Do this from the **Studio** page for the `classification-example-app`:

1. Select the **Filter** icon in the top right corner of the page.
2. Select **Ground Truth**.
3. Select **Ground truth** + **is** + `employment` from the dropdown menu.



4. Select the **checkmark (✓)** to filter viewable data for documents already labeled `employment`.

Use Snippet view

You can choose the most efficient view of the data.

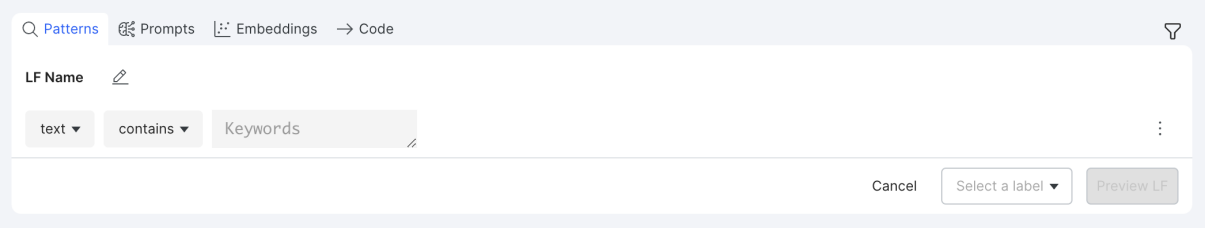
1. From the **Record view** menu, choose **Snippet view** to see a small chunk of text from each document.

Create, assess, and improve the first labeling function

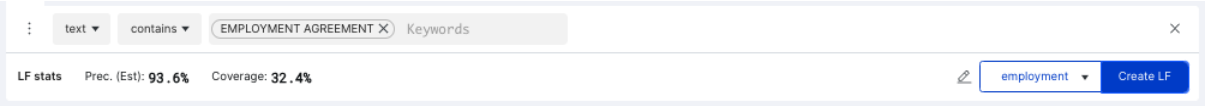
As you scan the documents in the `employment` set, notice that most start with `EMPLOYMENT AGREEMENT`. Let's create an LF that labels all documents containing the text `EMPLOYMENT AGREEMENT` as `employment`:

1. In the text box in top middle pane with the description **Type anything to start creating an LF**, type `/`.
2. Select **Keyword** under the **Pattern Based** section. This brings up the dialogue **text** + **contains** + **Keywords**.

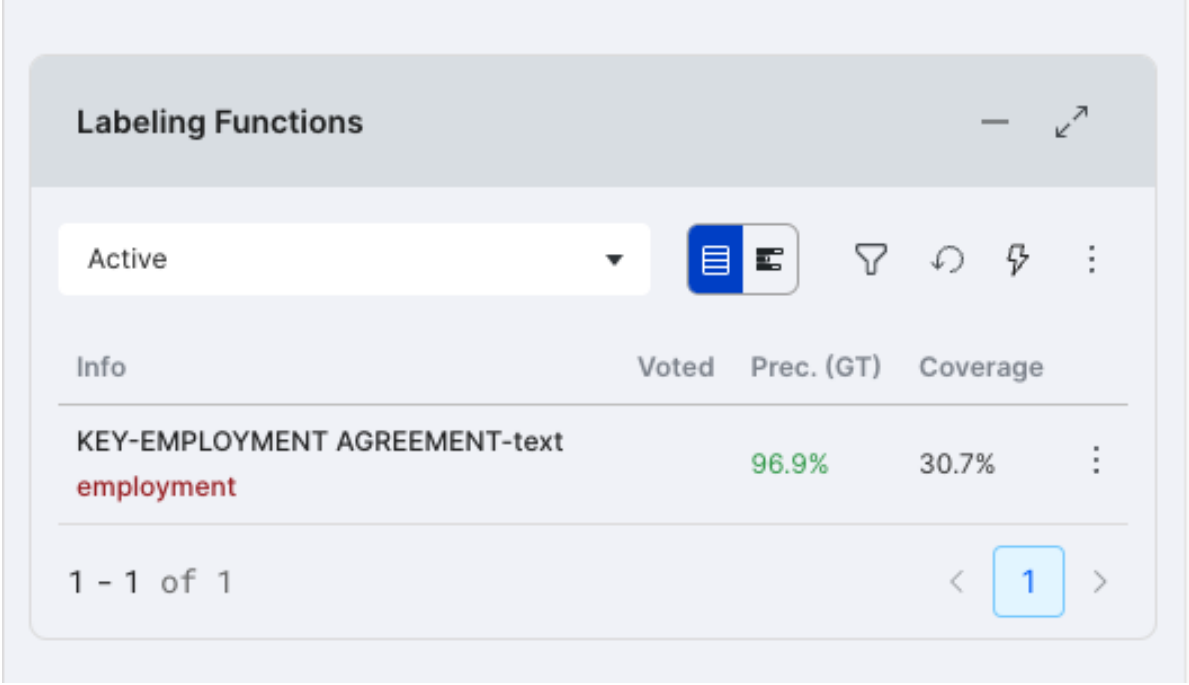
Copyright © 2025 Snorkel AI, Inc. All rights reserved



3. In the **Keywords** field, enter `EMPLOYMENT AGREEMENT`.
4. From the **Select a label** on the right side of the LF builder, select `employment`.
5. Select **Preview LF**. The **Record view** now shows documents that match the LF. Snorkel Flow also displays metrics for this LF:

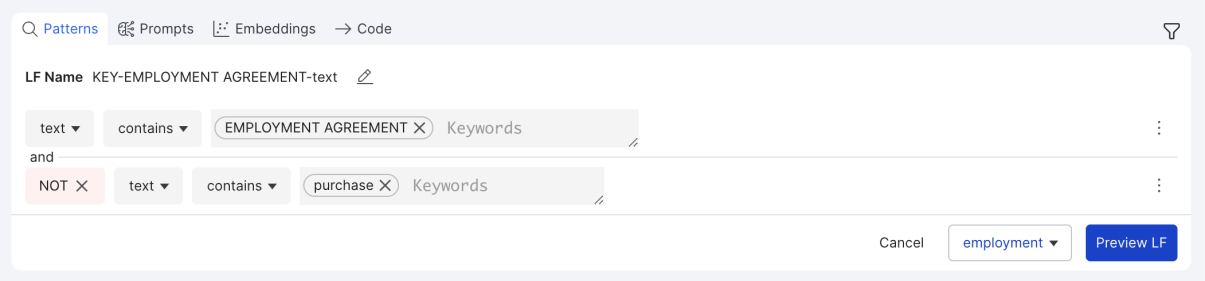


   - **Prec. (Est):** Measures the precision of the LF for your current data split. That is, it measures how often the LF correctly labels the documents, based on the ground truth. In this example, the precision is **93.6%**, which is highly accurate. Your results may vary.
   - **Coverage:** The % data in your split that fit this criteria. In this example, the coverage is **32.4%**. Your results may vary.
6. Select **Create LF** to add this LF to the active LFs for the model you're going to train. You can see this LF added to the **Labeling Functions** pane.



7. To achieve higher precision, incorporate a second criterion for the `EMPLOYMENT AGREEMENT` LF to exclude all documents that contain the text `purchase`. Select the `KEY-EMPLOYMENT AGREEMENT-text` label from the **Labeling Functions** menu.
8. To build a multi-part LF, select the three-dot menu next to the first factor. Select **Add another builder (and)**. From the three-dot menu, select **Negate this builder** to use the **NOT** option.

Copyright © 2025 Snorkel AI, Inc. All rights reserved



9. In the **Keywords** field, enter `purchase`.

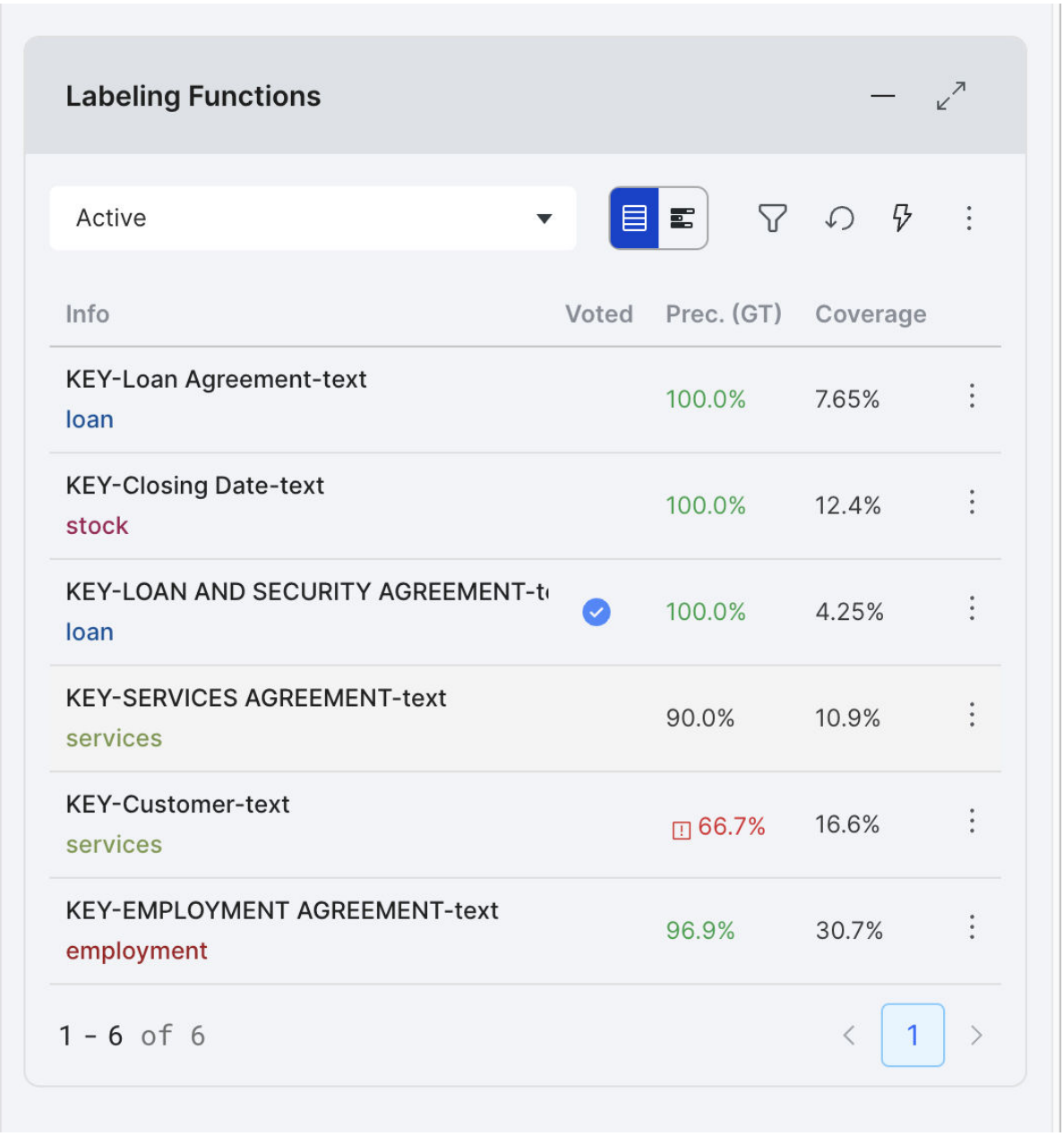
Create five more labeling functions

Repeat Steps 1-6 in the previous section to create five more labeling functions for the SEC filings dataset. Snorkel Flow can auto-detect the label after you select **Preview LF** for each LF. Create these LFs:

LF Type	Keywords	Label
Keyword	`Customer`	services
Keyword	`SERVICES AGREEMENT`	services
Keyword	`LOAN AND SECURITY AGREEMENT`	loan
Keyword	`Closing Date`	stock
Keyword	`Loan Agreement`	loan

Once you've created each of these LFs, you will see the six active LFs in the **Labeling Functions** pane on the left side of the page. Note that your values may vary from the example.

Copyright © 2025 Snorkel AI, Inc. All rights reserved



> TIP
> Learn more about Creating good labeling functions.

Now more of your data has programmatically added labels. Next, you'll create a model from this larger training dataset.

Train a machine learning (ML) model

Now that you've created several LFs for your model, you are ready to train your first ML model! After training the model, you will then compare the model's label predictions with the expert-assigned labels for the `valid` data split.

1. Expand the **Models** pane and select **Train a model**.
2. Choose **Fast model**, which is significantly faster than the other options. It runs a subset of the training data through the model and provides a good baseline to gauge this model's results. Select **Train Fast Model** to begin model training for the SEC filings dataset.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

A status bar appears that shows the model training % progress.

Once the model is finished training, it updates the **Models** table with the most recent model metrics.

Assess model accuracy for the valid split

Model assessment is the key to achieving high-quality results with your custom ML model. In this example, your goal is to achieve at least **90 accuracy** and **90 F1 Macro** for the `valid` data split.

Here are some example results from running a **Fast model** with the existing LFs:

	Dev	Valid
Accuracy	91.00%	84.84%
F1 Macro	88.67	82.51

- **Accuracy:** 84.84%
- **F1 Macro:** 82.51

These results are a good start, but below the desired **90 Accuracy** and **90 F1** success metrics when evaluating the model against the pre-labeled `valid` data split.

Next, you will use the **Analysis** pane to identify which data points and labels your model struggles to predict.

Analyze model errors and results

After running the first ML model, you can view the model performance in the **Analysis** pane.

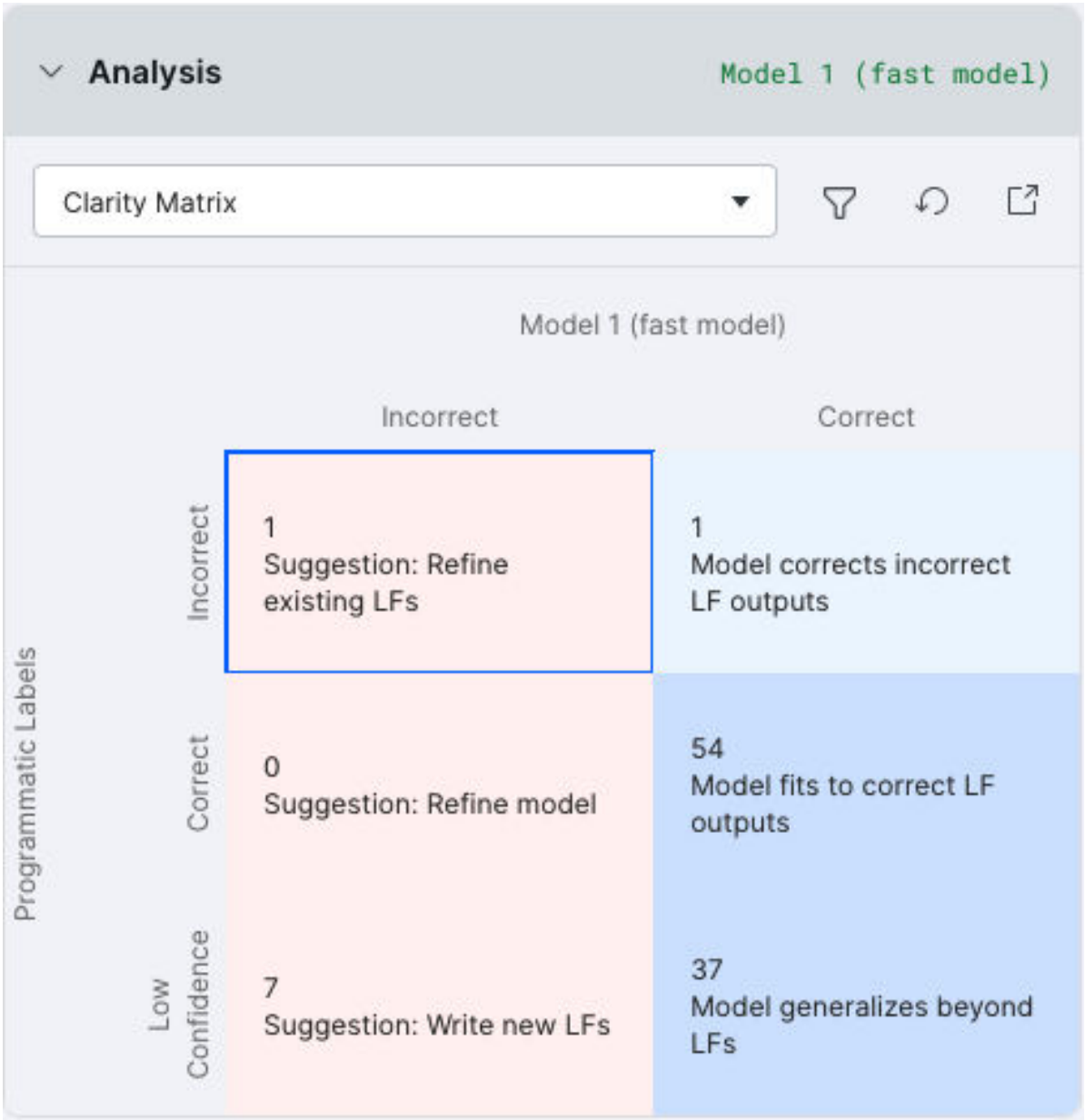
This tutorial uses the two most common error analysis visualization tools, the **Clarity Matrix** and **Confusion Matrix** to analyze and improve the example document classification model. To learn about all of Snorkel Flow's analysis visualization tools, read Analysis: Rinse and repeat.

Interpret results from the Clarity Matrix

The **Analysis** panel shows the **Clarity Matrix** by default.

The **Clarity Matrix** table compares labels created by LFs with labels created by the model, and compares both of them to the `valid` data split. Example **Clarity Matrix** output:

Copyright © 2025 Snorkel AI, Inc. All rights reserved



Explanation of the **Clarity Matrix** table:

LF	Model	Definition	Suggestion
Incorrect	Incorrect	The model learned the wrong thing from the LF.	Refine the LF for accuracy.
Incorrect	Correct	The model extrapolated correctly, despite the incorrect LF.	No action.
Correct	Incorrect	The model was inaccurate despite correctly-labeled data.	Refine the model.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

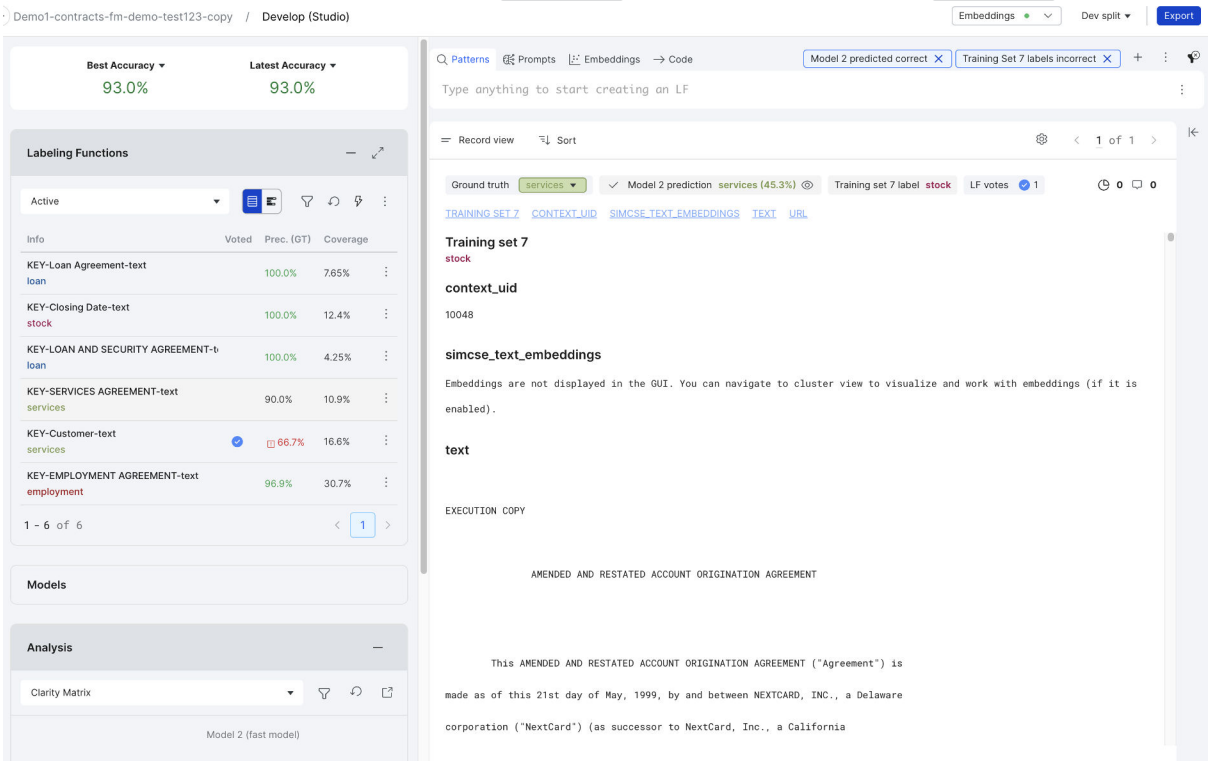
LF	Model	Definition	Suggestion
Correct	Correct	The model trained correctly from the LF-labeled data.	No action.
Low confidence	Incorrect	The model extrapolated incorrectly.	Write more LFs to give the model more training data.
Low confidence	Correct	The model extrapolated correctly.	No action.

Select any cell in the table to view the corresponding data points in the **Record view**.

(Optional) Use the Clarity Matrix to refine an incorrect LF

These next steps show how you can use the **Clarity Matrix** to improve your LFs and the subsequent iteration of your model. Your output may be different from the example, so you may not be able to follow these steps exactly. In that case, just read through the steps so you can learn more about how to improve LFs.

1. From the **Clarity Matrix**, select the **Refine existing LFs** cell. You can see the specific document that the LF labeled incorrectly. In the **Record view**, the **Ground truth** is `employment` but the **Model 1 prediction** and **Training set 1 label** are for `services`. So, we should look for a `services` LF.



Copyright © 2025 Snorkel AI, Inc. All rights reserved

2. Expand the **Labeling functions** pane. You can see that the `KEY-Customer-text` label **Voted** incorrectly and has low **Prec. (GT)**. This makes it clear that this is the LF that needs to be refined.
3. Incorporate a second criterion for the `Customer` LF to exclude all documents that contain the text `EMPLOYMENT AGREEMENT`. Select the `KEY-Customer-text` label from the **Labeling Functions** menu.
4. Select the three-dot menu next to the first factor. Select **Add another builder (and)**. From the three-dot menu, select **Negate this builder** to use the **NOT** option.
5. In the **Keywords** field, enter `EMPLOYMENT AGREEMENT`.
6. Select **Preview LF**.
7. Select **Save LF**.

(Optional) Use the Clarity Matrix to expand LF coverage

Continue to use the **Clarity Matrix** to write additional LFs. Your output may be different from the example, so you may not be able to follow these steps exactly. In that case, just read through the steps so you can learn more about how to expand your LF coverage.

1. From the **Clarity Matrix**, select the **Write new LFs** cell.
2. From the **Record view** dropdown, select **Snippet view**.
3. Look for patterns in the documents that none of the existing LFs apply to. In this example, there are several `stock` documents that need an LF.
4. Let's add another LF that addresses more of the `stock` documents:

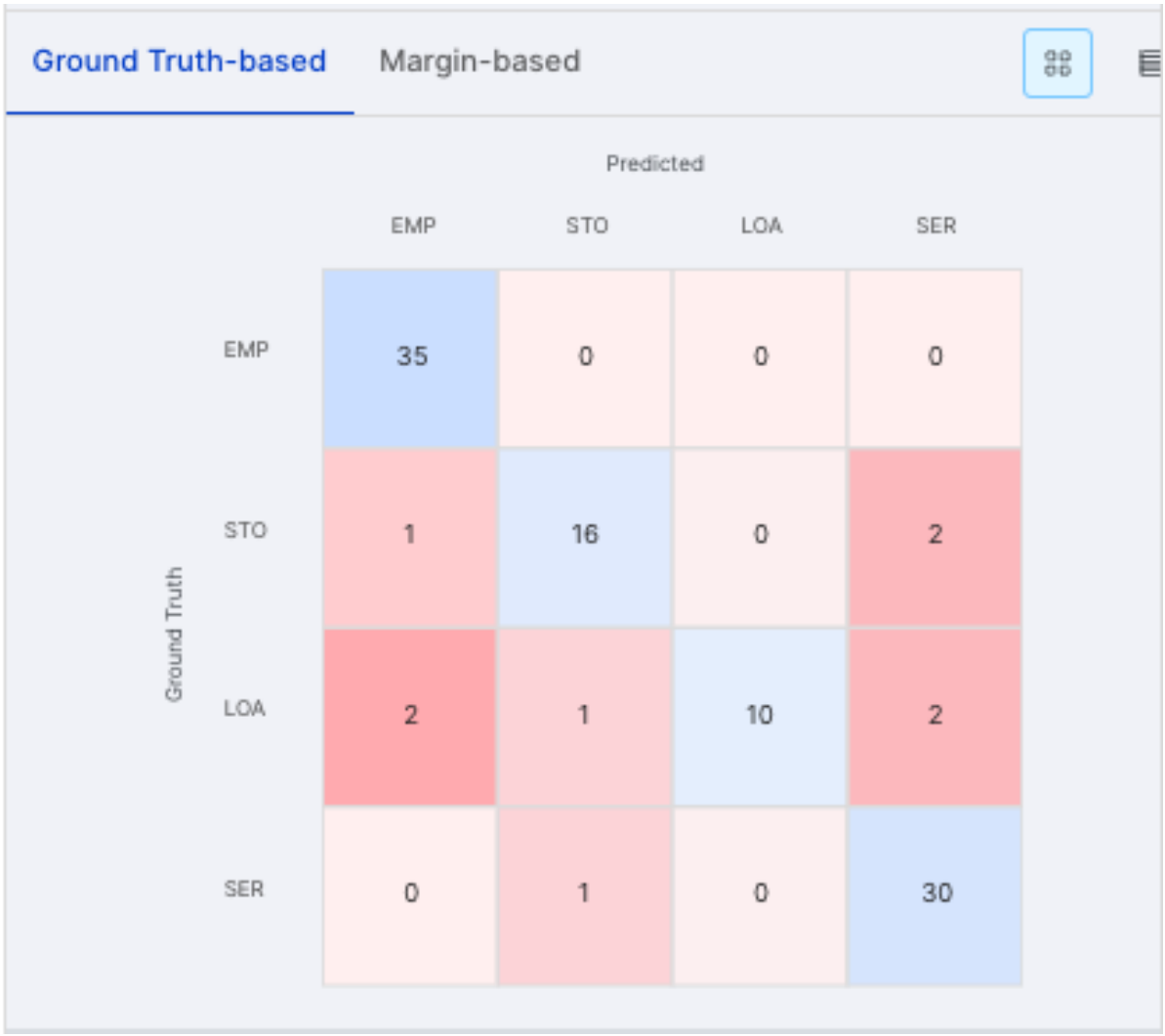
LF Type	Keywords	Label
Keyword	`STOCK PURCHASE AGREEMENT` NOT `LOAN AGREEMENT` + `EMPLOYMENT AGREEMENT`	stock

Interpret results from the Confusion Matrix

Select the **Confusion Matrix** from the dropdown menu in the **Analysis** pane.

The **Confusion Matrix** compares every actual label with every predicted label. For example, select the cell in the `LOA` row and the `SER` column to view documents that are actually `loan` agreements in our `valid` dataset, but were classified as `service` agreements.

Copyright © 2025 Snorkel AI, Inc. All rights reserved



(Optional) Use the Confusion matrix to expand LF coverage

Use the **Confusion Matrix** to write additional LFs. Your output may be different from the example, so you may not be able to follow these steps exactly. In that case, just read through the steps so you can learn more about how to expand your LF coverage.

1. From the **Confusion Matrix**, select the cell in the `LOA` row and the `SER` column.
2. From the **Record view** dropdown, select **Snippet view**.
3. Look for patterns in the documents that none of the `loan` LFs apply to.
4. Let's add another LF that addresses more of the `loan` documents:

LF Type	Keywords	Label
Keyword	`Loan Party`	loan

Copyright © 2025 Snorkel AI, Inc. All rights reserved

(Optional) Train a new model

Now that you've refined and added more LFs, train another fast model by following the steps in the **Train a machine learning (ML) model** section. Compare the results of this model with your first one and note any improvements or regressions.

Next steps: Iterate to improve model performance

After completing this tutorial, you now understand the most common features and capabilities of the Snorkel Flow platform and are ready to work towards achieving your model training success metrics!

For further learning, you can improve the model performance for this dataset and task by using the **Analysis** visualizations to tweak your LFs and models for better results. In this way, improving and adapting your model becomes unblocked from the need for hand-labeled training data and instead becomes fast, iterative, and analysis-driven.

As you continue to iterate on your results, we recommend using the following Snorkel Flow features:

- Customize your model's architecture, hyperparameters, and hyperparameter searches
- Write Search based LFs
- Make use of the additional analytical views described in Analysis: Rinse and repeat

Copyright © 2025 Snorkel AI, Inc. All rights reserved

Hierarchical configuration for classification and extraction tasks

In this page, you will learn how to create a hierarchical task (classification, extraction, etc.).

We will demonstrate how to use Snorkel Flow for hierarchical classification problems by building upon the application templates described in the Document classification: Classifying contract types tutorial.

Create a dataset and application

Follow the dataset and application creation from the Document classification: Classifying contract types tutorial.

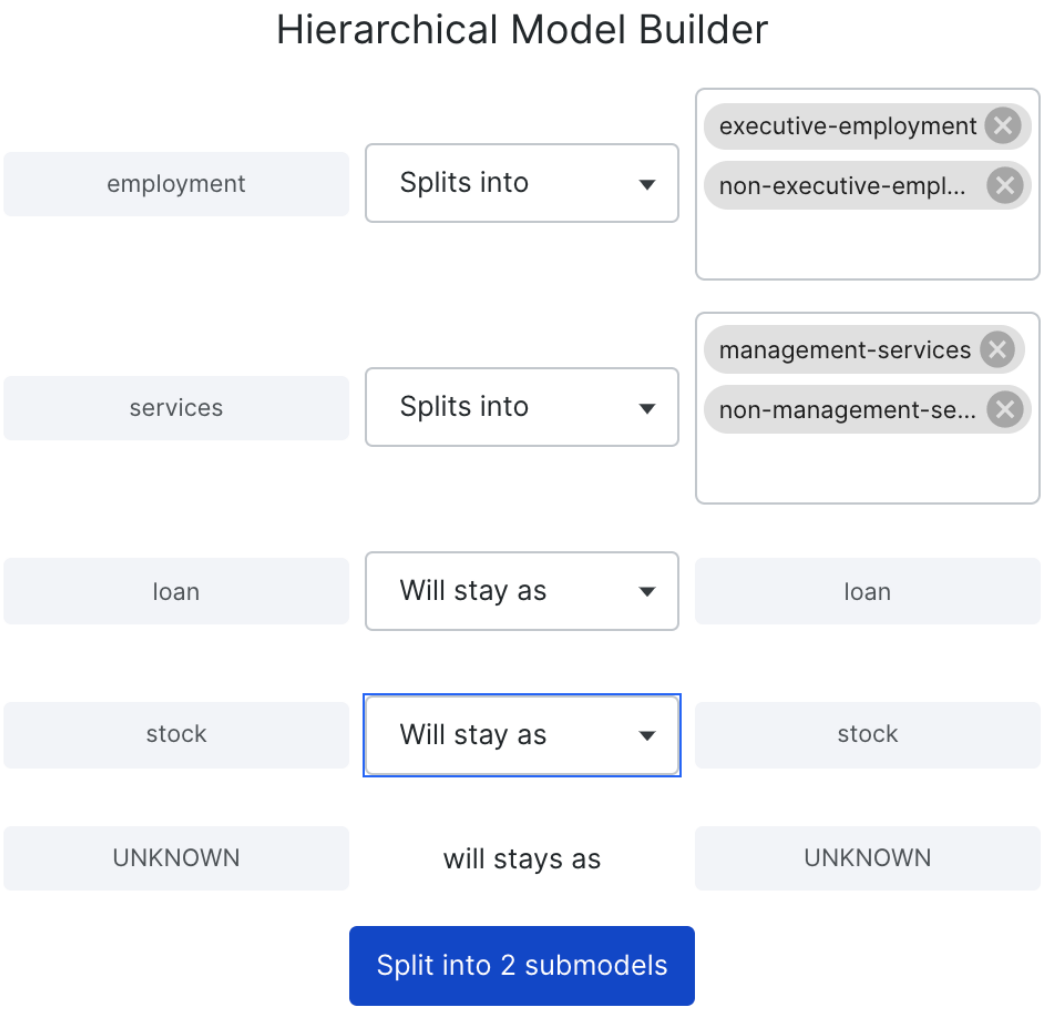
Create hierarchical configuration

After loading the datasources and creating the application template, we can create the hierarchical configuration by:

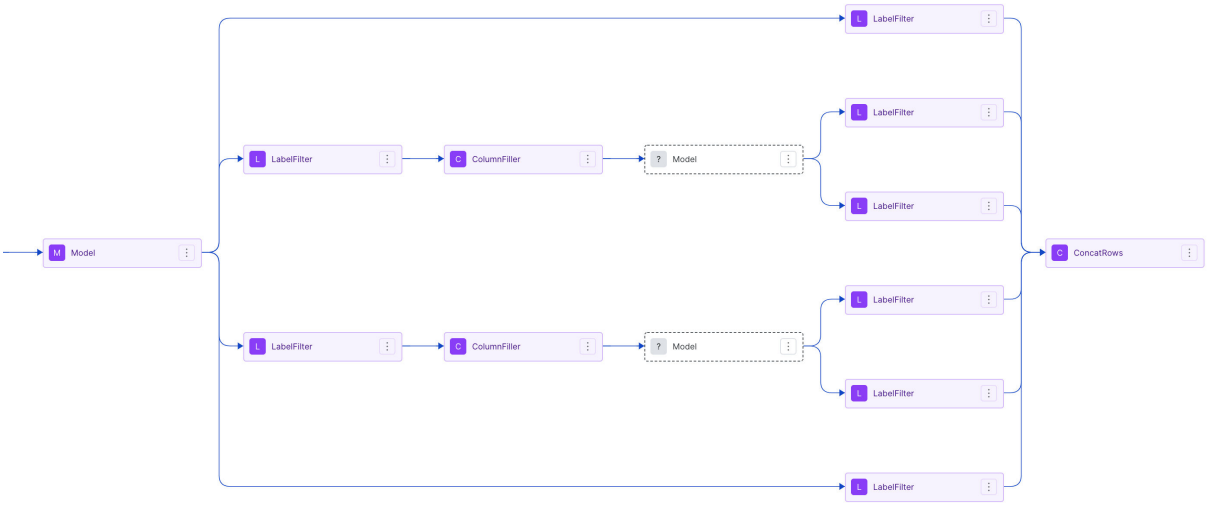
- Selecting the dropdown bar at the model node (the three-dot next in the model node)
- Select **Build hierarchical model**. This selection will prompt an interface for inputting sub-labels
- Click **Split into submodels** once you input all sub-labels
- Snorkel Flow will create a hierarchical DAG for the appropriate template

For example, we can split “services” into 2 sub-classes “management-services” and “non-management-services”; splitting “employment” into 2 sub-classes “executive-employment” and “non-executive-employment”. The two classes “loan” and “stock” can stay as they are.

Copyright © 2025 Snorkel AI, Inc. All rights reserved



The expected output will look as follows:



About the DAG:

- The `Model` nodes are where we iteratively write labeling functions, train the models, and analyze performance.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

- The `LabelFilter` and `ColumnFiller` will ensure Snorkel Flow passes the predictions into the correct models.
- The `ConcatRows` will aggregate the predictions across all models and create the resulting dataframe. This node is also the last node of the hierarchical task.

Iteration and development

The iteration process will happen at every `Model` node. Each model will perform a classification/extraction/etc task for each sub-labels group (e.g., “management-services” and “non-management-services”, “executive-employment” and “non-executive-employment”). You can upload the corresponding ground truths, write labeling functions, train models, and analyze performance.

Final predictions

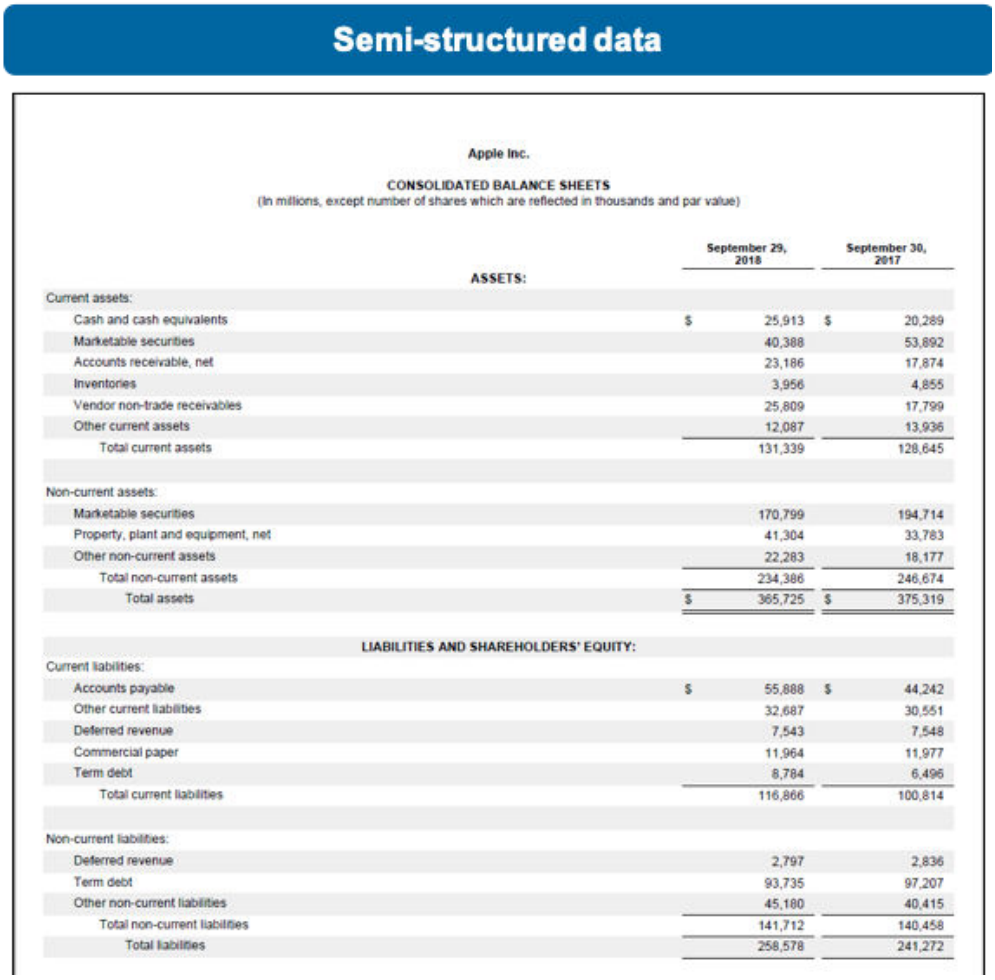
The `ConcatRows` node will output a dataframe, with the `preds_str` column containing the final prediction.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

Extraction from PDFs: Extracting balance sheet amounts

In this tutorial, you will use Snorkel Flow to extract balance sheet amounts from the SEC filings of public companies.

Snorkel Flow supports the extraction of structured information from semi-structured documents such as PDFs, HTML, and docx files. Common use cases extract information from financial filings, insurance claims, and medical reports. These documents are converted to a special format called "Rich Doc" in the platform. This conversion allows you to leverage text, layout, and image modalities to write labeling functions (LFs) and train models.



In this tutorial, you will learn how to:

1. Upload data into Snorkel Flow
2. Create an application
3. Add ground truth
4. Define labeling functions
5. Train a model

Copyright © 2025 Snorkel AI, Inc. All rights reserved

Upload data

Snorkel Flow can ingest data from a range of storage options, including cloud storage, databases, or local files. For PDF extraction applications, the input data must include the following fields:

Field name	Data type	Description
`uid`	int	A unique id mapped to each row. This is standard across all Snorkel Flow datasources.
`url`	str	The file path of the original PDF. This will be used to access and process the PDF.

For this tutorial, use the provided dataset from in AWS S3 Cloud Storage. Start by creating your new dataset.

1. Select the **Datasets** option from the left menu.
2. Select the **+ Upload new dataset** button on the top left corner of your screen.
3. In the **New Dataset** window, enter the following information:
   - **Dataset name:** Enter `balance-sheets-dataset`.
   - **Select a data source:** Select **Cloud Storage** as your data source.
   - **File path** and **Split:** Add these datasets:

File path	Split
`s3://snorkel-native-pdf-sample-dataset/splits/train.csv`	train
`s3://snorkel-native-pdf-sample-dataset/splits/valid.csv`	valid
`s3://snorkel-native-pdf-sample-dataset/splits/test.csv`	test
4. Select **Verify data source(s)** to run data quality checks to ensure data is cleaned and ready to upload.
5. After clicking **Verify data source(s)**, select `uid` from the **UID column** dropdown box. The `uid` is the unique entry ID column in the data sources.
6. Select **Add data source(s)**.

Once the data is ingested from all three data sources, you can create an application with your PDF extraction dataset. For more information about uploading data, see Data upload.

Create an application

There are two categories of PDF documents:

Copyright © 2025 Snorkel AI, Inc. All rights reserved

- **Native PDFs** are documents that were created digitally. These PDFs can be parsed without additional processing.
- **Scanned PDFs** are created from scans of printed documents. These PDFs don't have the metadata that we need to parse layout information. Scanned PDFs require additional preprocessing with an optical character recognition (OCR) library.

The application creation process is slightly different for these two formats. This tutorial uses native PDFs.

You will extract the numbers from the Consolidated Balance Sheets provided in public company filings, and classify these numbers into the following classes: `ASSETS`, `LIABILITIES` and `EQUITY`.

1. Select the **Applications** option in the left-side menu and select **Create application** to create a new application. Enter the values provided in the table:

Stage	Field	Value
Start	Application name	`line-item-classification`
Data	Dataset	`balance-sheets-dataset`
Label schema	Data type	PDF
	Task type	Extraction
	PDF type	Native PDF
	PDF URL field	`url`

   You selected the dataset created in the previous step, defined the data and task type, and specified where the PDFs can be found (`url`).

2. Select **Generate Preview** to generate and display a preview sample on the right from the input documents.

3. Edit the **Label schema** table:

4. 1. Select **Add new label** and add `ASSETS`. Next, add `LIABILITIES` and `EQUITY` to the table as new labels. These are the entities to extract.
   2. Edit the `NEGATIVE` label and rename it to `OTHER`. Any data point that does not fall into one of the positive classes will be labeled as `OTHER`.
      Do not edit the `UNKNOWN` label. This is used for unlabelled data.
   3. Select **Next**.

5. In the **Preprocessors** pane, select the following preprocessing operations to perform on the dataset:
   - **Split docs into pages:** Yes
   - **Page split method:** PageSplitter
   - **PageSplitter window_size:** 0
   - **Extraction method:** NumericSpanExtractor from an existing extractor
   - **NumericSpanExtractor field:** rich_doc_text

Copyright © 2025 Snorkel AI, Inc. All rights reserved

6. Select **Commit** for the `PageSplitter` and the `NumericSpanExtractor` operators.
   Operators perform transformations on your input data. You can add other operators in the background to pre-compute useful features to help define labeling functions. For more information on these operators, see PDF-based operators. The `PageSplitter` and the `NumericSpanExtractor` transformations are visualized in the **Preview sample** pane on the right side of your screen.
   - The `PageSplitter` operator splits the documents into pages. This split helps us decrease memory usage, retrieve metadata more efficiently, and improve latency.
   - The `NumericSpanExtractor` operator is used to extract all numeric values as candidate spans from the raw text. The **Preview sample** pane highlights the numeric values.
7. Click **Next** to start a job that runs the preprocessing operators on the entire dataset.
8. In **Development Settings**, select **Yes** using `span_text` to **Compute embeddings**.
9. Select **Go To Studio**.

To set up a Scanned PDF application, you would select the **Scanned PDF, no need to run OCR** option for PDF type. For more information about preprocessing scanned PDFs, see Scanned PDF Guide.

Add ground truth

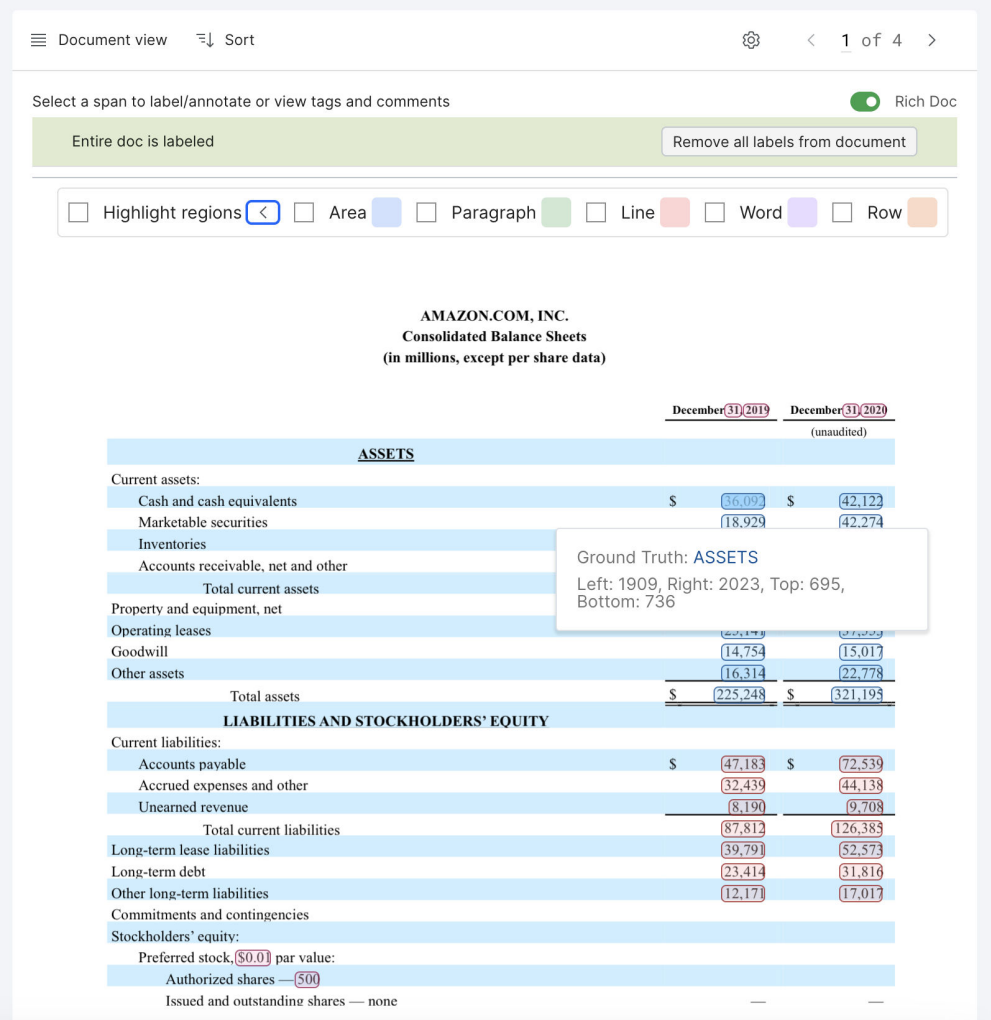
Snorkel Flow allows you to label your data with programmatic labels. First, you must establish ground truth with annotations to validate the performance. When you start a new project, you can annotate your data using our Annotation Studio. For this tutorial, you will annotated data.

1. Select **Overview** in the left-hand menu.
2. Select **View Data Sources** to view the data sources that we have added. In this view, you can see the total number of data points that are extracted for each data source.
3. Select **Upload GTs**.
4. Enter the following values:
   - **File path:** `s3://snorkel-native-pdf-sample-dataset/native_pdf_ground_truth.csv`
   - **File format:** `CSV`
   - **Label column:** `label`
   - **UID column:** `x_uid`
5. Select **Add**.
6. Refresh the page after uploading the ground truth.
   The `# of GT labels` column has non-zero values.
7. Click **Go To Studio** to return to the **Studio** page.

Review data in **Develop (Studio)**

Snorkel Flow extracts candidate spans from the raw text of the document using a heuristic. Common span types include numbers, dates, and email addresses. You can define labeling functions to assign labels to the candidate spans in the document. The labels are aggregated and used to train a classification model. The spans are highlighted with bounding boxes and color-coded by label.

Copyright © 2025 Snorkel AI, Inc. All rights reserved



1. Use your cursor to hover over any word in the document. You can see the bounding box coordinates of the word as measured in pixels.
2. Select **Rich Doc** toggle to see the raw text extracted from the document.
3. Select the arrow to expand **Highlight regions**. This allows you to highlight the bounding boxes of the different regions in the document: row, word, line, paragraph, and area.
   Develop (Studio) shows the `dev` split. The `dev` split is a sample of the train split used for iterating on labeling functions.
4. Select the **Dev split** dropdown in the top toolbar.
5. Select **Resample data**.
6. Set the **Sample size** to 4, and select **Resample dev split**.

Next, you'll define your labeling functions.

Define labeling functions

A labeling function (LF) is a programmatic rule or heuristic that assigns labels to unlabeled data. LFs are the key data programming abstraction in Snorkel Flow. Each labelling function votes on whether a data point has a particular class label. The Snorkel Flow label model is responsible for estimating LF accuracies and aggregating them into training labels.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

For PDF extraction applications, we add a few text fields that are useful for defining LFs using the `RichDocSpanRowFeaturesPreprocessor` operator. These fields are based on heuristics as defined below:

Field	Description
`rich_doc_row_text_inline`	text in the same row as the span
`rich_doc_row_header`	text that is the furthest to the left of the span
`rich_doc_inferred_row_headers`	text that is above the span and indented to the left
`rich_doc_row_text_before`	text in the row before the span
`rich_doc_row_text_after`	text in the row after the span

You can use these columns to define text-based LFs. You can also use the word bounding boxes (`top`, `left`, `bottom` and `right`) to define location-based LFs.

1. Create new labeling functions.
    1. In **Patterns**, select your LF template by typing `/`, followed by the pattern, such as `Keyword`, `Regex`, or `Numeric`.
    2. Enter the requirements for the labeling function.
    3. **Select a label** from the dropdown menu.
    4. Select **Preview LF** to see the precision and coverage of your LF.
    5. Select **Create LF**.
       The new LF shows up in the **Labeling Functions** pane.

Using the steps above, add these LFs to your application:

LF Template	Settings	Label	Explanation
Keyword	`rich_doc_inferred_row_headers` [CONTAINS] `Assets`, `Cash`, `Land`	`ASSETS`	The LF checks if the inferred row header of the span contains the keywords Assets, Cash or Land.
Keyword	`rich_doc_row_text_inline` [CONTAINS] `Liabilities`, `Payable`, `Debt`	`LIABILITIES`	The LF checks if the text in the same as the span contain the keywords Liabilities, Payable or Debt.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

LF Template	Settings	Label	Explanation
Regex	`rich_doc_row_header` matches pattern `Total.{1,15}equity`	`EQUITY`	The LF checks if the row header of the span matches the regex pattern.
Numeric	`right` [<=] `1200`	`OTHER`	The LF checks if the span's right boundary is less than or equal to 1200 pixels i.e. if the span is on left side of the page.

Some LF templates are available for only PDF extraction applications. These LFs allow us to combine the text and layout information to define LFs.

Using the steps above, add these LFs to your application:

LF Template	Settings	Label	Explanation
Span Regex Proximity Builder	If the span is up to [6] [LINE(s)] [AFTER] the regex pattern `current assets:`	`ASSETS`	This LF specifies that spans up to 6 lines after the expression “current assets:” will be labeled `ASSETS`.
Span Regex Row Builder	If the span is [0] rows before and [5] rows after the regex pattern `current liabilities:`	`LIABILITIES`	This LF specifies that spans up to 5 rows after the expression “current liabilities:” will be labeled `LIABILITIES`.
Rich Doc Expression Builder	Evaluate `Common stock` with `SPAN.left > PATTERN1.right and SPAN.top >= PATTERN1.top`	`EQUITY`	This LF specifies that spans to the right and below the expression “Common stock” will be labeled as `EQUITY`.

For more information on Rich Doc builders, see Rich document LF builders.

You can also encode your custom labeling function logic using the Python SDK. For examples using the `RichDocWrapper` object, refer to the SDK reference.

Train a model

You can aggregate the newly defined labeling functions to train a model.

1. In the **Models** pane, select **Train a model**.
2. For the **Model architecture**, select **Logistic Regression (TFIDF)**.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

3. To include the text fields used to define labeling functions, select **Input Fields** and select `rich_doc_inferred_row_headers`, `rich_doc_row_text_inline`, `rich_doc_row_header`, and `right` from the dropdown.
4. Leave the default settings in **Train options**.
5. Select **Train custom Model** to generate labels using the LFs and train a logistic regression model using the labels.

Once the model training is completed, you will see your first model in the **Models** pane.

Snorkel Flow offers default configurations for several commonly used models. For more information on models, see Model training.

Snorkel Flow enables users to iteratively generate better labels and better end models. For information about getting metrics and improving model performance, see Analysis: Rinse and repeat.

Conclusion

You have set up and iterated on a PDF extraction application in Snorkel Flow. To learning more about PDF extraction features in Snorkel, see this documentation:

- Learn how to work with scanned documents with our OCR guide.
- Read more about PDF-based Operators.
- Check out more examples of Rich Document Builders.

The Snorkel Flow Python SDK provides greater flexibility to define custom operations. Select **Notebook** on the left-side pane to access the SDK. See the SDK reference to learn more about custom operators and custom labeling functions.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

(Beta) Named Entity Recognition from PDFs: Extracting patent data

> NOTE
> This is a beta feature available to customers using a Snorkel-hosted instance of Snorkel Flow. Beta features may have known gaps or bugs, but are functional workflows and eligible for Snorkel Support. To access beta features, contact Snorkel Support to enable the feature flag for your Snorkel-hosted instance.

In this tutorial, you will use Snorkel Flow to extract patent data such as title, patent number, date of patent, assignee, and filed date from PDFs. This example highlights the information to be extracted in different colors:

Copyright © 2025 Snorkel AI, Inc. All rights reserved

(12) **United States Patent**
Shazeer et al.

(10) **Patent No.: US 10,452,978 B2**
(45) **Date of Patent: Oct. 22, 2019**

(54) **ATTENTION-BASED SEQUENCE TRANSDUCTION NEURAL NETWORKS**

(71) Applicant: **Google LLC**, Mountain View, CA (US)

(72) Inventors: **Noam M. Shazeer**, Palo Alto, CA (US); **Aidan Nicholas Gomez**, Toronto (CA); **Lukasz Mieczyslaw Kaiser**, Mountain View, CA (US); **Jakob D. Uszkoreit**, Portola Valley, CA (US); **Llion Owen Jones**, San Francisco, CA (US); **Niki J. Parmar**, Sunnyvale, CA (US); **Illia Polosukhin**, Mountain View, CA (US); **Ashish Teku Vaswani**, San Francisco, CA (US)

(73) Assignee: **Google LLC**, Mountain View, CA (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

(21) Appl. No.: **16/021,971**

(22) Filed: **Jun. 28, 2018**

(65) **Prior Publication Data**

US 2018/0341860 A1 Nov. 29, 2018

Related U.S. Application Data

(63) Continuation of application No. PCT/US2018/034224, filed on May 23, 2018.

(60) Provisional application No. 62/541,594, filed on Aug. 4, 2017, provisional application No. 62/510,256, filed on May 23, 2017.

(51) **Int. Cl.**

G06F 15/18	(2006.01)
G06N 3/08	(2006.01)
G06N 3/04	(2006.01)

(52) **U.S. Cl.**
CPC ***G06N 3/08*** (2013.01); ***G06N 3/04*** (2013.01); ***G06N 3/0454*** (2013.01)

(58) **Field of Classification Search**
CPC G06F 3/015
USPC 706/15, 45
See application file for complete search history.

(56) **References Cited**

PUBLICATIONS

Luong et al., Effective Approaches to Attention-based Neural Machine Translation, (2015) Conf. on Empirical Methods in Natural Language Processing at p. 1412-1421 (Year: 2015).*

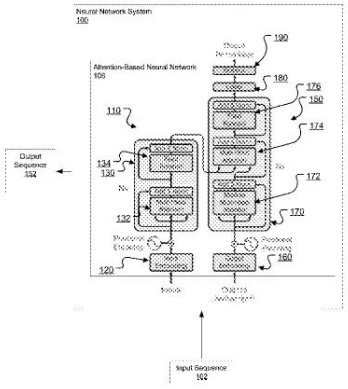
(Continued)

Primary Examiner — David R Vincent
(74) *Attorney, Agent, or Firm* — Fish & Richardson P.C.

(57) **ABSTRACT**

Methods, systems, and apparatus, including computer programs encoded on a computer storage medium, for generating an output sequence from an input sequence. In one aspect, one of the systems includes an encoder neural network configured to receive the input sequence and generate encoded representations of the network inputs, the encoder neural network comprising a sequence of one or more encoder subnetworks, each encoder subnetwork configured to receive a respective encoder subnetwork input for each of the input positions and to generate a respective subnetwork output for each of the input positions, and each encoder subnetwork comprising: an encoder self-attention sub-layer that is configured to receive the subnetwork input for each of the input positions and, for each particular input position in the input order: apply an attention mechanism over the encoder subnetwork inputs using one or more queries derived from the encoder subnetwork input at the particular input position.

30 Claims, 3 Drawing Sheets



There are many challenges to extracting the information:

- Some information come in different formats. For example, "patent number" comes in "US x,xxx,xxx B2" for newer patents, but could come in "x,xxx,xxx" for older patents.
- Older patents come in scanned PDFs and the location of these information in x-y pixel coordinates varies among documents.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

- Some information might be missing. For example, "patent number" and "assignee" are missing in the case of a pending patent.
- Patents issued in different countries come in completely different formats and layouts.

In Extraction from PDFs: Extracting balance sheet amounts tutorial, you first define a span extractor and work with extracted spans, which works well when the information to be extracted comes in the same format, such as numeric values.

For this tutorial, you have to extract title, assignee, patent number, etc., which all come in different formats. For this use case, the approach gives you better flexibility because you don't need to define a span extractor for a particular format.

In this tutorial, you will learn how to:

1. Upload data into Snorkel Flow
2. Annotate the data
3. Create an application
4. Define labeling functions
5. Train a model

Upload data

Snorkel Flow can ingest data from a range of storage options, including cloud storage, databases, or local files. For PDF extraction applications, the input data must include the following fields:

Field name	Data type	Description
`uid`	integer	A unique ID mapped to each row. This is standard across all Snorkel Flow datasources.
`url`	string	The file path of the original PDF. Used to access and process the PDF.

For this tutorial, use the provided dataset from in AWS S3 Cloud Storage. Start by creating your new dataset.

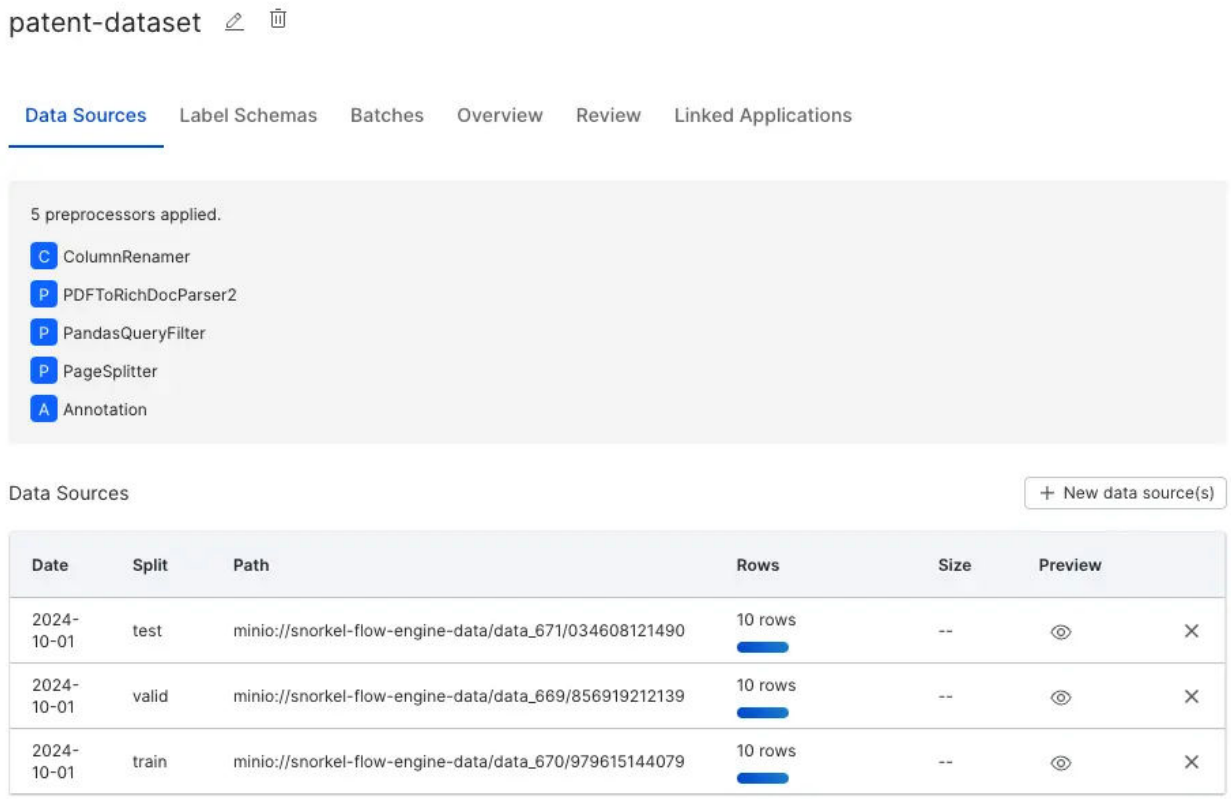
1. Select **Datasets** from the left menu.
2. Select the **Upload new dataset** button on the top right corner of your screen.
3. In the **Upload new dataset** window, enter the following information:
   - **Dataset name:** Enter `patent-dataset`.
   - **Enable multi-schema annotations:** Select this option.
   - **Select a data source:** Select **Cloud Storage** as your data source.
   - **File path** and **Split:** Add these datasets:

Copyright © 2025 Snorkel AI, Inc. All rights reserved

File path	Split
`s3://snorkel-public/patent1p/0-9.csv`	train
`s3://snorkel-public/patent1p/10-19.csv`	train
`s3://snorkel-public/patent1p/20-29.csv`	valid

4. Select **Verify data source(s)** to run data quality checks to ensure data is cleaned and ready to upload.
5. After clicking **Verify data source(s)**, select `uid` from the **UID column** dropdown box. The `uid` is the unique entry ID column in the data sources.
6. Define data type as follows:
   - **Data type:** PDF
   - **PDF type:** Native PDF
   - **Primary URL field:** `url`
7. Select **Add data source(s)**.

Once the data is ingested from all three data sources, you'll have a dataset like this:

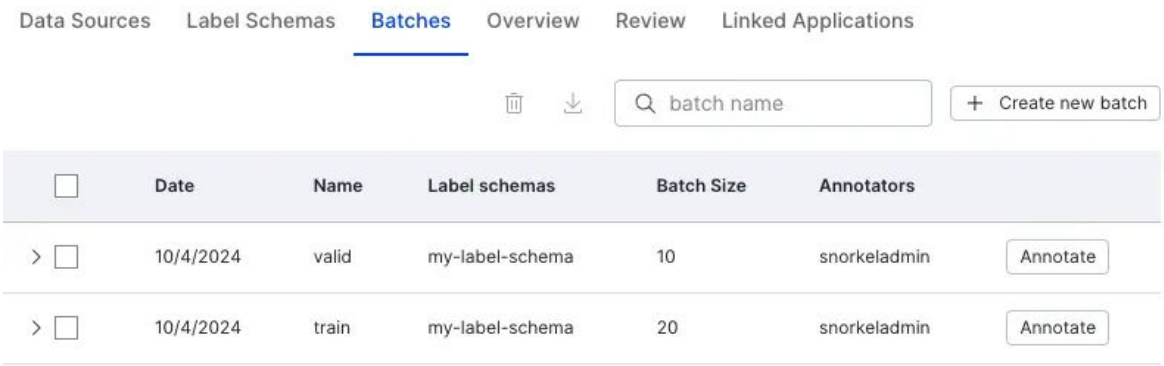


Annotate the data

Next, let's annotate the data.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

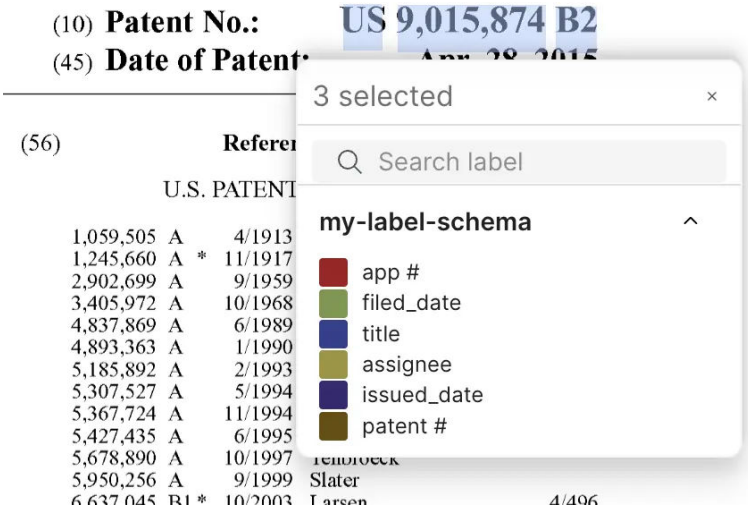
1. Select **Datasets** and search for your `patent-dataset`.
2. Select **Label Schemas** > **Create new label schema**.
3. Fill out the form as follows:
   - **Label schema name:** Enter `my-label-schema`.
   - **Data type:** Select `PDF`.
   - **Task type:** Select `Extraction`.
   - **PDF type:** Select `Native PDF`.
   - **PDF URL field:** Select `rich_doc_pdf_url`.
   - **Labels:** Add the following labels, using **Enter** for each new label:
   - Keep `UNKNOWN`
   - Keep `NEGATIVE`
   - Add `app #`
   - Add `filed date`
   - Add `title`
   - Add `assignee`
   - Add `issued date`
   - Add `patent #`
4. Select **Add label schema**.
5. Create a batch for each split.
   1. Select **Batches**.



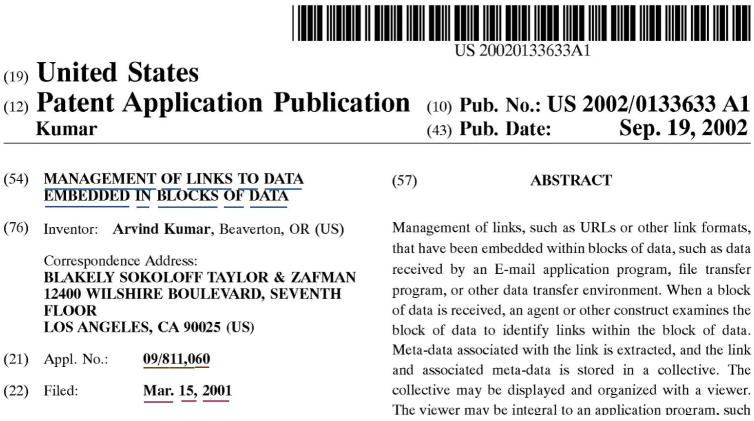
   2. Select **Create new batch**.
   3. Enter the details for your batch:
      - **Batch name:** Use your split.
      - **Label schema:** Use `my-label-schema`.
      - **Split:** Use one of the splits you created.
      - **Data points:** Select `All`.
      - **Sample percentage:** Use `100`.
      - **Assign to:** Select yourself as the annotator for this tutorial.
6. Go into the batches and annotate the data.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

For example, these words next to "Patent No.:" can be annotated as "patent #".



Some documents are missing some information. For example, this document is missing the patent number and the issued date because this is a pending patent.



7. Commit the annotations.

For more about annotating, see Walkthrough for annotators

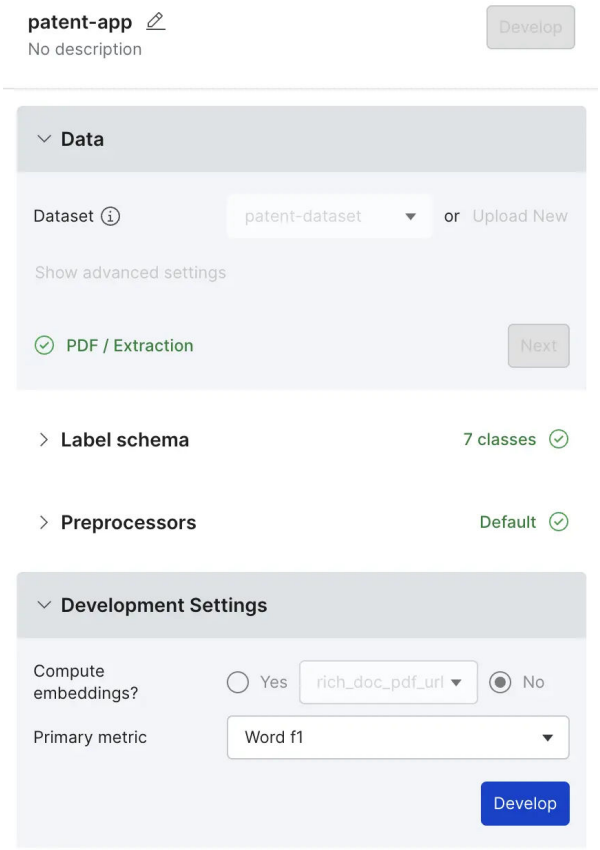
Create an application

Next, create an application on top of the dataset with the ground truth labels that you committed in the previous step.

1. Select **Linked Applications** in your dataset viewer, and select **Create application**.
2. Enter or select the following values:
   - **Application name:** `patent-app`
   - **Dataset:** `patent-dataset`
   - **Label schema name:** `my-label-schema`
3. Select **Next** in the "Label schema" stage. You will see a progress message like `Processing data source 1 / 3: starting.` Wait for this message to disappear before continuing.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

4. In **Development Settings**, keep `No` at the **Compute embeddings**, and then select **Develop**.



Review data in Studio

In **Develop (Studio)**, you'll review the data and write labeling functions.

First, let's resample the dev to have more pages.

1. Select the **Dev split** dropdown in the top toolbar.
2. Select **Resample data**.
3. Set the **Sample size** to 10 and **Min per class** to 0, and select **Resample dev split**.

> **NOTE**
> Pages are randomly sampled across PDFs in the dev split.

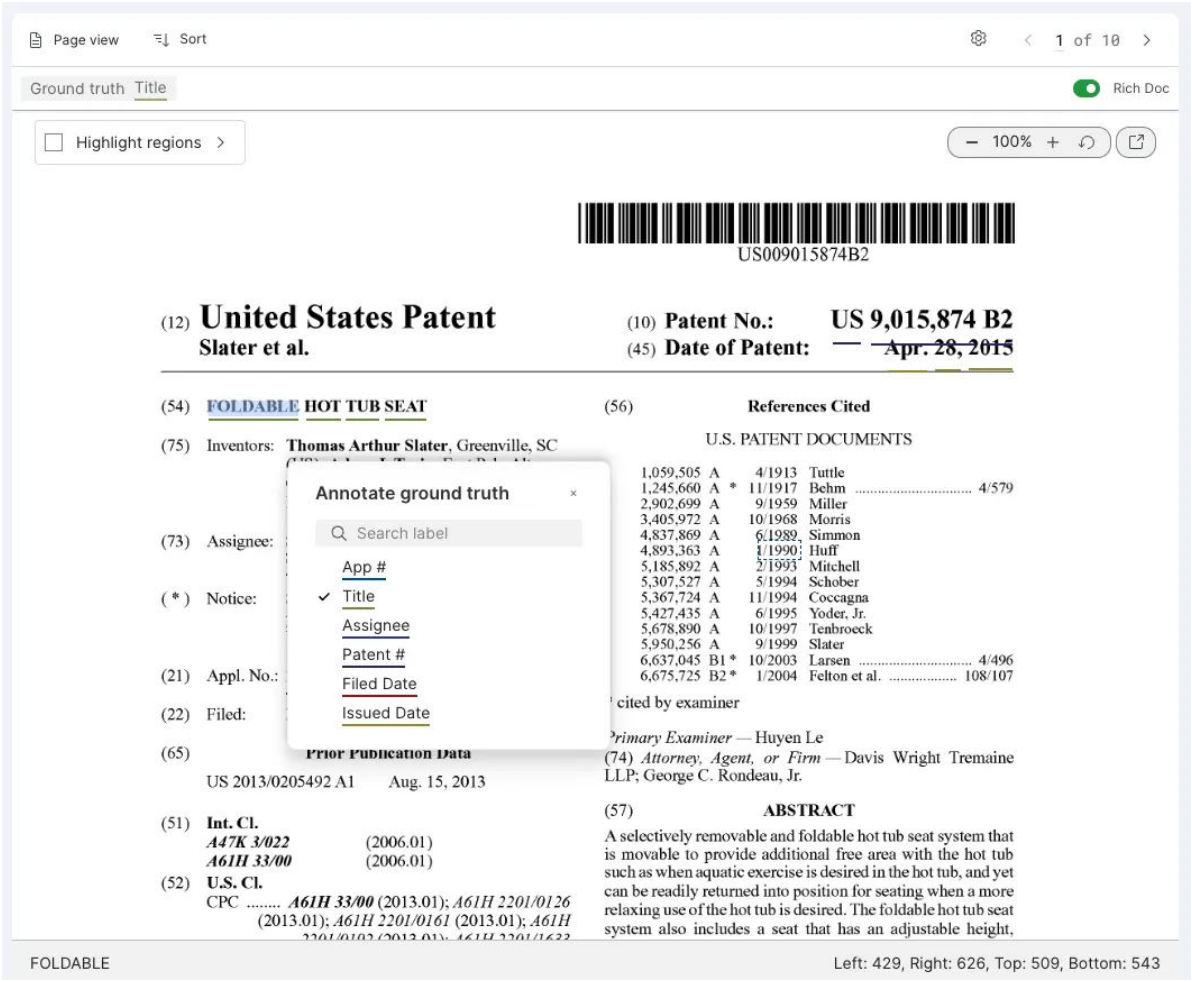
You should be able to see words that are highlighted with colored underlines if you've successfully committed annotations.

1. Select any word in the document. You can see the bounding box coordinates of the word as measured in pixels at the bottom of the screen.

2. Explore different view options in Studio:
   - Toggle **Rich Doc** off to see the plain text extracted from the document.
   - Select the arrow to expand **Highlight regions**. This allows you to highlight the bounding boxes of the different regions in the document: Area, Paragraph, Line, Word, and Row.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

- Zoom in and out of the document using the button at the top right corner.
- Pop up a modal to see the document of all pages using the button next. This is useful to see pages before and after the page you are looking at in Studio.



Next, you'll define your labeling functions.

Define labeling functions

1. Create new labeling functions.
2. In **Patterns**, select your LF template by typing `/`, followed by the pattern, such as `Word-based Expression` and `Word-based Regex`.
3. Enter the requirements for the labeling function.
4. **Select a label** from the dropdown menu.
5. Select **Preview LF** to see the precision and coverage of your LF.
6. Select **Create LF**.

The new LF shows up in the **Labeling Functions** pane.

Using the steps above, add these LFs to your application:

Copyright © 2025 Snorkel AI, Inc. All rights reserved

LF Template	Settings	Label	Explanation
Word-based Expression Builder	Label words satisfying expression: `PATTERN.right < WORD.left and abs((WORD.top + WORD.bottom)/2 - (PATTERN.top + PATTERN.bottom)/2) < 50` with regex pattern `patent (no\.|number)`	`patent #`	This LF labels words that are on the right and on the same row as "patent (no.|number)" as `patent #`.
Word-based Expression Builder	Label words satisfying expression: `PATTERN.right < WORD.left and abs((WORD.top + WORD.bottom)/2 - (PATTERN.top + PATTERN.bottom)/2) < 50` with regex pattern `Date of patent:`	`issued date`	This LF labels words that are to the right and on the same row as "Date of patent:" as `issued date`
Word-based Expression Builder	Label words satisfying expression: `PATTERN.right < WORD.left and abs((WORD.top + WORD.bottom)/2 - (PATTERN.top + PATTERN.bottom)/2) < 50 and WORD.right < 1275` with regex pattern `Filed:`	`filed date`	This LF labels words that are to the right and on the same row as "Filed:" as `filed date`, as long as it is on the left half of the page boundary
Word-based Expression Builder	Label words satisfying expression: `PATTERN.right < WORD.left and abs((WORD.top + WORD.bottom)/2 - (PATTERN.top + PATTERN.bottom)/2) < 50 and WORD.right < 1275` with regex pattern `appl\. no\.:`	`app #`	This LF labels words that are to the right and on the same row as "appl. no." as `app #`, as long as it is on the left half of the page boundary
Word-based Expression Builder	Label words satisfying expression: `PATTERN.right < WORD.left and abs((WORD.top + WORD.bottom)/2 - (PATTERN.top + PATTERN.bottom)/2) < 50 and WORD.right < 1275` with regex pattern `assignee:`	`assignee`	This LF labels words that are to the right and on the same row as "assignee:" as `assignee`, as long as it is on the left half of the page boundary

Copyright © 2025 Snorkel AI, Inc. All rights reserved

> NOTE
>
> The top-left corner x-y coordinate of each page is always (0, 0), and the bottom-right corner is (2250, 3300) when the page is in the letter size (8.5" x 11").

You can also encode your custom labeling function logic using the Python SDK. For examples using the `RichDocWrapper` object, refer to the SDK reference.

Train a model

You can aggregate the newly defined labeling functions to train a model.

1. In the **Models** pane, select **Train a model**.
2. For the **Model architecture**, select **LayoutLM**.
3. Leave the default settings in **Train options** and **Model options**.

Once the model training is completed, you will see your first model in the **Models** pane.

Model metrics

Snorkel Flow provides word-level F1 (micro), precision, and recall metrics for trained models.

Metrics are computed at the word-level rather than the document-level. For example, there are ten words in total that are all labeled as `app #`, eight words are predicted correctly as `app #`, and two words are predicted incorrectly as `patent #`, then the precision is 80%. It doesn't matter which document those ten words belong to.

If there is any labeled word in a page, all the unlabeled words in the same page are *assumed* to be labeled with the negative class. For example, in the patent example above, some words are labeled as `title`, `app #`, etc. All the unlabeled words in the same page are assumed to be labeled with `NEGATIVE`.

If there is no labeled word in a page, then all the words in the page remain unlabeled and they are ignored when computing the model metrics.

Words that are labeled and/or predicted as the negative class are ignored when computing the model metrics.

You can improve the performance of your model with these options:

- Increase the `num_train_epochs` in **Model options**.
- Add LFs for `NEGATIVE` class.

LF Template	Settings	Label	Explanation
Word-based Expression Builder	Label words satisfying expression `PATTERN.bottom < WORD.top and WORD.right < 1275` with regex pattern `Prior publication data`	`NEGATIVE`	This LF labels words that are below "Prior publication data" and are in the left half of the page as `NEGATIVE`.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

LF Template	Settings	Label	Explanation
Word-based Expression Builder	Label words satisfying expression `PATTERN.bottom < WORD.top and WORD.right > 1275` with regex pattern `References`	`NEGATIVE`	This LF labels words that are below "References" and are in the left half of the page as `NEGATIVE`.
Word-based Expression Builder	Label words satisfying expression `WORD.right < 1275 and PATTERN.top <= WORD.bottom` with regex pattern `inventors` AND another regex pattern `WORD.right < 1275 and WORD.bottom < PATTERN.top` with regex `assignee`	`NEGATIVE`	This LF labels words that are below "inventors" but above "assignee" as `NEGATIVE` to prevent inventors from being predicted as assignee.

Snorkel Flow enables users to iteratively generate better labels and better end models. For information about getting metrics and improving model performance, see Analysis: Rinse and repeat.

Although it is not supported in the user interface, you can preview and create prompt LFs through the SDK:

Copyright © 2025 Snorkel AI, Inc. All rights reserved

```
# preview a prompt LF
sf.preview_prompt_lf(
    node,
    model_name='openai/gpt-4o',
    model_type='word_text2text_qa',
    prompt_text = 'Task: Identify and list all exact answers to the question
found within the document. Return the answers separated by a semicolon (";"),
with no additional commentary not found in the document. Question: What is the
patent number mentioned in the document? Note that the patent number is only
given to an awarded patent and different from the application number.',
    primary_text_field='rich_doc_text',
    label='patent #',
    split='dev'
)

# create a prompt LF
sf.create_prompt_lf(
    node,
    model_name='openai/gpt-4o',
    model_type='word_text2text_qa',
    prompt_text = 'Task: Identify and list all exact answers to the question
found within the document. Return the answers separated by a semicolon (";"),
with no additional commentary not found in the document. Question: What is the
patent number mentioned in the document? Note that the patent number is only
given to an awarded patent and different from the application number.',
    primary_text_field='rich_doc_text',
    label='patent #',
    lf_name='patent_LF'
)
```

For more about prompt LFs, see Foundation model suite.

For more about model configuration, see Configuring external models.

What's next

You have set up and iterated on a PDF extraction application in Snorkel Flow. To learning more about PDF extraction features in Snorkel, see these topics:

- Learn how to work with scanned documents with our OCR guide.
- Read more about PDF-based Operators.

The Snorkel Flow Python SDK provides greater flexibility to define custom operations. Select **Notebook** on the left-side pane to access the SDK. See the SDK reference to learn more about custom operators and custom labeling functions.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

Information extraction: Extracting execution dates from contracts

We will demonstrate how to use Snorkel Flow for a simple text information extraction application: extracting the execution date from a set of loan agreements found in SEC filings. The execution date is the date on which a contract has been signed by all the involved parties.

In Snorkel Flow, we follow one classic approach to information extraction in which the first step is to identify a large, high-recall set of candidate spans: spans of text that might be execution dates. We then use the rest of the Snorkel Flow application to classify these candidate spans as `POSITIVE` (i.e. actually an execution date) or `NEGATIVE` (i.e. some other date), similar to how we classified documents in the previous walkthrough.

In this quick start example, we consider all dates as candidate spans. We use Snorkel Flow's built-in **date candidate extractor** to do this.

While most contract files have only one execution date, each document can contain multiple dates, such as the date it was restated/amended/terminated, or the date a reference document was executed.

1. For this demo dataset, the URL field may contain the execution date for some documents. We recommend you do *not* use this field for writing labeling functions or training end models.
2. This demo dataset is available via S3 or upon request.

Adding data sources

Before we can create an application we must create a dataset and upload data sources for each split: **train**, **test**, and **valid**. We can also upload additional data sources for each split over time. Data points in the `valid` and `test` splits come with ground truth labels. The train split is partially labeled and we'll use Snorkel Flow to label the rest of it.

First, we'll create a dataset named `loan-agreement-dataset` by clicking on the **Datasets** tab and then the **Add dataset** button. Then we'll individually add the data sources for each split by clicking **Add Data Source**. We can now pass the **File path** (see S3 paths below), and **Split** (see below) for each data source accordingly. Next, we'll hit **Verify Data Source(s)** and set **UID Column** to `uid`. Finally, we'll select **Add data source(s)** which will start the data ingestion process.

- **Test**: The held-out `test` split contains ground truth labels for evaluation purposes. While not strictly necessary, we recommend a final evaluation against ground truth (i.e. expert annotated) labels, as we have provided with these splits. Also note that you should not look at any of these data points, or else you risk biasing your evaluation procedure!
  - `s3://snorkel-loan-agreement/dropped_loan_agreement.test.parquet`
- **Valid**: Similar to `test`, the `valid` split contains some ground truth labels for evaluation purposes. This split can be used to tune your ML model.
  - `s3://snorkel-loan-agreement/dropped_loan_agreement.valid.parquet`
- **Train**: Data in this split, which does not require the use of any labels, helps guide the development of our LFs and feeds into our end model. We will add two data sources for `train`. During the development of LFs on the **Label** page, a subset of the `train` split is randomly sampled to create a `dev` split. By default, 10% of the `train` split (up to 4000 samples when sampling by doc for

Copyright © 2025 Snorkel AI, Inc. All rights reserved

applications with spans), will be used for the `dev` split, which can be updated later through resampling. We include a small number of them so that you can look at them during development to get an idea of the problem (see description of the development split below). In the subsequent steps, you will use Snorkel Flow to label the rest of this training split!
  - `s3://snorkel-loan-agreement/dropped_loan_agreement.train.parquet`
  - `s3://snorkel-loan-agreement/dropped_loan_agreement.dev.parquet`

Dataset details

Each original data point is a document with the content in the text field and the url of where the document is hosted in the URL field. For this application, we used built-in pre-processors in Snorkel Flow to extract dates from documents. Since we extracted candidates from the text field, we added several fields to the data to keep track of the spans.

- `char_start`: The index of the first character of the span in the document.
- `char_end`: The index of the last character of the span in the document.
- `context_uid`: The uid of the document from which the span was extracted. All spans that have the same context_uid come from the same document.
- `span_text`: The content of the candidate span.
- `span_field`: The field which the candidates were extracted from. In this case, it's the text field.
- `span_preview`: The span is highlighted with its context. The length of the context is specified in the load processors. In this case, `span_preview` captures 200 characters to the left and to the right of the span.
- `span_preview_offset`: The index of the first character of the span preview in the document
- `initial_label`: The label is assigned to a span at creation time (if available). After the initial upload, the label for each span comes from Snorkel Flow.

Create an application

Once you are done uploading each data source split you will create an application. Click on the **Applications** tab and select **New application**. Select the Text Extraction application template and click **Create application**. Name our application `contract-execution-date`. Also, select all fields under **Specify required input fields**. The **Unknown Label** and **Labels** sections are already filled out for us. Finally, select `text` as the **Extraction field** and hit **Save**.

Create an operator DAG in application studio

We will now add our **Span Extractor** to pull out dates via regex in our newly created **Text Extraction Block**. In this example, we show how you can write a regex-based extractor for your use case. Snorkel Flow also comes with built-in extractors (e.g. DateExtractor that you can use out-of-the-box).

Click on the **Expand 6 nodes** button to view the application DAG, and then select the **SpanExtractor** operator at the very beginning and the **RegexSpanExtractor**. Copy the following regex in the **regex** box, set the **Field** to `text`, and save by selecting **Commit**:

Copyright © 2025 Snorkel AI, Inc. All rights reserved

```
(?:[0-9]{1,2}(?:st|nd|rd|th)?(?: day)?(?: of)?.?(?:(?:Jan(?:.|uary)?)|
(?:Feb(?:.|ruary)?)|(?:Mar(?:.|ch)?)|(?:Apr(?:.|il)?)|May|(?:Jun(?:.|e)?)|
(?:Jul(?:.|y)?)|(?:Aug(?:.|ust)?)|(?:Sep(?:.|tember)?)|(?:Oct(?:.|ober)?)|
(?:Nov(?:.|ember)?)|(?:Dec(?:.|ember)?)),?.?[12][0-9]{3})|(?:(?:
(?:Jan(?:.|uary)?)|(?:Feb(?:.|ruary)?)|(?:Mar(?:.|ch)?)|(?:Apr(?:.|il)?)|May|
(?:Jun(?:.|e)?)|(?:Jul(?:.|y)?)|(?:Aug(?:.|ust)?)|(?:Sep(?:.|tember)?)|
(?:Oct(?:.|ober)?)|(?:Nov(?:.|ember)?)|(?:Dec(?:.|ember)?)).?[0-9]{1,2}
(?:st|nd|rd|th)?(?: day)?(?: of)?,?.?[12][0-9]{3})
```

Lastly, click on the **Model** operator.

Add span-level ground truth

On the **Overview** page, we get a high-level view of our application. We can also add any ground truth labels we have on this page. In the bottom third of the page you'll notice an **Upload GTs** button. After clicking on it you'll be prompted to provide the following information to import ground truth from a file:

- **File path**: `s3://snorkel-loan-agreement/loan-execution-date-gt.csv`
- **File format**: CSV
- **Label column**: `_gt_label`
- **UID column**: (leave this blank)

If you click on the **View Data Sources** button on the bottom left of the dashboard you can view the full set of data points and ground truth across splits:

	# of data points	# of GT labels	# of docs
train	4977	0	467
train (dev)	226	226	24
valid	372	372	31
test	162	162	25

We can view our data by clicking on the **Go to Studio** button on the top right.

View data at the document level

In case you have no ground truth labels in the train split and need some to help provide feedback while iterating on LFs, you can annotate directly within **Develop (Studio)**. All the examples in this quickstart include ground truth labels, so you can skip annotating if you prefer. See Ground truth annotations for more details.

For extraction-related tasks, there is both a **Record** and **Document** view, selected by clicking on the corresponding buttons above the dataviewer. Record view shows one candidate span at a time—this view is often useful because this candidate span is the actual object you are labeling with your labeling functions, and the model you will train. The document view shows one document at a time, which may

Copyright © 2025 Snorkel AI, Inc. All rights reserved

(and usually will) contain multiple candidate spans—this view is often useful to get a broader view of the context around the candidate spans.

Use the bottom arrow keys to iterate through documents and the top arrows to iterate through candidates in a given document. See Ground truth annotations for more details.

Resampling documents

Before we write LFs we will resample more documents. Click the **Dev set** dropdown in the top toolbar, then select the **Resample data** option.

Set a sample size of `100` and make sure you are sampling by docs. You can also set an optional random seed at the bottom of the box. Click **Resample dev split**.

Write labeling functions in label

In **Develop (Studio)**, you will notice two main panes:

- The left pane with accordions to
  - Manage labeling functions
  - Manage models
  - Conduct analysis
- The right pane for viewing the data

You can use the various **Filters** at the top to choose what subset of data to look at. For example, you can choose to view all samples whose ground truths (GT) are `POSITIVE`.

Note

The label for this task is on the span level, i.e. it is the label for a candidate span (not the whole document), which means it only decides whether a candidate date span (highlighted in the **Record** and **Document** viewers) is the execution date of that document or not.

One heuristic is that if a date is repeated multiple times throughout the document, it's likely to be the execution date. Let's create a **Span Count Builder** to express that. Click into the search bar at the top and choose a **Span Count Builder** from the **Span Based** submenu. Populate the template with the following information and click **Preview LF**, view results, and then click **Create LF**.

- Occurs: `>=`
- Count: `7`
- Label: `POSITIVE`

Another basic but powerful type of labeling function is one that reasons keywords around the span. This is similar to the keyword builder for document classification, except includes some basic positional logic relative to the candidate span. For example, you might notice that the dates that follow `is dated` are probably the execution date: we can create a labeling function with the **Span Context Builder** under **Span Based** for that.

- Span is `RIGHT` of `is dated` within `5` words
- Label: `POSITIVE`

You might also want to try out the **Duplicate LF** feature (accessible if you click the three dots near the LF name) to duplicate the current LF for faster LF creation.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

Tip: Regex Builder with {{span}} macro

The **Regex Builder** under **Pattern Based** can be powerful if you're comfortable with basic regular expressions. It can be combined with the macro `{{span}}`, a special token that refers to the extracted span. For example, if a span appears within this context `Closing Date shall mean {{span}}`, then it probably means the end date and not the execution date. You will also notice that the matches for this regex are highlighted in the **Data sources** tab, which can help with regex iterations.

- If field: `span_preview`
- Contains the regular expression: `\"Closing Date\" shall mean {{span}}`
- Then Label: `NEGATIVE`

Tip

Tip

Use filters to balance the classes

If, for example, we notice that most of our LFs are for `NEGATIVE` class, we could use the filters to look for patterns for `POSITIVE`. In fact, you can use the filters to look at all the data points whose ground truths are `POSITIVE` and are not yet covered by any labeling function.

Click the **Filters** icon in the top right. Select **Advanced Filters** and under **Recommended Filters** select **Show Uncovered Examples**. This filter is just a shortcut for `Ground Truth` is `POSITIVE` and `All LFs` (under **Labeling Functions** submenu) vote `UNKNOWN`.

Click around to explore your data and get more ideas for your labeling functions. Try to write at least one LF for each class that has 90%+ precision and 5%+ coverage. Here are some LFs you can try:

- Span-Based LFs > Span Context Builder
  - Span is `RIGHT` of `[0-9]` within `2` words
  - Label: `NEGATIVE`
  - Click the three dots to the left of **Span Context Builder** and **Advanced Settings** and enable **Regex** functionality
- Span-Based LFs > Span Context Builder
  - Span is `LEFT OR RIGHT` of `and` within `4` words
  - Label: `NEGATIVE`
- Span-Based LFs > Span Context Builder
  - Span is `RIGHT` of `EXECUTION VERSION` within `3` words
  - Label: `POSITIVE`
- Span-Based LFs > Span Context Builder
  - Span is `RIGHT` of `Contract Date:` within `3` words
  - Label: `POSITIVE`
- Span-Based LFs > Span Context Builder
  - Span is `LEFT` of `(the "Effective Date")` within `5` words
  - Label: `POSITIVE`

Copyright © 2025 Snorkel AI, Inc. All rights reserved

Analyze labeling function performance

See **Analyze Labeling Function performance** in Document classification: Classifying contract types.

Data exploration and visualization with in-platform notebook

The in-platform notebook in Snorkel Flow can be a powerful platform for analyzing and visualizing data to come up with new ideas for labeling functions, using either the built-in functions or external libraries.

For example, we can create a boxplot of the distribution of `char_start` to help us notice that most of the execution dates appear near the beginning of the document.

```
import matplotlib.pyplot as plt
# This dataset filter only looks at data points with no LF labels
df = sf.get_dataset(node, all_lfs_filter="UNKNOWN")
df.boxplot(column=["char_start"], by="GT", grid=False)
plt.ylim([0,500])
```

We see that we can use the condition that the span occurs in the first 350 characters to label the span as `POSITIVE`. We can use the **Span Location Builder** under **Span Based** labeling function builders where:

- In: `chars`
- Start index: `0`
- End: `350`
- Label: `Positive`

LF development with in-platform notebook

The In-platform Notebook interface allows you to develop custom labeling functions and explore your data using Jupyter. For this walkthrough, we'll cover one type of visualization and custom LF.

Say we want to look for whether other years (like 2005, 1995, etc.) are mentioned near the extracted span. We can write an LF that counts how many times the string `20` appears in the `span_preview` field.

```
from snorkel.labeling.lf import labeling_function

@labeling_function(name="my_nb_lf")
def lf(x):
    if x.span_preview.count("20") >= 2:
        return "NEGATIVE"
    else:
        return "UNKNOWN"

sf.add_code_lf(node, lf, label_str="NEGATIVE")
```

Train a simple model

You can train a model on any training set. For extraction tasks, we offer **Span Aware** model configs, in addition to all default model configs used for classification. Span Aware models train and predict the extracted spans and their surrounding context. Additionally, Span Aware models provide options for

Copyright © 2025 Snorkel AI, Inc. All rights reserved

classifying spans, such as masking the span text to prevent overfitting. Additional details on models can be found in Model training.

> NOTE
> The models with the * next to them are computationally intensive, and we recommend training them on GPU if possible.

Model training in the SDK

We have also included an example Jupyter notebook that uses the SDK to load the dataset and its generated labels to train a model. You can use a notebook like this one to test out other models that are not yet supported in the app or try different settings.

You can use `extraction_model_examples.py` for this task, which can be found under **Notebook** > **File Browser** > `examples` Python SDK to train an external model for this task.

Span and document level metrics

In the information extraction setting, we often have two levels at which we'd like to evaluate the final model performance:

- **Span level**: This is how many candidate spans were classified correctly – e.g. in our setting, how many dates we correctly marked as either `POSITIVE` (an execution date) or `NEGATIVE` (not an execution date).
- **Document level**: This measures our performance at extracting the correct value(s) from the overall document – e.g. the execution date of the document.

In many settings, such as this one, we care less about classifying each date correctly (the span level score), and instead just want to find the single execution date of the document with high accuracy (the document level score).

After the model finishes training, you can see the span-level metrics for the trained model on the non-train splits. Span-level metrics summarize the binary classification metrics over the extracted candidate data points. This is a good proxy for model training, but not the metric that we care about.

Inferring document ground truth via SDK

Since we have manually uploaded ground truth via CSV file, we will first need to explicitly infer the document ground truth (this is not the case when annotating spans in the platform). Inferring document ground truth using SDK is necessary for loading document-level metrics, so refresh after running the SDK steps to see document-level metrics in the UI. Open the **Notebook** interface and run the following lines:

```
import snorkelflow.client as sf
# NODE = <insert your node id here>
sf.get_inferred_document_ground_truth_from_span_ground_truth(NODE,
split="dev")
sf.get_inferred_document_ground_truth_from_span_ground_truth(NODE,
split="valid")
sf.get_inferred_document_ground_truth_from_span_ground_truth(NODE,
split="test")
```

Copyright © 2025 Snorkel AI, Inc. All rights reserved

Document-level metrics

These metrics measure how well our model did at identifying the correct execution date for each document. To calculate these metrics, we first need a way to determine which span with a positive predicted label in the document is the execution date. In Snorkel Flow, we abstract this as two basic operations:

- **Normalization**: Transforming the raw text of the candidate spans to a canonical form of the object we actually care about, e.g. a date.
- **Reduction**: Picking a single extraction from the many different candidates our model has made positive / negative predictions over.

In Snorkel Flow, **Normalizers** and **Reducers** perform this process of aggregating predictions across spans into the one execution date associated with the document. To add them we will need to go back to the **Application Studio** page where we defined our application DAG, or we can get there also via a short-cut by clicking on the app name in the navigation breadcrumbs across the top of the application.

Normalizers

- We will use the **DateSpanNormalizer**, which converts all spans into the YYYY-MM-DD format. At the very end of the DAG you will notice an **IdentitySpanNormalizer** operator, which just copies the span text as-is, and an empty **Reducer** placeholder. Replace the **IdentitySpanNormalizer** with a **DateSpanNormalizer** by clicking on the button with three vertical dots next to the normalizer and hit **Uncommit**. Now, click on the placeholder and add the **DateSpanNormalizer** operator, and hit **Commit**.

Reducers

- The **DocumentMostConfidentReducer** will select the date with the highest probability of being a positive span as being the execution date for a given document. Click on the **Reducer** operator placeholder and follow the same steps as above.
- You can also try the **DocumentFirstReducer**, which will select the first positively-predicted span in the document. This particular reducer is a good heuristic for this application since execution dates tend to appear near the beginning of the document.

View errors and iterate

See **View errors and iterate** in Document classification: Classifying contract types.

View model predictions

You can view the model predictions for the spans in the **Document view** in **Develop (Studio)**. Click the gear icon on the top right corner and select **Color Span by** the model of interest, (e.g. “run #0”). This will change the span highlighting to be based on the model prediction. If you hover over individual spans, you can see the predicted class with the confidence associated with the prediction and the ground truth label.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

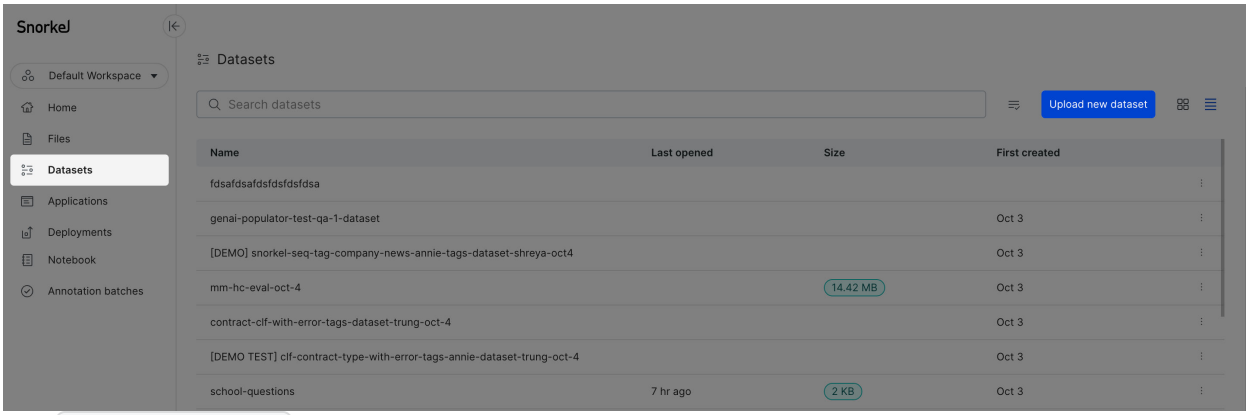
Multi-label classification: Predicting tags of posts on Stack Exchange

We will demonstrate how to use Snorkel Flow for a simple multi-label classification problem: predicting tags that were added to posts on Stack Exchange, specifically on the Cross Validated board which covers: "statistics, machine learning, data analysis, data mining, and data visualization." Each data point contains the text of the posted question title, question body, answer, some generic metadata; as well as tags which will be our independent variable. Given that a datapoint may contain 0 to N tags where N is the number of possible tags, we will use Snorkel Flow's Multi-Label Application Template.

- How to create a multi-label classification application
- How to write labeling functions for multi-label datasets

Create a dataset

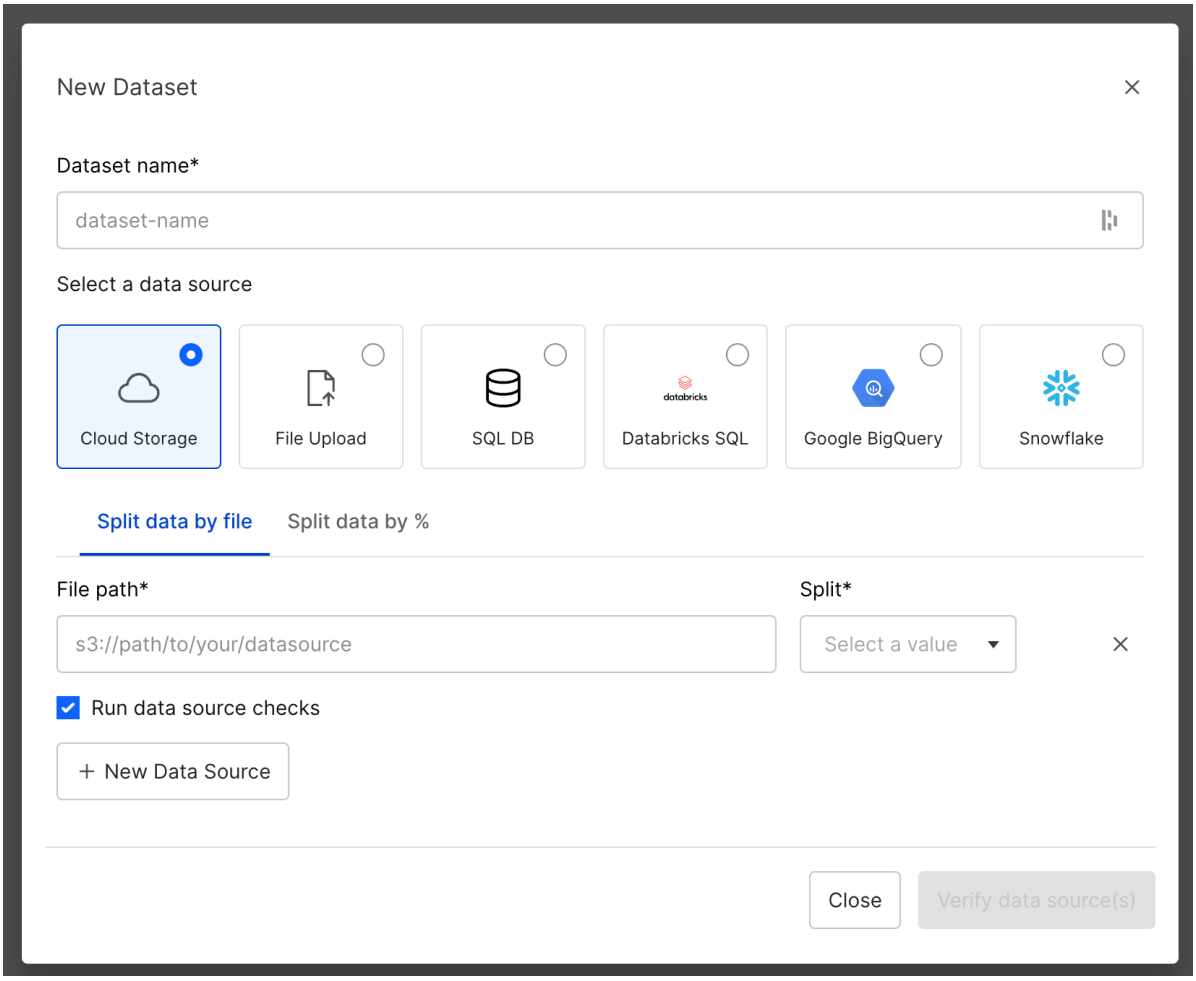
Navigate to the **Datasets** page from the sidebar menu. Click on **+ New dataset**.



Use `stats-questions` as your **dataset name** and the following parquet-based **data sources** with the specified splits. Use `Id` as the **UID column**.

split	path
train	`s3://snorkel-statsquestions-dataset/train_n20.parquet`
valid	`s3://snorkel-statsquestions-dataset/valid_n20.parquet`
test	`s3://snorkel-statsquestions-dataset/test_n20.parquet`

Copyright © 2025 Snorkel AI, Inc. All rights reserved



Multi-label label format

In general, multi-label labels have a unique format that must be maintained to be able to upload, export, and transfer labels on the platform. Multi-label labels are a dictionary that has the label classes as keys, and the "vote" assigned to each class as values. In contrast to other libraries, which only permit a binary vote on each class, Snorkel Flow also permits you to *abstain* from voting on a class (your classes can be *partially* labeled). The "vote" values then are any of `"PRESENT"`, `"ABSENT"`, or `"ABSTAIN"`.

Example:

```
'{
 "r": "PRESENT",
 "regression": "PRESENT",
 "bayesian": "ABSTAIN",
 "probability": "ABSENT",
 "time-series":"ABSTAIN"
}'.
```

Importing and exporting ground truth, training set labels, and model predictions

- Importing Ground Truth

Copyright © 2025 Snorkel AI, Inc. All rights reserved

- Import Ground Truth by file: your labels should be JSON-serialized (ie. they should be strings, not raw JSON objects). You can use SDK sf.import_ground_truth to upload your Ground Truth by file.
- Add Ground Truth for your list of datapoints using SDK function sf.add_ground_truth. deserialize the labels to be in their original data structure:
  - ```
    x_uids = ["doc::0"]
    labels = ['{ "r": "PRESENT", "regression": "PRESENT", "bayesian":
    "ABSTAIN", "probability": "ABSENT", "time-series":"ABSTAIN" }'.]

    # This will fail since the labels are JSON serialized as strings
    sf.add_ground_truth(node, x_uids, labels, user_format=True)

    # This will succeed since the labels have been deserialized into their
    original data structures
    deserialized_labels = [json.loads(label) for label in labels]
    sf.add_ground_truth(node, x_uids, deserialized_labels,
    user_format=True)
    ```

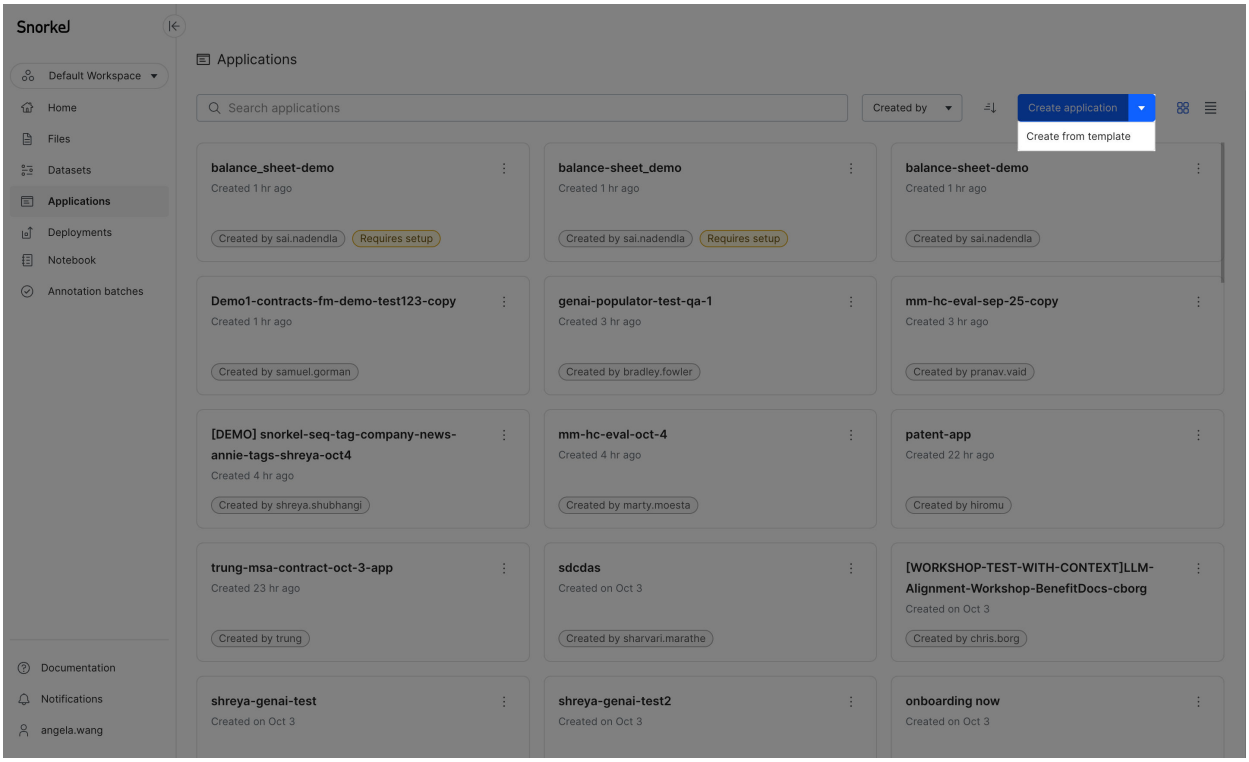
- Exporting training set labels
  - Use the SDK function sf.get_training_set as below, this will give you a dataframe containing all fields for training, and a field named `training_set_labels`:
  - ```
    sf.get_training_set(NODE_UID, uid=TRAINING_SET_UID, user_format=True)
    ```

- Exporting model predictions
  - Use the SDK function sf.get_predictions as below, this will give you a dataframe containing two columns: `preds` and `probs`:
  - ```
    sf.get_predictions(NODE_UID, model_uid=MODEL_UID, user_format=True)
    ```

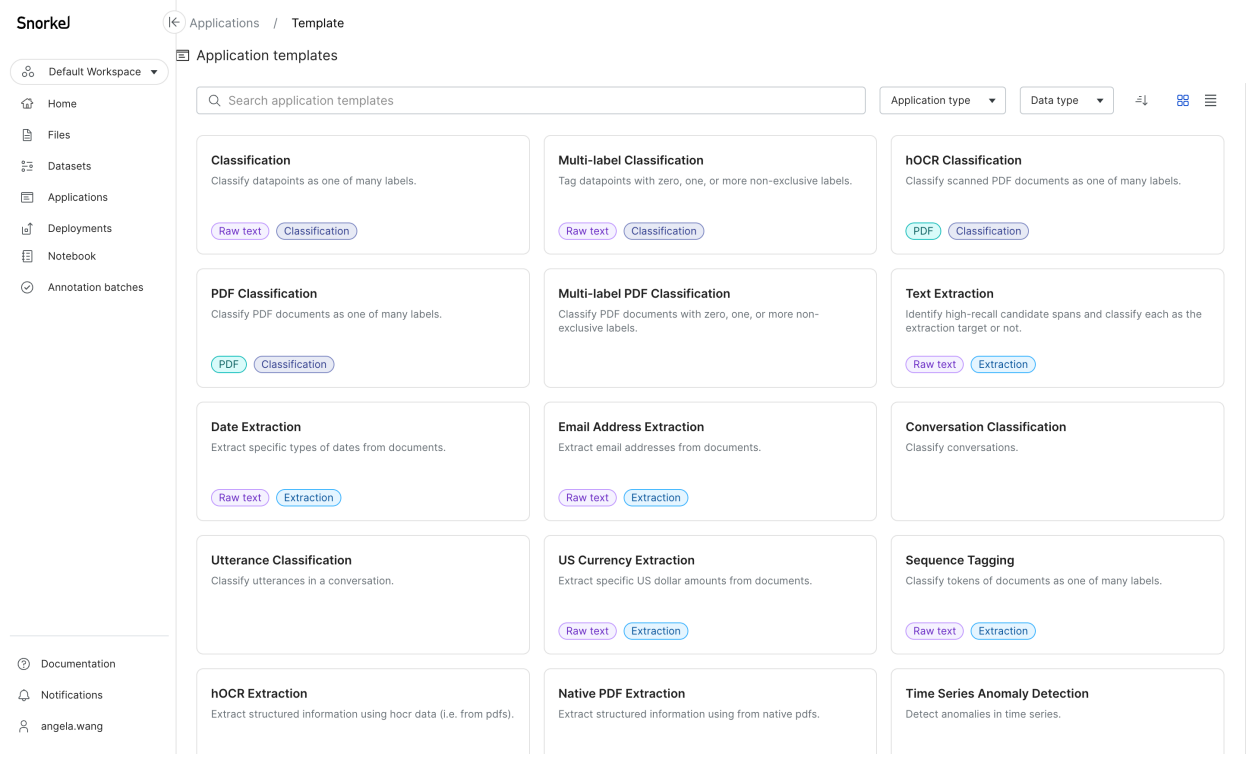
Create a multi-label classification application

In **Applications** page, click on the dropdown menu in the **Create application** button and select **Create from template** in the dropdown.

Copyright © 2025 Snorkel AI, Inc. All rights reserved
    ```

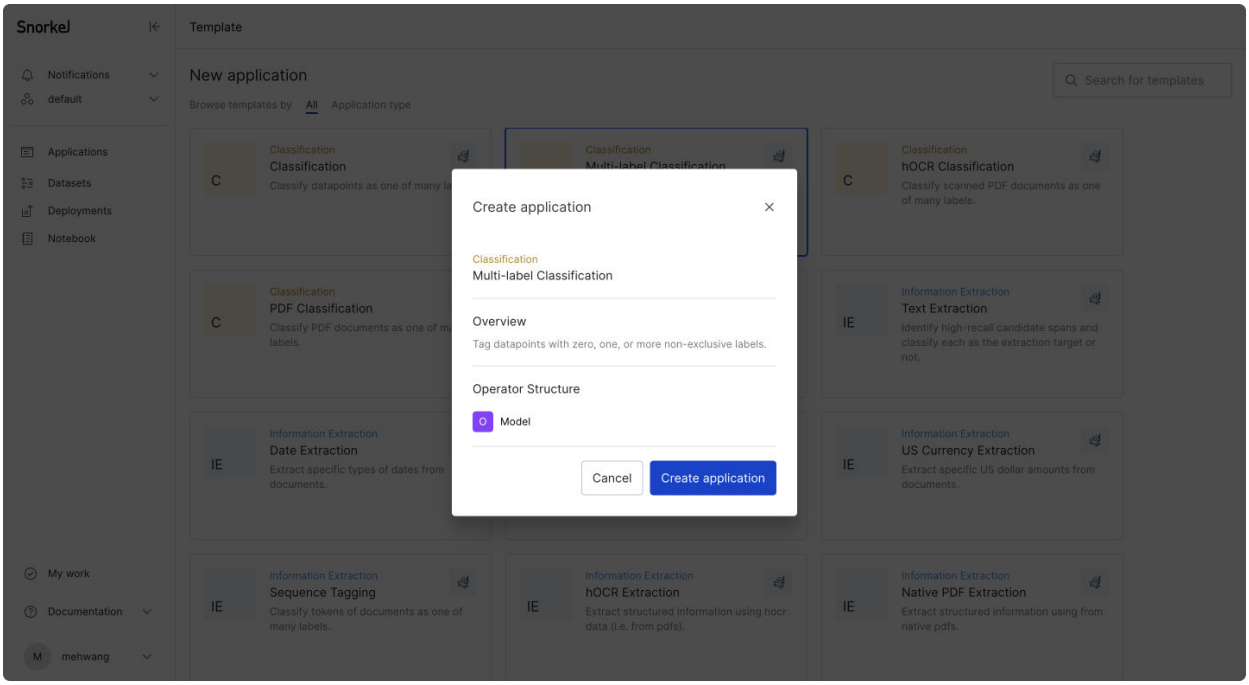


Select **Multi-label Classification** template on the template page.



Click the **Create application** button.

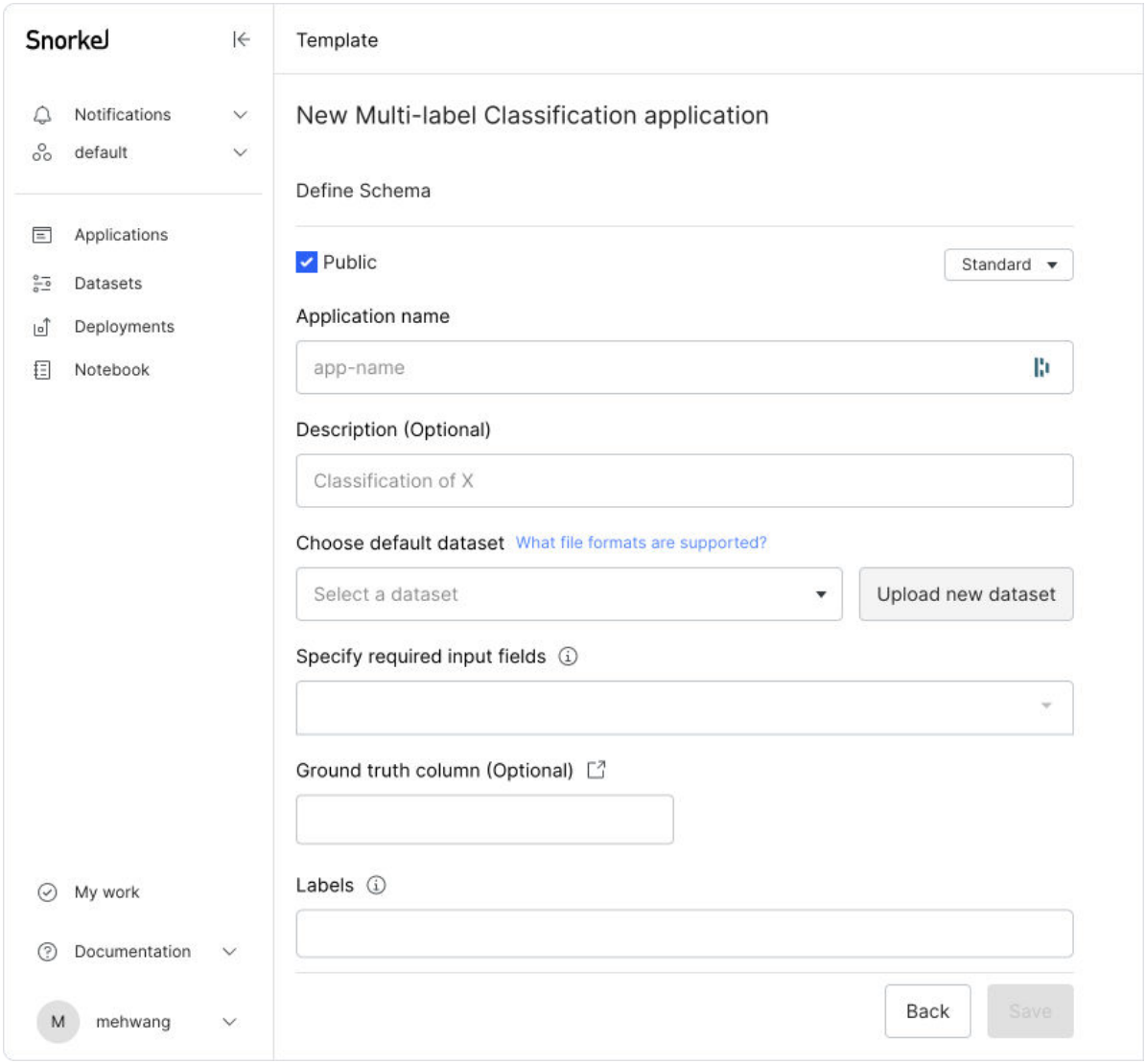
Copyright © 2025 Snorkel AI, Inc. All rights reserved



Fill out the fields as described below:

- **Application name**: `question-tag-clf`
- **Description (Optional)**: `Multi-label classification task`
- **Choose default dataset**: `stats-questions` (created above)
- **Specify required input fields**: Click on **Select all**
- **Ground Truth column (Optional)**: `Tag`
- **Labels**: `bayesian`, `time-series`, `r`, `regression`, `probability`

Copyright © 2025 Snorkel AI, Inc. All rights reserved



Data splits

This task consists of three data splits: `train`, `valid`, and `test`. Data points in the `valid` and `test` splits have ground truth labels assigned to them. The `train` split is partially labeled and we'll use Snorkel Flow to label the rest of it.

	train	valid	test
# data points	25,664	3,667	7,333
# ground truth labels	9,468	1,340	2,627

Copyright © 2025 Snorkel AI, Inc. All rights reserved

Write labeling functions

Filters

Note how filters operate somewhat differently to search for matching labels in the multi-label setting. When searching for ground truth, LF, model prediction, or training set votes, we can select from “present”, “absent”, or “abstain” for a particular class. That is, to find any doc where the “bayesian” tag exists, we use the filter `Ground truth is "present" on "bayesian"`. In addition to just searching for a particular class, filtering can be applied on any/all classes. These filters will check that any/all non-abstain values match the condition.

- **Incorrect predictions:** If there is at least one mistake in predicting the presence or absence of a label when GT is not abstained
- **Correct predictions:** When all classes of non-abstained prediction matches all classes of non-abstained GT, it's considered correct.
- **Unknown predictions:** If all the classes of a datapoint are abstained, the prediction is considered as "UNKNOWN."

LF builders for multi-label classification

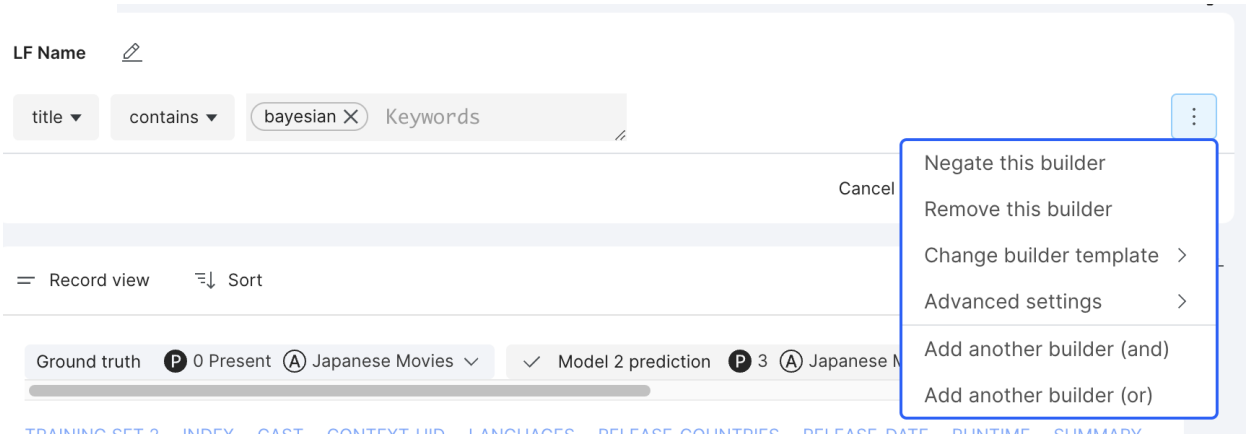
Our data comes with three relevant text fields:

- `Title`: title of the post
- `Body_q`: the body of the asker’s question
- `Body_a`: the body of the top responder’s answer

Note that for multi-label, each class can give its own vote for a single datapoint, e.g. a `Title` like “Time Series Analysis in R” could have a ground truth label of: `time-series: PRESENT`, `r: PRESENT`, `bayesian: ABSENT`, `regression: ABSENT`, `probability: ABSENT`. As such, labeling functions vote in a similar format, where each class can vote either “PRESENT”, “ABSENT”, or “ABSTAIN”.

In a single-label setting, voting for one label implicitly tells our model that the other labels are absent. In a multi-label setting, this is not true since the presence of one label doesn’t give any information about the other labels. Therefore, we should try to write labeling functions over both “PRESENT” examples and “ABSENT” examples for each class. If a class only has “PRESENT” votes, the model will naively learn to vote “PRESENT” on every example!

Below are some sample labeling functions you can try out. You can combine multiple labeling functions using **Add another builder (and)** or **Add another builder (or)** from the LF builder dropdown.



Copyright © 2025 Snorkel AI, Inc. All rights reserved

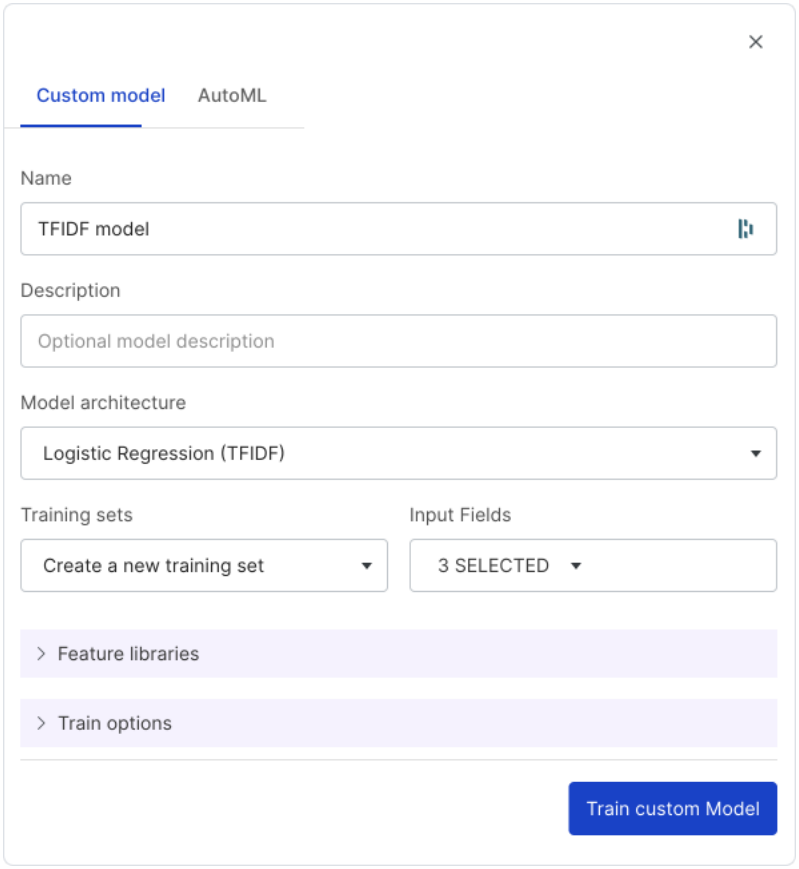
- KEY-bayesian
  - Pattern Based LFs > Keyword Builder
    - `Title` `CONTAINS` `["bayesian"]`
  - `OR`
  - Pattern Based LFs > Keyword Builder
    - `Body_q` `CONTAINS` `["bayesian", "MCMC", "GMM"]`
  - Label: `bayesian: PRESENT, time-series: ABSENT`
- KEY-regression
  - Pattern Based LFs > Keyword Builder
    - `Body_q` `CONTAINS` `["linear regression"]`
  - `OR`
  - Pattern Based LFs > Keyword Builder
    - `Title` `CONTAINS` `["regression"]`
  - Label: `regression: PRESENT`
- KEY-time-series
  - Pattern Based LFs > Keyword Builder
    - `Body_q` `CONTAINS` `["time series"]`
  - Label: `time-series: PRESENT, bayesian: ABSENT, regression: ABSENT, probability: ABSENT`
- RGX-probability
  - Pattern Based LFs > Regex Builder
    - `Body_a` `probabili`
  - `AND`
  - Pattern Based LFs > Regex Builder
    - `Title` `probabili`
  - Label: `probability: PRESENT, time-series: ABSENT`
- RGX-R
  - Pattern Based LFs > Regex Builder
    - `Body_q` `\bin R\W`
  - Label: `r: PRESENT, time-series: ABSENT`

After saving an LF, you'll see the macro-averaged score for that LF across all voted classes. The `View Incorrect` button can be used to iterate on the LF.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

Train a model

Model training is allowed without covering every class with present and absent votes, though it is recommended for the reasons above. After creating a labeled dataset, try training a simple `Logistic Regression (TFIDF)` model. For input fields, use the same fields we used for writing labeling functions: `Title`, `Body_q`, `Body_a`.

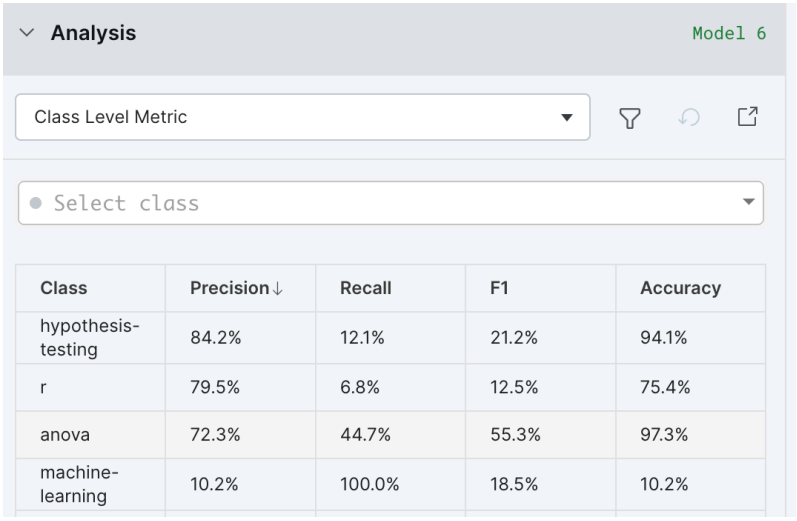


View errors and iterate

In a single-label setting, we have a single analysis page that covers all labels. In the case of multi-label, we have per-class analysis sections for each label class.

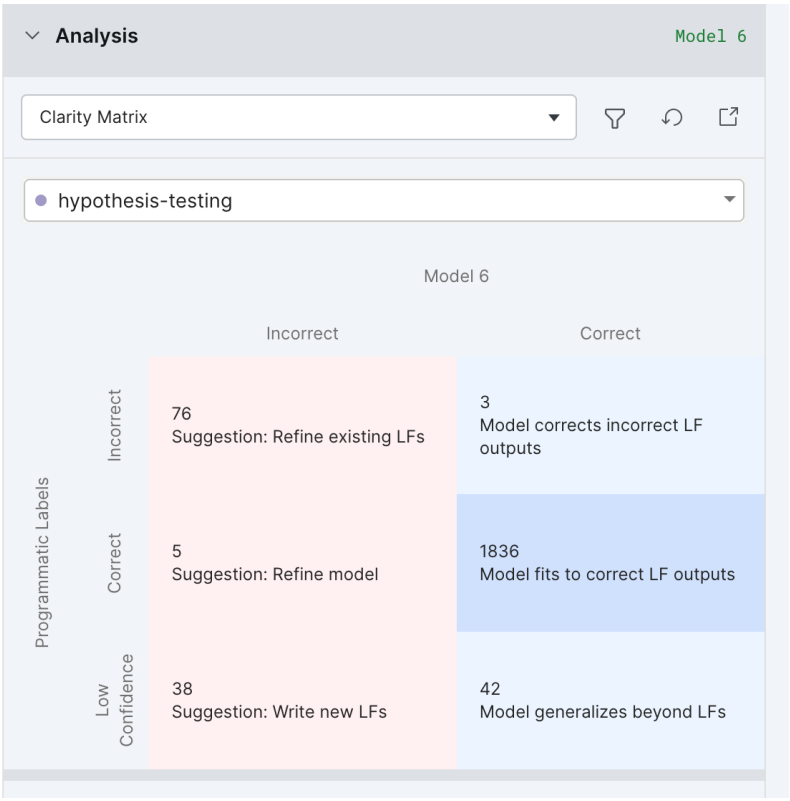
Class level metric: This table displays the classic metrics (accuracy, recall, precision, and F1) for each individual class and is shown by default when entering the Analysis section for the first time.

Copyright © 2025 Snorkel AI, Inc. All rights reserved



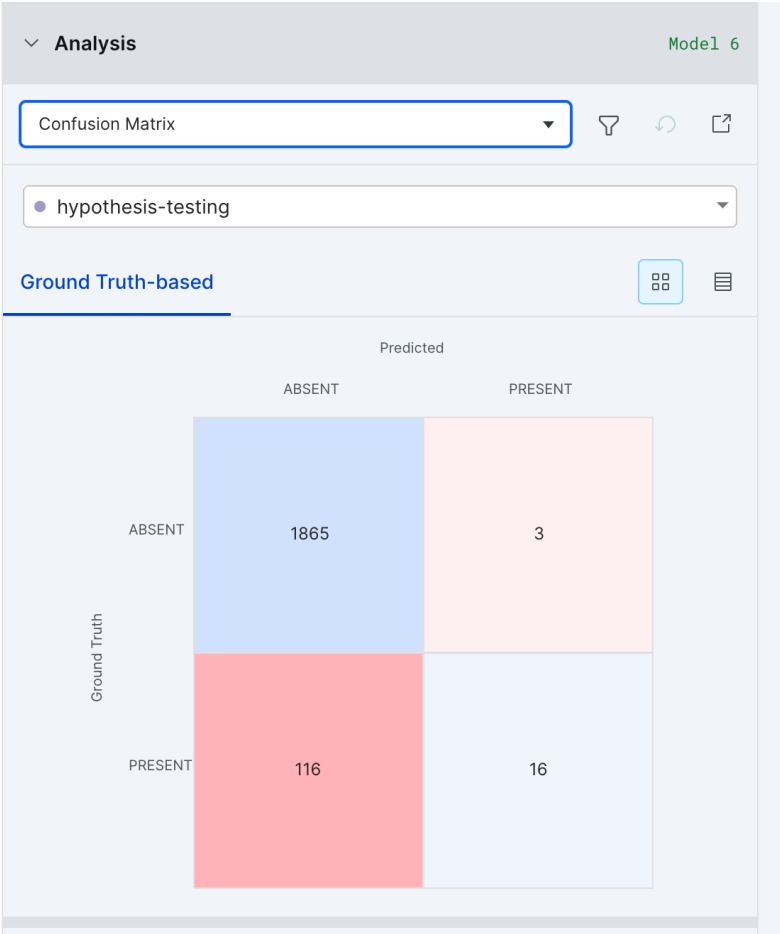
Recommended analysis flow is to sort by any metric of interest in the class level metric table to determine the class to work on. Then select that class in the class selector to proceed to do per class analysis with other analysis tools of your choice like clarity matrix, confusion matrix etc.,. Repeat these steps for each class until a desired model performance is reached.

Clarity matrix: only contains datapoints relevant to the current class. Every data point with ground truth is represented exactly once in each per-class clarity matrix.



Confusion matrix: only contains datapoints relevant to the current class.

Copyright © 2025 Snorkel AI, Inc. All rights reserved



Metrics

Snorkel Flow provides a set of metrics that help power our analysis suite and guide iteration.

- Standard metrics: In a traditional multi-class classification problem, the definitions of various metrics (such as accuracy, precision, and recall) align with their standard textbook definitions. In a multi-label setting, we instead opt to take the macro-averaged forms of these metrics, meaning we calculate per-class metrics over all classes and then take the average. We only calculate these metrics over points where neither the ground truth nor the prediction are `ABSTAIN`s.
- Aggregated metrics: Multi-label classification also supplies aggregate set-valued metrics, described in the platform as `any-present` and `all-present`. For a data point to receive a score under these metrics, the ground truth and prediction must overlap (non-abstain) on at least one class; and at least one of them must vote `PRESENT` on a class in the "overlap". If one of the two votes abstains, we do not count that class in our metric calculation. Of the classes in the "overlap", `any-present` is true when both the ground truth and prediction vote `PRESENT` on at least one class, and `all-present` is true when both the ground truth and prediction vote `PRESENT` on all the same classes.
    - Example:

Copyright © 2025 Snorkel AI, Inc. All rights reserved

```
ground_truth: {"Class0": "PRESENT", "Class1": "ABSENT", "Class2":
"ABSTAIN"}
prediction: {"Class0": "PRESENT", "Class1": "PRESENT", "Class2":
"ABSENT"}
ALL: False, since not all PRESENT votes line up among points where
neither abstained.
ANY: True, since one PRESENT vote lines up among points where neither
abstained.
ground_truth: {"Class0": "PRESENT", "Class1": "ABSENT", "Class2":
"ABSENT"}
prediction: {"Class0": "ABSTAIN", "Class1": "ABSENT", "Class2": "ABSENT"}
ALL: Not counted, since classes where neither abstained do not have any
PRESENT votes.
ANY: Not counted, since classes where neither abstained do not have any
PRESENT votes.
```

- Inter-annotator agreement metric: In contrast to the macro-averaged form of the aforementioned standard metrics, the inter-annotator agreement metric, often quantified using Krippendorff's Alpha, is evaluated by considering all classes simultaneously. For a given datapoint, the labels assigned to all classes must precisely match between the ground truth and the predicted values in order to be considered accurate.

Known limitations of multi-label classification

- End model does not support use LF as features
- Label model can not be used as end model
- Custom metrics are not supported

Copyright © 2025 Snorkel AI, Inc. All rights reserved

Sequence tagging: Extracting companies in financial news articles

We will demonstrate how to use Snorkel Flow to extract mentions of companies in financial news articles with state-of-the-art sequence tagging technology. In this tutorial, you will learn:

- How to create a sequence tagging application
- How to write labeling functions for sequence tagging application

Known limitations with sequence tagging

The following items are known limitations and are planned to be added in future releases.

- Active learning.
- Reviewer workflows in Annotation Studio.
- Labeling function conflict in Studio data viewer filters.
- Margin distance filter.
- Advanced filters (composing with "OR," "ALL/ANY LF Voted")
- Sequence Keyword LF supports at most ten keywords.
- Sequence Fuzzy Keyword LF and Sequence Word Vector LF support at most three keywords.
- Custom metrics.

Create a dataset

First, we will create a dataset named `company-dataset` by clicking on the **Upload new dataset** button in the **Datasets** page. Ensure that the **Enable multi-schema annotations** checkbox is selected. Then we will add the following data sources for each split:

- **Train**: `s3://snorkel-financial-news/mini_train.csv`
- **Valid**: `s3://snorkel-financial-news/mini_valid.csv`

After adding these two data sources, click the **Verify Data Source(s)** button. Fill out the following fields:

- **UID Column**: `index`
- **Data type**: `Raw text`
- **Task type**: `Sequence tagging`
- **Primary text field**: `text`

Finally, click the **Add data source(s)** button. For more information about uploading data, see Data upload.

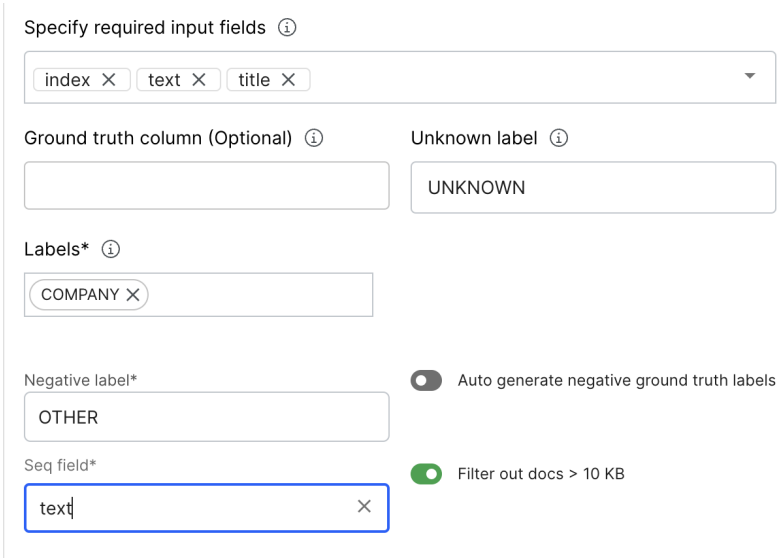
Create an application

In **Linked Applications** tab, click the **Create from Template** button from the **Create Application** dropdown menu. Select **Sequence tagging** and **Create Application**. Fill out the following fields:

- **Application name**: `company-ner-seq-tagging`
- **Choose default dataset**: `company-dataset` (created above)
- **Specify required input fields**: Select `text`, `context_uid`, and `title`
- **Ground truth column**: leave as blank

Copyright © 2025 Snorkel AI, Inc. All rights reserved

- Negative label: `OTHER`
- Labels: `COMPANY`
- Seq field: `text`



Note that the **Negative Label** (`OTHER` in this case) is equivalent to the `O` (non-entity tokens) in BIO notation.

Adding ground truth

Ground truth format

Here is an example ground truth file:

__DATAPOINT_UID	start	end	label
doc::1	29	40	COMPANY
doc::1	228	239	COMPANY
doc::2	395	406	COMPANY

Spans cannot be empty (`char_start` must be smaller than `char_end`). Overlapping or duplicating spans are not allowed. Negative spans (`OTHER`) are automatically inferred. Refer to `snorkelflow.client.add_ground_truth()` for more information.

By default, two preprocessors are added in the DAG: `TextSizeFilter` filters out documents larger than 10 KB. `AsciiCharFilter` filters out the non-ascii characters from the documents. This may result in some changes in offsets when exporting ground truth out of Snorkel Flow.

Upload ground truth file

On the **Overview** page we get a high-level view of our application. We can also add any external ground truth labels we have on this page. On the center of the dashboard you'll notice a **Upload GT** button. After clicking on it you'll be prompted to provide the following information to import ground truth from a file:

Copyright © 2025 Snorkel AI, Inc. All rights reserved

- File path: `s3://snorkel-financial-news/aligned_mini_gts.csv`
- File format: `CSV`

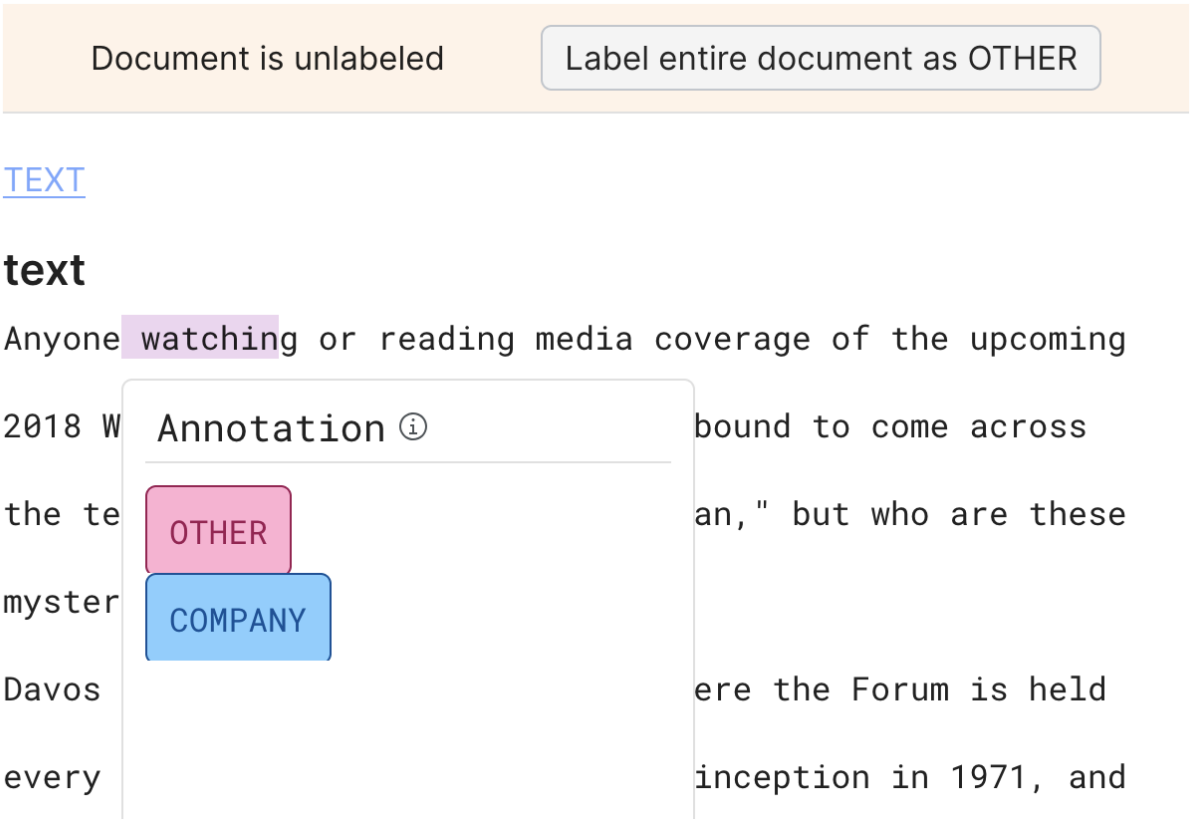
If you click on the **View Data Sources** button on the bottom left of the dashboard you can view the full set of data points and ground truth across splits:

	# of data points	# of GT labels
train	2000	78
valid	100	66

In the sequence tagging application, each document is represented as a datapoint. The # of GT labels refers to the number of documents with one or more spans annotated with GT. We can now view our data by clicking the **Develop** button on the top right.

Annotate ground truth

You can also annotate ground truth labels in the **Develop** page directly by highlighting a span using either a double click or drag select mechanisms and assigning the label from the drop-down menu.



Write labeling functions

LF builders for sequence tagging application

Back on the **Label** page, you will notice two main panes:

Copyright © 2025 Snorkel AI, Inc. All rights reserved

- The left pane for creating labeling functions and viewing labeling function statistics
- The right pane for viewing the data

Here are some LFs you can try:

- SKEY-facebook
  - Sequence Pattern Based LFs > Sequence Keyword Builder
    - `text` Contains a keyword in `["facebook"]`
  - Label: `COMPANY`
  - Vote all the mentions of "facebook" as `COMPANY`
- SRGX-Inc-Ltd-Corp
  - Sequence Pattern Based LFs > Sequence Regex Builder
    - `text` Contains the regular expression `[A-Z]+[a-z]*,? (Inc|Ltd|Corp)\.?`
  - Label: `COMPANY`
  - If a capitalized word is followed by "Inc" or "Ltd" or "Corp," vote the whole phrase as `Company`
- SRGX-Shareof
  - Sequence Pattern Based LFs > Sequence Regex Builder
    - `text` Contains the regular expression `(?<=Shares of)\s([A-Z][a-z]+\s){1,5}`
  - Label: `COMPANY`
  - Vote 1 to 5 capitalized words, as many times as possible, after the phrase "Shares of" as `Company`
- SRGX-CEOCFO
  - Sequence Pattern Based LFs > Sequence Regex Builder
    - `text` Contains the regular expression `\w+(\'|\')?s? (?=(CFO|CEO|President))`
  - Label: `COMPANY`
  - Vote the word before "CFO", "CEO" or "President" as `Company`
- SED-f500
  - Sequence Pattern Based LFs > Sequence Entity Dict Builder
    - `text` Contains the patterns from this file `s3://snorkel-workshop-data/financial-news/f500_ticker_key_fixed.json`
    - This file contains a dictionary of Fortune 500 company names, stock tickers, and their aliases.
  - Label: `COMPANY`
- SRGX-lower
  - Sequence Pattern Based LFs > Sequence Regex Builder
    - `text` Contains the regular expression `\b[a-z]+\b`

Copyright © 2025 Snorkel AI, Inc. All rights reserved

    - Label: `OTHER`
    - Vote all the lower case words as `OTHER`
- SRGX-wsww
    - Sequence Pattern Based LFs > Sequence Regex Builder
        - `text` Contains the regular expression `\(\w+\)`
    - Label: `OTHER`
    - Vote all the words in the parentheses, along with the parentheses as `OTHER`
- SSPROP-adj-adv-verb
    - Sequence Pattern Based LFs > Sequence Spacy Prop Builder
        - If the tokens in this Spacy field `doc` is tagged with any of `ADJ`, `ADV`, `VERB`
    - Label: `OTHER`
- SNER-DATE-PERSON
    - Sequence Pattern Based LFs > Sequence NER Builder
        - If the spans in this NER field `doc` is tagged with any of `DATE`, `PERSON`
    - Label: `OTHER`

LF metrics for sequence tagging application

After saving an LF, you'll see the score for that LF. Note that in the sequence tagging application, the metrics for LFs are computed at the character-level.

LF development with in-platform notebook

The in-platform notebook interface allows you to develop custom labeling functions and explore your data using Jupyter. Here is an example of a custom LF. You can find details on other available helper functions in the Python SDK.

We can use the pre-defined dictionary derived from the 2016 Fortune 500 list to build a custom LF function to label all the Fortune 500 Companies as `COMPANY`:

Copyright © 2025 Snorkel AI, Inc. All rights reserved

```
from snorkelflow.studio import resources_fn_labeling_function

def get_taxonomy():
    import fsspec
    import json
    taxonomy_file = "s3://snorkel-financial-news/f500_ticker_key_fixed.json"
    with fsspec.open(taxonomy_file, mode='rt') as f:
        taxonomy = json.load(f)
    return {"taxonomy": taxonomy}
@resources_fn_labeling_function(name="lf_taxonomy", resources_fn=get_taxonomy)
def lf_taxonomy(x, taxonomy):
    import re
    spans = []
    for ticker, companies in taxonomy.items():
        pat = '\s(' + '|'.join(companies).replace('\\', '') + ')\s'
        for match in re.finditer(pat, x.text):
            char_start, char_end = match.span(1)
            spans.append([char_start, char_end])
    return spans

sf.add_code_lf(node, lf_taxonomy, label="COMPANY")
```

You can also create a multi-polar code LF (see Multi-polar LFs for more information) using the `doc` field that contains entity-level prediction from Spacy, e.g., if an entity is tagged as `ORG`, label the corresponding span as `COMPANY`, and if tagged as `DATE`, label as `OTHER`:

```
@labeling_function(name="my_multipolar")
def sample_multi_polar_code_lf_sequence(x):
    ents = x.doc['ents']
    spans = []
    for ent in ents:
        char_start = ent["start"]
        char_end = ent["end"]
        label = ent["label"]
        if label == 'ORG':
            spans.append([char_start, char_end, "COMPANY"])
        elif label == 'DATE':
            spans.append([char_start, char_end, "OTHER"])
    return spans

lf = sf.add_code_lf(node, sample_multi_polar_code_lf_sequence,
is_multipolar=True)
```

Create a label package and label your dataset

See the **Create a label package and label your dataset** of the Document classification: Classifying contract types.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

Train a model

You can train a model on any training set. For sequence tagging applications, we offer the `DistillBERT` model for its compactness and relatively fast training time while preserving 95% of BERT's performance. If run on a CPU, this model may take several minutes or more to finish. We also offer **BERT Tiny** which is about 2x faster than **DistillBERT** while preserving 90% of BERT's performance. Additional details on models can be found in Model training.

Here are some details about parameters that are specific to `Distillbert` models:

- **fraction_of_negative_subsequences_to_use_for_training**: Determines how many negative span tokens to show while training. This parameter helps reduce model bias for the negative label class when there is class imbalance between positive and negative label classes.
- **stride_length** and **max_sequence_length**: These correspond to the transformer **tokenizer** parameters stride and max_length respectively.

Span, overlap span, and token level model metrics

In the sequence tagging setting, we offer three levels of metrics to evaluate the final model performance:

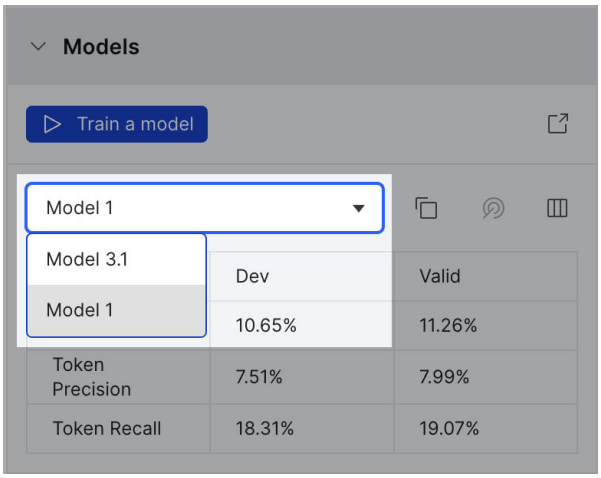
- **Word level**: Measures how many words are labelled correctly with respect to the ground truth.
- **Word Entity level**: Measures how many groups of words are labelled correctly as entities, e.g. "Advanced Braking Technology Ltd" is considered as one entity when calculating the metrics.
- **Overlap Span level**: Measures how many predicted and ground truth spans overlap more than 50% as measured by Jaccard similarity.

For example, given a ground truth span "Advanced Braking Technology Ltd" and a predicted span "Braking Technology Ltd," our recall scores for each measure are:

- `word_recall` = 3/4 = 0.75
- `word_entity_recall` = 0/1 = 0.0
- `overlap_span_recall` = 1/1 = 1.0

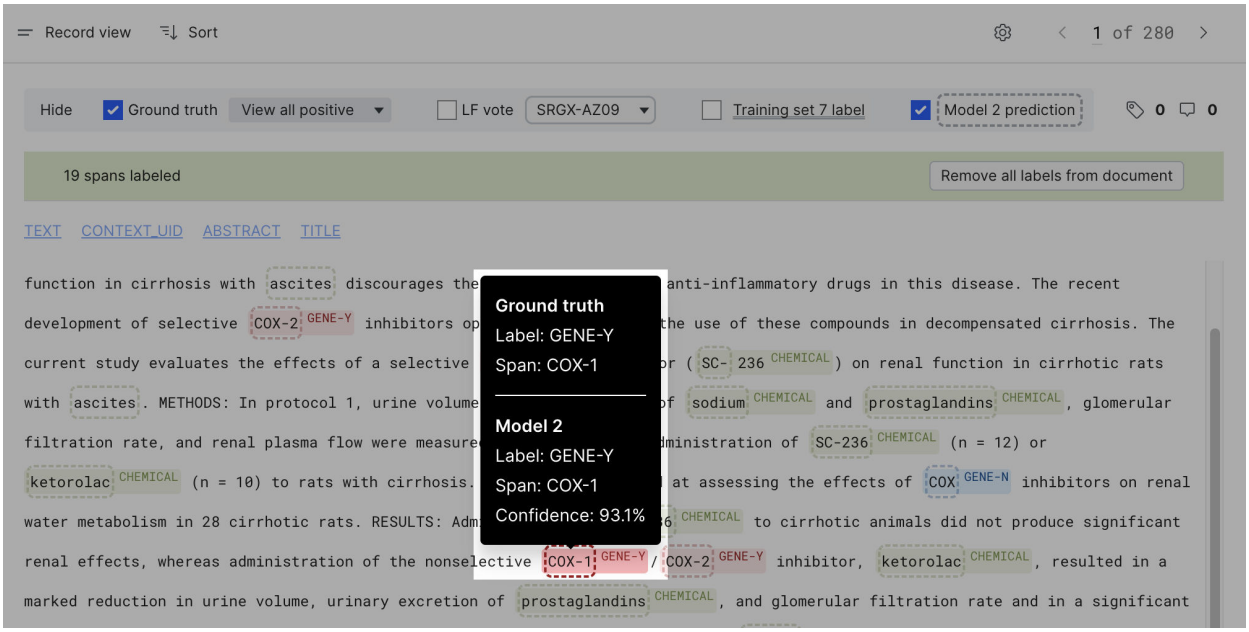
View model predictions

You can view the model predictions on the **Label** page. On the models pane, in the drop-down model selector menu, select the `Model ID` of interest, (e.g. "1").



This will change the span highlighting to be based on the model prediction. If you hover over individual spans, you can see the predicted class with the confidence associated with the prediction.

Copyright © 2025 Snorkel AI, Inc. All rights reserved



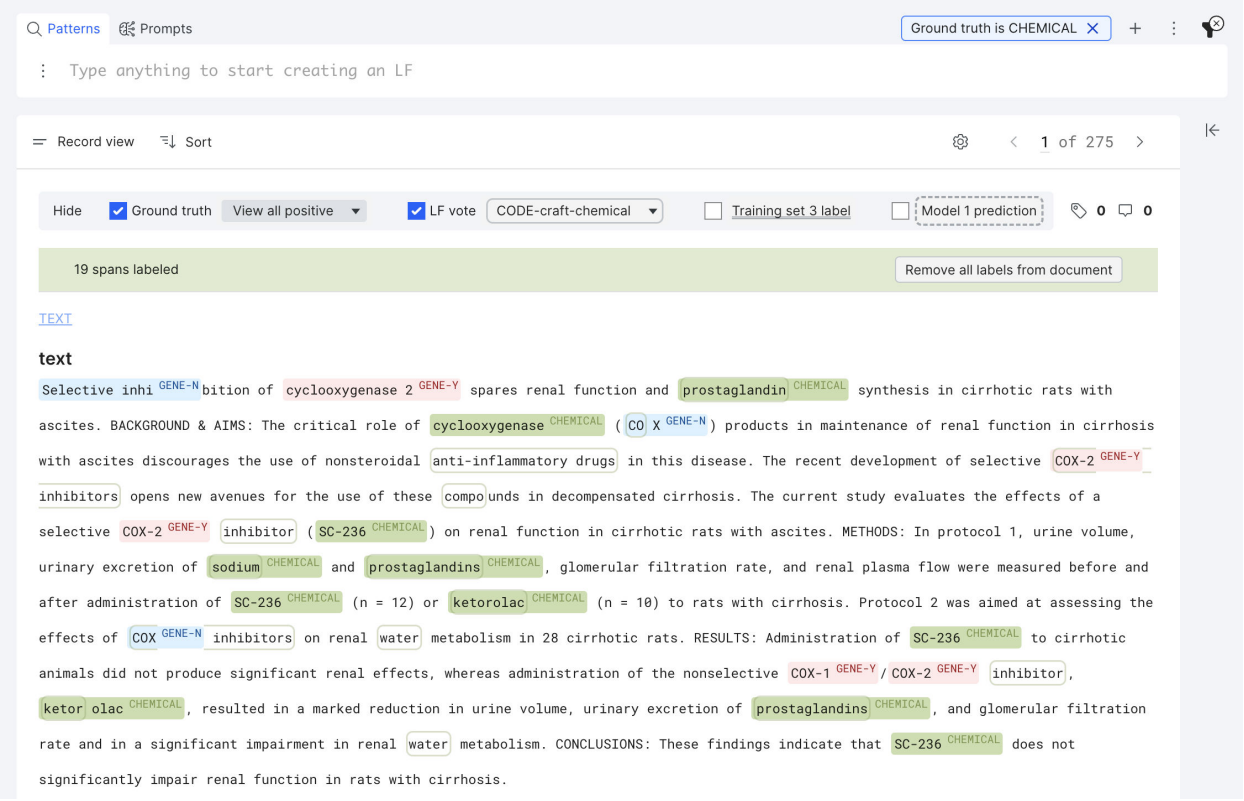
View errors and iterate

See **View errors and iterate** in Document classification: Classifying contract types. Note that in the sequence tagging application, all analysis metrics are computed at the span level, which includes the **Confusion Matrix**, **Clarity Matrix**, **Label Distribution**, **Class Level Metrics**, **Error Correlation,** and **PR curve**.

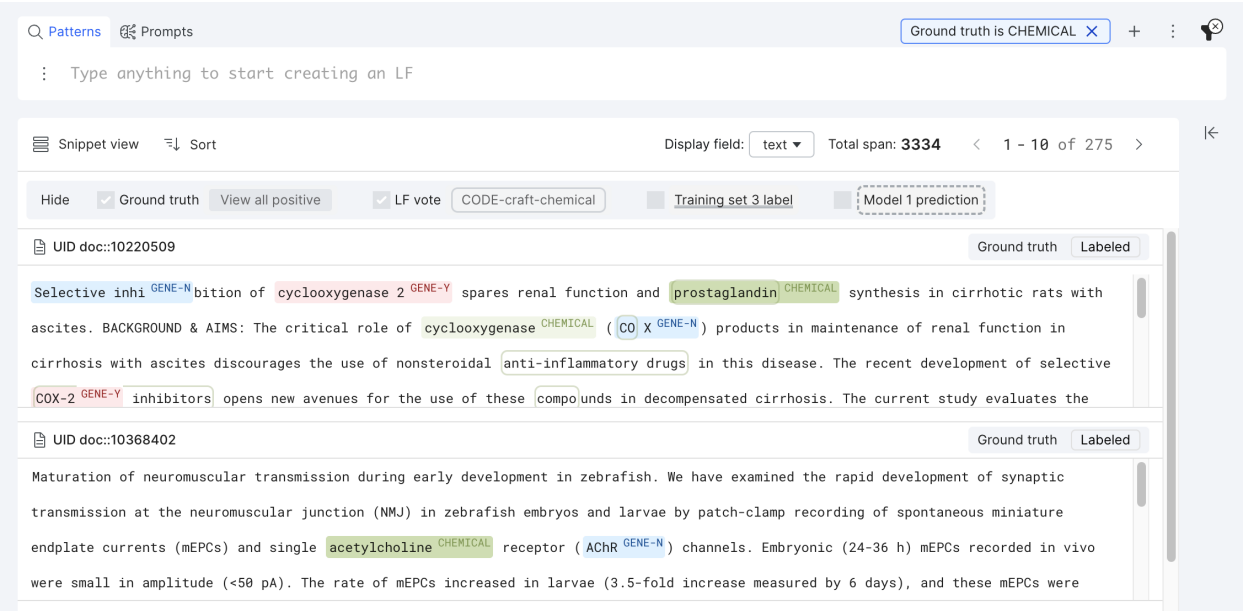
Filters and span highlighting in record view and snippet view

The filters are applied at the span level and automatically select the span type(s) to highlight among four types of spans (Ground truth, LF vote, Training set vote, and Model prediction). By default, ground truth is always highlighted. The spans matching the filter are presented in full color, whereas the spans that are not matching the filter are faded out in **Record view**.

Copyright © 2025 Snorkel AI, Inc. All rights reserved



In the **Snippet view**, the highlight control pane is not editable, but spans are still viewable by scrolling each snippet:



Correctness filter

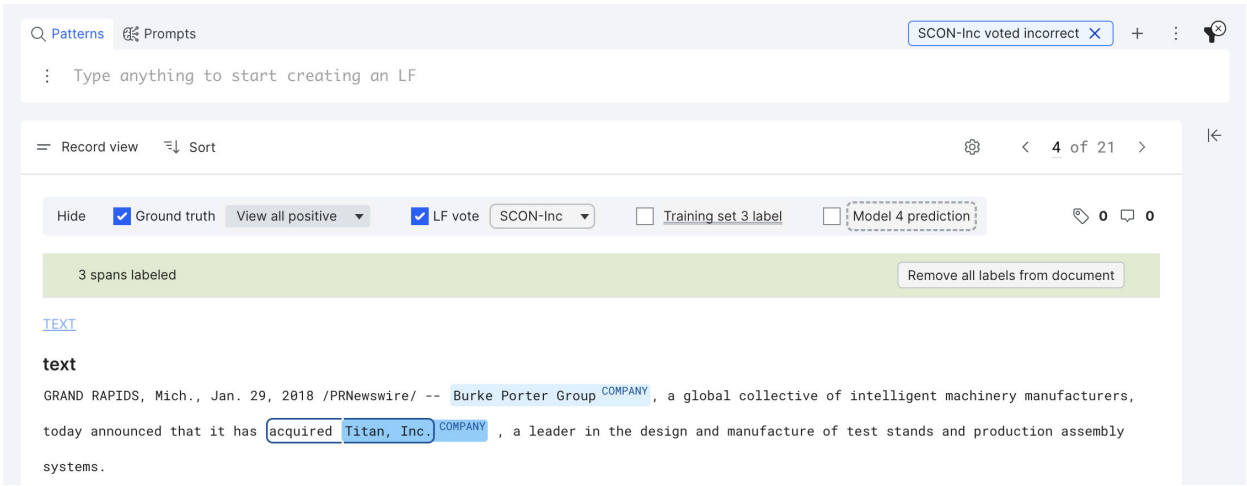
The correctness filter for LF and model/training set spans have different criteria. For model or training set spans to be correct, the exact match on both the boundary and label with ground truth is required. This is aligned with the **span-level metrics**. For LF spans, the correctness criteria is more relaxed: if an LF span only matches a part of a ground truth span, it is correct, as one can always write more LFs to cover the remaining part.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

LF correctness filter

An LF span is correct as long as it is a substring of a ground truth span with the same label. For example, in the screenshot below, both “Spring Bancorp, Inc.” and “Washington Bankshares, inc.” are correct.

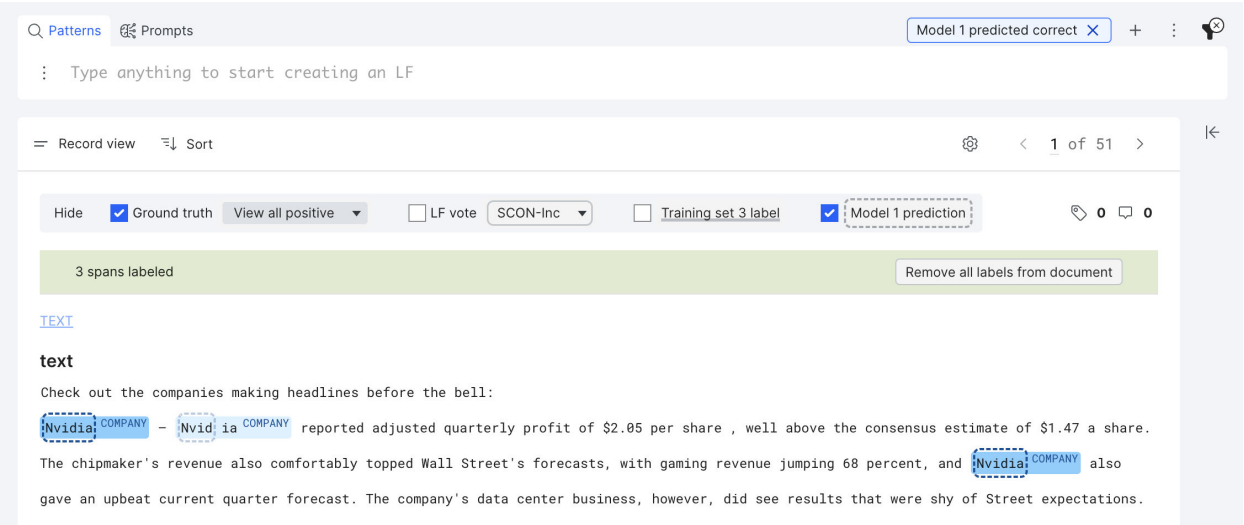
An LF span is incorrect as long as it votes on any character that is not in the selected class. For example, the LF voted incorrect on the “acquired ” (whose ground truth is `OTHER`)



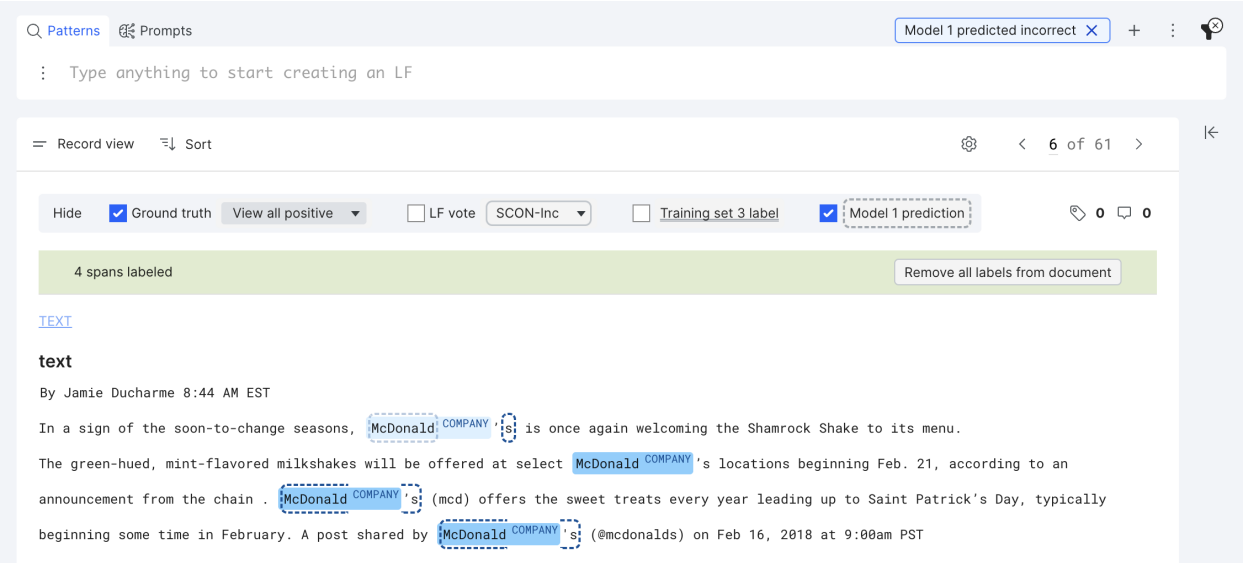
Model and training set correctness filter

For a model or training set span to be correct, it has to match the ground truth span on both the label and the boundary. For example, in the screenshot below, only the first “Nvidia” and last “Nvidia” are correct model predictions and thus in focus. The second “Nvidia” is incorrect due to a mismatch in the span boundary and thus in transparent state.

Copyright © 2025 Snorkel AI, Inc. All rights reserved



There are three types of incorrect model/training set spans: (1) False negative (e.g., the second "McDonald," the ground truth is `COMPANY` but model predicted `OTHER`) (2) False positive (e.g., in the second paragraph, the model predicted "s" as `COMPANY`, but the ground truth is `OTHER`) (3) a model prediction span has an incorrect boundary (e.g., the last two "McDonald's")



Dataset and application requirements

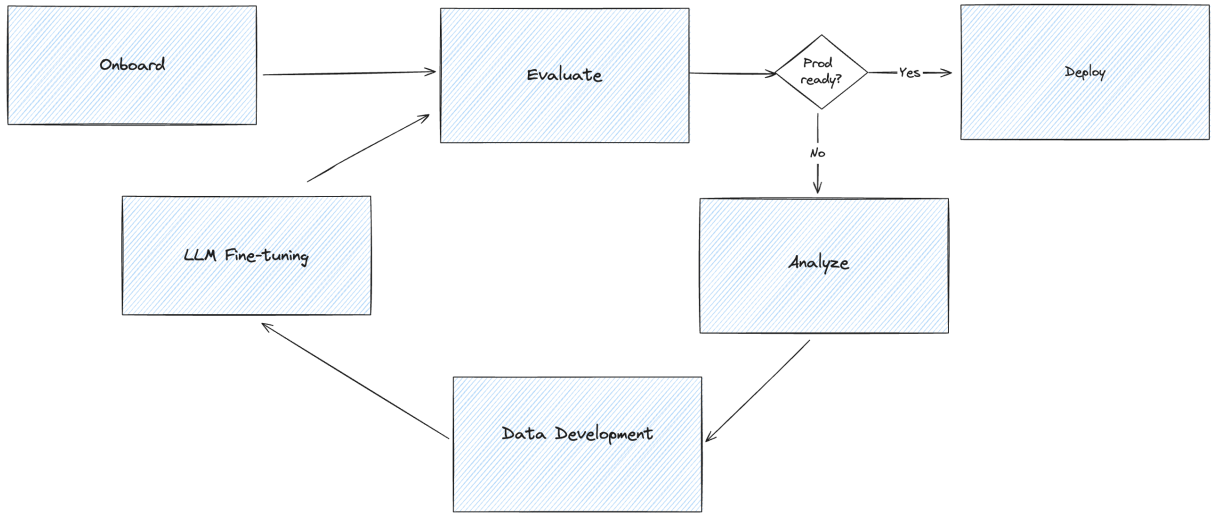
- Label classes: A maximum of 25 positive label classes can be specified
- Labeling functions: A maximum of 200 labeling functions can be defined
- Total dataset size: A maximum of 250 MB total text data (i.e., data in the text field) across splits can be specified
- Studio dataset size: A maximum of 15 MB total text data (i.e., data in the text field) can be loaded in Studio
- Each text field: The maximum allowable size for the text field in each data point is 10 KB

Please contact a Snorkel representative for recommendations if any of these limitations are a concern.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

LLM fine-tuning and alignment tutorial

In this tutorial, you can use Snorkel to evaluate the current performance of a generative AI system. Snorkel Flow allows developers to programmatically align a large language model (LLM) to domain-specific preference and objectives. After analysis, programmatically curate a high quality, diverse training dataset that's passed to an LLM for fine-tuning. Generated responses are brought back to Snorkel Flow for response quality labeling, error analysis, and iterative development.



This tutorial uses a dataset with ~600 financial service copilot instructions, retrieved context, and responses. A sample record is included below:

Copyright © 2025 Snorkel AI, Inc. All rights reserved

```
{
    "questions": "How does a balance transfer affect my credit score?",
    "responses": "A balance transfer can affect your credit score in several
ways, both positively and negatively:\n\n1. **Credit Utilization Ratio**: When
you transfer debt from one or more credit cards to a single card, you're
likely...",
    "prompt_prefix": "You are an expert in financial credit monitoring
services and support. Given the context information and no prior knowledge,
think step by step and answer the query. If you need to create a table, use
markdown language to create the table. Remain concise during the dialogue and
only answer information for which you have relevant factual context.",
    "rc_title_text": [
        {
            "TITLE": "Impact on Credit Utilization Ratio",
            "RELEVANT_CONTEXT": "When you execute a balance transfer, you move
existing debt from one credit card to another, typically to take advantage of
lower interest rates or promotional offers. This transfer can have an
immediate impact on your credit utilization ratio, which is the amount of
credit you're using compared to your total available credit. For instance, if
you transfer a $5,000 balance to a card with a $10,000 credit limit and leave
the original card's balance at zero, you improve your overall utilization
ratio, as long as you don't use the freed-up credit to incur more debt. Lower
utilization ratios are generally seen as positive by creditors and can lead to
an improvement in your credit score. However, if the balance transfer
significantly increases the balance on a single card, your score could
temporarily drop due to the higher utilization rate on that card, even though
your overall utilization might be lower."
        },
        {
            "TITLE": "Effect of Hard Inquiries",
            "RELEVANT_CONTEXT": "Applying for a new credit card to facilitate
a balance transfer can result in a hard inquiry on your credit report. Each
hard inquiry can potentially lower your credit score..."
        }
    ]
}
```


Create the LLMFineTuning application

1. Import the dataset:

Copyright © 2025 Snorkel AI, Inc. All rights reserved

```
# imports

import snorkelflow.client as sf
from snorkel.labeling.lf import labeling_function
import snorkelflow.sdk as sfs
from snorkelflow.sdk.fine_tuning_app import FineTuningApp, QualityDataset
from snorkelflow.types.finetuning import (
    AnnotationStrategy,
    FineTuningAppConfig,
    FineTuningColumnType,
    LLMFineTuningLabelingConfig,
)
from snorkelflow.types.source import ModelSourceMetadata
from snorkelflow.sdk import Dataset
from snorkelflow.sdk.slices import Slice
import pandas as pd
from sklearn.model_selection import train_test_split
ctx = sf.SnorkelFlowContext.from_kwargs(
    api_key=None, # change this to your API key if you are superadmin and
want to use its privileges
    workspace_name="default", # change this to work with a non-default
workspace
)
```

2. Create the application configuration.

> NOTE
>
> Only the **instruction** and **response** `FineTuningApp` class in the SDK documentation.

```
#Create an fine-tuning app config and map data fields
app_config = FineTuningAppConfig(
    column_mappings= {
        "questions": "instruction",
        "responses": "response",
        "rc_title_text": "context",
        "prompt_prefix": "prompt_prefix"
    }
)
app_name = 'mm-copilot-test'
FineTuningApp.create(app_name, app_config)
ft_app = FineTuningApp.get(app_name)
```

```
Train split: s3://snorkel-public/fine_tuning_tutorial_train.csv
```

```
Valid split: s3://snorkel-public/fine_tuning_tutorial_valid.csv
```

Copyright © 2025 Snorkel AI, Inc. All rights reserved

```
#onboard data from s3 and couple it to a model source object

import boto3
import pandas as pd
from io import StringIO

# Initialize a session using boto3
s3_client = boto3.client('s3')

# Specify your bucket name and file key
bucket_name = 'snorkel-public'
files = ['fine_tuning_tutorial_train.csv',
'fine_tuning_tutorial_valid.csv']
dfs = []
for f in files:
    # Fetch the file from S3
    response = s3_client.get_object(Bucket=bucket_name, Key=file_key)
    status = response.get("ResponseMetadata", {}).get("HTTPStatusCode")

    if status == 200:
        csv_content = response["Body"].read().decode('utf-8')
        data = StringIO(csv_content)
        df = pd.read_csv(data)
        dfs.append(df)
        print("CSV loaded successfully.")
    else:
        print(f"Failed to fetch the file from S3. Status code: {status}")

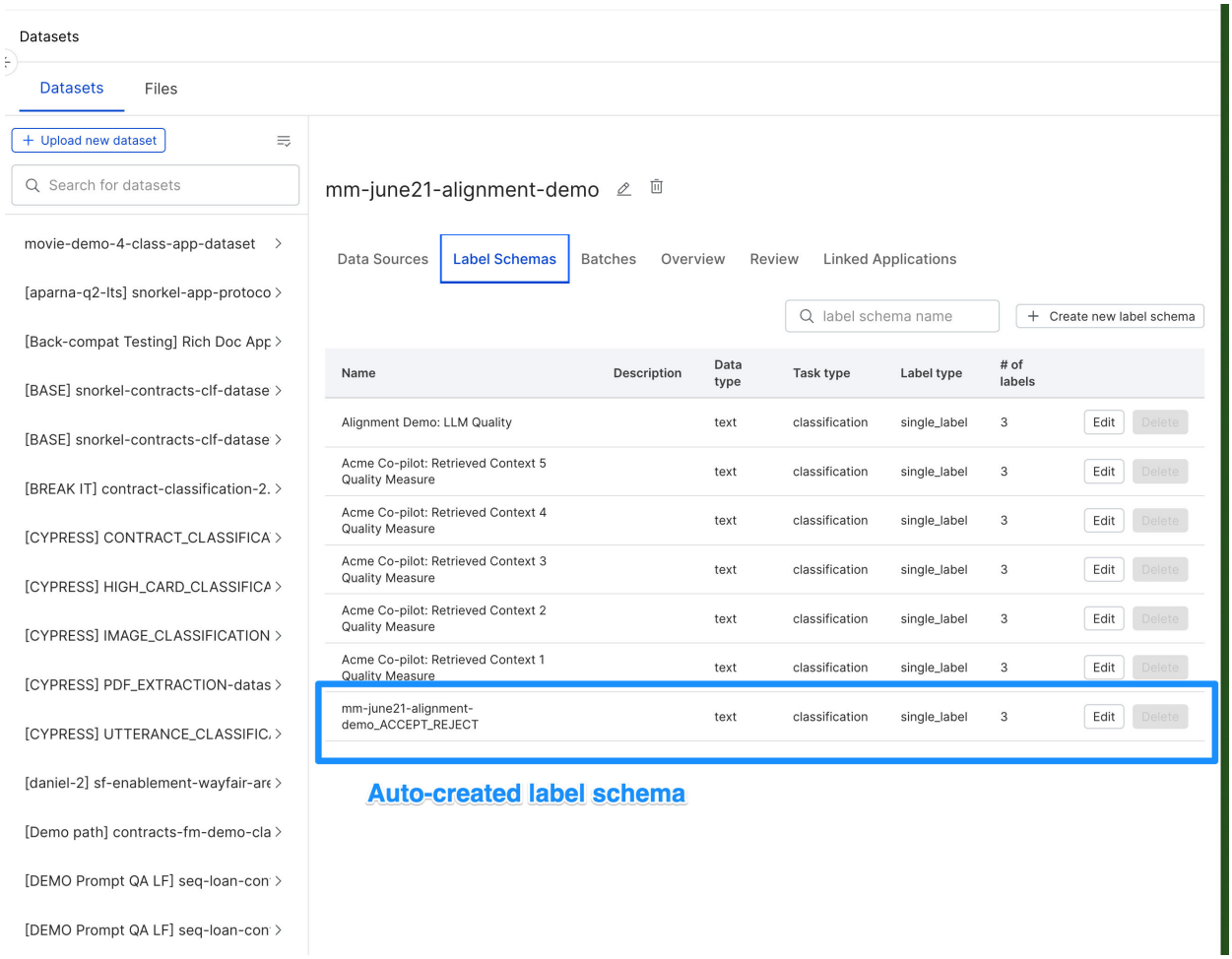
train_df = dfs[0]
valid_df = dfs[1]

model_source_uid = ft_app.register_model_source("model_iteration_name",
    metadata=ModelSourceMetadata(model_name = "model_iteration_name"))
['source_uid']

ft_app.import_data(train_df, "train", model_source_uid, sync = True )
ft_app.import_data(valid_df, "valid", model_source_uid, sync = True )
```

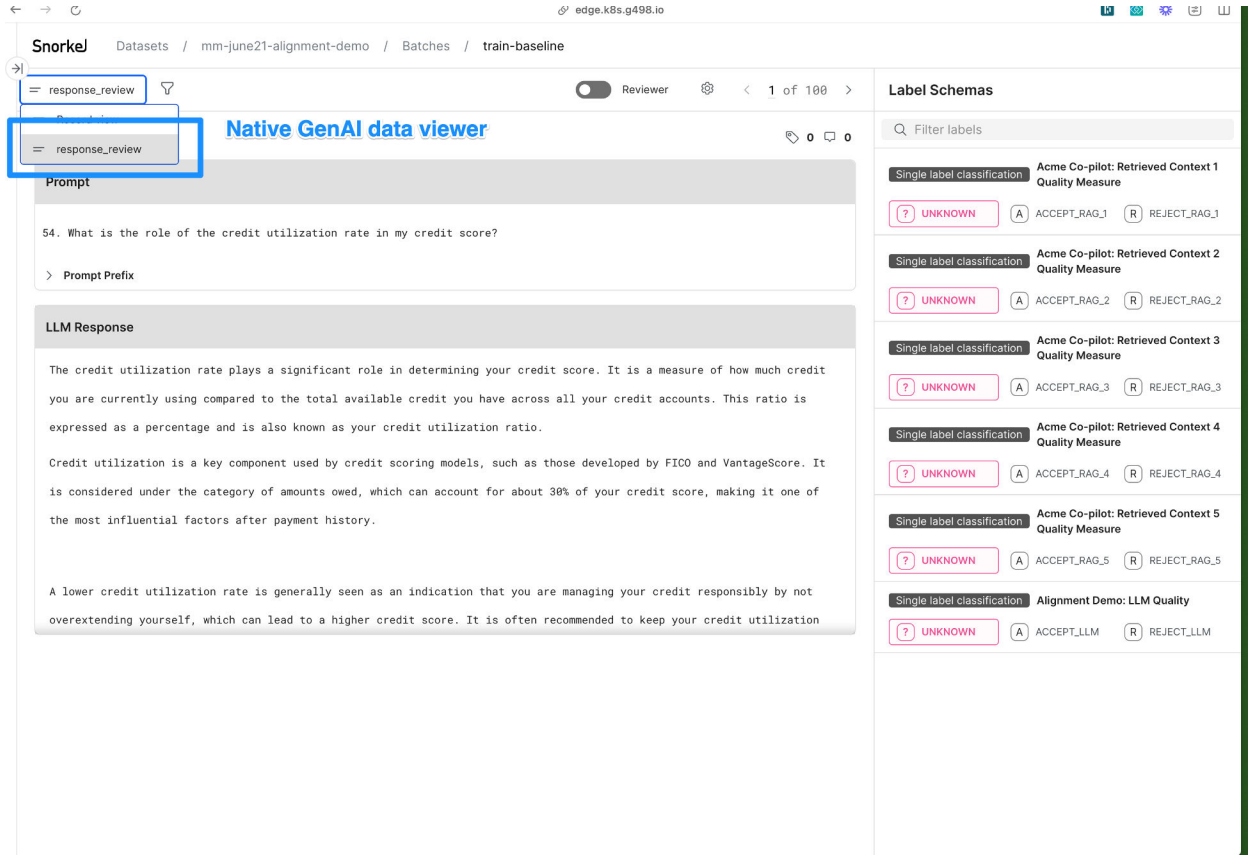
This workflow introduces a **model source** object, which maps data sources in the parent dataset to different iterations of fine-tuning. For example, data coming from the **llama3-8b-instruct** base model has a different model source id than data coming from a **fine-tuned llama3-8b** model. This object distinguishes metrics across evaluations and determine the data a user is developing on.

Copyright © 2025 Snorkel AI, Inc. All rights reserved



A FineTuning dataset and application have been created. An LLM quality label schema and single response dataset view are automatically generated.

Copyright © 2025 Snorkel AI, Inc. All rights reserved



For more information on label schemas and dataset views, see SDK documentation for **label_schemas** and **dataset_views**.

Evaluate

Now that data is onboarded, assess the performance of the current system. Snorkel Flow recommends two methods of evaluation:

- Manual evaluation via domain experts
- Programmatic evaluation via a quality model

Manual evaluation is a great starting point. However, with each fine-tuning iteration, the ground truth must be discarded because new responses are generated. Build a quality model in Snorkel Flow to solve for this issue.

Manual evaluation via domain experts

Assign domain experts with batches of data to evaluate. Snorkel recommends labeling the quality of the generated response and the quality of the retrieved context.

1. Create additional label schemas for retrieved context objects with this helper function:

Copyright © 2025 Snorkel AI, Inc. All rights reserved

```
def create_finetuning_label_schemas(dataset_name: str,
    include_llm: bool, include_rag: bool, num_contexts: int):
    #create the llm quality label schema
    d = sfs.Dataset.get(dataset_name)
    base = "Acme Co-pilot: "
    if include_llm:
        d.create_label_schema(
            name = base + "LLM Quality Measure",
            data_type = "text",
            task_type = "classification",
            multi_label = False,
            label_map = ["ACCEPT_LLM", "REJECT_LLM"]
        )
    if include_rag:
        for i in range(1,num_contexts+1):
            ii = str(i)
            d.create_label_schema(
                name = base + "Retrieved Context " + ii + " Quality
Measure",
                data_type = "text",
                task_type = "classification",
                multi_label = False,
                label_map = [("ACCEPT_RAG_"+ii), ("REJECT_RAG_" + ii)]
            )

# Don't create the llm quality label schema because the FineTuningApp
# already creates it.
create_finetuning_label_schemas(dataset_name = app_name,
    include_llm = False, include_rag = True, num_contexts = 5)
```

2. Assign batches of data to domain experts to review.
   Snorkel recommends creating an annotation guide, detailing characteristics of "good" and "bad" responses, to ensure the collection of highly-accurate ground truth.

3. Annotators leverage Generative AI data viewers to interact with data in a more intuitive format. See the example below for creating dataset views:

```
duid  = ft_app.dataset_uid
sf.create_dataset_view(
    dataset= duid,
    name="GenAI Data Viewer",
    view_type="single_llm_response_view",
    column_mapping={"instruction": "questions", "response": "responses",
    "context": "rc_title_text"},
    label_schemas = [list of label_schema uids that you want to enable the
view for]
        )
```

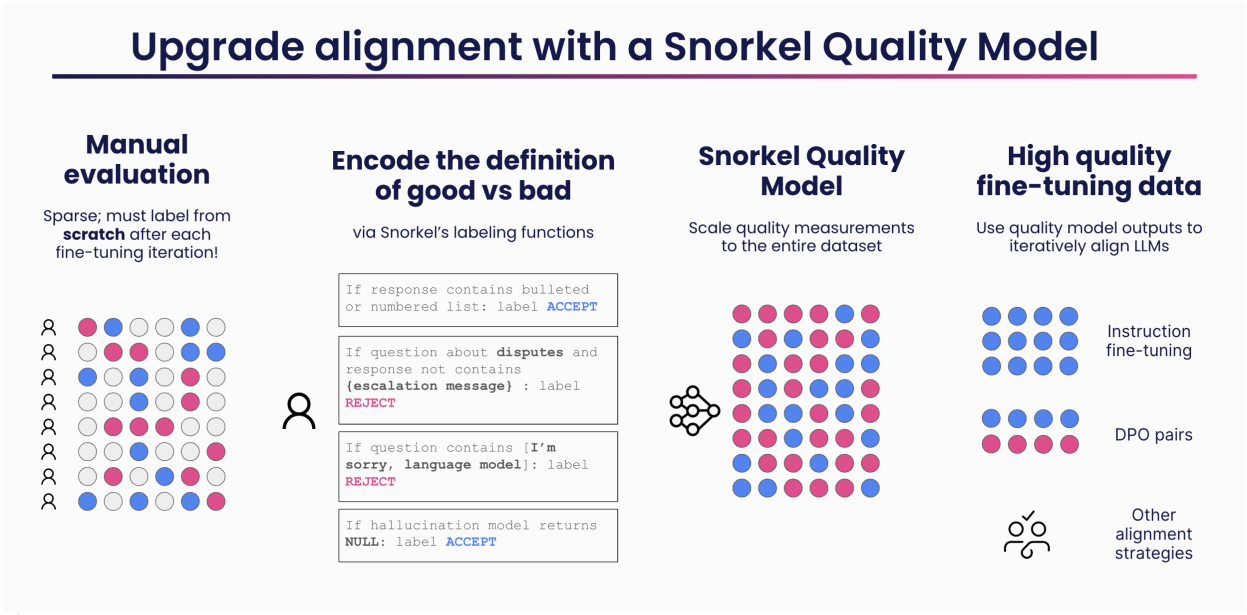
Copyright © 2025 Snorkel AI, Inc. All rights reserved

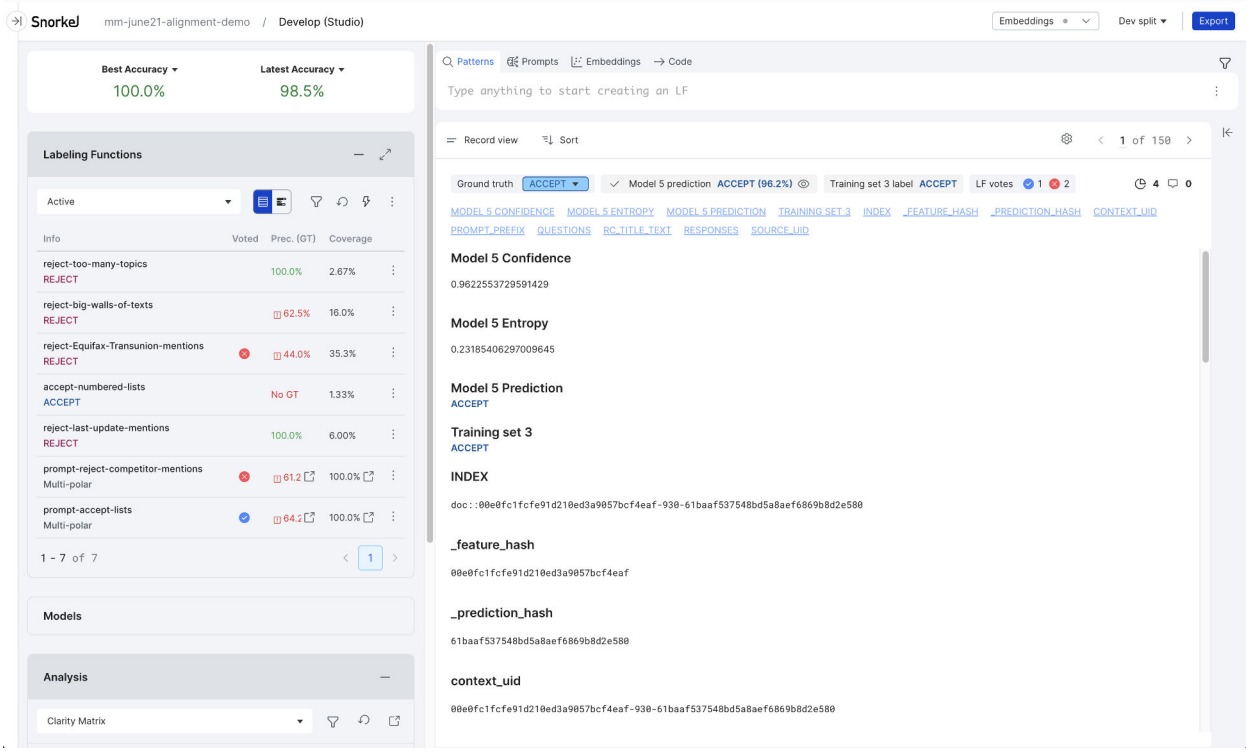
4. After batches have been completed, commit annotations as dataset ground truth for the relevant label schemas.

Programmatic evaluation via a quality model

Encode the definition of **good** and **bad** via labeling functions (LFs). Use these LFs to train a quality model, which serves these functions:

- Identify good and bad instruction for response data at scale. These instructions filter data to create a curated dataset that is passed to an LLM for fine-tuning.
- Automatically evaluate new responses from a fine-tuned LLM.





Copyright © 2025 Snorkel AI, Inc. All rights reserved

Analyze

With the manual and programmatic quality measurements have been collected, it's time to collate them into an evaluation to analyze and address error modes. Analysis takes these forms:

- Global
- Slice-wise

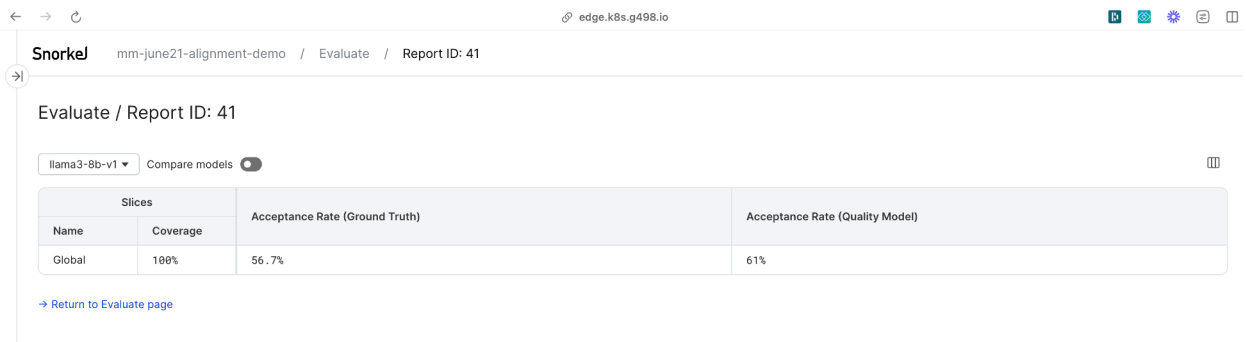
Snorkel Flow's **Evaluations Module** synthesizes manual and programmatic measurements of quality across your dataset. This approach gives developers a single, comprehensive view into their model's performance.

The global evaluation report is a great start, but lacks the fidelity needed to robustly understand performance gaps and understand where the developer needs to focus to improve performance. To solve for these limits, slicing functions are unit tests for your data. Slicing functions allow you to categorize the types of interactions you care about, measuring quality over each data slice.

Global analysis

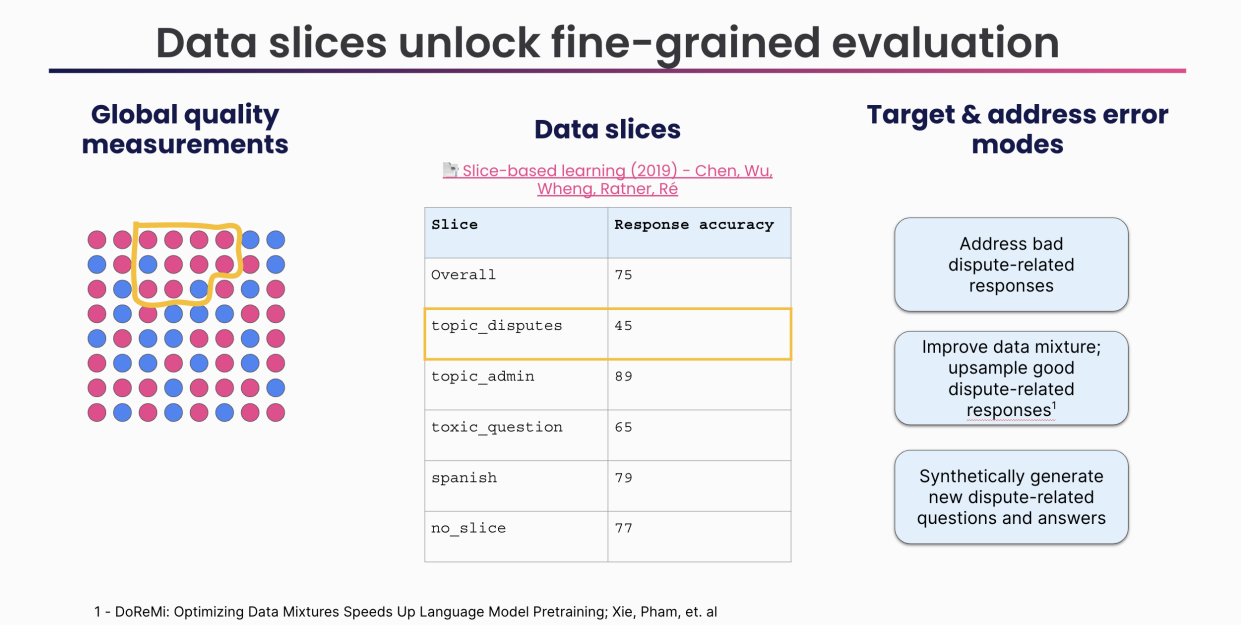
Combine the quality measurements from domain experts and a trained quality model to assess performance:

```
# Create a global evaluation report
ft_app.create_evaluation_report(
    split="train",
    quality_models = [{Insert quality model name}],
    slices = None
)
```



Copyright © 2025 Snorkel AI, Inc. All rights reserved

Slice-wise analysis



Write slicing functions

Slicing functions leverage the same operators as labeling functions. Use keywords, regular expressions, model, and LLM prompts to identify data slices of interest:

```
node = ModelNode.get(node_uid=ft_app.model_node_uid)

def apply_slice(slicing_fn):
    df = node.get_dataframe()
    slice_mask = df.apply(slicing_fn, axis=1)
    df_sliced = df[slice_mask]
    x_uids= list(df_sliced.index)
    slice_name = slicing_fn.__name__
    slice_percent = len(df_sliced) / len(df)
    print(f"Applied '{slice_name}' slice (n={len(df_sliced)},
{slice_percent*100:.2f}%)")
    return df_sliced, x_uids
```

```
import re
def topic_admin(x):
    # Data slice to capture when users are asking
    #about things like resetting passwords or usernames.
    pattern = r"\b(password|email|username)\b"
    match = re.search(pattern, x.responses, re.IGNORECASE)
    return bool(match)

admin_df, admin_x_uids = apply_slice(topic_admin)
admin_slice = Slice.create(name="topic_admin", dataset=ft_app.dataset_uid)
admin_slice.add_x_uids(admin_x_uids)
```

Copyright © 2025 Snorkel AI, Inc. All rights reserved

Create a slice-wise evaluation

Combine quality measurements with data slices to get a fine-grained view into the system's current performance:

```
# Create a global evaluation report
ft_app.create_evaluation_report(
    split="train",
    quality_models = [{Insert quality model name}],
    slices = [slice_1.slice_uid, slice_2.slice_uid, slice_3.slice_uid, ...]
)
```

Built-in evaluation metrics

Each evaluation report comes with two types of built-in metrics:

- Acceptance Rate (Ground Truth) is calculated based on the ground truth collected manually from domain experts.
- Acceptance Rate (Quality Model) is calculated based on the labels generated by the Quality Models. If you pass multiple Quality Models into the evaluation report, Snorkel Flow creates multiple metrics. For each Quality Model, the metric is derived from the latest committed version. If there are no committed versions, the metric defers to the latest trained version.

Datapoint-level versions for each metric are also available in the evaluation report.

Custom Evaluation Metrics

You can register your own metrics defined through custom Python functions.

1. Define a custom Python function:

```
# Preview a sample of the dataframe
sample_df = ft_app.get_dataframe(split="train")
sample_df.head()

# Write the custom Python function
# Note that you can use all columns in the sample dataframe except the GT
column
# Example Python function:
def short_response_ratio(df: pd.DataFrame) -> float:
    if len(df) == 0:
        raise ValueError(f"No samples found")
    short_response_count = (df["responses"].str.len() <= 2000).sum()
    return short_response_count / len(df)

# Test the metric function with the sample dataframe
sample_score = short_response_ratio(sample_df)
```

2. Register the custom metric with the FineTuningApp:

Copyright © 2025 Snorkel AI, Inc. All rights reserved

```
# Register the custom metric with the FineTuningApp
ft_app.register_custom_metric(
    metric_name = "Short Response Ratio",
    metric_func = short_response_ratio,
)

# Create evaluation report
# All custom metrics registered with the FineTuningApp are automatically
added to the report
ft_app.create_evaluation_report(split="train")
```

3. If needed, update an existing custom metric registered with the FineTuningApp:

```
ft_app.register_custom_metric(
    metric_name = "Short Response Ratio",
    metric_func = short_response_ratio,
    overwrite = True,
)
```

Create a curated fine-tuning dataset

If evaluations show system performance at or above production benchmarks, then developers can deploy the system into production. If not, you must further develop the LLM to reduce errors. Developers can leverage the quality model and slice membership to curate a high-quality, diverse training set that is passed to an LLM for fine-tuning.

1. Create a quality dataset, which is a subset of data from your global set.

```
# Create a curated dataset from a trained quality model
qd = ft_app.get_quality_dataset(model_uid = 1) # update to your end model
uid
qd_filtered = qd.filter(confidence_threshold=0.9, labels=["ACCEPT"])
good_x_uids = list(qd_filtered.get_data().index)
good_datasource_uids = list(qd_filtered.get_data()
['datasource_uid'].unique())
good_datasource_uids = [int(i) for i in good_datasource_uids]
```

2. Pass this quality dataset to an LLM service for model fine-tuning.

For more advanced filtering operations, see the `QualityDataset` class in Snorkel's SDK documentation.

Fine-tune and update the LLM

Use the curated quality dataset to fine-tune an LLM. Snorkel supports these methods of fine-tuning:

- Fine-tune using Snorkel's SageMaker Jumpstart integration
- Fine-tune via another third-party LLM provider

Copyright © 2025 Snorkel AI, Inc. All rights reserved

Fine-tune using Snorkel's SageMaker Jumpstart integration

1. Configure access to the SageMaker Jumpstart service:

```
#Code to create the sagemaker connection
AWS_ACCESS_KEY_ID = "aws::finetuning::access_key_id"
AWS_SECRET_ACCESS_KEY = "aws::finetuning::secret_access_key"
SAGEMAKER_EXECUTION_ROLE = "aws::finetuning::sagemaker_execution_role"
FINETUNING_AWS_REGION = "aws::finetuning::region"

sf.set_secret(AWS_ACCESS_KEY_ID, "{your_access_key}",
secret_store='local_store',
    workspace_uid=1, kwargs=None)
sf.set_secret(AWS_SECRET_ACCESS_KEY, "{your_secret_key}",
secret_store='local_store',
    workspace_uid=1, kwargs=None)
sf.set_secret(SAGEMAKER_EXECUTION_ROLE,
"arn:aws:iam::746568209548:role/service-role/AmazonSageMaker-ExecutionRole-
20220908T104212",
    secret_store='local_store', workspace_uid=1, kwargs=None)
sf.set_secret(FINETUNING_AWS_REGION, "us-west-2",
secret_store='local_store',
    workspace_uid=1, kwargs=None)
boto_session = boto3.Session(
    aws_access_key_id= "{your_access_key}",
    aws_secret_access_key="{your_secret_key}",
    region_name="us-west-2"
)
sagemaker_client = boto_session.client('sagemaker')
sagemaker_runtime_client = boto_session.client('sagemaker-runtime')
sagemaker_session = Session(
    boto_session=boto_session,
    sagemaker_client=sagemaker_client,
    sagemaker_runtime_client=sagemaker_runtime_client
)
```

2. Set up the training job configuration:

Copyright © 2025 Snorkel AI, Inc. All rights reserved

```
finetuning_configs = {
    "epoch": "1",
    "instruction_tuned": "True",
    "validation_split_ratio": "0.1",
    "max_input_length": "1024",
    "chat_dataset": "False"
}
training_configs = {
    "instance_type": "ml.p3dn.24xlarge or insert instance type"
}
column_mapping = {
    FineTuningColumnType.INSTRUCTION: "questions",
    FineTuningColumnType.RESPONSE: "responses",
    FineTuningColumnType.CONTEXT: "rc_title_text",
    FineTuningColumnType.PROMPT_PREFIX: "prompt_prefix"
}
external_model_trainer = ExternalModelTrainer(
    column_mappings=column_mapping,
    finetuning_provider_type=FinetuningProvider.AWS_SAGEMAKER
)
```

To request an LLM provider or base model, reach out to your Snorkel support team. Snorkel currently supports Low Rank Adaptation(LoRA)-based fine-tuning over these base models:

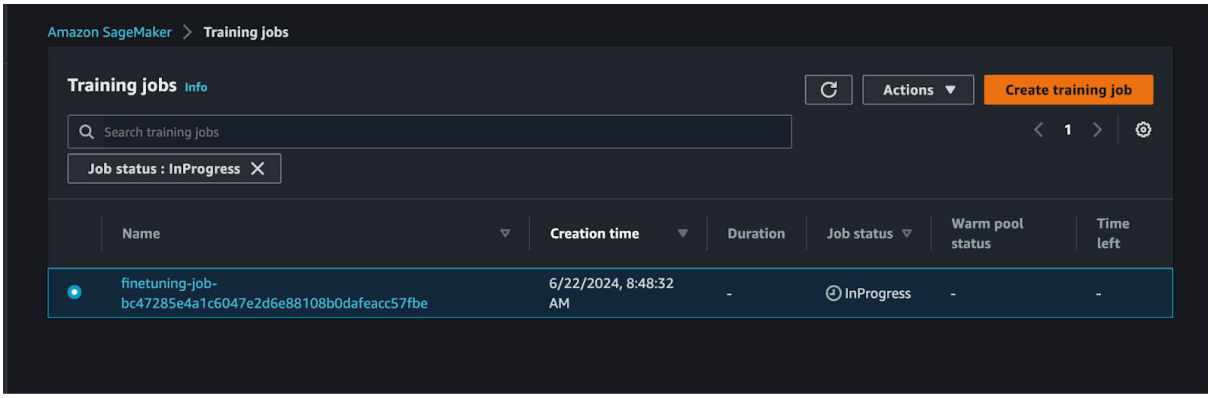
- Mistral-7B
- Llama3-8b-instruct

3. Upload training data to Sagemaker and begin the fine-tuning job:

```
external_model = external_model_trainer.finetune(
    base_model_id="meta-textgeneration-llama-3-8b-instruct",
    base_model_version="2.*",
    finetuning_configs=finetuning_configs,
    training_configs=training_configs,
    datasource_uids= good_datasource_uids,
    # to filter on x_uids
    x_uids=good_x_uids,
    # Set sync=False to return a job id and release the notebook kernel
    sync=True
)
```

4. To verify fine-tuning progress, log in to the SageMaker tenant, navigate to **Training Jobs**, and view **InProgress** jobs.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

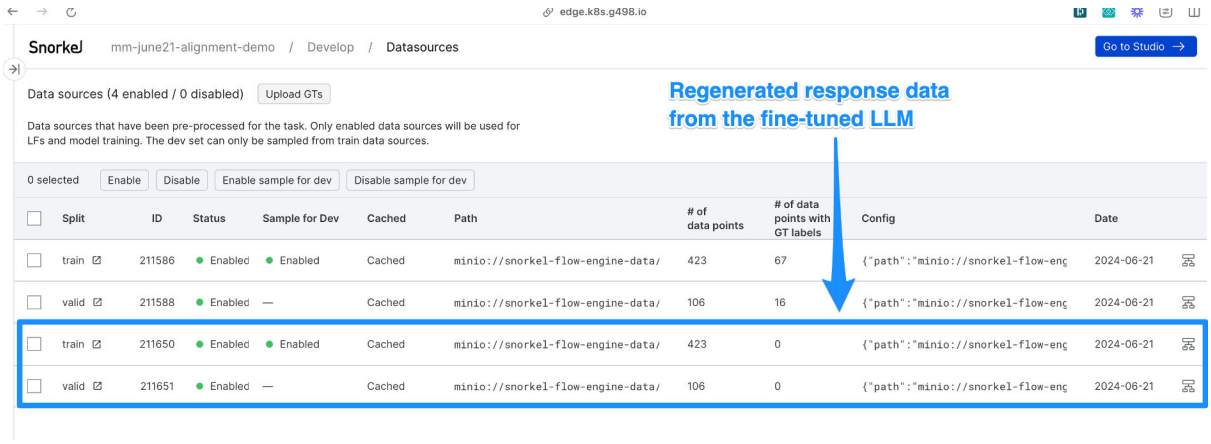


5. After fine-tuning is complete, run the dataset's instructions and optional `relevant_context` against the fine-tuned model to generate new responses:

```
DATASOURCES_FOR_INFERENCE = datasource_inference_uids
source_uid = external_model.inference(
    datasource_uids=DATASOURCES_FOR_INFERENCE,
    # to filter on x_uids, uncomment here
    # x_uids=x_uids,
    deployment_config = {"instance_type": "ml.g5.2xlarge"},
    # Set sync=False to return a job id and release the notebook kernel
    sync=True,
)
```

6. Upload the regenerated responses as a new datasource in Snorkel Flow.

7. After the responses are regenerated, view application datasources to see the newly created and enabled datasources:



Fine-tune via third-party LLM provider

If the fine-tuning and inference service is not natively integrated into Snorkel Flow, you can enable fine-tuning with a third-party LLM Provider.

1. Create a fine-tuning dataset, and export the dataset from Snorkel:

Copyright © 2025 Snorkel AI, Inc. All rights reserved

```
# Create a curated dataset from trained quality model predictions and model
confidences

qd = ft_app.get_quality_dataset(model_uid=1) # update to your quality model
uid
qd_filtered = qd.filter(confidence_threshold=0.9, labels=["ACCEPT"])
qd_filtered.to_csv('fine_tuning_set.csv')
```

2. Export the entire dataset from the application to regenerate:

```
df = ft_app.get_data()
df.to_csv("instruction_data.csv")
```

3. Fine-tune an LLM outside of the Snorkel platform.

4. Regenerate responses using the fine-tuned LLM outside of the Snorkel platform.

5. Import the fine-tuned dataset with new generations:

```
# Upload new datasources to Snorkel, consisting of the original instruction
set and newly generated responses.
train_df = pd.read_csv('./train_new_responses.csv')
valid_df = pd.read_csv('./valid_new_responses.csv')

# configure the LLM provider, model iteration and upload new data
model_source_uid = ft_app.register_model_source("llama3-8b-finetunev1",
metadata=ModelSourceMetadata(model_name = "llama3-8b-finetunev1"))
['source_uid']
ft_app.import_data(train_df, "train", model_source_uid, sync = True )
ft_app.import_data(valid_df, "valid", model_source_uid, sync = True )
```

After completing these steps, users will be able to evaluate and develop data from the newly fine-tuned LLM.

Re-evaluate

After fine-tuning is complete, evaluate the fine-tuned model. A manual approach requires engaging domain experts for another round of labeling. With Snorkel's quality model, you can get immediate feedback on the performance globally and across data slices.

> NOTE
>
> Quality model predictions are not automatically generated for new data.

1. Commit the desired quality model to the model node. Note the model UID.

2. Acquire the new data in the notebook and run it through the current application graph.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

```
model_source_uid = {new model source requiring predictions}
app_name = {application name}
qm_model_uid = {uid of quality model}
node_uid = {uid of model node}
ft_app = FineTuningApp.get(app_name)

new_data = parent = ft_app.get_dataframe(source_uids = [1095])
results = sf.execute_graph_on_data(application=app_name,target_uids=
[model_node], df=new_data)[model_node]
```

3. Once quality model predictions for new data have been acquired, map the dataset `x_uids` to model predictions and probabilities.

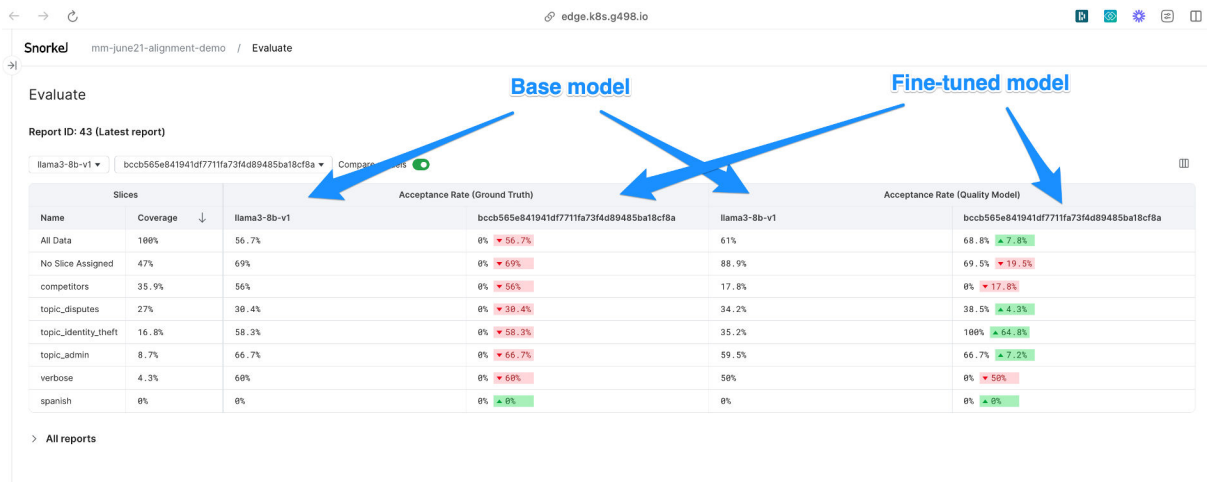
4. Upload those to the quality model as new predictions.

```
results['mapped_x_uids'] = results['cont_uid'].map(lambda x:
new_data[new_data['cont_uid'] == x].index[0])
sf.add_predictions(
    node = model_node,
    model_uid = qm_model_uid,
    x_uids = results['mapped_x_uids'].tolist(),
    predicted_labels = results['preds'].tolist(),
    predicted_probs = results['probs'].tolist()
)
```

5. Create a new evaluation report and make notice of QM predictions for the new model iteration data.

6. View the performance of the fine-tuned model against the base model.

```
ft_app.create_evaluation_report(
    quality_models = ['quality model name']
)
```



Copyright © 2025 Snorkel AI, Inc. All rights reserved

Conclusion

Snorkel's fine-tuning workflow enables developers to programmatically align an LLM to domain-specific preference and objectives. The annotation, evaluation, and development modules enable faster LLM fine-tuning via the programmatic approach.

For any questions, contact your account team or support@snorkel.ai.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

Evaluate GenAI output

Snorkel Flow enables enterprise AI teams to assess AI-generated output with evaluations that are:

- Specialized
- Fine-grained
- Actionable

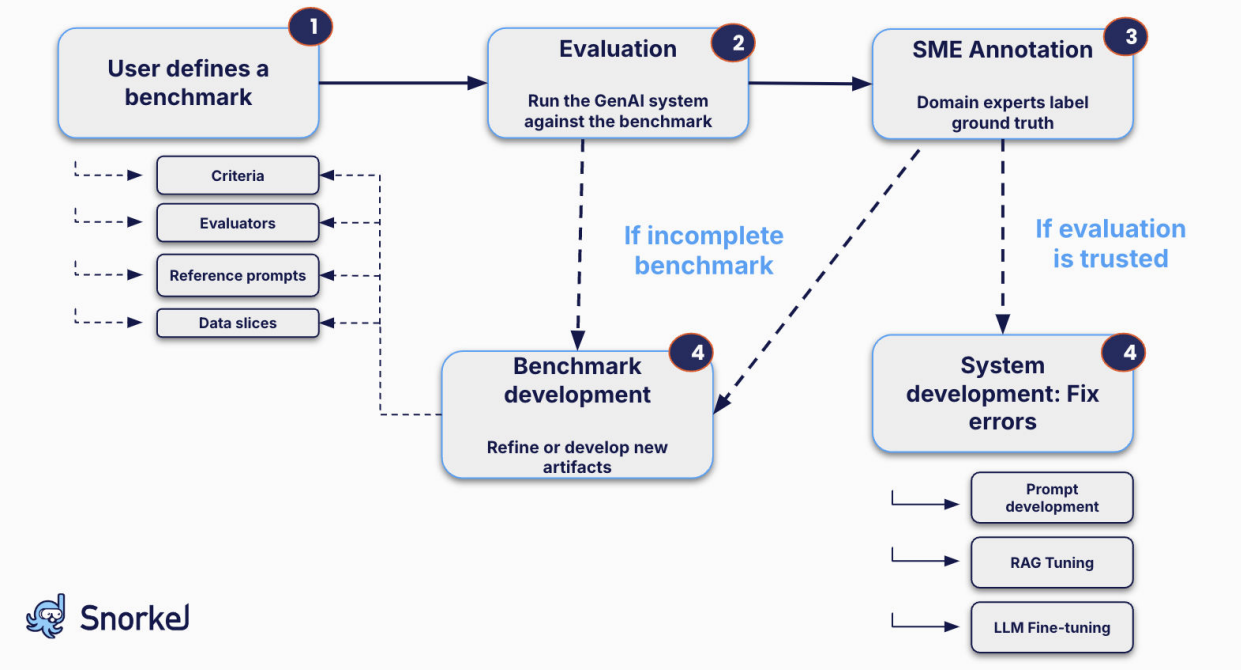
This tutorial walks you through an end-to-end workflow for how to set up an evaluation pipeline for GenAI-generated output, using Snorkel's Evaluation feature set.

The majority of the steps use Snorkel's SDK, which are combined in a single Python notebook for ease of use.

This tutorial describes the example scenario, the prerequisite data setup, and the output you'll achieve. The notebook itself contains the documentation and the code to complete the functional example use case.

Process overview diagram

Here's a diagram that shows the phases of the evaluation workflow. Each phase is discussed in detail in the following sections.



Prerequisites

- A Snorkel Flow instance
- A Superadmin API key for Snorkel Flow
- Amazon SageMaker with authentication secrets (used for fine tuning; you can skip that step in the notebook)

Copyright © 2025 Snorkel AI, Inc. All rights reserved

Download the notebook and resources

Download these files. Upload all of them to Snorkel Flow, in the **Notebook** section, in the same directory.

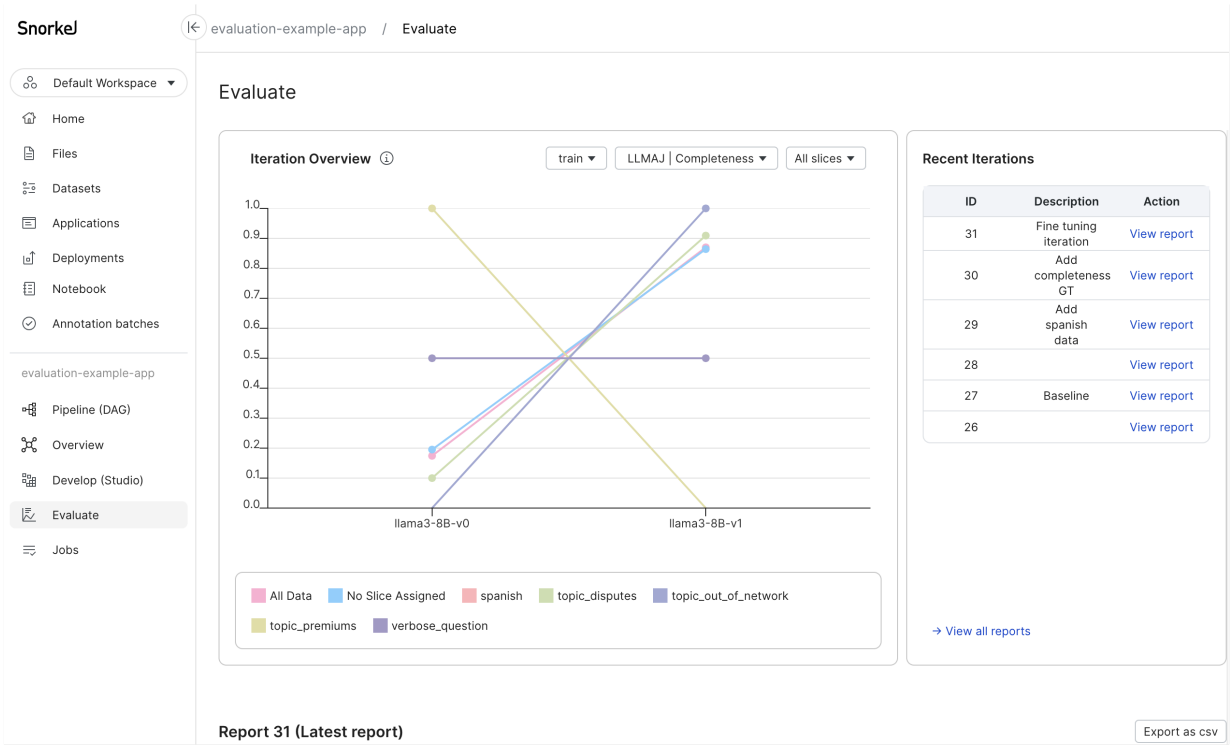
- eval-genai-notebook.ipynb
- eval-spanish.csv
- eval-train-1.csv
- eval-train-2.csv
- eval-valid.csv
- eval-lid.176.bin

Once you've uploaded the files and updated the required user inputs in the notebook, you can run the notebook.

Evaluate output from a health insurance chatbot

This example uses a health insurance chatbot as the application. The notebook walks through the four main phases of an evaluation workflow. This document discusses an overview of the use case, while implementation details and detailed guidelines and best practices are included in the notebook.

By the end of this example, you'll have reports for two generations of the chatbot, which you can view in the **Evaluate** page in the Snorkel Flow app.



Phase 1: Onboarding artifacts

The goal for your evaluation pipeline is for every version of your GenAI system to have an evaluation report associated with it. This notebook starts with a dataset (`eval-train-1.csv`) that represents a set of instructions and responses from the first version of the chatbot.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

You'll also be defining how you want to measure success.

Prepare your evaluation pipeline with these steps:

1. Import the necessary libraries and create the application scaffolding in Snorkel Flow.
2. Import your existing data from the first version of the health insurance chatbot and some labeled responses as ground truth.
3. Define individual criteria for a high-quality response. In this example, the criteria are:
   - Completeness
   - Correctness
   - Polite Tone
   - Retrieved Context Accuracy
   - `unit_test_contains_pii`
   - `unit_test_mentions_competitors`
4. Set up annotation batches to collect human feedback.
5. Define evaluators that measure the criteria programmatically.
6. Define data slices to benchmark the performance of the chatbot across different topics. In this example, the slices are:
   - Verbose questions
   - Questions written in Spanish
   - Disputes
     - Questions about out-of-network coverage
     - Questions about premiums

> NOTE
>
> - Remove any NA/None type values from your dataframes before uploading them to Snorkel.
> - This workflow introduces a **model_source** object, that maps data sources in the parent dataset to different experiments. For example, the initial data is from the `llama3-8B-v0` experiment, and it has a different model source id than data added later from the `llama3-8B-v1` experiment. This object distinguishes metrics across evaluations and determines the data a user is developing.

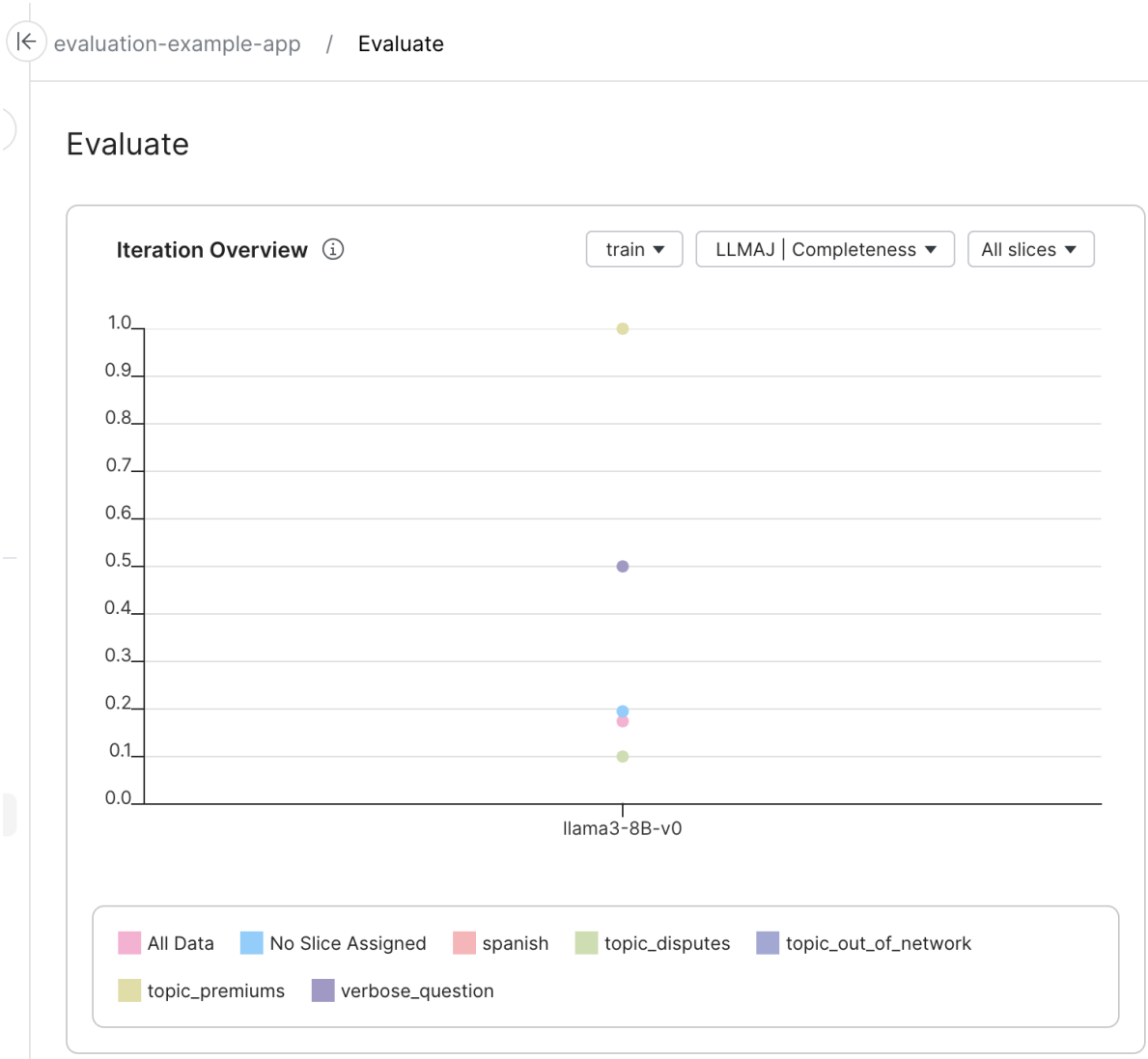
To read more about this phase, see Onboarding artifacts.

Phase 2: Creating the initial evaluation benchmark

In this phase, you'll create a simple single-criterion benchmark and a multi-criteria benchmark that uses the evaluators and data slices you defined in the previous phase. You'll create the first report that shows the performance of the chatbot for the first version of collected data.

Visit the **Evaluate** page for your app in the Snorkel Flow app to see the results.

Copyright © 2025 Snorkel AI, Inc. All rights reserved



To read more about this phase, see Creating the initial evaluation benchmark.

Phase 3: Refining the benchmark

Before you can trust your benchmark, you need to assess whether it's measuring what you want it to measure. In this phase, you'll refine the benchmark using several methods. The example notebook shows how to:

- Add additional reference prompts in the Spanish language
- Incorporate human feedback from the annotation batch for the Completeness metric

To read more about this phase, see Refining the benchmark.

About criteria and evaluators

Before you can measure response quality, you need to identify the dimensions of response quality that your organization cares about. You can measure each criterion using domain expert labels as ground truth or with Snorkel evaluators, which allow you to scale quality measurements and evaluate subsequent experiments with little involvement from your domain experts.

These are the criteria included in the notebook:

Copyright © 2025 Snorkel AI, Inc. All rights reserved

```
criteria = [
    {
        'name': 'Correctness',
        'description': "Correctness measures the accuracy and relevance of the
chatbot's response to the user's query. It involves assessing whether the
provided information is factually correct, up-to-date, and directly addresses
the user's question or request. Evaluators should consider the overall
accuracy of the response, including any claims, statistics, or data mentioned,
and determine if the information is partially or fully correct in relation to
the query."
    },
    {
        "name": "Explainability",
        "description": "Explainability refers to the clarity and
comprehensibility of the chatbot's response. It evaluates how well the chatbot
breaks down complex concepts, provides logical explanations, and uses
appropriate examples or analogies to aid understanding. Evaluators should
assess whether the explanation is clear for the target audience, has a logical
flow, and effectively communicates the intended information or instructions."
    },
    {
        "name": "Proper Tone",
        "description": "Polite Tone assesses the chatbot's ability to maintain
a respectful and courteous demeanor in its responses. Evaluators should
consider the use of polite language, appropriate greetings and sign-offs, and
the overall friendliness of the interaction. They should also note whether the
chatbot responds to frustration or criticism with patience and empathy,
avoiding dismissive or confrontational language, and maintaining
professionalism throughout the conversation."
    },
    {
        "name": "Retrieved Context Accuracy",
        "description": "This criterion evaluates how well the AI's response
aligns with and correctly uses retrieved information. Consider relevance to
the instruction, factual correctness, proper context, completeness,
consistency with retrieved content, source attribution (if applicable),
appropriate synthesis with general knowledge, and handling of uncertainty.
Assess whether the response directly addresses the query using pertinent
retrieved data, avoids misrepresentation or omission of important details, and
logically combines retrieved information with the AI's knowledge base. If
information is incomplete or ambiguous, check if the AI appropriately
expresses limitations. Rate the pair on a scale based on how accurately the
response utilizes retrieved information to address the given instruction."
    },
    {
        "name": "unit_test_contains_pii",
        "description": "This criterion assesses whether the chatbot's response
includes any information that could be used to identify a specific individual.
```

Copyright © 2025 Snorkel AI, Inc. All rights reserved

```
Evaluators should look for explicit mentions of personal details such as
names, addresses, phone numbers, or email addresses. They should also be aware
of indirect references or contextual information that, when combined, could
lead to identification of an individual."
    },
    {
        "name": "unit_test_mentions_competitors",
        "description": "This criterion evaluates whether the chatbot's
response mentions or discusses competing products, services, or companies,
especially when not directly relevant to the user's query. Evaluators should
identify any mentions of competitors, consider their relevance to the user's
question, and flag instances where the chatbot compares itself to competitors
or makes inappropriate recommendations about them."
    }
]
```

About simple and multiple-criteria evaluation reports

There are two ways to create evaluation reports:

1. As a child method of the fine-tuning application. Recommended for simple evaluation reports.
2. As part of the standalone `evaluation` module. Recommended for more complex, multi-criteria, multi-evaluator reports.

Let's examine the function signature for the FineTuningApp's `create_evaluation_report` method:

```
ft_app.create_evaluation_report?
```

Create an evaluation report for the quality dataset.

Parameters

split The split of the data to evaluate (if not provided, metrics will be computed for all splits) quality_models The quality models to evaluate (if not provided, the committed quality model or the most recently trained model will be used, in that order) finetuned_model_sources The finetuned model sources to evaluate (if not provided, all finetuned models associated with the datasources will be used) slices The slices to evaluate (if not provided, all slices in the given dataset will be evaluated)

Returns

Dict[str, Any] A dictionary containing the evaluation results

The current `create_evaluation_report` method is a child of the FineTuningApp, which means that it has access only to the ground truth label schema attached to the model node and any custom evaluators that the user has developed. By default, the function will try to calculate all available metrics for all available slices, but this can be customized in the function call.

The individual evaluator values are not stored at the datapoint level. Global and per-slice evaluator scores are stored as part of the evaluation dashboard.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

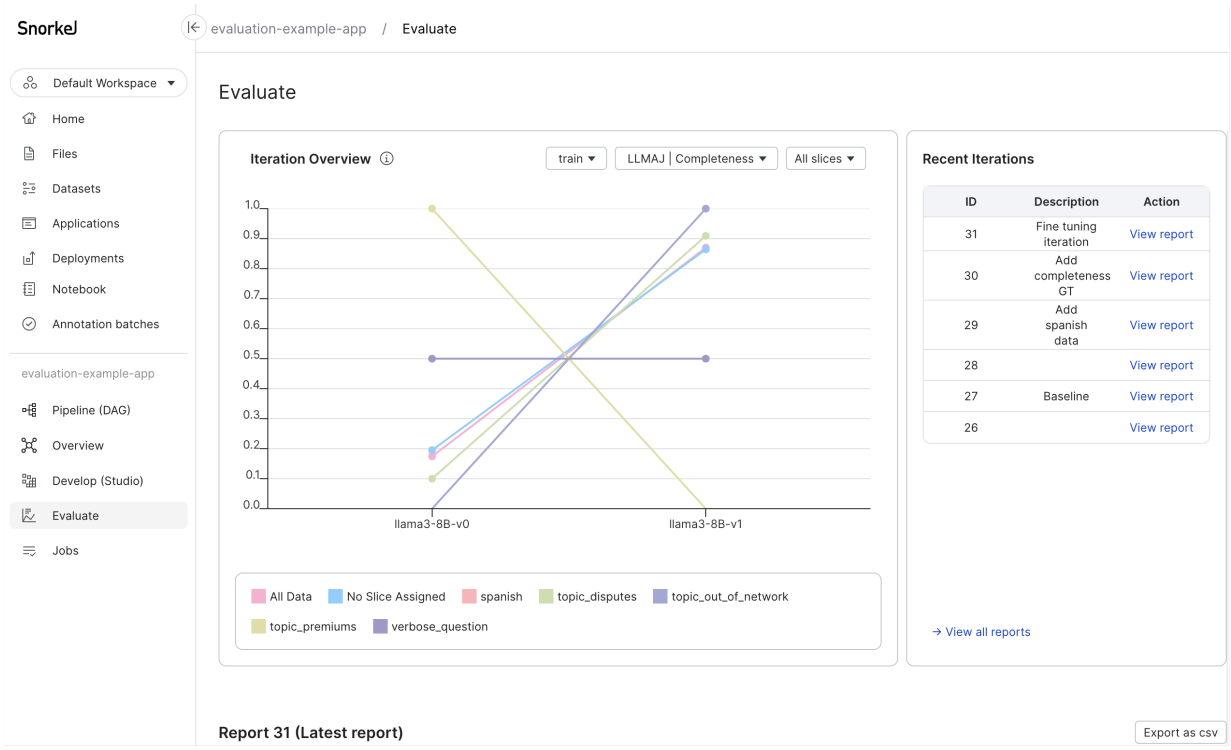
To create a more advanced version of the evaluation report, you can instead use the `evaluation` module. The `evaluation` module allows you to combine all of collected ground truth at the dataset level with all of your custom evaluators. If you're building complex custom evaluation dashboards, Snorkel recommends this approach.

Phase 4: Improving your GenAI application

Finally, with a trustworthy benchmark, you can now seek to improve the chatbot, using insights from the evaluation reports.

The example notebook shows how to:

1. Fine tune the chatbot using a curated dataset.
2. Upload a new set of instructions and responses from the second version of the chatbot (`eval-train-2.csv`).
3. Generate a new evaluation report to see the performance of the improved chatbot.
4. Visit the **Evaluate** page for your app in the Snorkel Flow app to see the updated results and compare performance across versions of the chatbot.



To read more about this phase, see Improving your GenAI system.

Next steps

Keep iterating on your GenAI application until its performance across the evaluation criteria meets your standards for production. You can read more about Snorkel's Evaluation feature set here.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

Notebook: eval-genai-notebook.ipynb

View and download the eval-genai-notebook.ipynb notebook:

Introduction

This notebook demonstrates how to set up an evaluation pipeline for GenAI-generated output, using Snorkel's Evaluation feature set.

The evaluation workflow has four main phases:

1. Onboarding artifacts
2. Creating the initial evaluation benchmark
3. Refining the benchmark
4. Improving your GenAI application

This notebook covers all four phases, using a small dataset of chatbot responses from an example GenAI chatbot application that answers questions about medical insurance for customers. For more about this example use case, read the Evaluate GenAI output (https://docs.snorkel.ai/docs/0.95/user-guide/use-cases/genai-evaluation-notebook) tutorial in the Snorkel documentation.

After running this notebook, you'll have a functional end-to-end benchmarking pipeline that allows you to track progress against multiple criteria for sequential generations of the chatbot response data.

Prerequisites

Have the following resources available before you run this notebook:

- A Snorkel Flow instance
- A Superadmin API key for Snorkel Flow
- Amazon SageMaker with authentication secrets (used for fine tuning; you can skip that step)

User input

Copyright © 2025 Snorkel AI, Inc. All rights reserved

Data preparation

Before uploading your data to Snorkel Flow, we recommend that you go through the basic preparation steps outlined in this page (in addition to any custom preprocessing needed for your unique application) to minimize any unexpected data processing issues when managing your applications. After you are finished preparing your data, follow the steps in Data upload to upload your dataset to Snorkel Flow.

> NOTE
> The examples on this page use data that is stored in Pandas DataFrame and Snorkel Flow's object storage (MinIO). However, Snorkel Flow supports data uploaded from other sources as well. See Supported data source types for more information.

Format data and ground truth for your ML task

The following pages walk through how to format your data and ground truth for different input data types and output tasks:

- Data format for different ML tasks
- Ground truth formats for different ML tasks

Check for NaN, -Inf, and Inf values

We recommend removing `NaN`, `-Inf`, and `Inf` values from your data before uploading. *Out of range float values are not JSON-compliant* errors are common with these types of values while working with data in Snorkel Flow.

There are many resources online that walk you through how to remove `NaN`, `-Inf`, and `Inf` values. Here is an example for sanitizing data in this format for Pandas DataFrames.

Check column types

We recommend checking the column types of your data to ensure that they are typed as expected. This can be done using `dtypes` for Pandas DataFrames.

> TIP
> Snorkel Flow treats columns titled `url` as URLs, which means the raw URL and the content hosted at the URL is displayed. Snorkel Flow renders the rich file format when you view this data for web pages, PDFs, images, and audio.

Make note of the label column name and label classes

If you have ground truth as a column in the data that you are uploading, make a note of the **label column** name and the **label classes**. You'll need to specify these while uploading data to Snorkel Flow.

(Optional) split data into train, valid, and test sets

You can optionally split your data into **train**, **valid**, and **test** sets before uploading it to Snorkel Flow. You many also choose to split by percentages during data upload to Snorkel Flow.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

- Train split: The train split does not require ground truth labels, but we recommend including a few ground truths so that you can utilize them during model development. You can also add ground truth labels in Snorkel Flow's Annotation Studio after you upload the dataset.
- Valid split: The valid split should contain ground truth labels for evaluation purposes. This split can be used to tune your ML model.
- Test split: The held-out test split should contain ground truth labels for evaluation purposes.

The following example shows how data can be shuffled and split into train, valid, and test sets. The data is stratified over class labels in the column named `"label_column"`, with 70% of the data in the train set, and 15% in each of the valid and test sets.

```
# Shuffle dataset
df = df.sample(frac=1, random_state=42)
# 70% / 15% / 15% split by class
df_train = df.groupby("label_column").sample(frac=0.7, random_state=42)
df_valid = df.drop(df_train.index).groupby("label_column").sample(frac=0.5,
random_state=42)
df_test = df.drop(df_train.index).drop(df_valid.index)
```

> **TIP**
>
> Be sure to check out our Tips for splitting and partitioning data guide when preparing data splits.

Export the data splits as Parquet or CSV files

If you are uploading data from a Pandas DataFrame via object storage or locally, first export the data to a Parquet or CSV file.

Snorkel recommends Parquet files instead of CSV files to avoid column type ambiguities. For example, defining if "123" is meant to be ingested as a `string` or `integer` type. Parquets explicitly store these column types; CSVs do not.

The following example shows how to export data from a Pandas DataFrame to Parquet files:

```
# Export to Parquet file
df_train.to_parquet("/path/to/my/dataset_name_train.parquet")
df_valid.to_parquet("/path/to/my/dataset_name_valid.parquet")
df_test.to_parquet("/path/to/my/dataset_name_test.parquet")
```

(Optional) Upload data to Snorkel Flow's object storage (MinIO)

Data can be uploaded from a variety of sources (see Supported data source types for more information). One option is to upload data from the MinIO object storage system deployment that comes with Snorkel Flow. This can be a good option if your data is stored locally and is larger than 100MB.

If you plan to use MinIO as a data source, you need to upload it to MinIO first. Follow the steps below:

Copyright © 2025 Snorkel AI, Inc. All rights reserved

1. **Access Snorkel Flow's object storage**. On the bottom of the left-side menu, click your user name, then click **Resources → MinIO object storage**. From there you can login to the MinIO console. For any issues with the access key, or for access issues with Kubernetes installations, contact your Snorkel administrator.

2. **Create a bucket to upload your data**. On the bottom right corner of your screen, click the **red +** button, then click **Create bucket**. From there, you can name your bucket.

   > NOTE
   >
   > Make sure that you use lowercase characters while naming your bucket. Using upper-case characters will result in an error.

   > NOTE
   >
   > MinIO does not create a folder at the specified location until a file is uploaded. If you plan to use a bucket programmatically and need the folder to exist, upload an empty file in the UI to create the folder.

3. **Upload your data**. Once your bucket is created, click the **red +** button again, then click **Upload file**. Select the files that you'd like to upload.

Make sure that you remember your file paths on MinIO, as you'll need them to create a data source on Snorkel Flow. The file path is in the form `minio://<bucket>/<path/to/file>`. For example, a file named `train.parquet` inside the bucket `mybucket` in MinIO would be referred to as `minio://mybucket/train.parquet`.

Supported data formats

Snorkel Flow does not support the DateTime data format. When you import data to Snorkel Flow, convert DateTime to String.

When you save a Snorkel Flow DataFrame to a `.csv` file, Snorkel Flow automatically converts unsupported data formats to supported data formats. For example, Snorkel Flow converts DateTime to String. If you import a `.csv` generated from another source, you must convert unsupported data types manually before importing the file to Snorkel Flow.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

Supported data source types

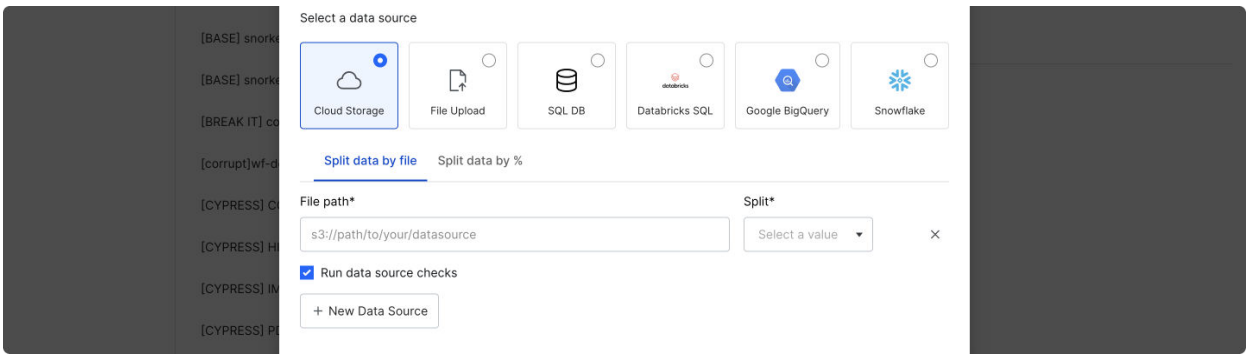
This page provides information about the data source types that Snorkel Flow supports. You can follow the steps outlined in Data preparation and Data upload to prepare and upload your dataset into Snorkel Flow.

Snorkel flow supports the following types of data sources: Cloud storage, local files, SQL, Databricks SQL, Google BigQuery, and Snowflake.

> **NOTE**
>
> For SQL, Snowflake, Google BigQuery, and Databricks SQL, your credentials are encrypted, stored, and managed by Snorkel Flow. Credentials are not logged or stored unencrypted.

Cloud storage

Select **Cloud Storage** to add data sources from Snorkel Flow's MinIO object storage or cloud services like AWS S3.



For the **File path** field, use:

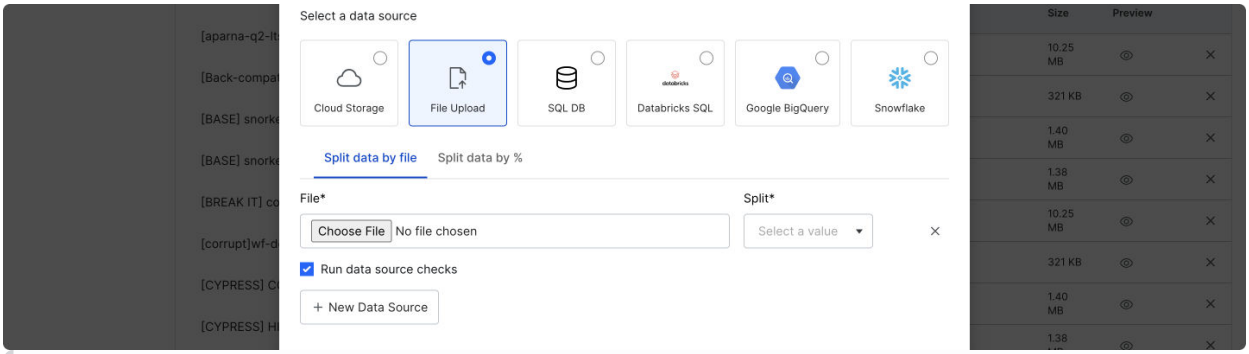
- `minio://` paths (e.g., `minio://mybucket/train.parquet`) for files in Snorkel Flow's MinIO object storage.
- `s3://` (e.g., `s3://mybucket/train.parquet`) paths for files in AWS S3.

> **NOTE**
>
> If you are using a cloud service like AWS S3 and your storage bucket is private, your Snorkel Flow instance will need to have access via an instance role. Please reach out to your administrator to configure access to private storage buckets.

Local file

Select **File Upload** to add data sources from your local machine. You must upload a Parquet or CSV file.

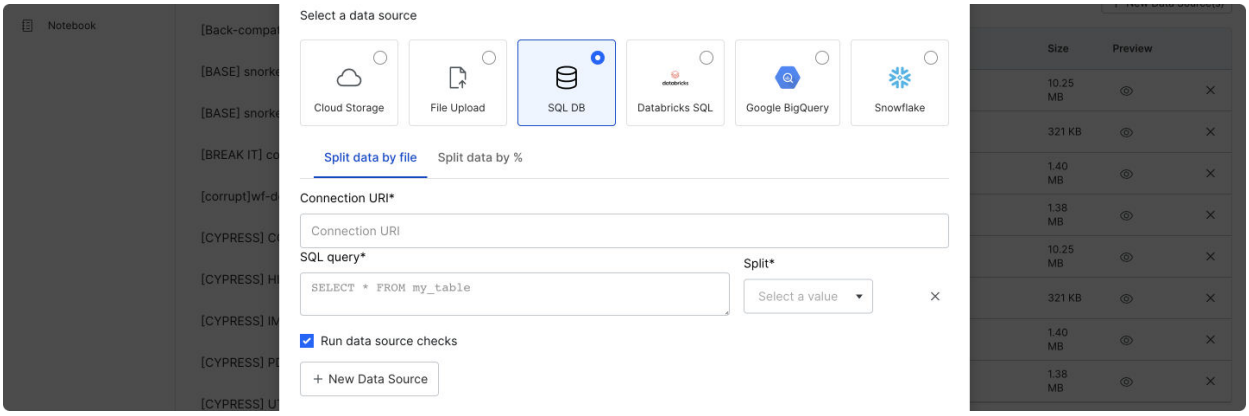
Copyright © 2025 Snorkel AI, Inc. All rights reserved



> **NOTE**
>
> Local data upload currently only supports data source files up to a maximum size of 100MB. If your data files are too large, you should instead use MinIO or a cloud storage service like AWS S3.

SQL

Select **SQL DB** to add data sources from queries against SQL databases like Postgres and SQLite.



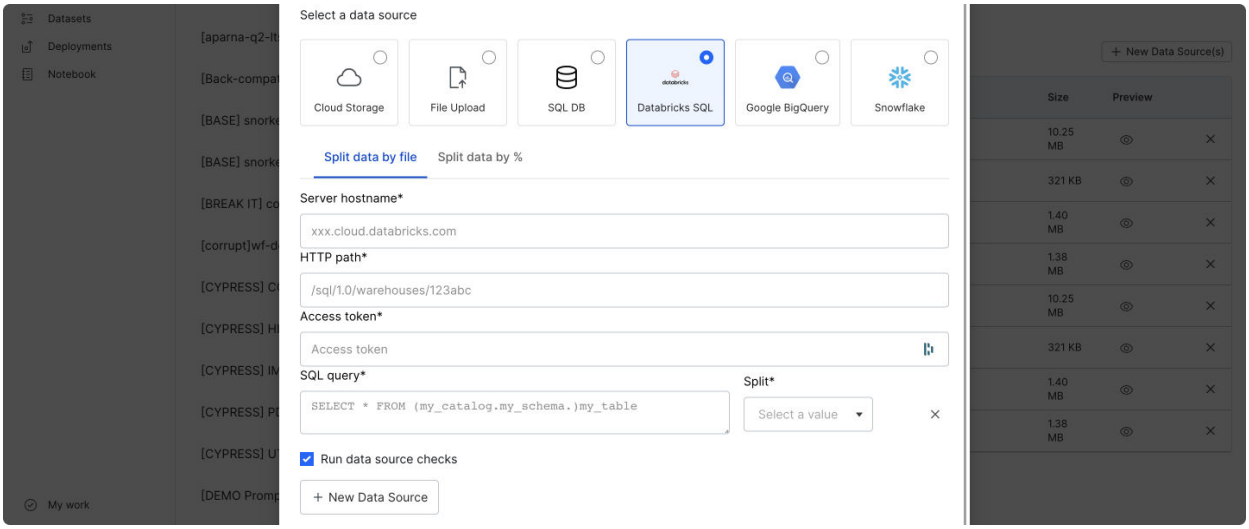
Fill out the required fields:

- **Connection URI**: A database connection string, for example, those that are used by SQLAlchemy.
- **SQL query**: A SQL query where each result row is a data point.

Databricks SQL

Select **Databricks SQL** to add data sources from queries against a Databricks SQL warehouse. Currently, all table columns are read.

Copyright © 2025 Snorkel AI, Inc. All rights reserved



Fill out the required fields:

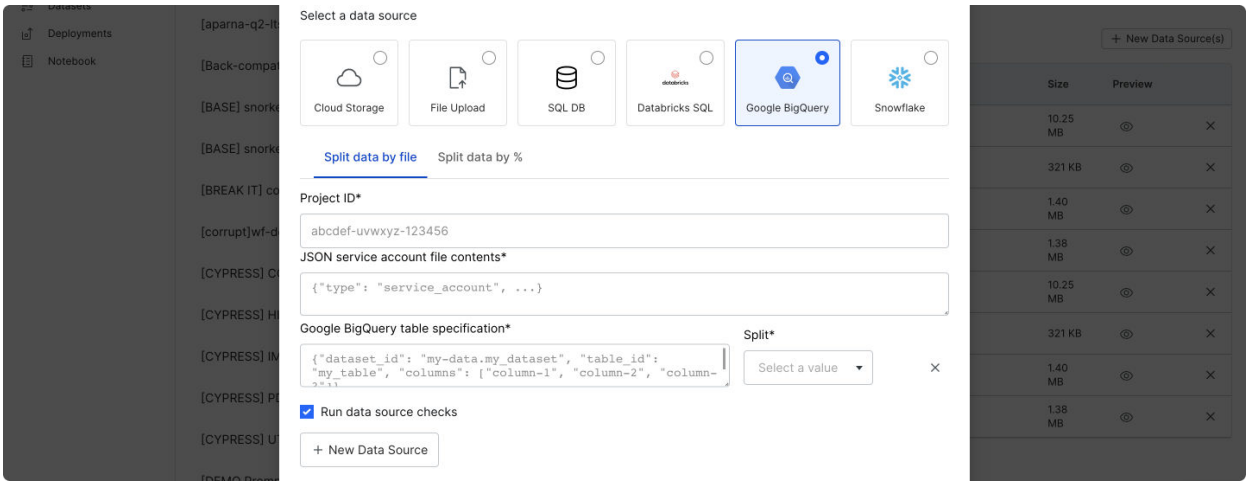
- **Server hostname**: The server hostname of your Databricks SQL warehouse.
- **HTTP path**: The HTTP path of your Databricks SQL warehouse.
- **Access token**: Your Databricks access token.
- **SQL query**: A Databricks query where each result row is a data point.

See the Databricks documentation for more details about these items.

Note
Your credentials are encrypted, stored, and managed by Snorkel Flow. Credentials are not logged or stored unencrypted.

Google BigQuery

Select **Google BigQuery** to add data sources from Google BigQuery tables. Currently, all table columns are read.



Fill out the required fields:

- **Project ID**: Your Google Cloud project ID.
- **JSON service account file contents**: The raw JSON contents of a key file that belongs to a service account with access to Google BigQuery. Please note that the service account requires

Copyright © 2025 Snorkel AI, Inc. All rights reserved

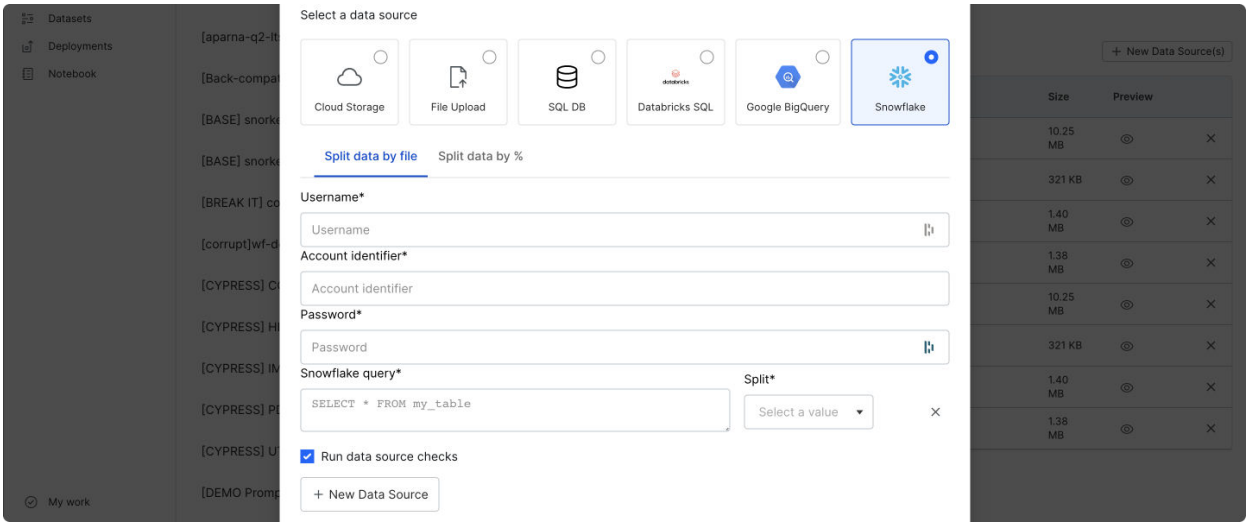
`roles/bigquery.readSessionUser` and `roles/bigquery.dataViewer` to read a table.

- **Google BigQuery table specification**: A JSON specification for the table columns to read, where each result row is a data point. The specification must have the following keys, and an example is provided below.
  - `dataset_id`: The BigQuery dataset ID.
  - `table_id`: The BigQuery table ID.
  - `columns`: The list of columns to include.

```
{
 "dataset_id": "bigquery-public-data.noaa_tsunami",
 "table_id": "historical_source_event",
 "columns": ["id", "year", "location_name"]
}
```

Snowflake

Select **Snowflake** to add data sources from queries against a Snowflake data warehouse.



Fill out the required fields:

- **Username**: Your Snowflake username.
- **Account identifier**: Your Snowflake account identifier.
- **Password:** Your Snowflake password.
- **Snowflake query**: A Snowflake query where each result row is a data point. Queries can specify the database, schema, and table, as in the example below.

```
SELECT c_name, c_address
FROM snowflake_sample_data.tpch_sf1.customer
LIMIT 10;
```

> NOTE
>
> The default warehouse that is specified for your Snowflake account will be used.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

Copyright © 2025 Snorkel AI, Inc. All rights reserved

Upload ground truth

This topic demonstrates how to upload ground truth (GT) in Snorkel Flow. There are two types of GT in Snorkel Flow: document-level GT and span-level GT.

Document-level ground truth

Document-level GT often applies to classification applications, where each document has one label.

Uploading from a dataset

In a Dataset, navigate to the **Data Sources** tab. From there, click on **Upload ground truth** button on the right. Add your ground truth file for upload.

> NOTE
>
> Only classification label schemas are supported for upload on the **Datasets** page.

Uploading from an application

The document-level GT can be added into the data source files, where we have a label column to indicate the GT label for each row of the data. During application creation, you need to specify the `Ground truth column` by the name of the label column.

You can also upload the document-level GT from the **Datasource** upload page in the **Model** pane.

Data format

In particular, the GT contains the following columns:

- `uid` (int): The uid of the document.
- `label` (str): The GT label of the document.

Span-level ground truth

Span-level GT often applies to information extraction applications, where each span has one label. For more, see the Information extraction: Extracting execution dates from contracts tutorial.

Uploading from an application

The span-level ground truth can only be uploaded from the **Datasource** upload page in the **Model** node.

Data format

The uploaded span-level GT must contain the following columns:

- `context_uid` (int): The UID of the document from which the span was extracted. All spans that have the same `context_uid` come from the same document. The `context_uid` is associated with the **UID column** when one creates the dataset. For more, see Data upload.
- `span_field` (str): The field where the spans were extracted from.
- `char_start` (int): The index of the first character of the span in the document.
- `char_end` (int): The index of the last character of the span in the document.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

- `_gt_label` (str): The GT label of the span.

As an example, below is an example ground truth for a single document with `context_uid = 0`, and extraction field of `text`:

gt_label	context_uid	span_field	char_start	char_end
class_1	0	text	17	28
class_2	0	text	37	46

> NOTE
> Only the spans already extracted by the candidate extractor can be recognized as GT. Both the span extracted and the span in the GT must have the same vaule of `context_uid`, `char_start` and `char_end`.

We recommend you to iterate on the extractor until you get as close as possible to 100% recall. One can evaluate the performance of the candidate extractor via `sf.get_candidate_extractor_metrics` SDK method. For more, see Candidate extractor scoring.

If you find most of your GT spans overlap with the extracted spans, you can consider resolving you GT to be exactly match with the extracted spans. For example, in the sentence `It's a lovely day.`, if the GT span is `day`, but the extracted span is `lovely day`, we can resolve the GT span to be `lovely day` and use the corresponding `char_start` and `char_end` of `lovely day` in the span level GT.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

Uploading a dataset

Snorkel Flow organizes data into **data sources** and **datasets:**

- **Data sources** are individual partitions of data points, such as the rows of an individual parquet file or resulting rows from an individual SQL query. Each data source is assigned to exactly one split (train, valid, or test).
- **Datasets** are collections of data sources, with each data source assigned to a specific split (train, valid, or test) within the dataset.

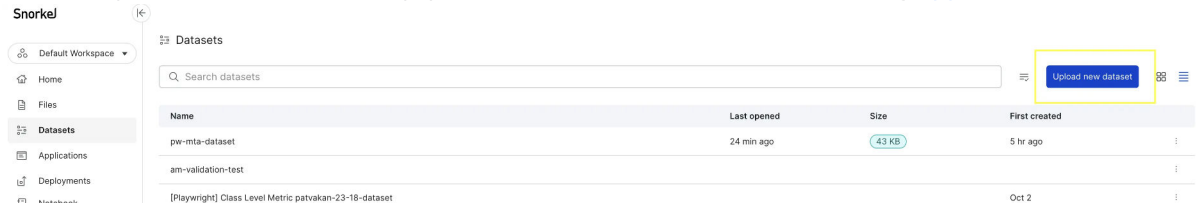
You can upload datasets to Snorkel Flow, starting with a single data source. Data sources can be added to datasets at any time.

Prerequisite

Before uploading your dataset to Snorkel Flow, prepare your data to minimize any unexpected data processing issues when managing your applications.

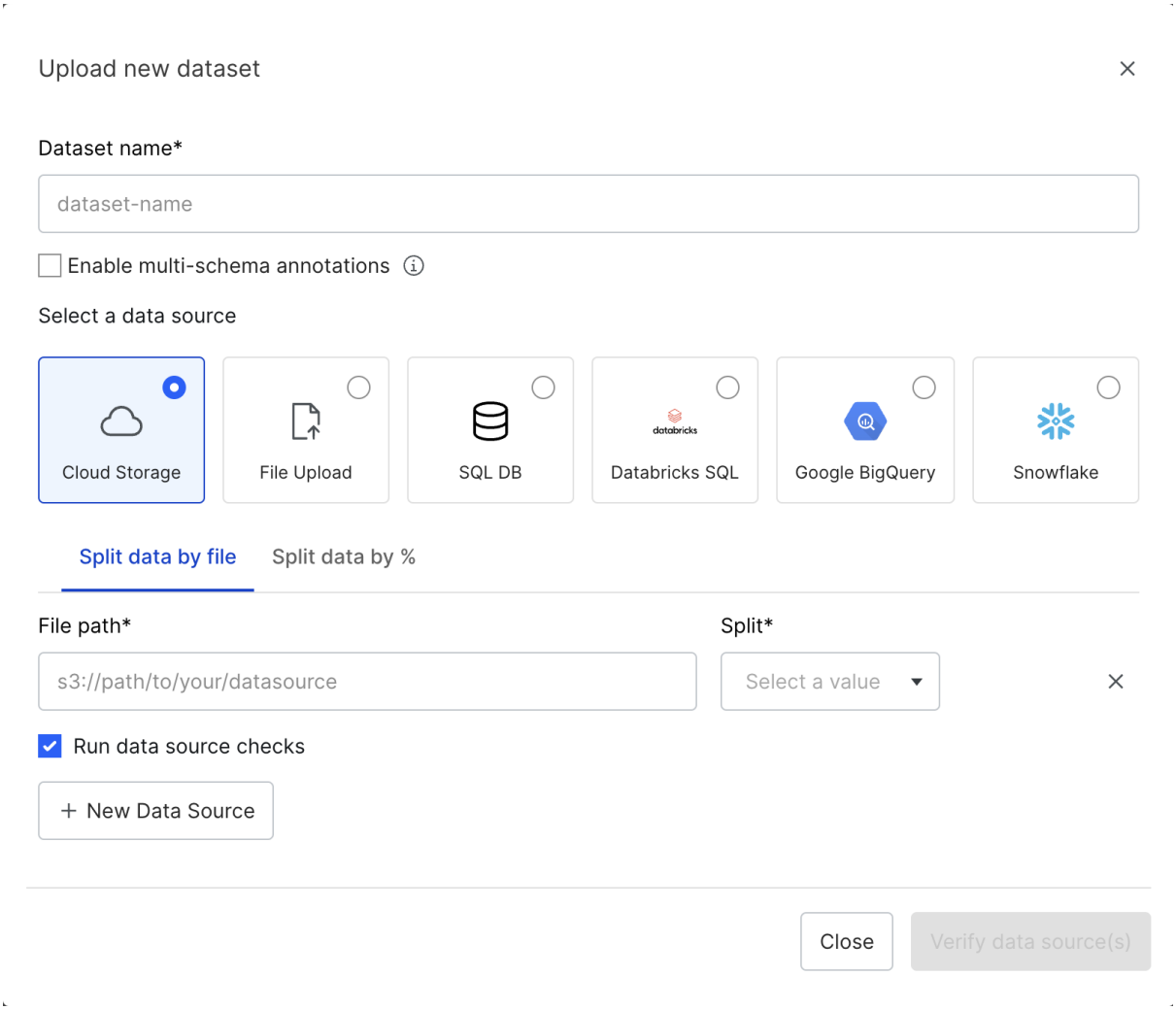
To upload a dataset

1. Select the **Datasets** option in the left-side menu, and then click **Upload new dataset** in the top right corner of your screen. Alternatively, you can also create a new dataset during application creation.

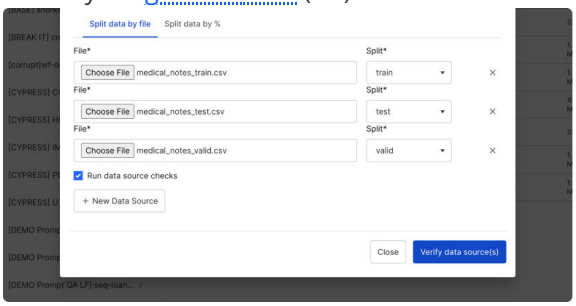


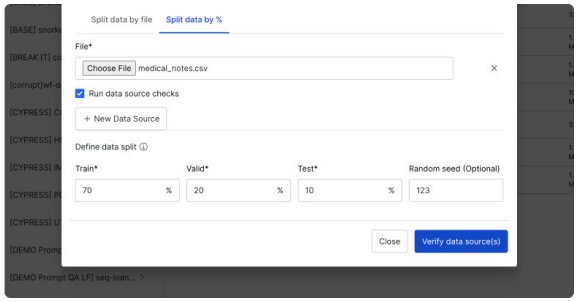
2. For the **New dataset**, enter a **dataset name** and select your **data source**. Learn more about supported data source types.

Copyright © 2025 Snorkel AI, Inc. All rights reserved



3. Assign splits to data sources. There are two ways to assign splits to data sources:
   - Manually with the **Split data by file** option.
   - Automatically with the **Split data by %** option. If you select this option, then you will also need to define your ground truth (GT) column.



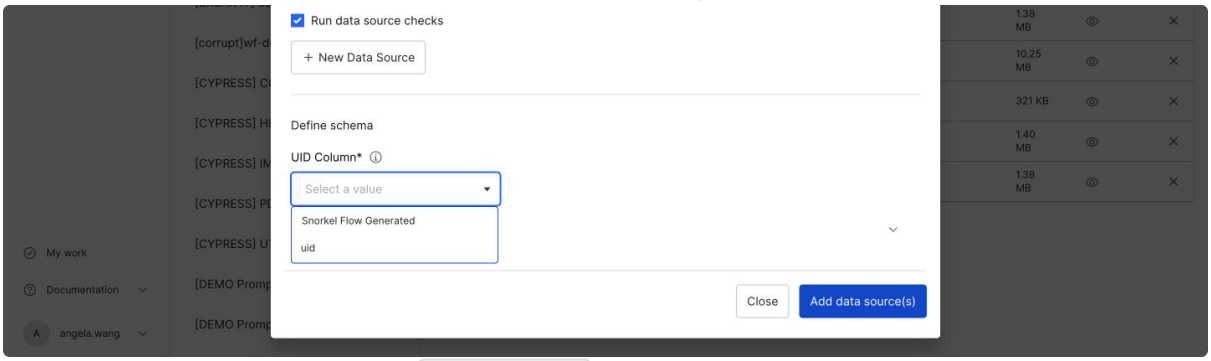


**Tip **If you have a large amount of unlabeled data, split your training data into smaller data sources. You can enable a subset of data sources for faster initial development before scaling up to all of your data.

4. After uploading your data sources, select **verify data source(s). Run data source checks** is selected by default to automatically run basic checks such as detecting NaN values.
   - If the verification succeeds, there is a green checkmark next to each data source.
   - If the verification fails, there are warnings and/or errors next to each data source.

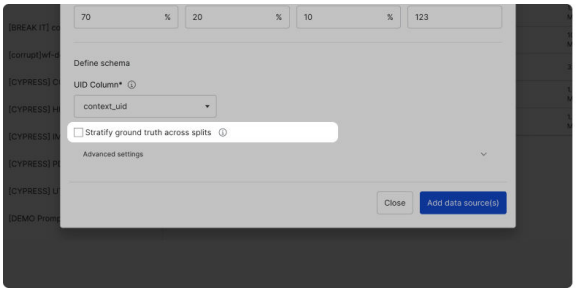
Copyright © 2025 Snorkel AI, Inc. All rights reserved

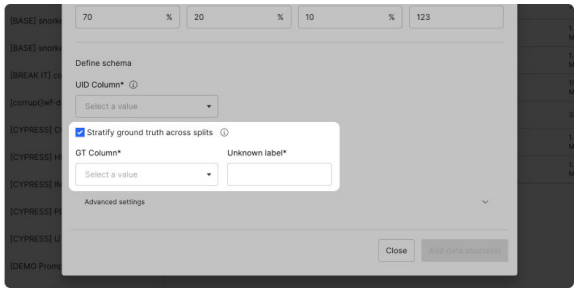
5. Once all of your data sources are verified, choose a **UID column** that is unique for each row across the dataset. This UID column can be text or an integer.If your dataset does not have that type of field, then choose **Snorkel Flow Generated** to have Snorkel Flow generate a UID column.



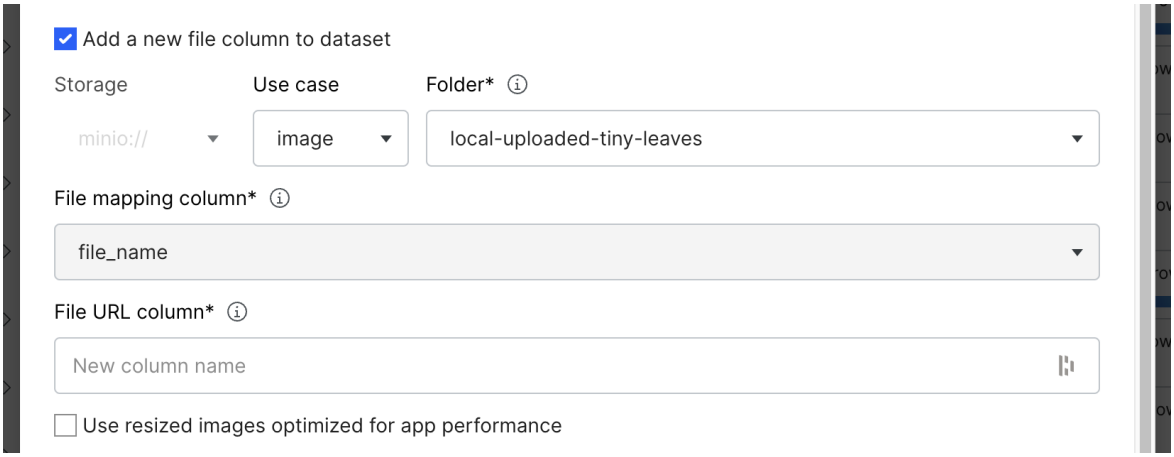
Once the dataset is created, a new `context_uid` column is added to your data. This column is populated with the selected **UID column** or the **Snorkel Flow Generated** UID.

6. If you chose to automatically split your data with the **Split data by %** option, you can **Stratify ground truth across splits**. This option ensures that the ground truth labels are evenly distributed across the splits.
   - If you opt in to stratify ground truth, provide the **GT column** and the value corresponding to UNKNOWN ground truth.
   - If you opt out, the data is split randomly.



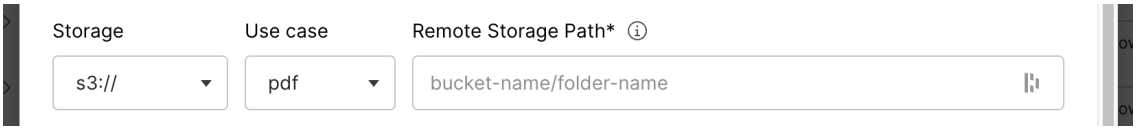


7. **[Optional]** When you are creating a dataset for a CV or PDF use case, you can reference uploaded files that are associated with each row's UIDs in the dataset.
   - Select the **Add a new file column to dataset** checkbox. This options shows the inputs to specify where your files are located and how to associate them with your dataset.



Copyright © 2025 Snorkel AI, Inc. All rights reserved

- **Storage:** This dropdown specifies where the relevant files currently reside. If you have uploaded your files directly through Snorkel Flow, they will reside in `minio://<workspace-id>/`.
    - **minio://:** If you specify `minio://` as your **Storage** option, enter the **Folder** input. Select this dropdown from the list of folders you have previously uploaded files to directly through Snorkel Flow.
    - **Remote Storage Path:** If you specify a remote storage service such as Amazon S3, enter the details:



      Supply a fully qualified storage URL or the path portion of it. Both of these input values are acceptable for S3:
        - `s3://my-bucket/path/to/files`
        - `my-bucket/path/to/files`
    - **File mapping column:** Select the column header from the uploaded data source that stores the filenames without extensions of the files that are associated with each row in the data source. For example, if you uploaded an image that is named `starfish_123.jpg`, then the row associated with that file should have a value of `starfish_123` in the column to select.
    - **File URL column:** Specify a name for the new column to be automatically inserted into the dataset. The new column is populated with URLs for the uploaded files.
    - **Optional: Use resized images optimized for app performance:** This option is valid for only image file types uploaded through Snorkel Flow. When files are uploaded through Snorkel Flow, smaller optimized versions are created automatically. If you would like to optimize application performance, check this box to reference the optimized versions instead of the full resolution images.
3. **Use case:** The dataset use case, such as image or PDF.

8. Select **Add data source(s)** to complete the dataset upload.

You can use this dataset in any application.

What's next?

- Using multi-schema annotations

Copyright © 2025 Snorkel AI, Inc. All rights reserved

Uploading files to file collections

User files are the assets related to data points in a given data source. For example, if you have a PDF application, the user files would be the actual `.pdf` files. For a computer vision (CV) application, the user files would be the `.jpg` or `.png` image files.

PDF and CV applications display these files throughout Snorkel Flow. When files are uploaded via Snorkel Flow, they reside on the same server that hosts the Snorkel Flow application.

Permissions

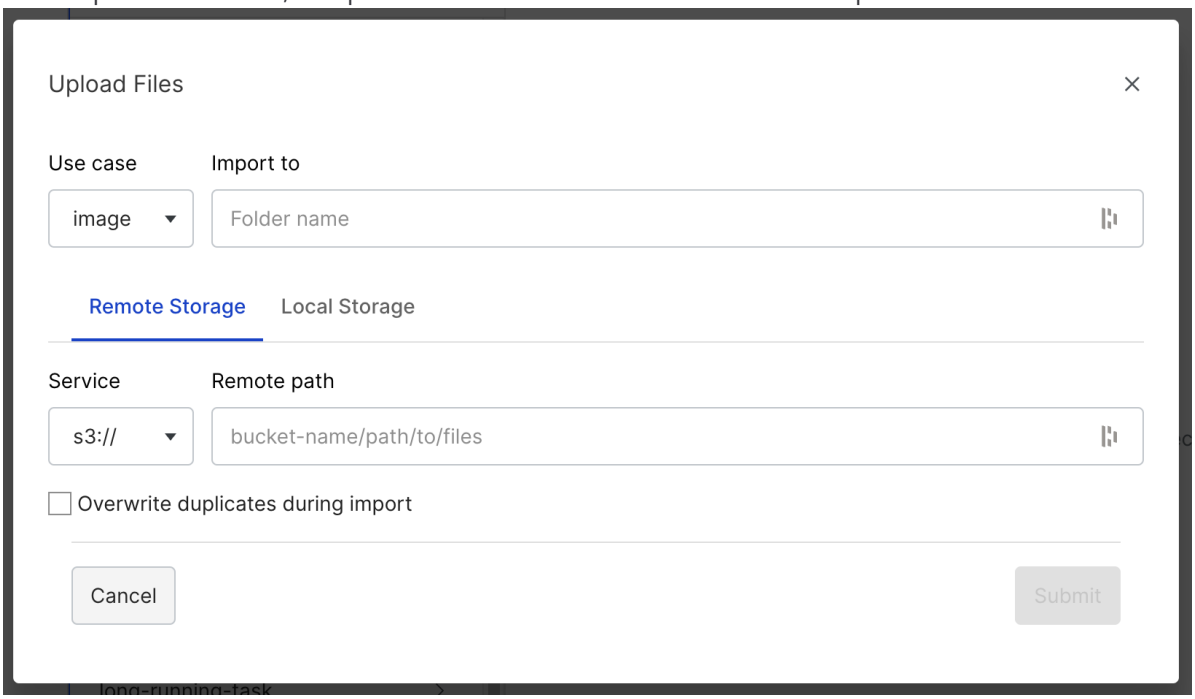
Instance-level permissions are required to upload user files:

- Direct file uploads from the computer that is running Snorkel Flow.
- Remote file transfer from a cloud storage provider such as Amazon S3 or Google Cloud Storage.

An administrator may disable upload permissions for the instance you are working on, which could prevent you from uploading or transferring user files.

To upload user files

1. To upload user files, navigate to the **Datasets** page.
2. Select the **+ Upload new files** button to open up the file upload modal.
3. In the upload file modal, complete the fields to transfer remote files or upload local files:

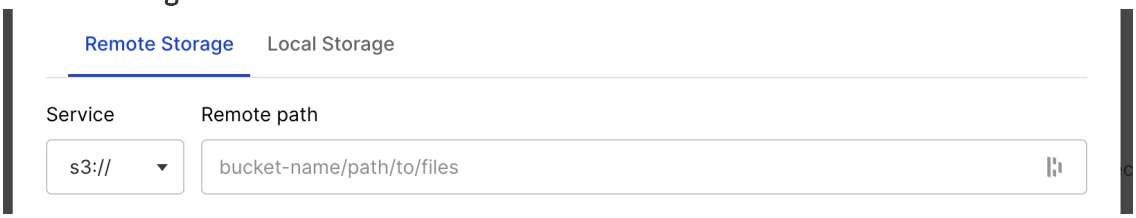


- **Use case:** Select the file type you want to upload from the dropdown. This restricts the file type that can be uploaded or transferred. For `image` applications, `.jpg`, `.jpeg`, and `.png` are allowed. For `pdf` applications, only `.pdf` files are allowed.
- **Import to:** Name the folder to which you want to upload files. If you enter an unused name, a new folder is created where the files are uploaded.
  If you enter the name of an existing folder, files are added to the existing folder. For existing

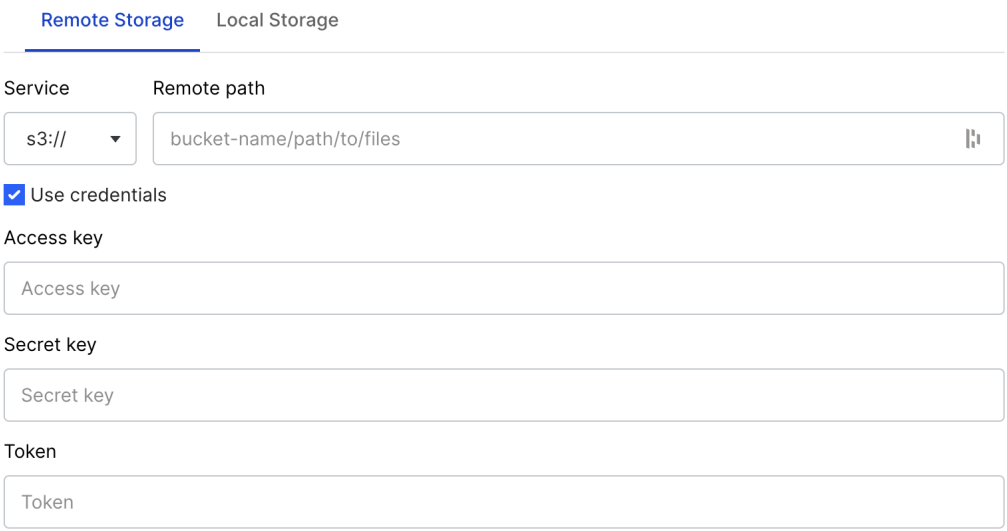
Copyright © 2025 Snorkel AI, Inc. All rights reserved

folders, you can select **+ Upload to folder** in existing folder in the file collections list. See Managing file collections.

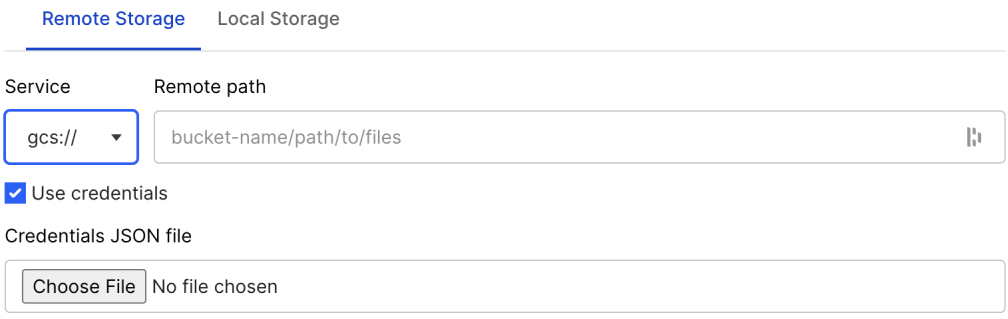
- Select the **Remote storage** and **Local storage** tabs:
  - Remote Storage



    - **Service:** Select the cloud storage provider that hosts your files: Amazon S3 or Google Cloud Storage.
    - **Remote path:** Enter the URL or path to the folder containing your remotely hosted files. For S3, you can include or exclude a protocol prefix with these acceptable inputs:
      - `s3://my-bucket/path/to/files`
      - `my-bucket/path/to/files`
    - **Use credentials:** To authenticate access to your remote bucket, select **Use credentials**:
      - **S3**: Input credentials in the **Access key**, **Secret key**, and/or **Token** fields.



      - **GCS**: Upload a credentials file.



  - Local Storage

Copyright © 2025 Snorkel AI, Inc. All rights reserved

- **Choose Files:** Select this input to open your operating system's file browser. Select individual or multiple files to upload.
  For direct uploads from your computer, you are limited to a maximum of 1,000 files and/or a total upload size of 1 GB at a time. These are not limitations on the folder that you upload to. You can upload subsequent files to the same folder using **+ Upload to folder** in existing folder in the file collections list. See Managing file collections.
  You cannot select directories, but you can use shortcuts to select all files in a given directory. For example, on MacOS, enter `CMD + A` to select all files inside a given directory.

- **Overwrite duplicates during import**
  - **Checked**: Any files you upload in the collection will replace existing files with the same name.
  - **Unchecked**: Any files you upload with the same name as existing files in the collection are not uploaded.

4. Select **Submit.**
   - If you selected the **Remote storage** option, your files are transferred in the background. You can check the status of this transfer using the **Jobs** icon:

   - If you selected the **Local Storage** option, your files are uploaded. A loading spinner appears, and the user interface is disabled.
     IMPORTANTDo not close the tab. Wait for an alert to confirm the upload finished successfully. It may take several minutes to finish uploading.

Once you have uploaded or transferred files to your Snorkel Flow instance, you can associate them with your data sources when uploading new datasets. For more information about viewing your uploaded files, see Manage file collections.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

Scanned PDF guide

Scanned PDF documents are created from scanned images of printed documents. They don't contain information about the text in the document and where it is located. A quick way to check if a PDF is scanned is by trying to search for words in the document. If no matches are found, the PDF is not machine-readable as is.

Scanned PDFs require additional preprocessing using an optical character recognition (OCR) library to derive the text and layout information from these documents. Snorkel Flow supports both running OCR in-platform and importing OCR results from external tools. This page the process for setting up applications with both approaches.

In-platform OCR

We support in-platform OCR with Tesseract and Azure Form Recognizer. Azure Form Recognizer performs better on average but requires a paid subscription. We recommend starting with Tesseract which is free and open source. It is usually sufficient for data that isn't too noisy or handwritten.

To get started with a PDF application, create data sources with the following fields and add a new dataset. See Data preparation and Data upload for more information about creating and uploading data.

Field name	Data type	Description
`uid`	int	A unique id that is mapped to each row. This is standard across all Snorkel Flow data sources.
`url`	str	The file path of the original PDF. This will be used to access and process the PDF.

Tesseract

Follow these steps to create an application that runs OCR with Tesseract:

1. Click the **Applications** option in the left-side menu, then click **Create application**. This brings up the New application guided flow.
2. Enter a name and optional description for your application.
3. In the Data accordion, select the scanned PDF dataset that you previously uploaded.
4. In the Label schema accordion, set the following options:
   - **Data type**: PDF
   - **Task type**: Extraction
   - **PDF type**: Scanned PDF, need to run OCR
   - **PDF URL field**: Select the column with URL of the original PDF.
5. Follow the application creation steps described in the PDF Extraction tutorial to finish setting up your application.

This setup runs tesseract OCR over your input PDFs and saves the output in the hOCR format. We ingest this and store the data using Snorkel's internal `RichDoc` representation.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

Azure Form Recognizer

To set up an application with Azure Form Recognizer, you need to use the Python SDK (this will be coming to the Snorkel Flow UI soon!). First, follow the steps above to set up an application with Tesseract. Then, follow these steps to modify the application:

1. Set up a Form Recognizer endpoint and an Azure blob storage container. Follow the steps in the Azure portal to do this.

2. Add secret keys from the SDK.
   The `AzureFormRecognizerParser` requires access credentials to the Form Recognizer endpoint and storage container. We can store these credentials using our Credential management tool. A superadmin user can add the credentials from the SDK:

   ```
   sf.set_secret("azure_fr_key", "YOUR_AZURE_FORM_RECOGNIZER_KEY")
   sf.set_secret("azure_connection_string", "YOUR_AZURE_CONNECTION_STRING")
   ```

3. In Snorkel Flow, select **Pipeline (DAG)** in the left-hand menu.

4. Ensure that you are in "EDIT" mode.

5. Select the three dots on the first node in the DAG.

6. Click **Add node before**, then click **ChangeColumns**.

7. Click the newly created ChangeColumns node, then under **Select Operator**, click `AzureFormRecognizerParser`.

8. Enter the following fields:

Field	Value
Pdf url field	`rich_doc_pdf_url`
Form recognizer endpoint	form recognizer endpoint url
Form recognizer key	azure_fr_key (set using secret store)
Result upload storage key	azure_connection_string (set using secret store)
Result upload container	the name of your container
Result upload blob prefix	leave this blank
Result upload overwrite	leave this enabled.

9. Remove the `HocrToRichDocParser` node (and `TesseractFeaturizer` if added).
   This operator is not needed once the Azure operator is added to the Application DAG.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

```
# Deleting HocrToRichDocParser
hocr_node = sf.get_node_uid(APP_NAME, "HocrToRichDocParser")[0]
sf.delete_node(hocr_node)

# Deleting TesseractFeaturizer
tesseract_node = sf.get_node_uid(APP_NAME, "TesseractFeaturizer")[0]
sf.delete_node(tesseract_node)
```

Importing OCR results

There are several other open-source and paid OCR tools available for use in the market. If the output of these tools are provided in a standard hOCR format, then it can be ingested for use in Snorkel Flow. The input data sources need to have an additional `hocr` column as input. Create data sources with the following fields and add a new dataset. See Data preparation and Data upload for more information about creating and uploading data.

Field name	Data type	Description
`uid`	int	A unique id that is mapped to each row. This is standard across all Snorkel Flow data sources.
`url`	str	The file path of the original PDF. This will be used to access and process the PDF.

`hocr`	str	a representation of the data in standard hOCR format. For example:
		`...`
		`<p class='ocr_par' lang='deu' title="bbox930">`
		`<span class='ocr_line' title="bbox 348 797 1482 838; baseline -0.`
		`<span class='ocrx_word' title='bbox 348 805 402 832; x_wconf 93'>`
		`<span class='ocrx_word' title='bbox 421 804 697 832; x_wconf 90'>`
		`...`

Follow these steps to create an application and import your OCR results:

1. Click the **Applications** option in the left-side menu, then click **Create application**. This brings up the New application guided flow.
2. Enter a name and optional description for your application.
3. In the Data accordion, select the scanned PDF dataset that you previously uploaded.
4. In the Label schema accordion, set the following options:
   - **Data type**: PDF
   - **Task type**: Extraction
   - **PDF type**: Scanned PDF, no need to run OCR
   - **PDF URL field**: Select the column with URL of the original PDF.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

- **hOCR field**: Select the column with the OCR predictions.

5. Follow the application creation steps described in the PDF Extraction tutorial to finish setting up your application.

The team at Snorkel has used the open-source libraries Tesseract and DocTR outside the platform previously, so we provide some sample code below. Snorkel does not make any specific recommendations for or against the available OCR tools. If you have an OCR tool that you would like to use, but have concerns about data ingestion, please reach out to the Snorkel support team.

Tesseract

Follow these steps to use the in-platform notebook to run OCR with Tesseract:

1. Install the Tesseract library locally. To use the script that we provide, you'll also need the convert utility function. On a Mac, you can use the following steps for installation:

```
`brew install tesseract`
`brew install imagemagick      # Verifying installation   which tesseract
which convert`
```

2. Run the shell script below to convert PDFs to images and run OCR on them:

```
# Script to parse PDF files to HOCR files
PDF_DATASET_DIR=pdf/
HOCR_DATASET_DIR=hocr/
TIFF_DATASET_DIR=tiff/

# Command to standardize filenames in a directory to 1.pdf, 2.pdf, ... n.pdf
# ls -v | cat -n | while read n f; do mv -n "$f" "$n.pdf"; done

for f in $PDF_DATASET_DIR*
do
  filename=$(basename -- "$f")
  filename="${filename%.*}"
  echo "Processing $filename"
  # Convert PDF to TIFF image
  convert -strip -depth 8 -alpha off -density 300 -quality 100 $f
$TIFF_DATASET_DIR$filename.tiff
  # Use Tesseract to convert TIFF image to HOCR file
  # psm 4 instructs Tesseract to parse as a single column
  tesseract -l eng --psm 4 $TIFF_DATASET_DIR$filename.tiff
$HOCR_DATASET_DIR$filename hocr
done
```

DocTR

Follow these steps to use the in-platform notebook to run OCR with DocTR:

1. Follow the instructions to install DocTR.

2. Run the python script below to run OCR over the scanned PDFs:

Copyright © 2025 Snorkel AI, Inc. All rights reserved

```
from doctr.io import DocumentFile
from doctr.models import ocr_predictor
from lxml import etree, html

# Load in the OCR model
model = ocr_predictor(pretrained=True)
# Load in the PDF
doc = DocumentFile.from_pdf("path/to/your/doc.pdf")
# Run OCR on the PDF
result = model(doc)
# View the OCR result on the PDF
result.show(doc)
# Export to hOCR - per page so you have to combine the strings
xml_output = result.export_as_xml()

# Get the first page's body
doc = html.fromstring(xml_output[0][0].decode("utf-8"))
body = doc.xpath("//*[@class='ocr_page']")[0].getparent()

# Adding only the page tag from each subsequent page
for page in xml_output[1:]:
    page_doc = html.fromstring(page[0].decode("utf-8"))
    page_body = page_doc.xpath("//*[@class='ocr_page']")[0].getparent()
    for child in page_body:
        body.append(child)

total_hocr = etree.tostring(doc, pretty_print=True, method='xml',
xml_declaration=True).decode("utf-8")
with open('x_doctr.hocr', 'w') as file:
    file.write(total_hocr)
```

Conclusion

This page discussed the different ways that you can process and ingest scanned PDF documents in Snorkel Flow. To learn more about using PDF documents in Snorkel Flow, please see our other documentation:

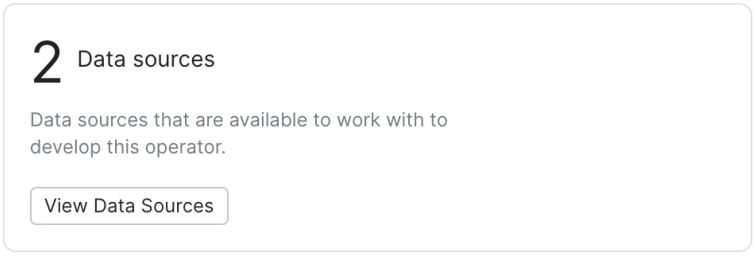
- Check out our PDF Information Extraction tutorial.
- Read more about PDF-based Operators.
- See more examples of Rich Document Builders.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

Manage the data sources for a model

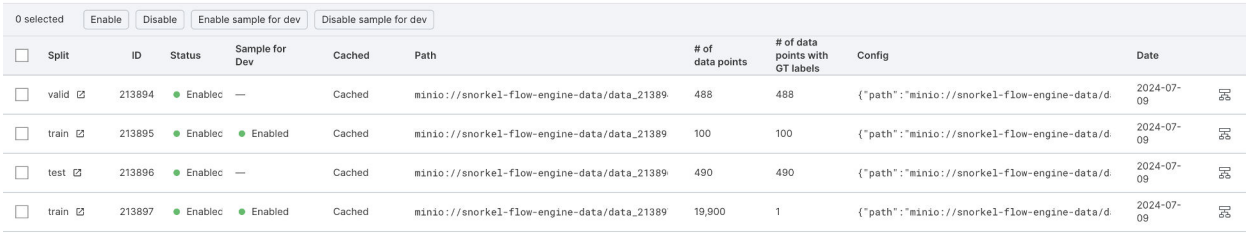
This article provides an overview of the Snorkel Flow features that allow you to manage the data sources for a model after app creation. For example, you might need to update the data sources for a model if your dataset contains data points that are not important for your use case. Or, you might need to upload ground truth for a data source.

To access this page, first navigate to the overview page for your app, then select the **View Data Sources** button.



Datasources Table

The main table on this page gives you a way to toggle the data sources that are enabled for a model. Every data source in your app is shown as a row in the table. The table also displays information about each of the data sources, including the count of data points, count of ground truth (GT) associated with each data source, the data source identifier, and the storage path for the data source file.



Select one or more data sources in the table and use the **Enable** and **Disable** buttons to enable/disable those data sources for this model. This will affect the data points that are used for training the model.

You can also enable or disable data sources from being used for dev split sampling. The dev split is a collection of data points that are used to build and evaluate the performance of our model and labeling functions (LFs). Because these datapoints are not used during training, you might want to exclude certain data sources from being used for dev split sampling.

Uploading ground truth

Select the **Upload GTs** button to upload GT for your model. This is done by specifying a path to a file that contains ground truth labels, or by uploading the file directly. Be sure to carefully follow the instructions in the dialog, as each task type has different requirements for the format of the uploaded file. See the Upload ground truth article for more information.

> **NOTE**
>
> Uploading new GT values will overwrite existing ones.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

Copyright © 2025 Snorkel AI, Inc. All rights reserved

Managing file collections

Snorkel Flow provides options for managing the files in your datasets. After uploading a dataset, select the **Files** tab to manage and view the associated files. Actions include:

- Organizing files into collections
- Checking file metadata
- Uploading additional files to specific folders.

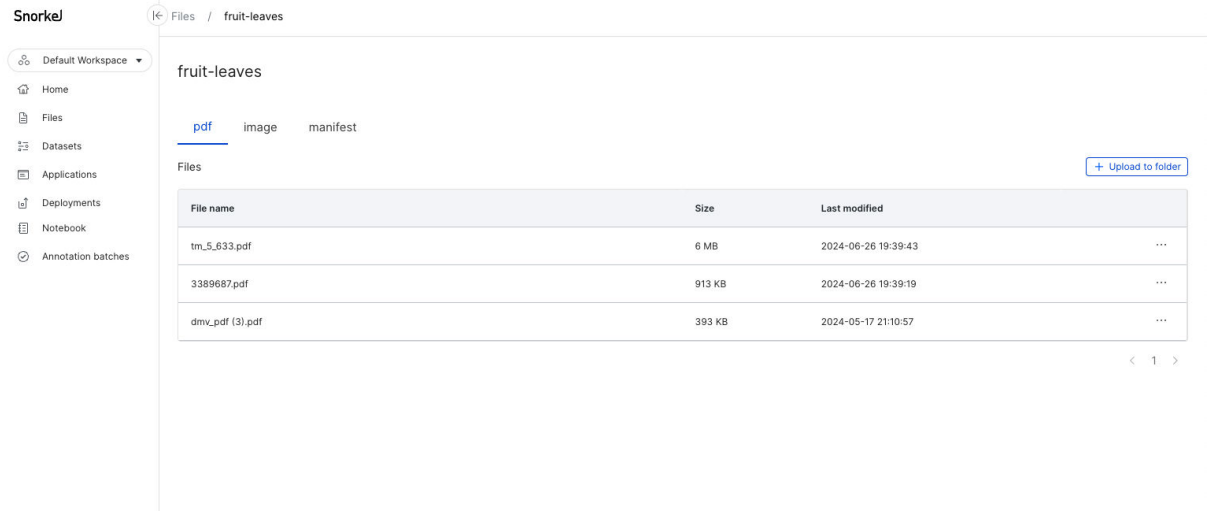
File management ensures that the correct files are linked to the relevant dataset for model training or data processing.

Once you have uploaded files into file collections, you can view and manage them on the **Datasets** page.

When you upload files, a manifest CSV is automatically created with a listing of the storage locations of all of the uploaded files. This file serves as an index of all the files you've uploaded, allowing you to easily reference their storage paths. The file can be downloaded and used in the creation of datasets. For more information see Data upload.

To view file collections

1. Select **Datasets** in the left-side menu to navigate to the **Datasets** page.
2. On the left side of your screen, you'll see your list of file collections. You can scroll the list or search for file collection by name.
3. Select the name of a file collection to show all the contained files and their metadata.



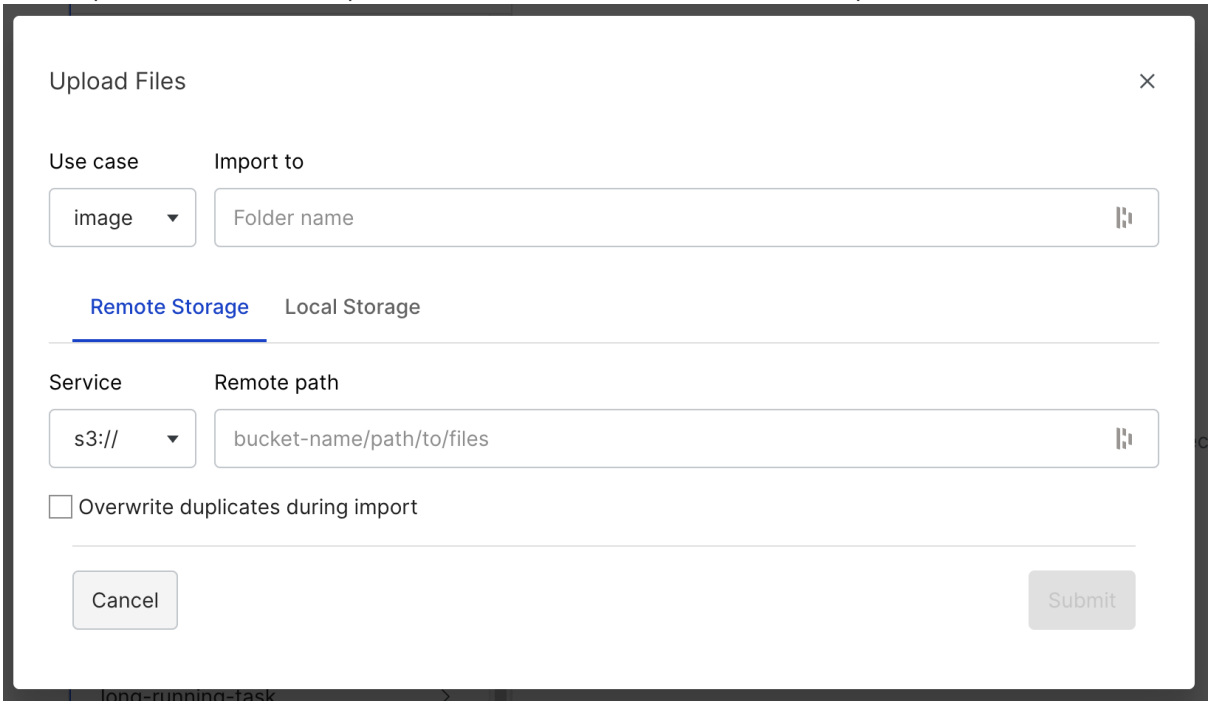
   Typically collections contain only one type of file, such as images or PDFs. If multiple types exist, the files are automatically sorted into subfolders by file type, which you can switch between by selecting the different file type tabs.

To upload additional files to a collection

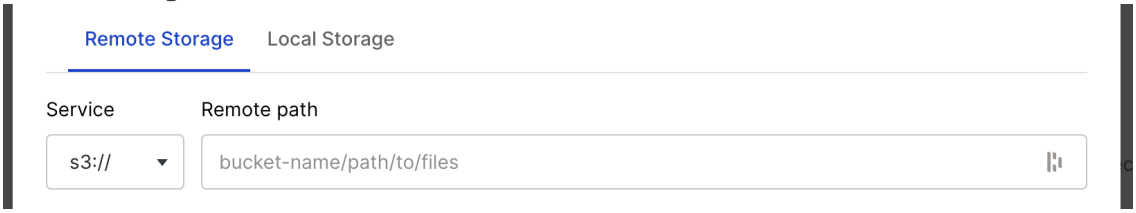
1. In the left-hand sidebar, select the file collection to which you want to upload.
2. Select **+ Upload to folder**, which will open the upload dialog.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

3. In the upload file modal, complete the fields to transfer remote files or upload local files:



- **Use case:** Select the file type you want to upload from the dropdown. This restricts the file type that can be uploaded or transferred.
  - For `image` applications, `.jpg`, `.jpeg`, and `.png` are allowed.
  - For `pdf` applications, only `.pdf` files are allowed.
- **Import to:** Name the folder to which you want to upload files.
  - If you enter an unused name, a new folder is created where the files are uploaded.
  - If you enter the name of an existing folder, files are added to the existing folder.
- Select the **Remote storage** and **Local storage** tabs:
  - **Remote Storage**



    - **Service:** Select the cloud storage provider that hosts your files: Amazon S3 or Google Cloud Storage.
    - **Remote path:** Enter the URL or path to the folder containing your remotely hosted files. For S3, you can include or exclude a protocol prefix with these acceptable inputs:
      - `s3://my-bucket/path/to/files`
      - `my-bucket/path/to/files`
  - **Local Storage**

Copyright © 2025 Snorkel AI, Inc. All rights reserved

- **Choose Files:** Select this input to open your operating system's file browser. Select individual or multiple files to upload.
  - For direct uploads from your computer, you can upload up to 1,000 files and/or a total of 1 GB in a single upload. This limit only applies to each individual upload, not to the folder you're uploading to. You can add more files to the same folder later by using the **+ Upload to folder** option in the file collections list.
  - You can't upload entire directories at once, but you can use shortcuts to select all files in a directory. For example, on MacOS, enter `CMD + A` to select all files inside a given directory.

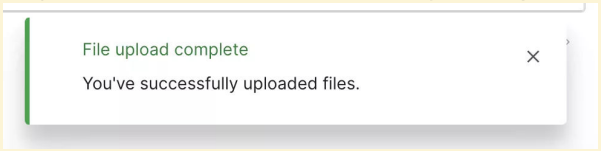
- **Overwrite duplicates during import**
  - **Checked**: Any files you upload in the collection will replace existing files with the same name.
  - **Unchecked**: Any files you upload with the same name as existing files in the collection are not uploaded.

4. Select **Submit.**

- If you selected the **Remote storage** option, your files are transferred in the background. You can check the status of this transfer using the **Jobs** icon:

- If you selected the **Local Storage** option, your files are uploaded. A loading spinner appears, and the user interface is disabled.

> ⚠ **WARNING**
> Do not close the tab. Wait for an alert to confirm the upload finished successfully. It may take several minutes to finish uploading.

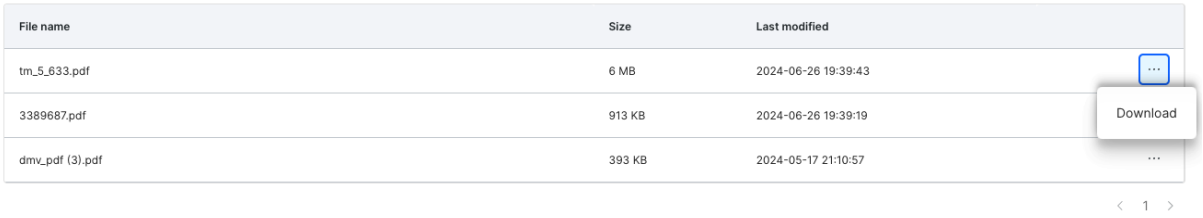


5. Once you have uploaded or transferred files to your Snorkel Flow instance, you can associate them with your data sources when creating new datasets.

To download files from a collection

1. In the left-hand sidebar, select the file collection from which you want to download.
2. Select the `...` icon next to an individual file to open the file options menu.
3. Select **Download**.

Copyright © 2025 Snorkel AI, Inc. All rights reserved



The file downloads to your browser's default download location.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

Working with MinIO

> ⚠ WARNING
>
> **Notice of deprecation**
>
> MinIO is scheduled for deprecation in future versions of Snorkel Flow through early 2025, with its functionality being replaced by **Snorkel File Service (SFS)**.
>
> - We will continue to proactively communicate changes regarding deprecation directly with customers and update documentation when appropriate. If you have concerns regarding current usage of MinIO and whether it will be supported by SFS, communicate with your Snorkel Success Manager.
> - The MinIO Console will be deprecated as of Q4 2024, but will still be accessible until the end of Q1 2025 to support backwards compatibility for older workflows. After Q1 2025, access to MinIO Console will be removed.
> - Instead, use the Files feature for uploading PDFs and images. Support for arbitrary file type upload and basic file management utilities within the Files feature will be provided by end of Q1 2025 to meet MinIO Console core feature parity.
> - For files that are not PDFs and images, you can continue to use Snorkel Flow's SDK. These SDK workflows are detailed below.
> - All SDK workflows documented here will be supported by SFS.

Overview

MinIO is an object store that is compatible with S3. A MinIO API is shipped with Snorkel Flow for data management across the platform.

It allows users to upload and download files that are accessible from both Notebooks within Snorkel Flow as well as in Operators.

Connecting to MinIO

Authentication to the Snorkel Flow MinIO is handled automatically if you are working in Snorkel Flow Notebooks. If you're accessing MinIO elsewhere, you will need to set the following environment variables `MINIO_URL`, `MINIO_ACCESS_KEY`, and `MINIO_SECRET_KEY`.

Using MinIO with the Snorkel Flow SDK

When working with the Snorkel Flow SDK, you first need to set the appropriate context to ensure code is being applied to the right workspace and application. To use MinIO, you need to define the workspace:

```
import snorkelflow.client as sf
ctx = sf.SnorkelFlowContext.from_kwargs(
    workspace_name="your.workspace",  # change this to the name of your
current workspace
)
```

This `sf` client object will be used in sections below

Copyright © 2025 Snorkel AI, Inc. All rights reserved

File upload

The SDK provides two explicit methods for uploading files (`upload_file`) and directories (`upload_dir`) using the `sf` client object defined at the top of this section. Both absolute and relative paths are supported. Examples are shown below:

```
# Upload a file from a local directory to MinIO
local_file_path = "/path/to/local/report.pdf"
remote_file_path = "minio://bucket/path/to/some/report.pdf"

uploaded_file_path = sf.upload_file(local_file_path, remote_file_path)
```

```
# Upload a directory from a local directory to MinIO
local_directory = "/path/to/local/directory"
remote_directory = "minio://bucket/upload/directory"

uploaded_dir_path = sf.upload_dir(local_directory, remote_directory)
```

File download

The SDK has two explicit methods for downloading files (`download_file`) and directories (`download_dir`) using the `sf` client object defined at the top of this section. Both absolute and relative paths are supported. Examples are shown below:

```
# Download a file from MinIO to a local directory
remote_file_path = "minio://bucket/path/to/some/report.pdf"
local_file_path = "/path/to/local/report.pdf"

sf.download_file(remote_file_path, local_file_path)
```

```
# Download a directory from MinIO to a local directory
remote_directory = "minio://bucket/upload/directory"
local_directory = "/path/to/local/directory"

sf.download_dir(remote_directory, local_directory)
```

List directory

To list files in a remote directory, use the `list_dir` method with the `sf` client object defined at the top of this section. Example is shown below:

```
remote_directory = "minio://bucket/upload/directory"

sf.list_dir(remote_directory)
```

Copyright © 2025 Snorkel AI, Inc. All rights reserved

Generic file operations

For file operations, use the SDK function `snorkelflow.utils.file.open_file`, which will return a file-like object. `open_file` works with both MinIO and local paths.

Reading a file from MinIO:

```
from snorkelflow.utils.file import open_file
with open_file("minio://bucket/path/to/some/file", mode="r") as f:
    data = f.read()
```

Writing a file to MinIO:

```
from snorkelflow.utils.file import open_file
with open_file("minio://bucket/path/to/some/file", mode="w") as f:
    data = f.write("Hello, World!")
```

Example use cases

Files in MinIO can be used as resources in custom Operators and labeling functions. This can be useful if you want to save computationally expensive outputs to a cached file such as Hugging Face model outputs.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

Labeling function example

```
import snorkelflow.client as sf
from snorkelflow.studio import resources_fn_labeling_function

ctx = sf.SnorkelFlowContext.from_kwargs(
    workspace_name="your.workspace",  # change this to the name of your
current workspace
)

# Replace "my_application" with the actual application name
application_name = "my_application"

# Fetch the model node for the given application
node = sf.get_model_node(application_name)

def get_minio_file():
    from snorkelflow.utils.file import open_file
    import json
    ext_file = "minio://bucket/path/to/some/resource.json"
    with open_file(ext_file, mode='r') as f:
        ext_resource = json.load(f)
    return {"ext_resource": ext_resource}

@resources_fn_labeling_function(name="sample_code_lf",
resources_fn=get_minio_file)
def lf(x, ext_resource):
    if x in ext_resource:
        return "LABEL"
    return "UNKNOWN"
sf.add_code_lf(node, lf, label="LABEL")
```

Custom operator class example

The file at the MinIO path should be opened within the `_compute_features` method.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

```
from typing import Any, Dict, Optional
from snorkelflow.operators.featurizer import Featurizer, OpProgressCallback

class CustomFeaturizer(Featurizer):
    """Preprocessor that retrieves model predictions from a file"""

    def __init__(self, file_path: str):
        self.file_path = file_path

        import pandas as pd
        from snorkelflow.utils.file import open_file
        self.saved_model_preds_dict: Optional[Dict] = None
        self.model_pred_df: Optional[pd.DataFrame] = None

    @property
    def input_schema(self):
        return {}

    @property
    def output_schema(self):
        return {"ext_model_predictions": str}
    def no_op_progress_callback(*args: Any, **kwargs: Any) -> None:
        pass
    def _compute_features(self, df: pd.DataFrame, callback: OpProgressCallback
= no_op_progress_callback) -> pd.DataFrame:
        # Open resource and cache it
        if self.saved_model_preds_dict is None:
            with open_file(self.file_path) as f:
                self.model_pred_df = pd.read_parquet(f)
            self.saved_model_preds_dict =
dict(zip(self.model_pred_df.context_uid, self.model_pred_df.preds))

        # Map predictions based on the context_uid
        df["ext_model_predictions"] =
df['context_uid'].map(self.saved_model_preds_dict)
        return df
```

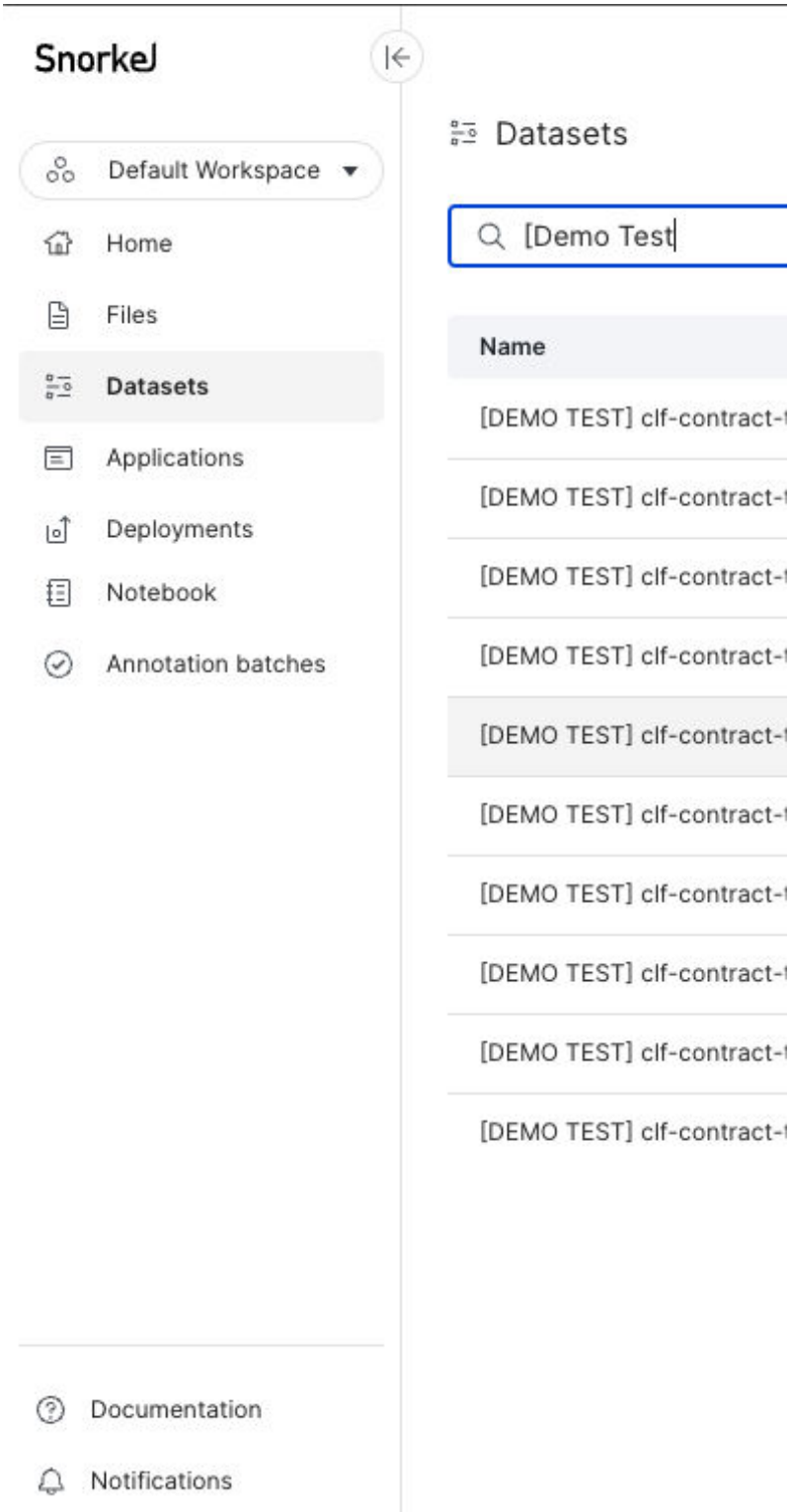
Copyright © 2025 Snorkel AI, Inc. All rights reserved

Re-split data

Sometimes, new data can be added to a dataset, or the data distribution can change in an application. When this happens, we may want to resplit the dataset in an application to move ground truth to different splits or to put new data into different splits. This creates a new application, copying everything from the previous application, and links the new dataset to the new application.

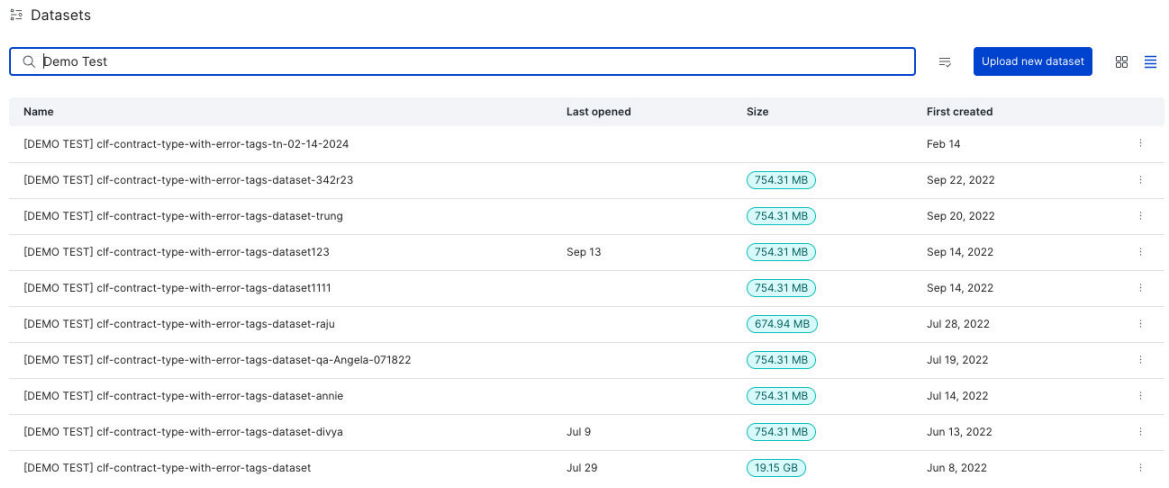
- Split Size: While split size is problem dependent, the default split percentages are 70/20/10 (`train`/`valid`/`test`).
- Usage:
    1. Select `Datasets` menu from the sidebar nav

Copyright © 2025 Snorkel AI, Inc. All rights reserved

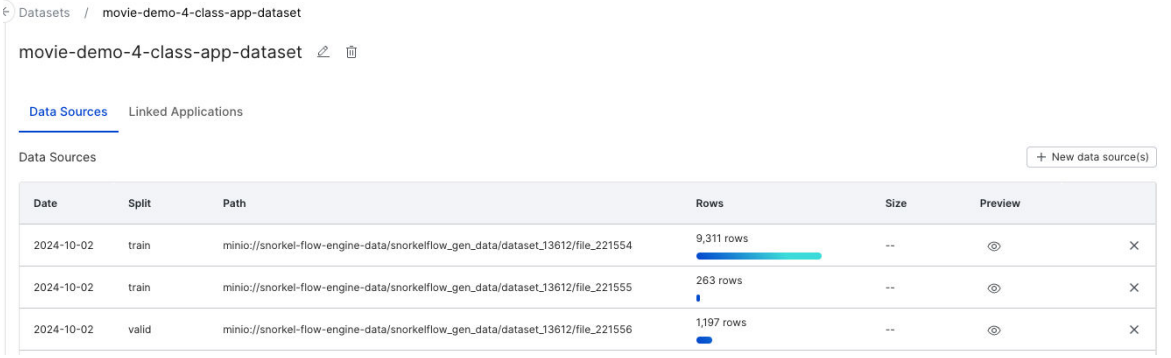


2. Select the dataset you want to re-split data on

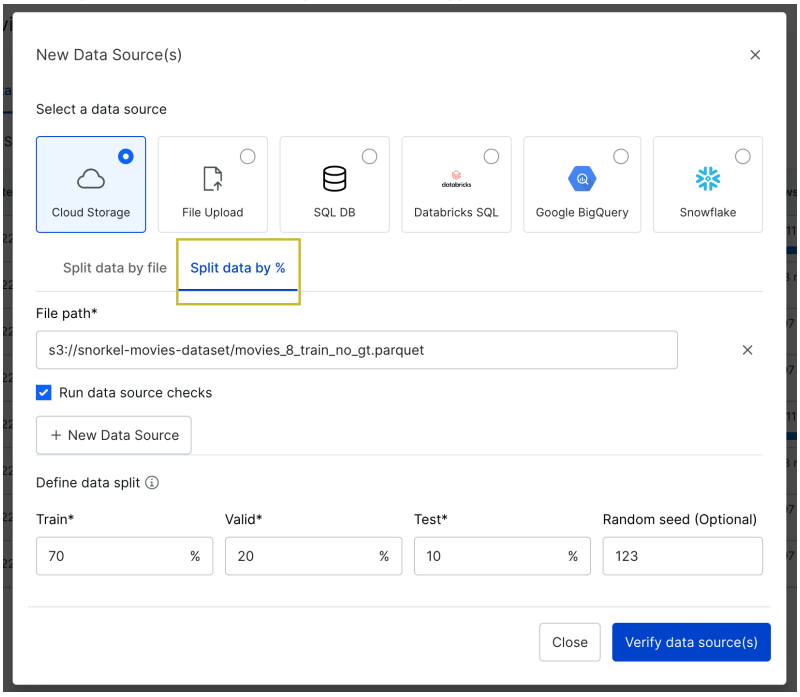
Copyright © 2025 Snorkel AI, Inc. All rights reserved



3. Click on the `New Data Source(s)` button



4. Select your data source upload strategy (cloud, file, Snowflake, etc.) and then click on `Split by %`



5. You'll see that there are defaults already set for `train`/`valid`/`test` but you can change them as per your requirement

Copyright © 2025 Snorkel AI, Inc. All rights reserved

- **SDK:** Snorkel Flow SDK supports resplitting data with the following functionality: `sf.resplit_datsources_by_percent(application_uid, datasource_uids)`. Additionally, an optional `split_random_seed` parameter, as well as an optional `split_pct` parameter, can be input to further specify a random seed or override the default split size. An example call might look like the following:

```
sf.create_new_application_with_resplit_datasources(
    application_uid=application_info["application_uid"],
    datasource_uids=datasource_uids,
    split_pct=json.loads(SplitWiseDistribution(
                          train=train_percent,
                          test=test_percent,
                          valid=valid_percent).json()))
```

Copyright © 2025 Snorkel AI, Inc. All rights reserved

Tips for splitting and partitioning data

When working with a new dataset, one of the most important steps is to create three representative splits of data.

- `train`: for LF development and training set creation
- `valid`: for model hyperparameter tuning
- `test`: for final evaluation

Given the nature of the task and the particularities of the data, it may not be appropriate to randomly split the data across these partitions.

General tips

- **Split size:** While split size is problem-dependent, a good starting breakdown is around 70/15/15 (`train`/`valid`/`test`).
- **Test split**: Make sure your `test` split contains the most reliable ground truth labels compared to other splits.
- **Hierarchical Data**: Create splits based on the hierarchical nature of your data. For example, many text extraction tasks are done at the "page level" but each page may correspond to a given document. In this case, your should split by a unique document identifier, and all of its individual pages in a given split.
- **Diverse data**: If the diversity of data is high, ensure that each split, to the degree possible, contains a representative "slice" of data from each source of diversity (i.e. stratified sampling). For example, if you want to extract fields from PDF reports, and you have noticed a dozen or so different report styles that account for 80% of the data. Ensure that each split contains some degree of each report style across `train`, `valid`, and `test`. While this is important for ensuring models can generalize, it's more critical to account for hierarchical sources of data leakage.

Snorkel Flow specific tips

These tips are relevant when uploading a dataset.

- **Dev split:** Snorkel Flow automatically generates a `dev` split from the `train` split. The default size is 10% of the total `train` size and has a hard limit of 10k examples (or 4k when sampling by documents for extraction tasks). When resampling a `dev` split close to the total dataset size, the system always reserves at least one sample for training. Because Snorkel does not support manually selecting your `dev` split, think carefully about how this could potentially cause data leakage issues. For example, if data from multiple documents are split by pages and `dev` is a random sample of pages from `train`.
- **Ensure each datasource file is < 100 MB:** If you have one large file, please repartition the file into several smaller files and ensure each file is **<100 MB** in size to reduce overhead.
- **Discrete Splits for Slices**: Consider creating multiple files for `valid` and `test` splits for specific "slices" of data, depending on the nature of your data or application. Snorkel Flow supports uploading and enabling/disabling multiple data sources per split, which is useful for quickly assessing model performance on a given segment or slice of data. What you choose to include in these slices will depend on the specific nature of each application.

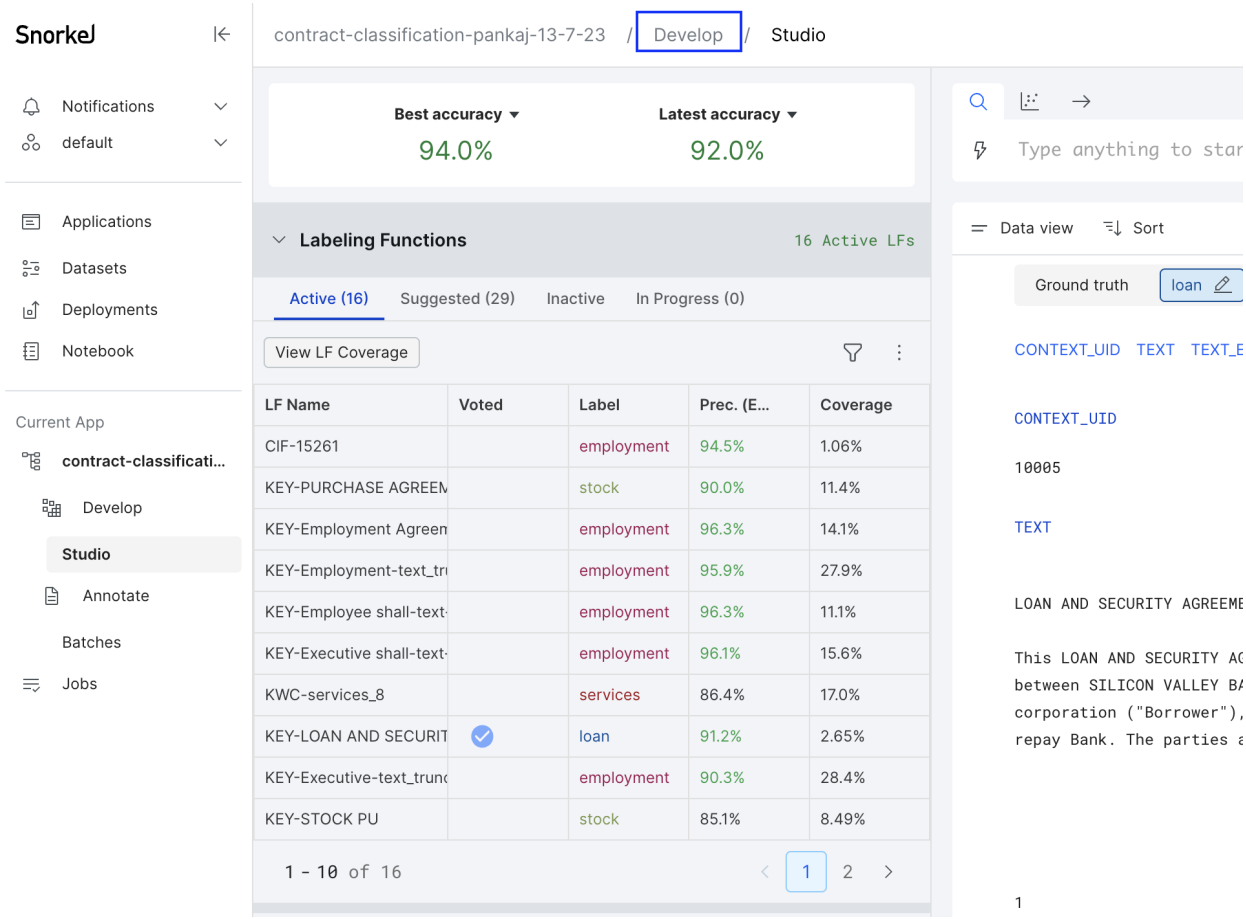
Copyright © 2025 Snorkel AI, Inc. All rights reserved

Training set overview: Review your training sets

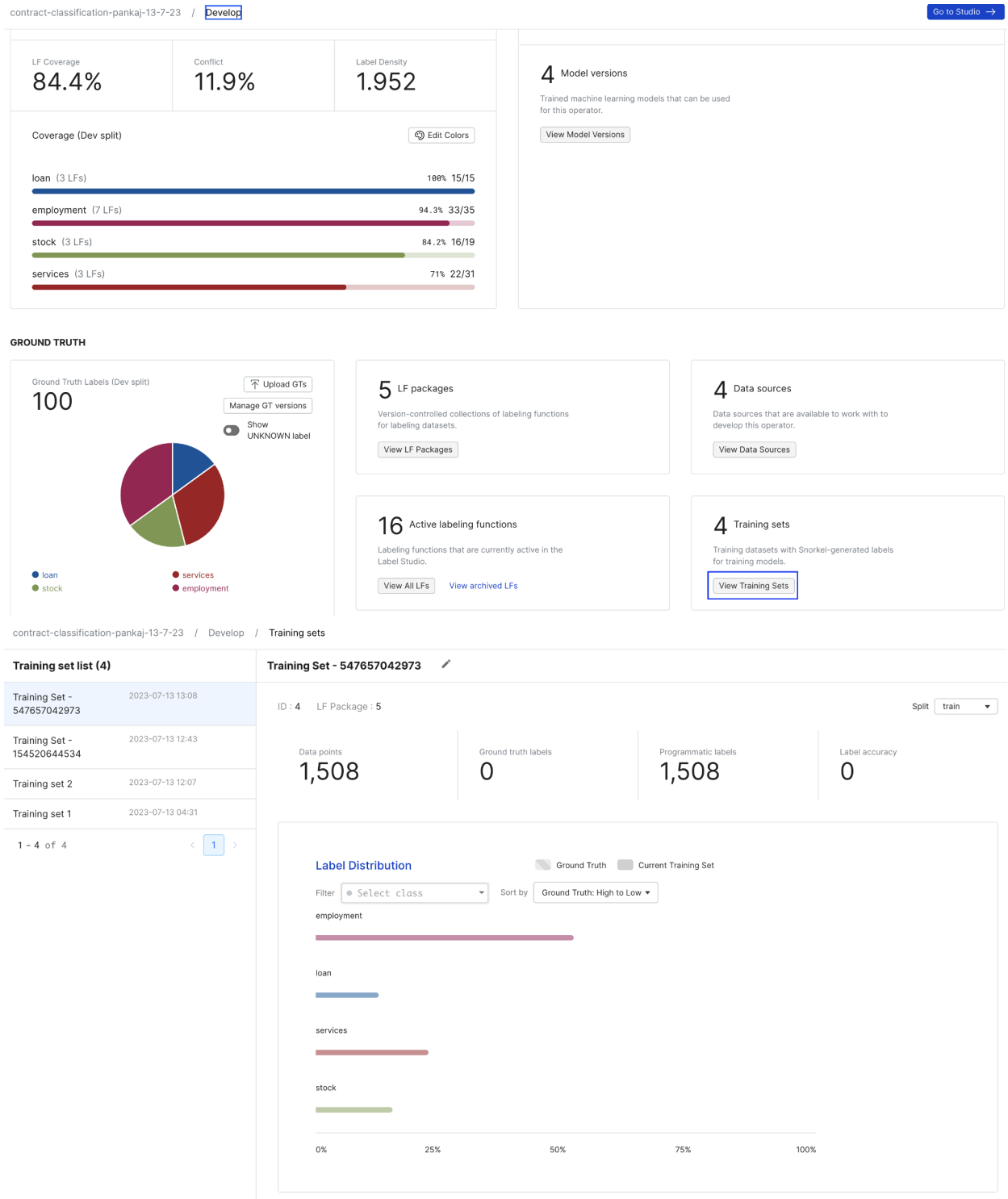
This page shows you a list of the training sets that have been created from your LF packages. For a given training set, you can edit its name and view summary statistics for its splits, including the number of data points, number of ground truth labels, number of programmatic labels, the programmatic label accuracy, and the label distribution.

Navigate to training set view

To navigate to the Training set view, open **Overview** > **Training sets**



Copyright © 2025 Snorkel AI, Inc. All rights reserved



Copyright © 2025 Snorkel AI, Inc. All rights reserved

Data format for different ML tasks

For each Application type in Snorkel Flow, there are **required** fields that Datasets must have depending on the Application type chosen for a use case. An error will appear if these fields are missing.

This document outlines those required fields that a Dataset must have for each Application type in Snorkel Flow.

Text

`Text` use cases in Snorkel Flow are defined as any use case working with raw text strings. Text data does not include PDFs, images, or any other file types not containing raw text strings.

Text use cases currently supported in Snorkel Flow are:

- `Classification`
- `Information extraction`
- `Sequence tagging`

Datasets for Text applications **only require a field (string) that contains text for each data point**. This is what will be annotated and predicted on by the ML model. **There are no requirements for the naming of the field**, and you will specify which field you want to use during application creation.

Due to this flexible requirement, any file type can be treated as a Text use case in Snorkel Flow as long as it is converted into the required format laid out in this section prior to uploading to a Dataset.

Once your data is in the correct format you can upload a Dataset.

Example table with required fields:

text_column *(can have any name)*
The quick brown fox jumps over the lazy dog
...

PDF classification and PDF information extraction

PDF Datasets **require a manifest file with a field (string) containing URLs that point to the location of each PDF**. These URLs can point to either MinIO or S3. **There are no requirements for the naming of the field**, and you will specify which field you want to use during application creation.

This manifest file can be created manually, or there are two options to create the manifest file within Snorkel Flow:

1. ***(Recommended)*** If the PDFs are stored in the user files section in Datasets, then a manifest file will automatically be created which can be downloaded and used directly in creating the Dataset. See our guide on uploading user files.

2. There is an SDK option if PDF files are first uploaded to MinIO. Using the `snorkelflow.ingest.docs.dirtree_to_parquet()` SDK helper function this will create a manifest file that can be downloaded from MinIO and used directly in creating the Dataset.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

Scanned PDFs

Scanned PDFs do not contain parseable text data and need to go through an OCR (optical character recognition), therefore, they will additionally **require a field (string) which contains a representation of the PDF data in hOCR format. It must be named 'hocr'**. This field can be optionally added manually prior to Dataset upload, or it can be created in the Application afterwards using one of Snorkel's built-in OCR preprocessors. Read more about working with Scanned PDFs in Snorkel Flow.

Once your data is in the correct format you can upload a Dataset.

Example table of a manifest file with the required fields:

url_col *(can have any name)*	hocr *(optional for scanned PDFs)*
minio://example_bucket/documents/report.pdf	[hocr data]
...	...

Computer vision

Computer vision Datasets **require a manifest file with a field (string) containing URLs that point to the location of each image**. These URLs can be either to MinIO or S3. **There are no requirements for the naming of the field**. The images must be PNG or JPEG and cannot be larger than 512 pixels.

This manifest file can be created manually, or there are two options to create the manifest file within Snorkel Flow:

1. If the images are stored in the user files section in Datasets, then a manifest file will automatically be created which can be downloaded and used directly in creating the Dataset. See our guide on uploading user files.
2. There is an SDK option using the `upload_images_to_MinIO` function which automatically:
   - validates whether images exist and can be opened
   - resizes images to a maximum resolution of 512 pixels
   - converts images to one of the supported image formats
   - uploads images to MinIO
   - outputs the corresponding MinIO image paths

Once your data is in the correct format you can upload a Dataset.

Example table of a manifest file with the required fields:

url_col *(can have any name)*
minio://example_bucket/images/photo1.png
...

Copyright © 2025 Snorkel AI, Inc. All rights reserved

Ground truth formats for different ML tasks

This document serves as a reference for the ground truth (GT) formats for different ML task types in Snorkel Flow. Use this guide before uploading your data to ensure you know the correct format needed for adding labels to your dataset. Proper formatting is crucial for the successful integration of your data within the Snorkel Flow platform.

Refer to the sections below for specific details on the GT format for each ML task type.

Text classification / PDF classification

For text and PDF classification tasks, each document requires a ground truth (GT) label at the document level. Each document has a single label. In this example, documents receive the label of `POSITIVE` or `NEGATIVE` sentiment.

You can provide the document-level GT as part of your data source file, which includes a column specifying the GT label for each row of data. When creating your application, make sure to set the `Ground truth column` to match the name of the label column in your data file.

Alternatively, you can upload the document-level GT directly from the `Datasource` upload page of your application or model. To do this:

1. Click the **Overview** menu on the left side of the Studio page.
2. In the Ground Truths section, click the Upload GTs button.
3. Provide the remote path of the GT file or select the GT file from your local machine.

The uploaded GT file must contain the following columns:

- UID column: The unique identifier for each document, in the format `doc::1`.
- Label column: The GT label for each document.

Below is an example ground truth for four documents:

uid	label
doc::1	POSITIVE
doc::2	NEGATIVE
doc::3	POSITIVE
doc::4	POSITIVE

Multi-label text classification

For multi-label text classification tasks, the ground truth (GT) for a document is represented as a JSON object. This JSON object maps each label to a corresponding label marker, which can be one of the following:

- `PRESENT`: Indicates the document is positively labeled for the given label.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

- `ABSENT`: Indicates the document is negatively labeled for the given label.
- `ABSTAIN`: Indicates the document is neither positively nor negatively labeled for the given label. This option is useful when the ground truth for a specific label is not known at the time of labeling.

Additionally, a special label, `_default`, can be used to represent missing labels in the mapping. This serves as a catch-all for any labels not explicitly specified, helping to reduce the memory required to store the ground truth.

For example, the following example of label markers

```
{
    "class_1" : "PRESENT",
    "class_2" : "ABSTAIN",
    "class_3" : "ABSENT",
    "class_4" : "ABSENT",
    "class_5" : "ABSENT"
}
```

is equivalent to:

```
{
   "class_1" : "PRESENT",
   "class_2" : "ABSTAIN",
   "_default" : "ABSENT"
}
```

By using `_default`, you can reduce the need to repeat the same value for multiple labels, making the JSON object more compact and memory-efficient.

Information extraction

For information extraction tasks, ground truth (GT) is provided at the span level, where each extracted span is associated with a specific label. The span-level GT can only be uploaded via the `Datasource` upload page for the relevant Application or Model. To do this:

1. Click **Overview** on the left-hand menu.
2. In the Ground Truths section, click the Upload GTs button.

The uploaded GT file must contain the following columns:

- `context_uid` (int): The unique identifier of the document from which the span was extracted. All spans with the same `context_uid` come from the same document. This is linked to the `UID column` specified when creating the dataset.
- `span_field` (str): The field name from which the spans were extracted.
- `char_start` (int): The index of the first character of the span in the document.
- `char_end` (int): The index of the last character of the span in the document.
- `_gt_label` (str): The ground truth label for the span.

Below is an example of ground truth for a single document with `context_uid=0`.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

_gt_label	context_uid	span_field	char_start	char_end
class_1	0	text	17	28
class_2	0	text	37	46

> NOTE
> Only spans that match the spans extracted by the candidate extractor can be recognized as ground truth. The GT file must have identical values for `context_uid`, `char_start`, and `char_end` as the candidate extractor's output. A GT span will not be used if its values do not exactly match to any of the extracted spans.

To verify which GT spans are included in the application after upload, you can use the `sf.export_ground_truth` SDK method.

For optimal results, you should iterate on the extracted ground truth until the recall is as close to 100% as possible. You can evaluate the candidate extractor's performance using the `sf.get_candidate_extractor_metrics` SDK method.

PDF information extraction

For PDF information extraction tasks, the span-level ground truth (GT) can only be uploaded via the Datasource upload page for the relevant Application or Model. To do this:

1. Click **Overview** on the left-hand menu.
2. In the Ground Truths section, click the Upload GTs button.

The uploaded GT file must contain the following columns:

uid	label
span::17,2,rich_doc_text,6e85bf3f0698497465102d9104bfb4fe,412,442	LABEL_VALUE

- UID column: The unique identifier (UID) of the span. This UID must match the data point UIDs exactly.
  - Example UID: `span::17,2,rich_doc_text,6e85bf3f0698497465102d9104bfb4fe,412,442` where:
    - `17` is the `context_uid` (int): The UID of the document from which the span was extracted.
    - `2` is the `page_id` (int): The page number within the document. This field is included if you select Split docs by page when creating a PDF application or manually add a Page splitter. Otherwise, it should be omitted, along with the associated comma (e.g., span::17,rich_doc_text,6e85bf3f0698497465102d9104bfb4fe,412,442).
    - `rich_doc_text` (str): the internal name of the field where the spans were extracted from.
    - `6e85bf3f0698497465102d9104bfb4fe` is the `span_field_hash_value` (str): the hash of the value of the field that the span is extracted from. Can be computed by the python code `hashlib.md5(span_field_value.encode("utf-8")).hexdigest()`.
    - `412` is the `char_start` (int): the index of the first character of the span in the document.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

    - `442` is the `char_end` (int): the index of the last character of the span in the document.
- Label column: The GT label of the document.

Ensure that all these elements are correctly formatted and included in your GT file to ensure proper processing and accurate extraction results.

Conversational AI

For Conversational AI, the ground truth (GT) can be added to the data source files within the field corresponding to metadata. The expected data format is a JSON file containing a list of objects, where each object represents a conversation. Each conversation object contains a list of sub-objects, where each sub-object represents an individual utterance.

Each conversation dictionary should include the following:

- Field: The string key under which list of utterances can be found.
- Utterances Path: The path to actual utterances if the input JSON has a nested path.

Each utterance dictionary should include:

- Speaker field: The string key for speaker of each utterance.
- Text field: The string content of the utterance.
- Metadata field: The dictionary potentially containing "GT".

Below is an example of the expected JSON format:

Copyright © 2025 Snorkel AI, Inc. All rights reserved

```
[
  {
    "turns": [
      {
        "speaker": "USER",
        "utterance": "I want to transfer $500 to XYZ.",
        "frames": {
          "GT": 0
        }
      },
      {
        "speaker": "SYSTEM",
        "utterance": "Okay your money was transferred.",
      }
    ]
  },
  {
    "turns": [
      {
        "speaker": "USER",
        "utterance": "I want to check my balance.",
        "frames": {
          "GT": 1
        }
      },
      {
        "speaker": "SYSTEM",
        "utterance": "Your balance is $100.",
      }
    ]
  },
]
```

In this example

- Field: "turns" – The key where the list of utterances is stored.
- Utterances Path: None – No nested path is needed in this example.
- Speaker Field: "speaker" – The key indicating the speaker of each utterance.
- Text Field: "utterance" – The key containing the text of each utterance.
- Metadata Field: "frames" – This field contains the ground truth, indicated by the key "GT".

After creating your dataset and application, you can import ground truth data using the SDK function `import_utterance_ground_truth`.

Sequence tagging

For sequence tagging tasks, the ground truth (GT) for a document is a JSON representation of a list of spans. Each span is represented as a 3-tuple of `(char_start, char_end, label)` consisting of:

Copyright © 2025 Snorkel AI, Inc. All rights reserved

- `char_start`: The start index of the span.
- `char_end`: The end index of the span.
- `label`: The label associated with the span.

Below is an example of a ground truth for a document when tagging the spans as `COMPANY` or `OTHER`:

```
[
    [0, 29, 'OTHER'],
    [29, 40, 'COMPANY'],
    [40, 228, 'OTHER'],
    [228, 239, 'COMPANY'],
    [239, 395, 'OTHER'],
]
```

Ensure that the following conditions are met:

- Spans cannot be empty (i.e., `char_start` must be less than `char_end`).
- Overlapping or duplicate spans are not allowed.
- The sets of character offsets (`char_start`, `char_end`) should be sorted in ascending order.

By default, the `AsciiCharFilter` preprocessor is added in the DAG, and filters out non-ASCII characters from documents. If your ground truth data was collected outside of SnorkelFlow, use the SDK function `align_external_ground_truth` to align it before ingestion into SnorkelFlow. To upload the ground truth file:

1. Click Overview on the left-hand menu.
2. In the Ground Truths section, click the Upload GTs button.
3. Provide the following information to import the ground truth from a file:

- `File path`: `s3://path_to_your_file.csv`
- `File format`: `CSV`
- `Label column`: `label` - The column containing the label.
- `UID column`: `x_uid` - The column containing the document UID, in the format doc::2005.

If your external ground truth data does not include negative ground truth labels, select the `Auto generate negative labels` option on the Upload GTs page. This option is also available during application creation. Alternatively, use the SDK function `add_ground_truth` to infer negative labels.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

Cold start scenarios

Starting a Snorkel Flow application without pre-labeled data is known as a "cold start." Snorkel Flow's features make cold start data development more approachable.

In this article you will learn practical strategies to successfully develop your data from a cold start, such as:

- Manual labeling
- Leveraging external resources like knowledge graphs
- Using existing models as labeling functions

Read through the following cold start approaches to see which ones fit your data.

Warming Up your Cold Start in Snorkel Flow

Sometimes, a cold start isn't as cold as it looks. While the data set itself may be raw and unlabeled, existing structures can help kick-start the process.

Users can leverage existing external knowledge sources, such as ontologies, taxonomies, or knowledge graphs, to derive label categories or relationships between entities. This can help to bootstrap the labeling process and reduce the amount of manual effort required.

Existing internal resources can help. If the organization has a mechanism for applying labels to this data set—be it a rules-based system or a model that doesn't perform up to the desired accuracy—they can use that as a starting point. Snorkel Flow accepts existing models and systems as labeling functions.

Snorkel Flow cold start blocker: Schema or ground truth?

Cold start problems typically lack one of two types of metadata:

- The data lacks a ground truth
- The data lacks a schema

Let's address each.

How to tackle a cold start when ground truth is missing:

Starting a Snorkel application without any labeled data can be a challenging task. While having a label schema defined helps, it's still necessary to generate labeled data to train a model.

The simplest solution to this problem is to label some ground truth data using Snorkel Flow's annotation suite. Subject matter experts within your organization can set aside a few hours to pull up examples, apply labels and use the comment feature to explain why they chose a particular label; data scientists can later use those comments to formulate labeling functions.

Your team can also tackle this problem in the reverse direction. Instead of asking subject matter experts to label randomly-extracted data in the annotation suite, data scientists can define an initial set of labeling functions—one per label in the schema—and then ask subject matter experts to verify or invalidate the applied labels from a random sample. This approach helps ensure that you label a sufficient number of examples per class.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

Addressing the challenge of no label schema

In some cases, users may not have a well-defined label schema when starting a Snorkel Flow application. This can be due to the complexity of the task or the lack of prior domain knowledge. For example, an online retailer wants to present optimized versions of its home page based on the kind of visitor seeing it, but they don't know how to categorize those visitors.

An organization can kick off the process by gathering stakeholders with domain knowledge to brainstorm and outline relevant attributes. From there, define your classes: likely buyers and likely browsers, for example. These categories should be collectively exhaustive, meaning that every example can be assigned to at least one category. In our online retailer example, the visitors should belong to only one category, but some problems require a multi-class approach.

The above ignores a potential blocker: despite their expertise, your stakeholders may not have a deep enough understanding of the data to know what buckets are appropriate. That's where machine learning may help. Unsupervised learning techniques, such as clustering or topic modeling, can help group examples into meaningful categories. These categories can then be used as your initial labels.

Once your team agrees on a set of labels—whether through raw human intellectual power or with the assistance of unsupervised learning—they can begin the process of building ground truth, as described in the previous section.

Refine and iterate

Once your team has decided on a set of labels and begins building out your probabilistic data set with labeling functions and weak supervision in Snorkel Flow, your schema may change. That's okay. Snorkel Flow is built to handle that.

Conclusion

Snorkel Flow provides tools to capture and build the initial metadata for your data, whether that metadata comes from existing resources, human expertise, or unsupervised learning.

From a cold start, Snorkel Flow helps you expand and refine your labeling process with confidence and quick iteration. As you continue the data development process, you'll reach the point where you can deploy your new machine learning application.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

Create application: A guided flow

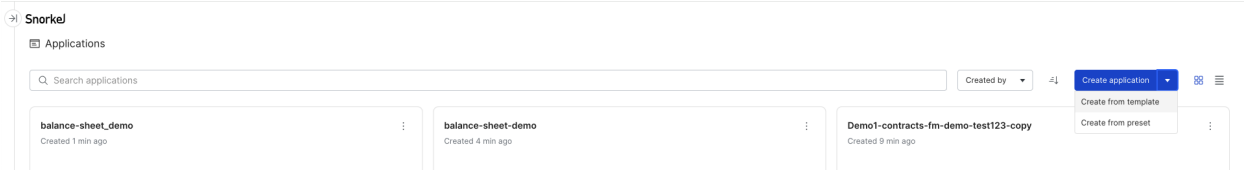
This page walks the process of creating a Snorkel Flow application. The guided experience currently supports the following data and task types:

- Raw text classification
- Raw text extraction (candidate-based or sequence tagging)
- PDF classification
- PDF extraction (candidate-based)
- (Beta) PDF extraction (word-based)

> **NOTE**
>
> This guided experience is best suited for **single-task applications**. To create an application that involves multiple model nodes, custom preprocessing modules, and/or a hierarchical structure, choose the legacy Create from template option in the dropdown.

Entry point

In the left-side menu, click **Applications**, then click the **Create application** button.

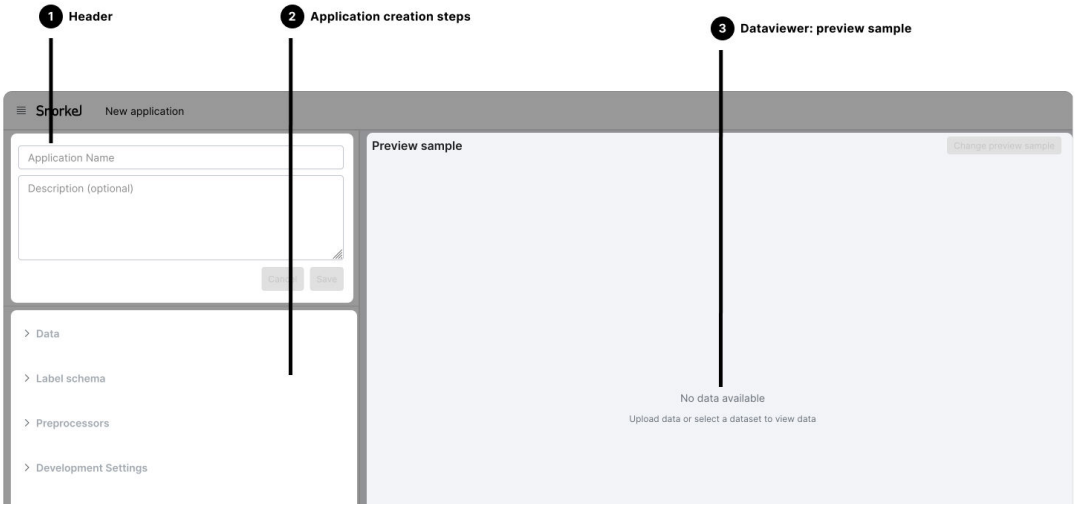


Page overview

The application creation page includes three main sections:

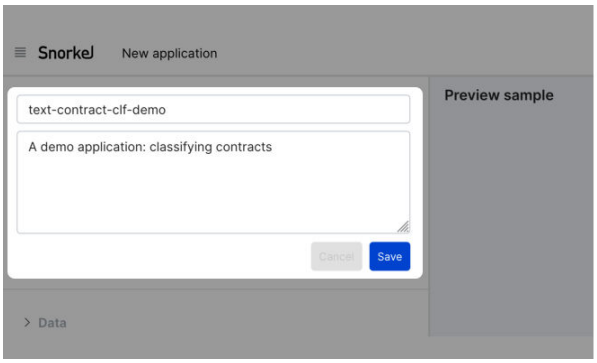
1. **Header**: add an application name (required) and description (optional)
2. **Application creation steps**: expand the accordions to provide the required inputs
   - Data
   - Label schema
   - Preprocessors
   - Development Settings
3. **Dataviewer: preview sample**: view a small sample (20 data points) of the dataset and if applicable, see data changes per preprocessor settings

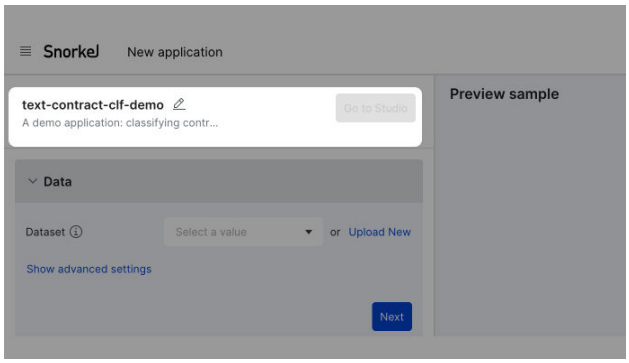
Copyright © 2025 Snorkel AI, Inc. All rights reserved



Step 1: Add application name and description

On the top-left of the page (the **Header** section), enter an **Application Name** and optionally a short **Description** of the purpose of the application. Click **Save** to move on to Step 2: Define Data. Click the **pencil** icon next to the application name to edit the name or description at any time.

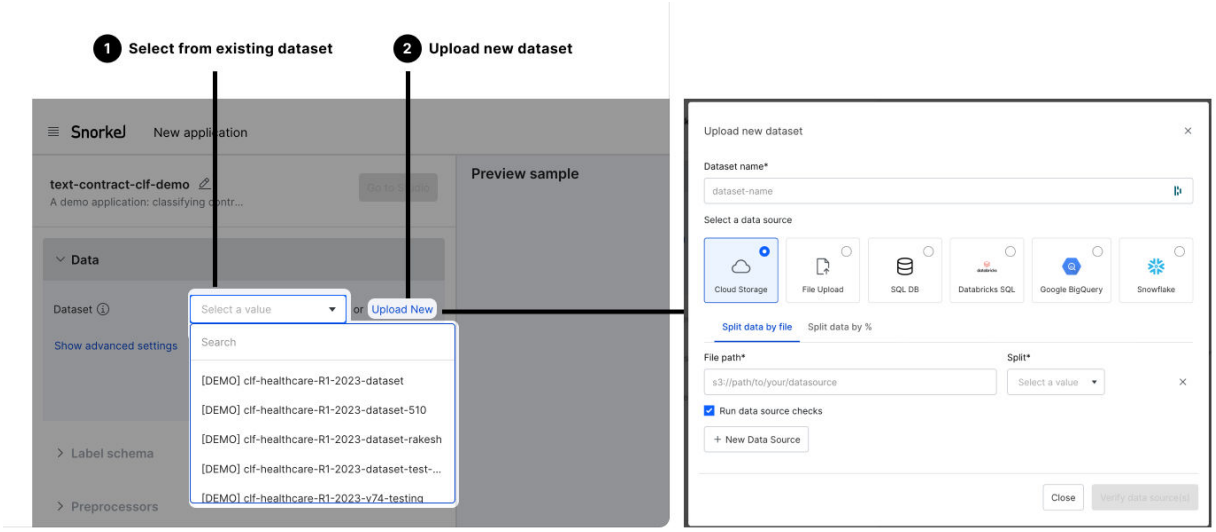




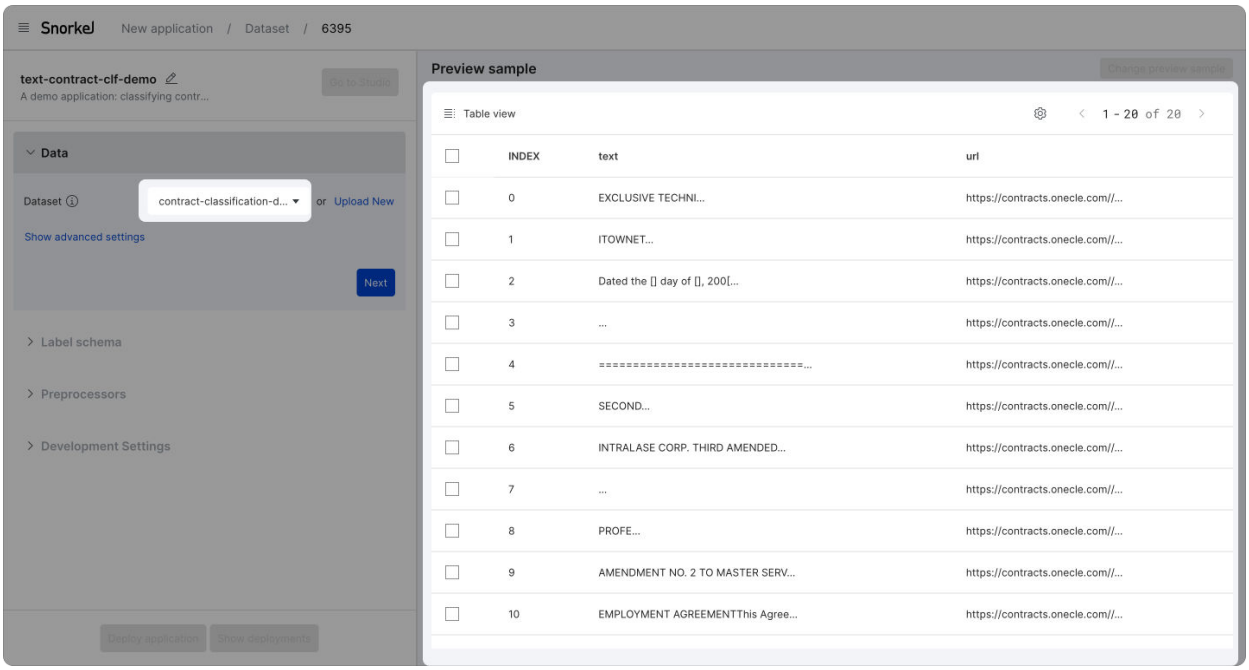
Step 2: Define data

In the **Data** accordion, select the dataset that you wish to use for this application. You can choose from existing datasets in your workspace in the dropdown, or click **upload new** to upload a new dataset (see Data upload for more information).

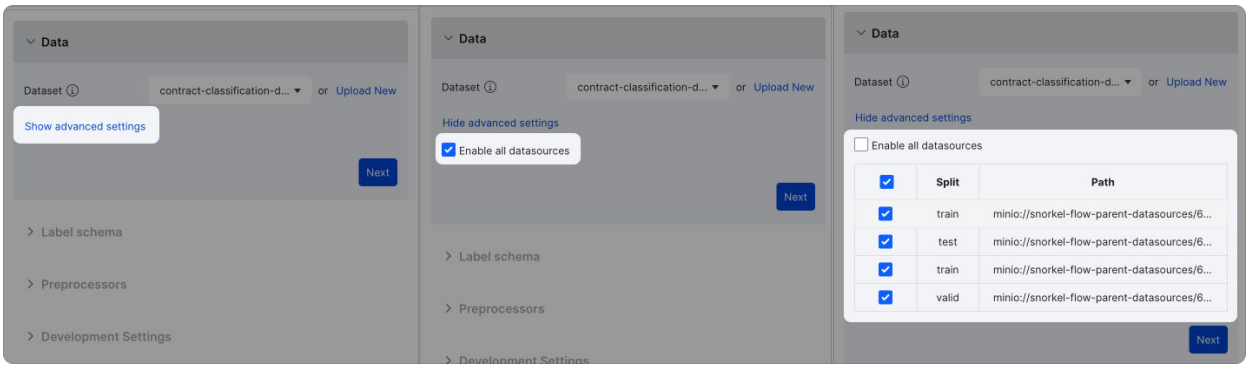
Copyright © 2025 Snorkel AI, Inc. All rights reserved



Once a dataset is selected, 20 data samples from your dataset will be loaded in the **Preview sample** pane. If you click **Table view**, you then have the option to view the sample data in Snippet view or Record view. For more information about the data views that are available, see Dataviewer: Display control pane.



In addition, **Show advanced settings** includes rare operations such as altering data sources (the default is to enable all data source in the selected dataset).



Copyright © 2025 Snorkel AI, Inc. All rights reserved

Confirm that your selections are correct as they cannot be changed later. Then click **Next** to confirm the selections and move on to Step 3: Define label schema. This action also creates and saves the application.

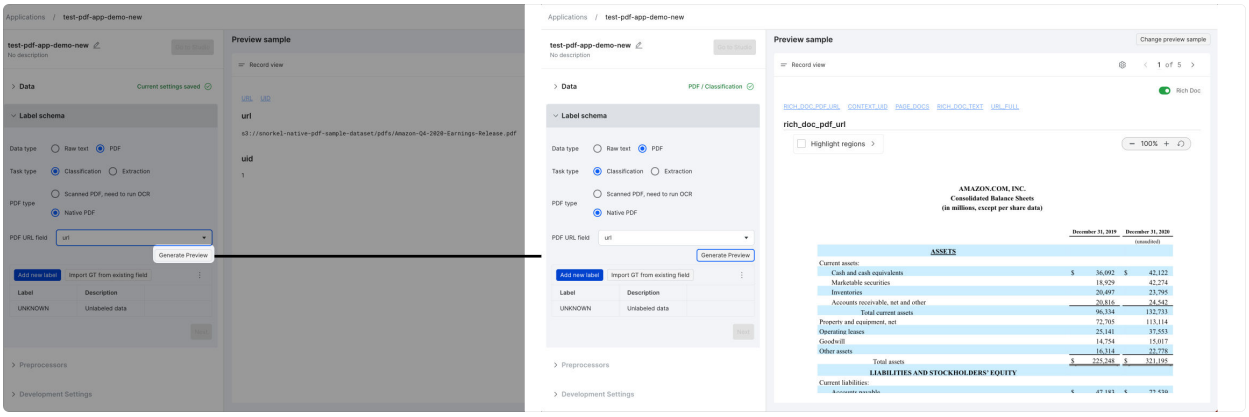
Step 3: Define label schema

In the **Label schema** accordion, first complete the data and task type inputs. Additional options may be required based on your data and task type selection.

Option	Data and/or task type	Description
Primary text field	raw text	The primary text field for labeling.
IE type	raw text extraction	The desired format for information extraction. **Candidate-based**: generating candidate spans using a heuristic (e.g. regex matcher), and then classify each candidate as a target class or "other" (i.e. false positive). **Sequence-tagging**: token-level tagging over spans.
Filter out docs > 10MB	raw text extraction sequence tagging	Select to skip documents larger than 10MB in the dataset to optimize performance.
PDF type	PDF	The nature of PDF data. This selection sets the default preprocessors to run in the Preprocessor step. **Native** PDFs are created digitally and **Scanned** PDFs are typically created from scans of printed documents.
PDF URL field	PDF	The field with a URL link to the PDF file.
hOCR field	PDF - scanned PDF, no need to run OCR	The field with hOCR value. This assumes that you have already ran preprocessed your data with an optical character recognition (OCR) library.
Image URL Field	Image	The field with a URL link to the image file.

If your data type is PDF or Image, you can click **Generate Preview** to view the rendered PDF/image in the Preview sample.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

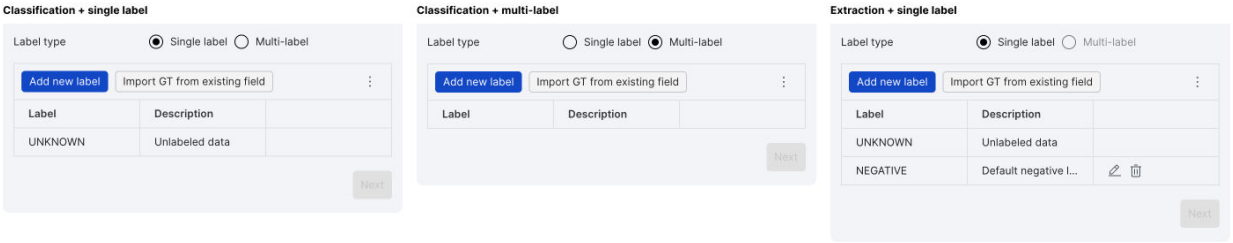


Based on your selections above, you can now complete the following actions:

- Specify label type
- Add new labels
- (Optional) edit or delete labels
- (Optional) import ground truth (GT) from file

Specify label type

Choose either Single label (i.e., the target is a single choice) or Multi-label (i.e., the target is zero or more choices) label type. Upon selection, default labels will be loaded per task and label type setting.



Add new labels

Depending on the task and label type, one or more custom labels are required to complete the initial schema setup (you can edit this later). There are two ways to add labels:

- **Add new label** allows you to add one custom label at a time.
- **Import GT from existing field** allows you to import the unique strings from a field in the dataset.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

(Optional) Edit or delete labels

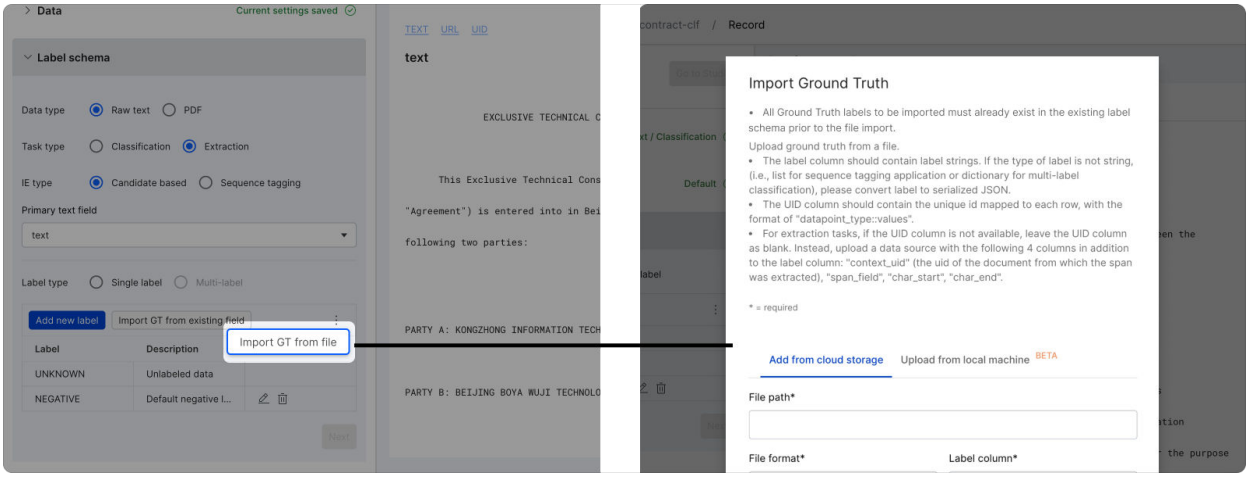
To edit or delete labels:

- Click the **pencil** icon to edit the label name and description.
- Click the **trashcan** icon to delete a custom label. For all data under the deleted label, the Ground Truth value is reassigned to **UNKNOWN**.

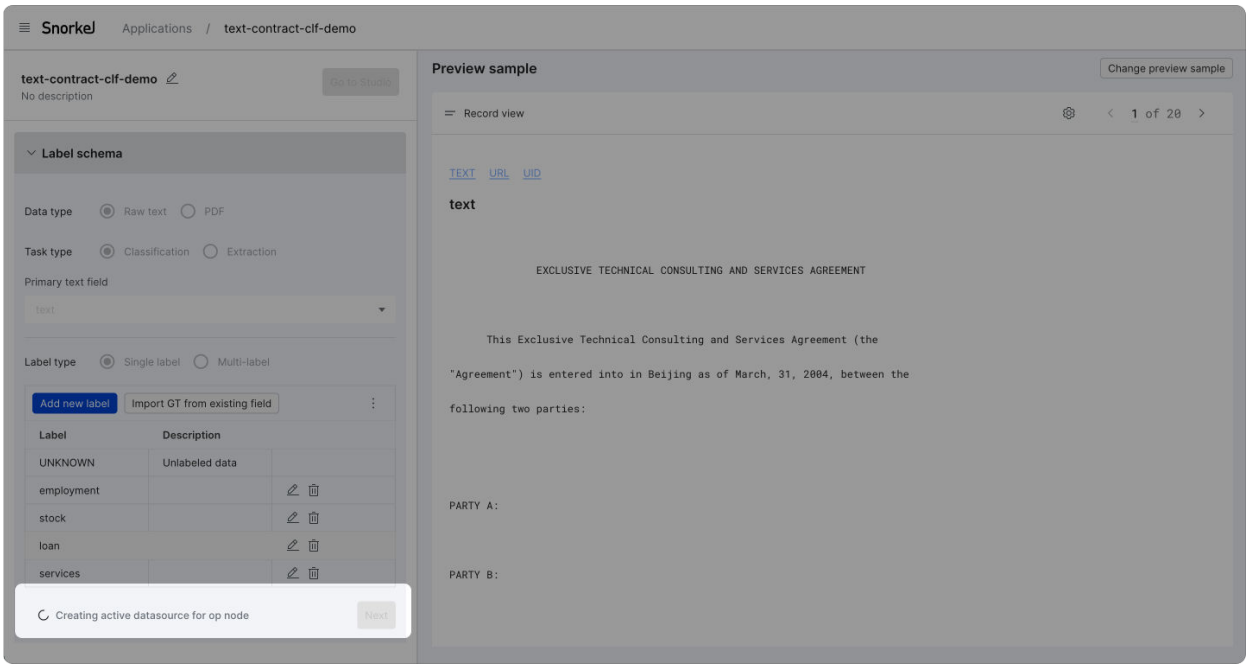
(Optional) import GT from file

Once the label schema is specified, you can import the GT of the dataset to this application by uploading a file that connects the data UID and label names. Click the three dot overflow menu and select **Import GT from file**. In the modal, you can either add the file via **cloud storage** or from a **local machine**.

Copyright © 2025 Snorkel AI, Inc. All rights reserved



Click **Next** to confirm the label schema setting. This will kick off preprocessing on the full dataset (previously preprocessing was done on the preview sample only), which may take a few minutes. Refer to the progress bar at the bottom of the accordion to see the status of the operation.



Upon completion, click **Next** again to move on to next step: Step 4: Set up preprocessors (optional)

Step 4: Set up preprocessors (optional)

Snorkel Flow provides preset preprocessors based on the data type and task type that you selected. If additional user input is not required, then you will be automatically brought to Step 5: Development settings.

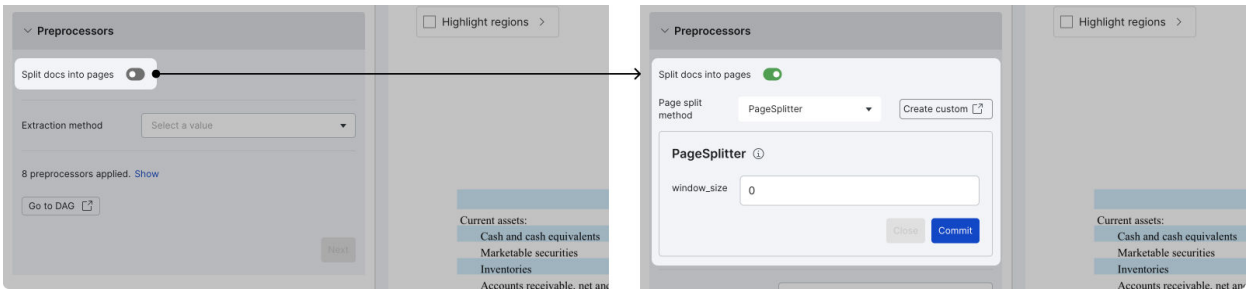
Data & task types	Additional user input required?
Raw text classification	No
Raw text extraction (candidate-based)	Yes: select the extraction method

Copyright © 2025 Snorkel AI, Inc. All rights reserved

Data & task types	Additional user input required?
Raw text extraction (sequence tagging)	No
PDF classification	Yes: option to split docs to pages (highly recommended)
PDF extraction (candidate-based)	Yes: select the extraction method, and option to split docs to pages (highly recommended)
(Beta) PDF extraction (word-based)	Yes: see (Beta) Extraction from PDFs extracting patents

Split docs into pages (PDF-only)

Splitting docs into pages is a highly recommended operation for PDF data types to reduce file size and proceeding iteration time. Upon turning on the option, the default page split method **PageSplitter** is selected. If you want to use a custom method, click **Create custom** and follow the documentation that opens in a new window to setup custom operator in the SDK. Any custom operator that is uploaded to the workspace will be available in the dropdown.

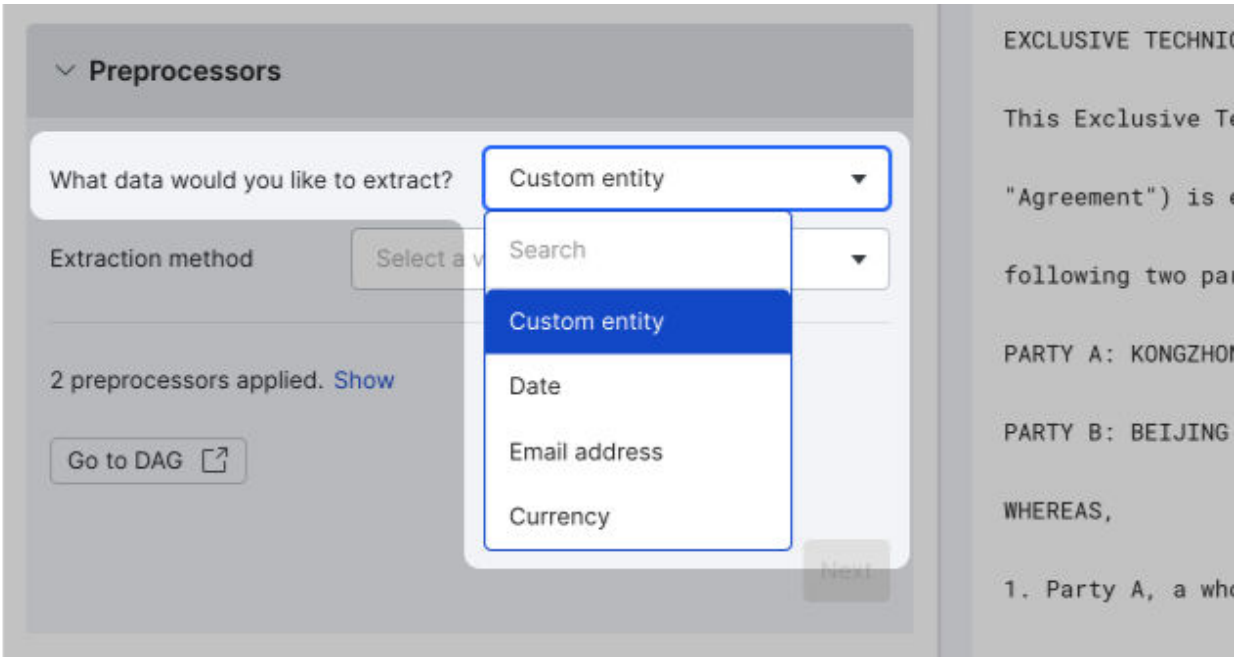


Click **Commit** to confirm the page splitting operation. Note that this step is **currently non-reversible**.

Select an extractor

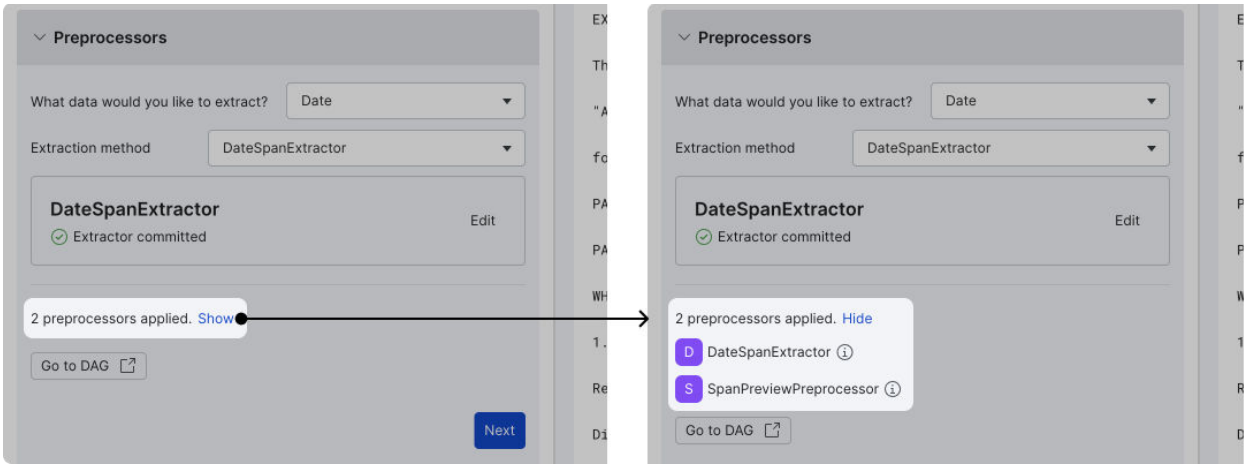
For raw text extraction, in the **What data would you like to extract?** option, choose one of the common data types to auto-select a built-in extractor method, or choose **Custom entity** and then select from Snorkel's full library of extraction methods. Additionally, you can also use custom extractors.

Copyright © 2025 Snorkel AI, Inc. All rights reserved



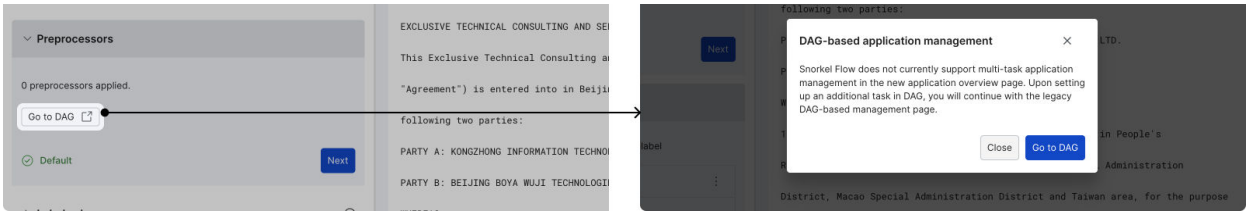
Once the extraction method is selected, fill out the required fields and then click **Commit**. You can change the extractor and its settings later.

After all required fields are completed, you can see the full list of preprocessors by clicking **Show**, which is next to **# preprocessors applied**.



Transition to DAG-first workflow

If you want to perform advanced edits (e.g., add custom preprocessors or add blocks), click **Go to DAG** to perform advanced actions directly in the DAG. Note that this action will transition this application to the legacy template-based setup and is **not reversible** (i.e. you cannot return to the guided interface).



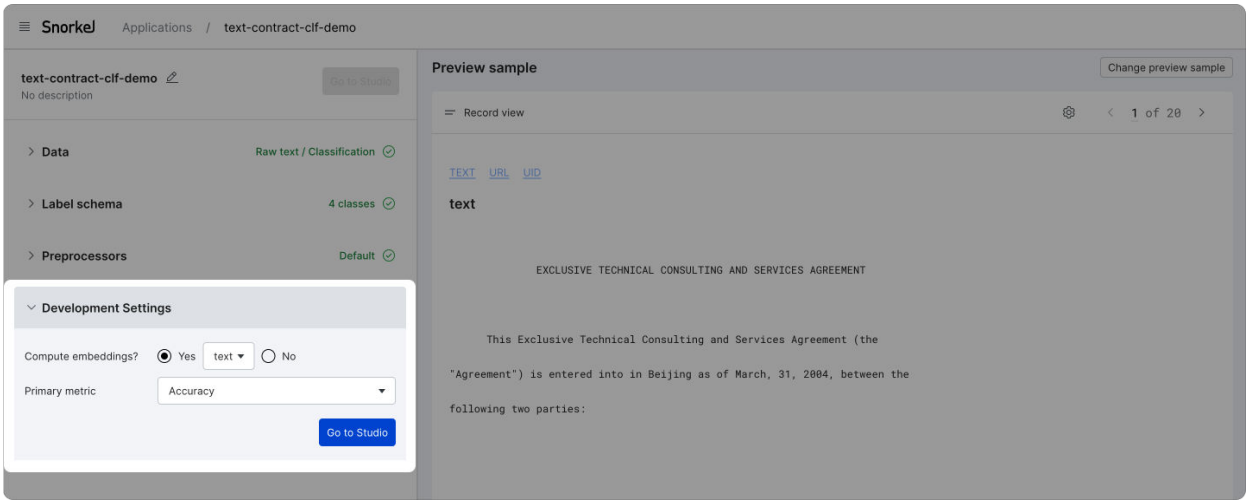
Click **Next** to move on to Step 5: Development settings.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

Step 5: Development settings

In the **Development Settings** accordion, confirm or edit the following default settings:

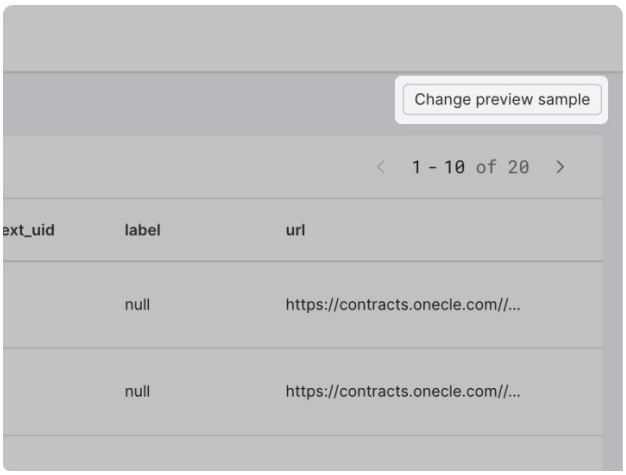
- **Compute embeddings**: The default setting is **Yes**, with the field selection being the primary text field. For more information about using embeddings across your end-to-end workflow, see Utilizing embeddings.
- **Primary metric**: The default metric varies depending on your data and task type selection.

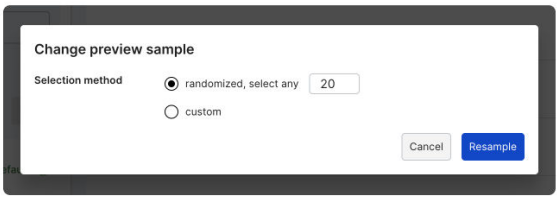


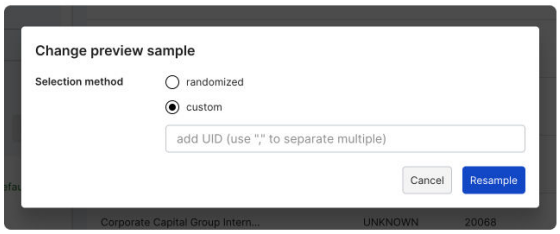
Click **Go to Studio** to complete application setup and start developing your model!

Resample the preview sample

If you want to resample the preview data during the setup process, click **Change preview sample** on the top-right corner of your screen, then select the desired resample method.





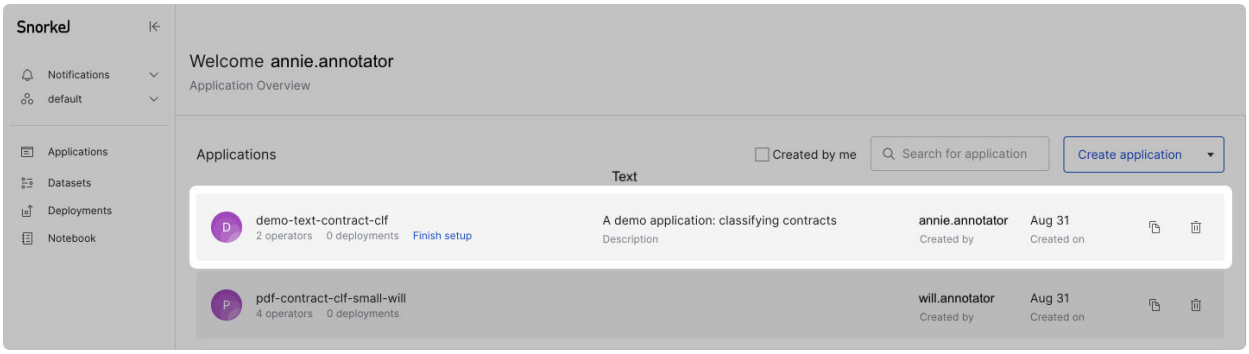


Return to the application setup page

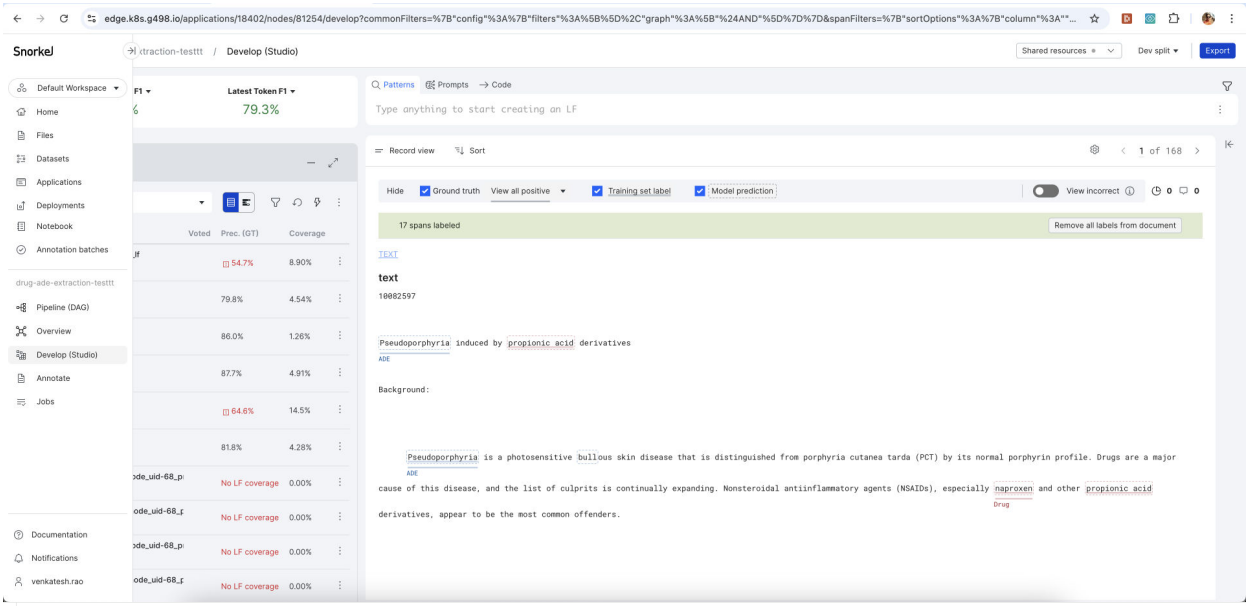
Any initiated application (i.e., passing Step 2: Define data) can be found in the applications list on the **Application** page. You can get to the **Application** page at any time by clicking **Applications** in the left-side menu.

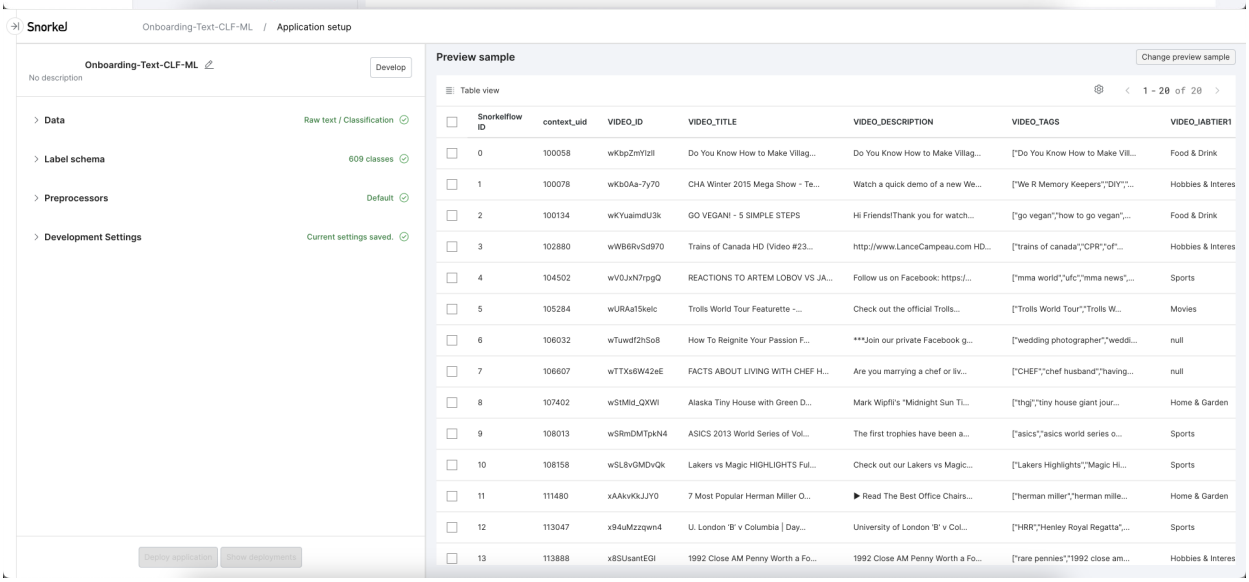
For applications with unfinished setup, clicking the application brings you to the application setup page.

Copyright © 2025 Snorkel AI, Inc. All rights reserved



For applications with completed setup, click the **application name** in the left-side menu to return to the application setup page.





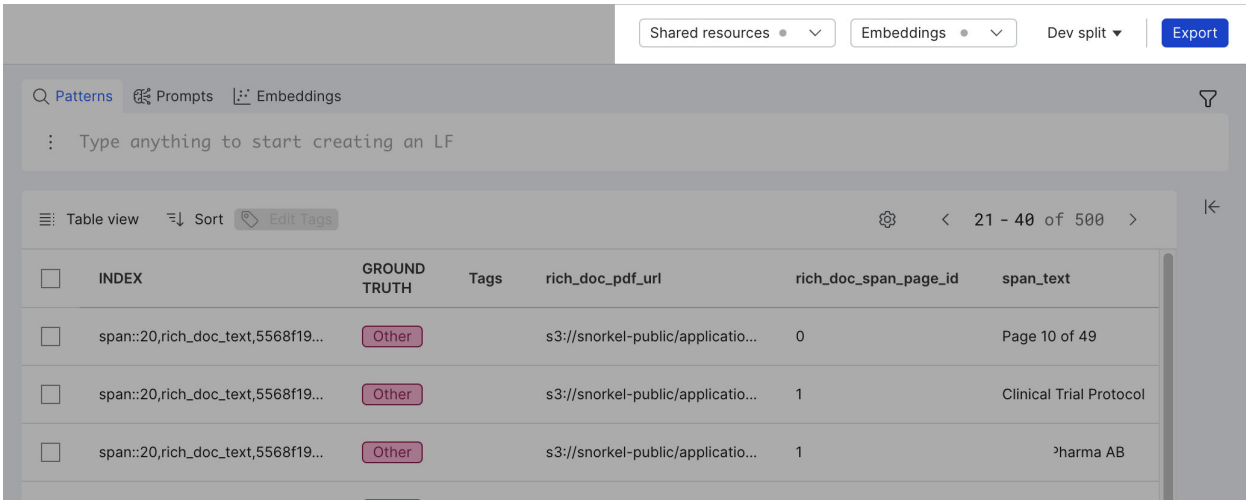
Copyright © 2025 Snorkel AI, Inc. All rights reserved

Application data control pane

This page walks through the functionality and settings of the application data control pane.

The application data control pane includes controls for the following actions:

- Allocate dedicated resources
- Compute embeddings
- Select data splits and resample data
- Export data



Allocate dedicated resources

If you feel that your dataset is particularly large and is slowing down your application, you can click **Shared resources** to control the resources allocated per application. Dedicated resources caches your dataset in memory where it can be fetched and operated on quickly. For more information about dedicated resources, see Using dedicated resources to speed up applications with large datasets.

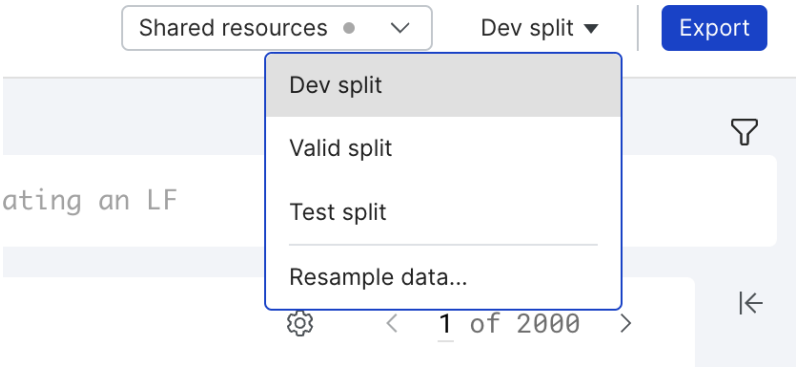
Compute embeddings

If available for your use case, you generate embeddings using SimCSE or RAG. You can use these embeddings to create labeling functions (LFs) during model development. For more information about embeddings, see Utilizing embeddings and Embedding based cluster LFs.

Select data splits and resample data

Use the data splits dropdown to change the active data split or resample data.

Copyright © 2025 Snorkel AI, Inc. All rights reserved



Select split

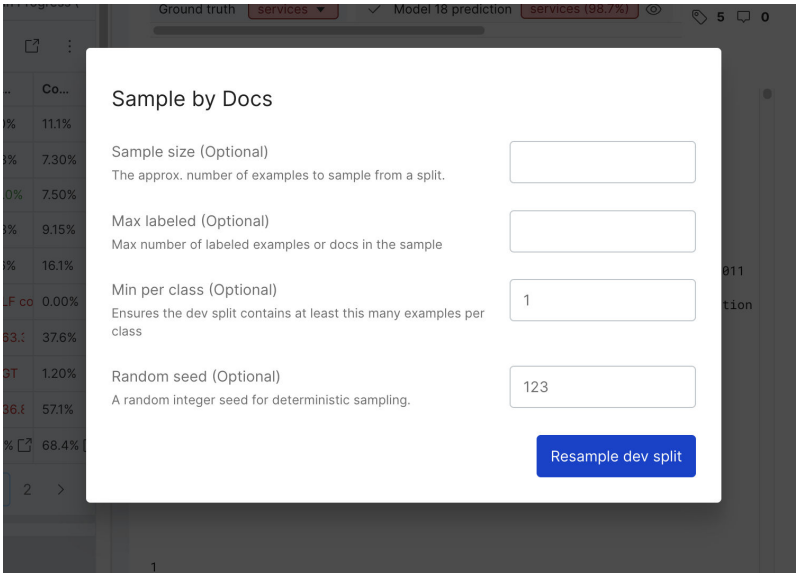
Click the split dropdown to change which split is currently enabled in Application Studio.

Resample data

The longer you iterate, the more likely your LFs are to be tuned to the particular data points that you've labeled in your dev set (i.e., overfitting). For this reason, it's a good practice to periodically refresh your dev set by resampling your dev set from the train split.

Click **Resample data** to open a modal with various resampling options:

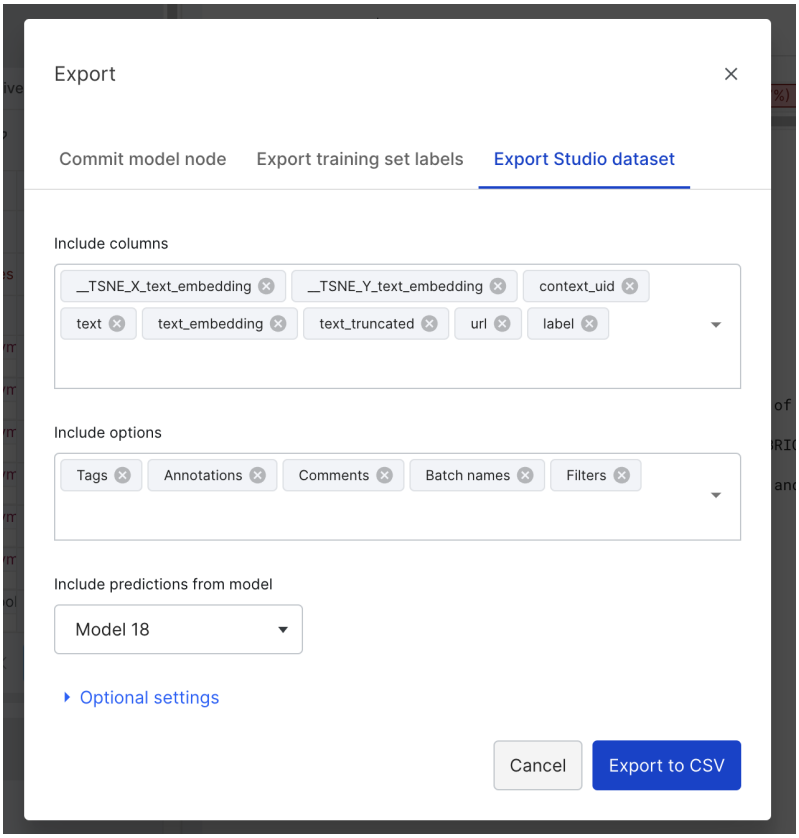
- **Sample size**: The approximate number of data points to sample from a split.
- **Max labeled**: The maximum number of labeled data points to include in the sample.
- **Min per class**: The minimum number of data points per class to include in the sample.
- **Random seed**: A random integer seed to use for deterministic sampling.



Export data

Click **Export** to open a modal with various export options.

Copyright © 2025 Snorkel AI, Inc. All rights reserved



Commit model node

Click **Commit model node** to save a specified model into the application pipeline. You must commit your model if you want to deploy your application. For more information about deployments, see Deploying Snorkel-built models.

Export training set labels

Click **Export training set labels** to export a specified training set into a CSV file. You can specify which columns to export as well as one or more splits to include data from.

Export Studio dataset

Click **Export Studio dataset** to export your dataset into a CSV file. You can specify what columns to export as well as which model to include predictions from.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

Create application from template

An alternative way to create an application in Snorkel Flow is through the templates. You may consider choosing this experience (as opposed to the new guided workflow) if you want to:

- Include more than one model nodes in your application;
- Establish a hierarchical structure;
- Customize preprocessing/post-processing modules beyond defaults; or
- Have the flexibility to perform any advanced action.

Application templates and label spaces

Each application template is exclusive to one label space. When you're working with a specific template, you can expect the GT formats to align with the label space associated with the application template.

All application templates that specify Multi label in their naming conventions use multi label as the label space. These are Multi-label Classification and Multi-label PDF Classification.

All Information Extraction tasks (applications that have the Information Extraction label on the application template cards) use single label spaces. These are US Currency Extraction, Native PDF Extraction, hOCR Extraction, Date Extraction, Email Address Extraction, and Text Extraction.

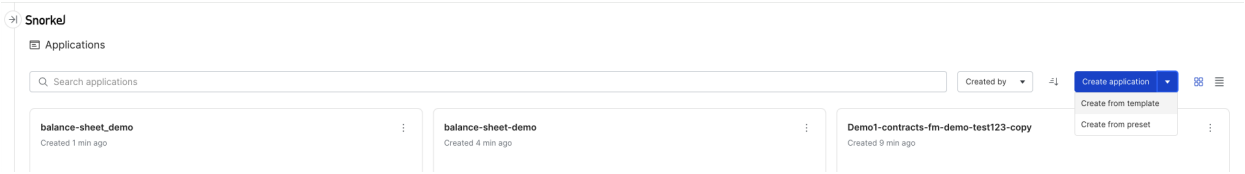
Sequence Tagging is the only application that uses sequence label spaces.

All other application templates also use single-label spaces. These are PDF Classification and plain Classification, Utterance and Conversation Classification, and Text Entity Classification.

Create application

To create a new application from a template:

- Click the **Applications** option in the left-side menu.
- Click the drop-down arrow on the **Create application** button (top right corner of your screen), then click **Create from template** to be brought to the list of pre-set application templates.

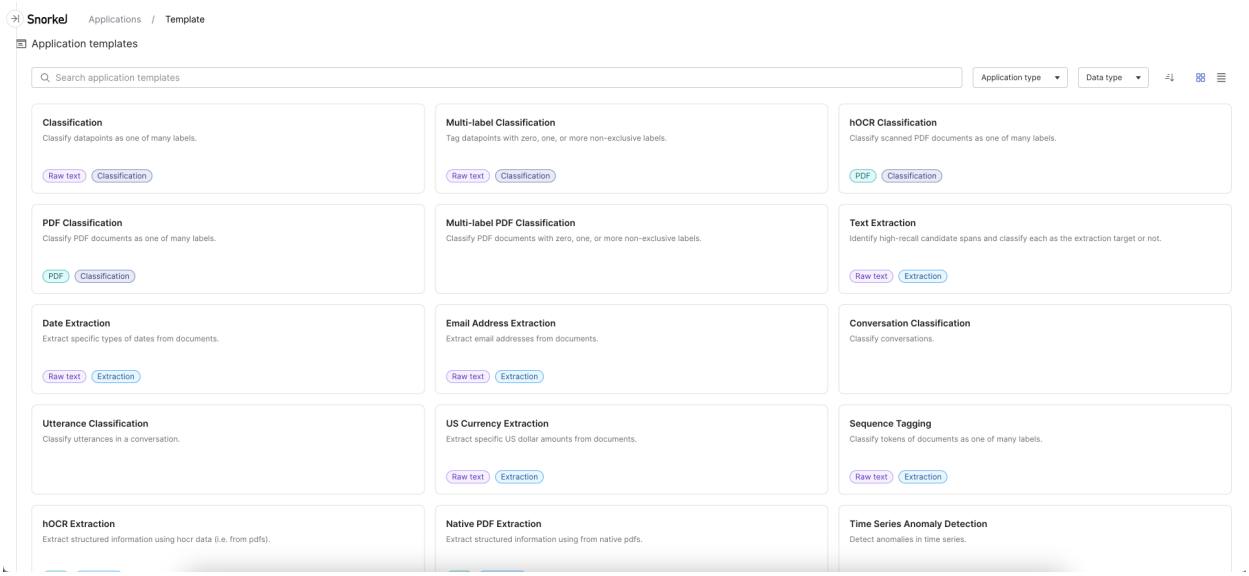


Application templates

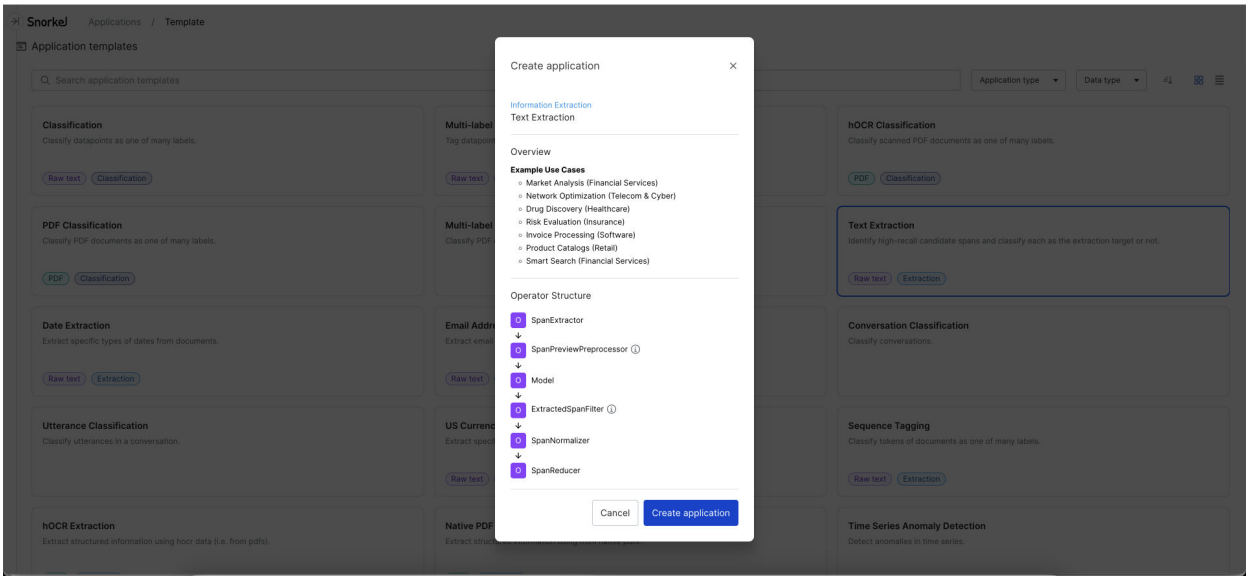
There are a total of **12 templates** that support various:

- **data formats** (raw text and PDF)
- **task types** (classification and extraction of various types)
- **label types** (single and multi-label)

Copyright © 2025 Snorkel AI, Inc. All rights reserved



Select any template to preview the default preprocessors setup before opening the template. Below shows the preview for the Information Extraction: Date Extraction template.



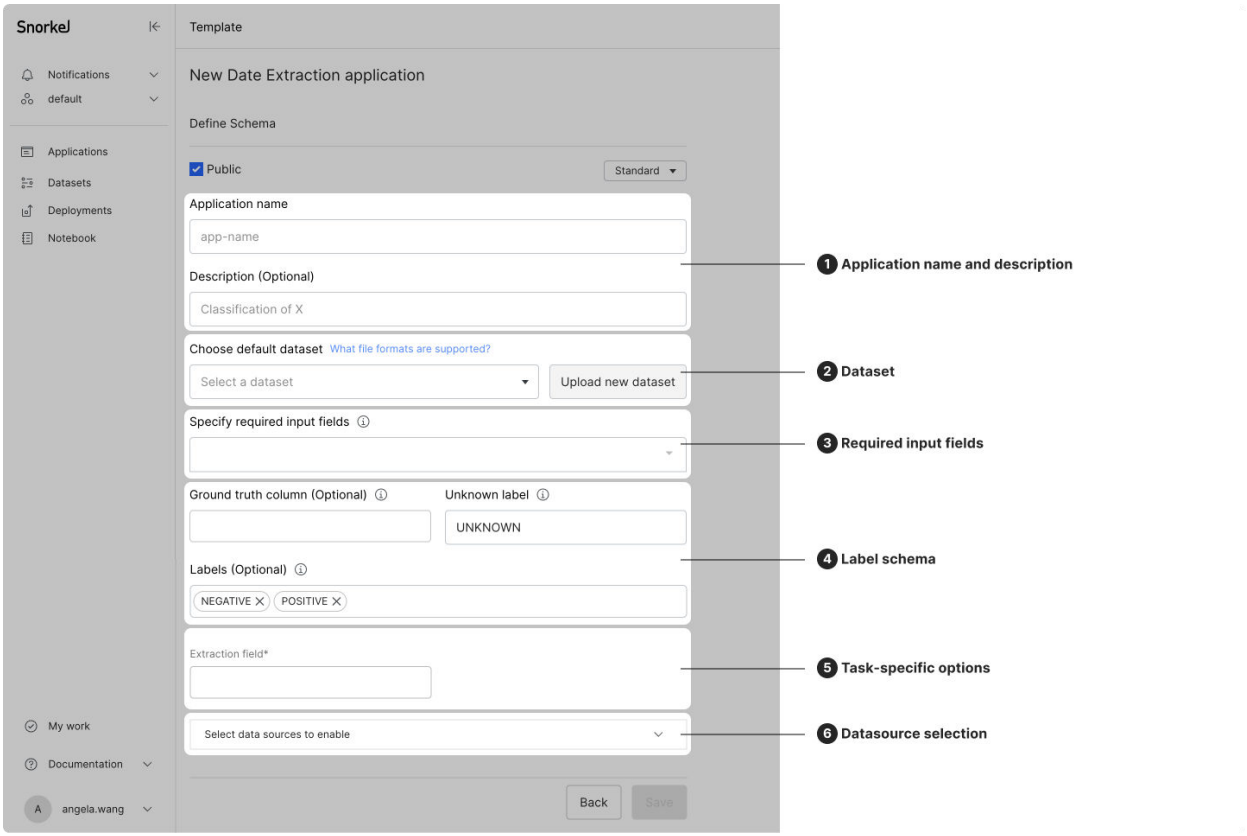
Template overview

The type of user inputs that are required vary across templates, but are in the following general categories:

- **Application name and description**
- Dataset: Either select an existing dataset in your workspace in the dropdown, or click **Upload new dataset** to upload a new dataset (see Data upload for more information).
- **Required input fields** (during application deployment)
- **Label schema definition**: ground truth (GT) column, name for an unknown label, and custom label names
- **Template-specific options**
- **Datasource selection**

Copyright © 2025 Snorkel AI, Inc. All rights reserved

Below shows the **New Date Extraction application** screen for the Information Extraction: Date Extraction template.



Template-specific options

Below is a table of all the template-specific options:

Option	Template	Description
Add embeddings	Single-label classification Single-label candidate-based extraction	Tuning this on allows you to use embedding related features such as Cluster view.
Split doc by page	PDF	Turning this on will add a PageSplitter placeholder operator to the application.
Run OCR on PDF	hOCR	Turning this on will add a hOCR placeholder operator to the application.
PDF field	Native PDF	Select the column in which the PDF URL or relevant info is stored.
Extraction field	Information extraction (excluding sequence tagging)	Select the column from which you want to extract information.

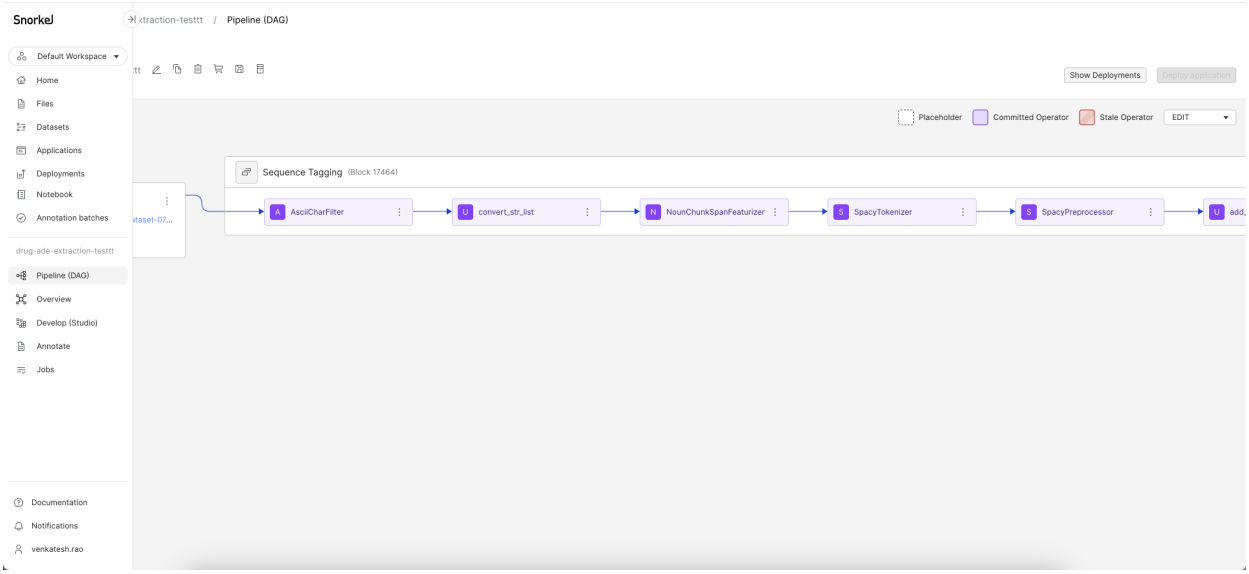
Copyright © 2025 Snorkel AI, Inc. All rights reserved

Option	Template	Description
Negative label	Sequence tagging	Customize name of the negative label.
Seq field	Sequence tagging	Select the column from which you want to extract information.
Auto-generate negative ground truth labels	Sequence tagging	This automatically assign Ground Truth to the negative label where it is currently missing.
Filter out docs > 10 KB	Sequence tagging	Turning this on will skip docs larger than 10 KB in the application to improve performance.

Launch application

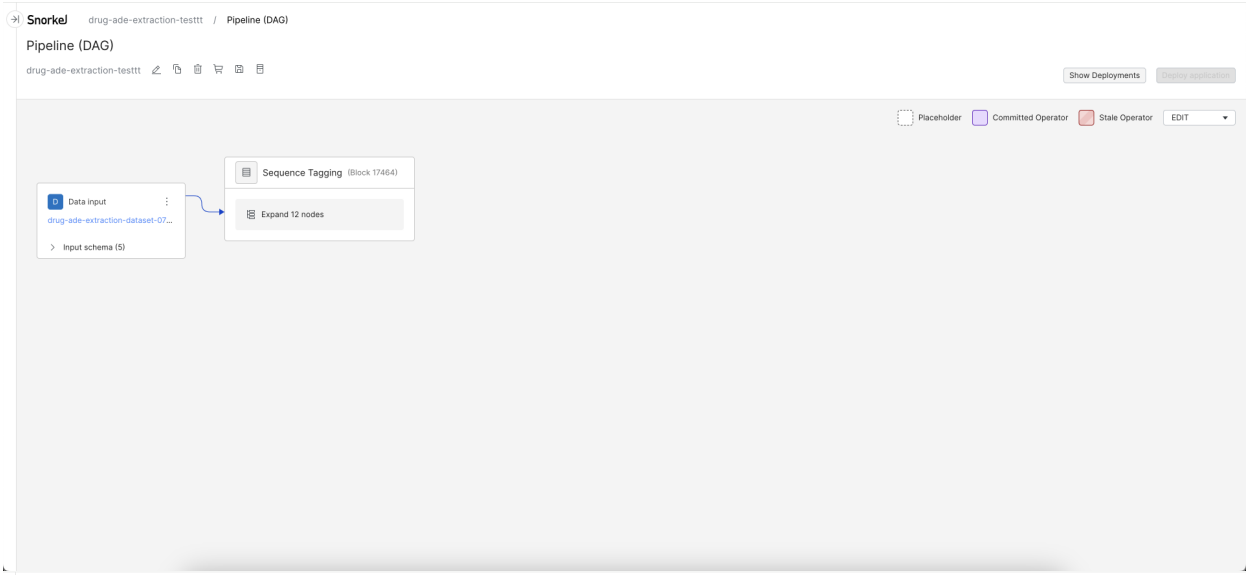
After you fill in the required fields, click **Create** to launch the application.

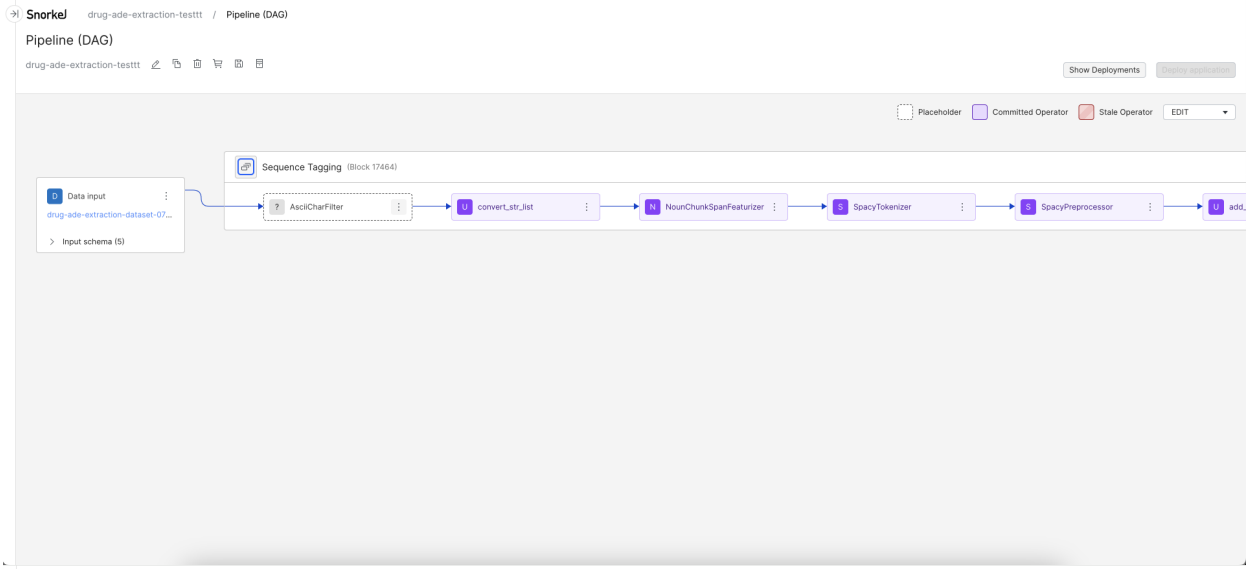
- If no additional input is necessary (e.g., the Classification template), you will land in the **Development Studio** and proceed with developing labeling functions and model iteration.
- If a template requires additional input to the preprocessors, you will be guided to the **DAG page** to complete the preprocessor setup.
  - Once all operators before the Model node are committed (the node is a purple color), click the **Model node** to enter the **Development Studio**.

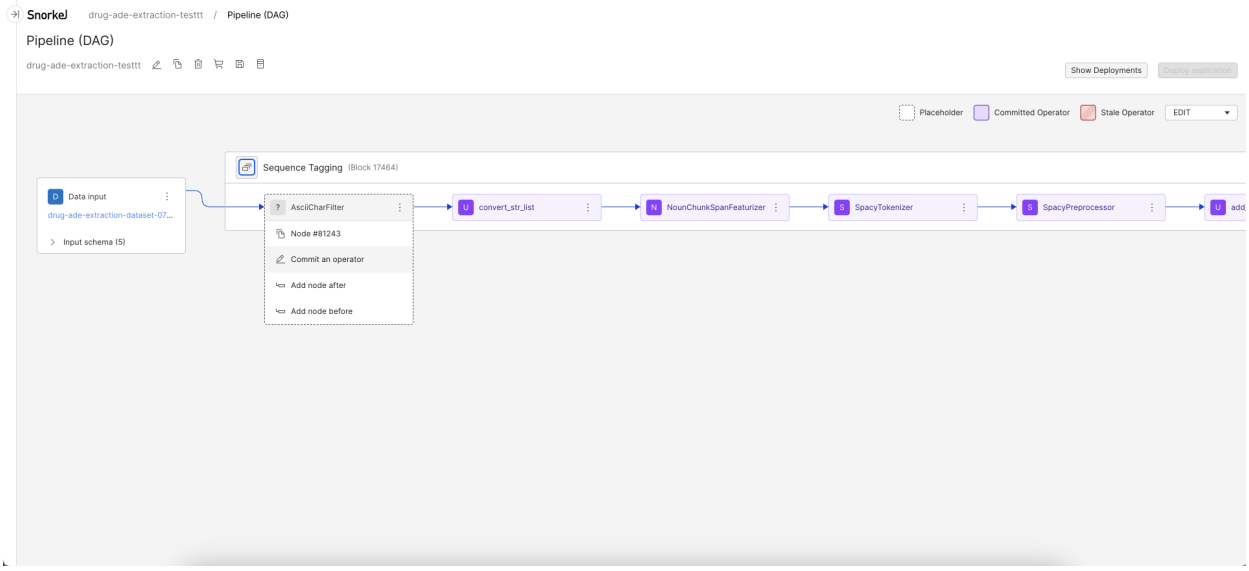


Below shows an example flow of how to define the operator that is used for the SpanExtractor node.

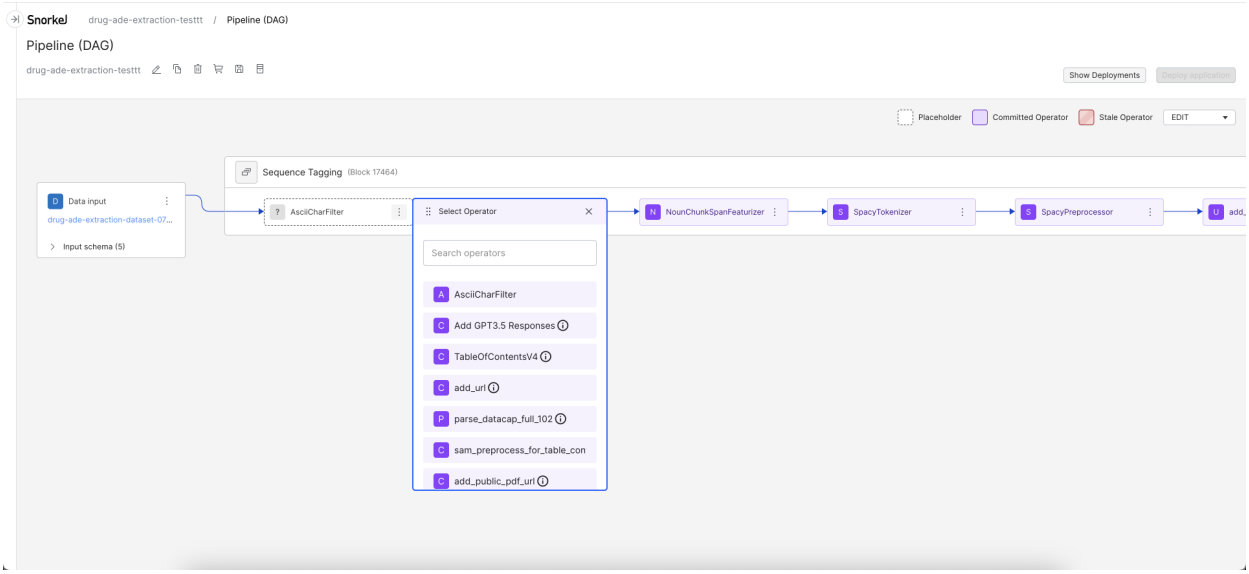
Copyright © 2025 Snorkel AI, Inc. All rights reserved







Copyright © 2025 Snorkel AI, Inc. All rights reserved



Copyright © 2025 Snorkel AI, Inc. All rights reserved

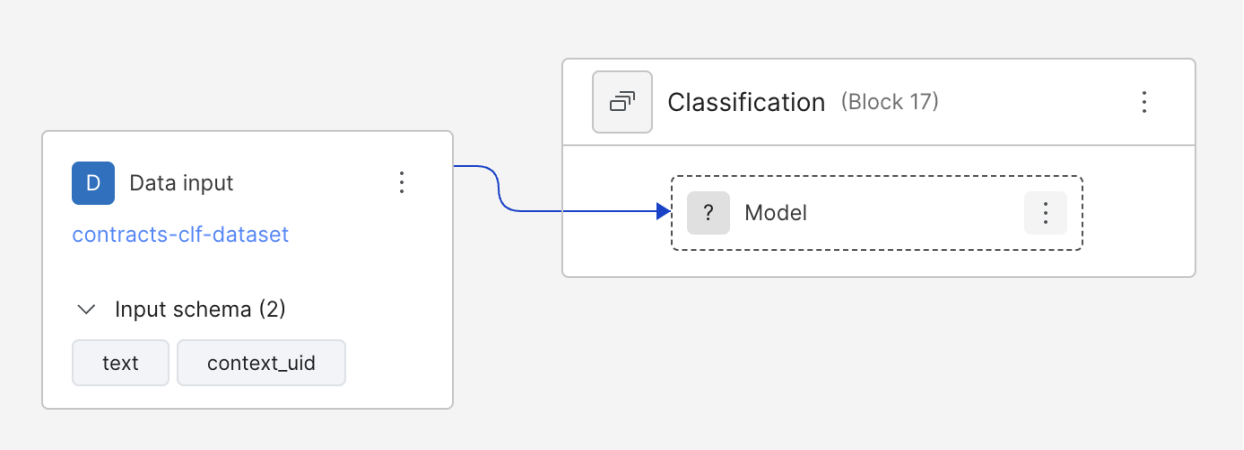
Developing and registering custom models

What you will learn:

- How to load a custom trained model into Snorkel Flow

In addition to built-in model templates, we can also train a custom model and use it as part of an application. To do so, we create a custom model class in a Notebook via the Python SDK.

In this example, you will work with a contract dataset and determine whether each contract can be classified as a loan, employment, service, or stock contract based on the `text` feature.



> NOTE
> You can access the **Pipeline (DAG)** in the left-hand menu. To go back to **Develop (Studio)** view, click on the Model in the DAG.

Develop a custom model

Workflows for developing custom models will typically include

1. Retrieving the data from the application
2. Training the model in Notebook
3. Defining a model class
4. Registering the class via SDK.

Retrieving data

In Notebook, retrieve the programmatically labeled training data that precedes the model node using the SDK method `get_node_data`. Specify `split="train"` to receive data from the train split and training_set_labels=True to receive the labels. Then, pop the labels to target y_train.

Note: training_set_labels requires a training set uid. If you don't have one, it can be created by calling sf.add_training_set(node_uid).

Copyright © 2025 Snorkel AI, Inc. All rights reserved

```
import snorkelflow.client as sf
df_train = sf.get_node_data(
    node=node_uid, # model node uid
    split="train",
    ground_truth=True,
    training_set_labels=True,
    training_set_filter_unlabeled=True, # filter out unlabeled data points
)
y_train = df_train.pop("training_set_labels")
```

To clean up the data, remove any columns like id column `context_uid` that are not needed for training. In this example, keep only the 'text' column. Make sure to remove the ground truth label columns if it's in the dataframe.

```
relevant_columns = list(df_train.columns.values)
relevant_columns.remove('context_uid')
relevant_columns.remove('url') # in the dataframe and not needed for training
relevant_columns.remove('ground_truth')

df_train = df_train[relevant_columns]
```

Training a model

Now that you've prepared your data from the model node, you're ready to train a model. A model of any choice is acceptable: e.g. Sklearn, PyTorch, Keras, etc. Feel free to experiment with a variety of custom models.

> NOTE
>
> The chosen model must be able to extract the weights and save them to disk.

To illustrate a simple example, lets train a Logistic Regression model from Sklearn.

You will first extract the values from the pandas dataframe, fit a `LogisticRegression()` model, and make the predictions on the training data. Since the text column contains string data, use `TfidfVectorizer` to perform text vectorization.

```
from sklearn.linear_model import LinearRegression
from sklearn.feature_extraction.text import TfidfVectorizer

vectorizer = TfidfVectorizer(max_features=5000)
X_train = df_train["text"].str.lower()
X_train = vectorizer.fit_transform(X_train).toarray() # vectorize the text
data

model = LogisticRegression() # model
model.fit(X_train, y_train) # train against the ground truth stored in y_train
```

Copyright © 2025 Snorkel AI, Inc. All rights reserved

Defining a custom model class

After training the model, you need to write a custom model class in Notebook that inherits from the base class `TrainedClassificationModelV2`. Within the class, several necessary methods need to be defined: `__init__`, `save`, `load`, `predict`.

Let's build up these methods individually.

`__init__` takes in a `model` parameter where you will pass in your trained model instance from above. Instantiate a model instance variable with this parameter. In addition, create a list of relevant columns (features) needed for the `predict` method.

Note: `relevant_columns` is a required list that dictates the input schema of the model. It must include all columns used by the `predict` method; otherwise, inference will fail.

```
def __init__(self, model):
    self.model = model
    self.relevant_columns = ['text']
```

`save` determines how the weights of the model will be extracted and saved to a file on disk. It takes in a required argument `dirpath`, which is the directory where the model will be saved. In this case, you simply pickle the model and dump it to the path. In addition, a manifest needs to be written to disk and stores meta-information regarding the model. Instantiate a `BaseManifest` object, pass in the class `type` and relevant `model_fields` (the relevant columns defined in `__init__`). Then, simply write the manifest json to the same directory.

> NOTE
>
> The only two required manifest fields are `type` and `model_fields`. However, the manifest can be additionally used to store other useful metadata for use in the other methods of the class definition.

```
def save(self, dirpath: Path):
    import pickle
    with open(dirpath / "model.pickle", "wb") as tf:
        pickle.dump(self.model, tf)

    # Need to also define operator input schema for manifest
    model_fields = self.relevant_columns
    from snorkelflow.models.cls_model import BaseManifest
    manifest = BaseManifest(type=self.__class__.__name__,
model_fields=model_fields)
    with open(dirpath / "manifest.json", "w+") as mf:
        mf.write(manifest.json())
```

The class method `load`, on the other hand, defines how you read the weights from a directory path and load an instance of the defined class. The method takes in the directory path `dirpath` to read from, and a `storage_options` argument. In this case, simply load the file from the path and unpickle the model. Then instantiate an instance of the class `ExampleClass` with the trained model.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

```
@classmethod
def load(cls, dirpath: Path, storage_options: Optional[Dict[str, Any]] =
None):
    import pickle

    from snorkelflow.utils import open_file
    model_path = os.path.join(dirpath, "model.pickle")
    with open_file(model_path, "rb") as tf:
        model = pickle.load(tf)

    return cls(model=model)
```

The last method `predict` dictates how you will run inference on the data with your trained model. As such, it takes in one argument `df`, the dataframe you will run inference on. Here, simply filter for the columns you need for prediction and call `.predict` and `.predict_proba` on the dataframe values.

> NOTE
>
> `predict` must return a tuple of classification predictions and probabilities. However, probabilities are optional, in which case you could return `..., None` instead.

```
def predict(self, df: pd.DataFrame):
    import pandas as pd
    processed_df = df[self.relevant_columns]
    return self.model.predict(processed_df.values),
self.model.predict_proba(processed_df.values)
```

The full code can be seen below

Copyright © 2025 Snorkel AI, Inc. All rights reserved

```
from pathlib import Path
from typing import Any, Dict, Optional

from snorkelflow.models.cls_model import (
    TrainedClassificationModelV2,
    trained_models_dict,
)

class ExampleClass(TrainedClassificationModelV2):
    def __init__(self, model):
        self.model = model
        self.relevant_columns = ['text']

    def save(self, dirpath: Path):
        import pickle
        with open(dirpath / "model.pickle", "wb") as tf:
            pickle.dump(self.model, tf)

        # Need to also define operator input schema for manifest
        model_fields = self.relevant_columns
        from snorkelflow.models.cls_model import BaseManifest
        manifest = BaseManifest(type=self.__class__.__name__,
model_fields=model_fields)
        with open(dirpath / "manifest.json", "w+") as mf:
            mf.write(manifest.json())

    @classmethod
    def load(cls, dirpath: Path, storage_options: Optional[Dict[str, Any]] =
None):
        import pickle

        from snorkelflow.utils import open_file
        model_path = os.path.join(dirpath, "model.pickle")
        with open_file(model_path, "rb") as tf:
            model = pickle.load(tf)

        return cls(model=model)

    def predict(self, df: pd.DataFrame):
        import pandas as pd
        from sklearn.feature_extraction.text import TfidfVectorizer

        vectorizer = TfidfVectorizer(max_features=5000)
        df = df[self.relevant_columns]

        df = df["text"].str.lower()
        df_num = vectorizer.fit_transform(df).toarray()
        return self.model.predict(df_num), self.model.predict_proba(df_num)
```

Copyright © 2025 Snorkel AI, Inc. All rights reserved

Testing the custom model class

Now that you've written a custom model class definition, you can test the class before registering with Snorkel Flow.

```
# Initialize the model instance
model_instance = ExampleClass(model=model)
# Test the save method
model_instance.save(resolve_data_path('minio://custom_model_test'))
# Test the load method
model_instance2 =
model_instance.load(resolve_data_path('minio://custom_model_test'))
# Test the predict method, check that results are the same before
saving+loading and after

model_instance2.predict(df)
model_instance.predict(df)
```

Registering the custom model class

Now that you've tested your custom model class, it's time to register the class with Snorkel Flow. Simply create an instance of the class and pass in the trained Sklearn model. Then, call the SDK method `sf.register_trained_model` and pass in this instance, the ID of the model node, and the description of the model.

```
model_instance = ExampleClass(model=model)
job_id = sf.register_trained_model(model=model_instance,
                                   node=node_uid,
                                   description="our newly created custom
model")["job_id"]
```

> NOTE
>
> If you're getting an error about a label class mismatch, double check that the label classes defined in the application are the same as in the training data. Pay special attention to any unknown labels in the training data.
> A server error may occur if the model class name is too long. Try shortening the name of the class.

Monitoring progress and completion

`models.register_trained_model` SDK method will output a Job ID for tracking. Using `poll_job_status`, you can track the status of the registration and prediction.

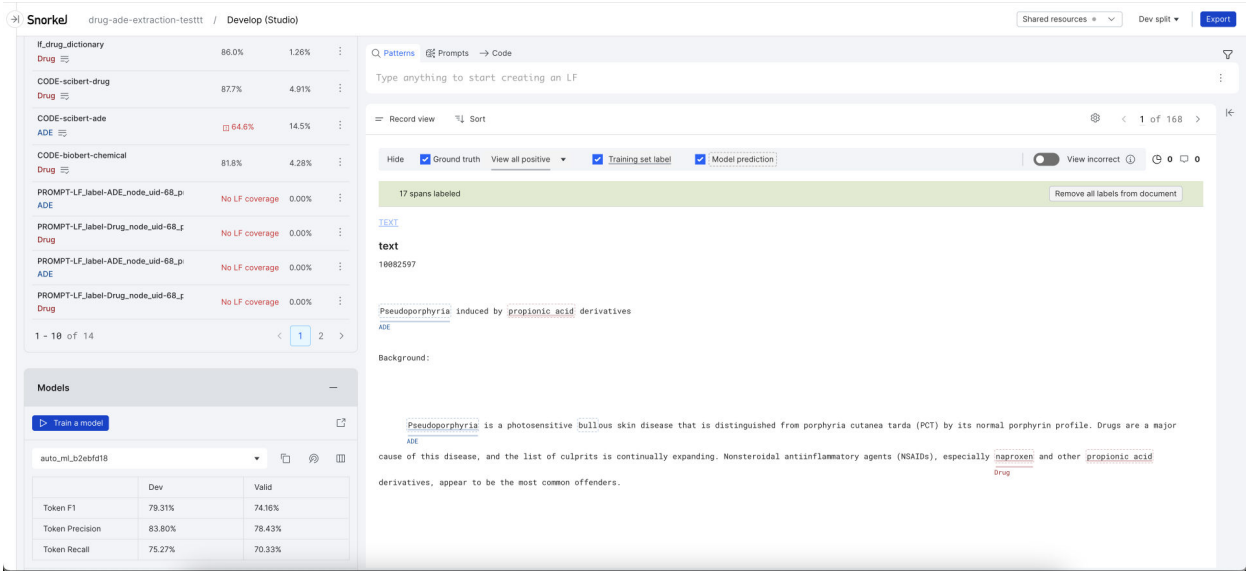
```
sf.poll_job_status(job_id)
```

Copyright © 2025 Snorkel AI, Inc. All rights reserved

Use a custom model

Now that registration has been completed, you can use the model in Snorkel Flow.

Once registration has finished, the model will appear on the Models tab in **Develop (Studio)**. You can see more info by clicking on the External Link button next to "Train a Model." It can then be committed just like any other trained built-in model and will appear in Application Studio. The model node in the DAG will now be purple.



Note: As of now, custom models are not supported for analysis.

Investigating issues

If you encounter any issues and want to understand the details, you could enable the debug mode of the Snorkel Flow SDK. For example, if you want to investigate issues when registering the custom-trained model, you can do the following:

```
ctx.set_debug(True)
sf.poll_job_status(job_id)
```

It will show the detailed error message and stacktrace which can help you fix the custom-trained model.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

Image Studio overview

Snorkel Flow supports an end-to-end machine learning workflow for image classification. Unlock the programmatic labeling workflow for your image datasets within Image Studio.

For more details about the features and capabilities of Image Studio, see the reference guide.

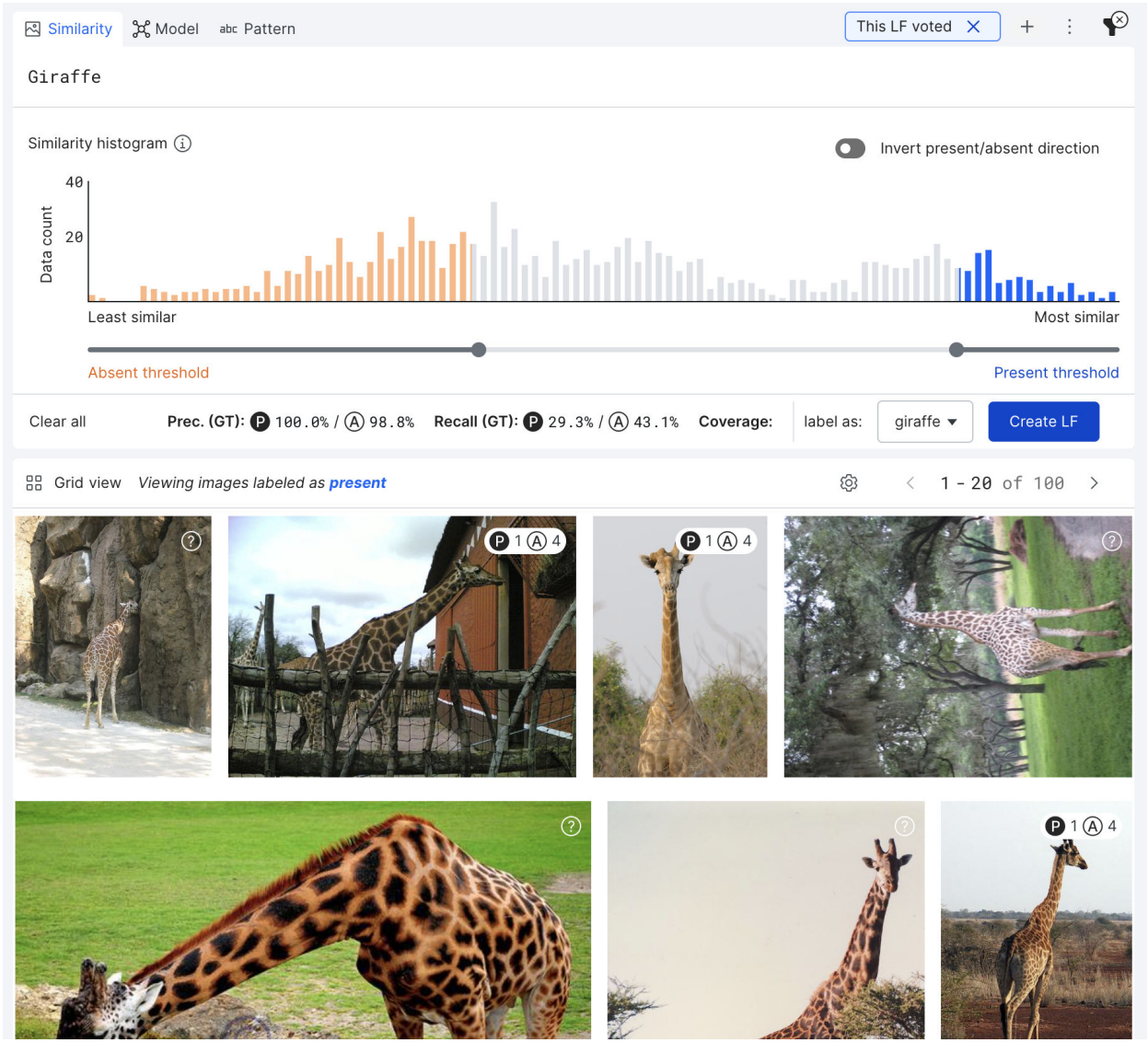


Image Studio offers several new tools for writing labeling functions for your data.

- T**ext-to-image similarity:** Compose a text string and query for images that match the query.
- **Image-to-image similarity:** Select a target image and query for image matches that are indexed by the similarity to the selected image. In addition, you can match against a cropped section of an image.
- **Model confidence:** Use a model that you have already trained and query for image matches indexed by the model's predictive inference confidence score.
- **Text pattern:** Use an assortment of pattern-based matchers to query against multi-modal text data that is associated with your images.

Image Studio also offers three new views from which you can explore and annotate your data.

- **Grid**: View multiple images in a gallery view to easily explore the images in your dataset.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

- **Image**: View an individual image and its associated metadata.
- **Data**: View image data along-side any text-based metadata.

Native features in Snorkel Flow Studio are ready to use with Image Studio, including LF management tools, view filters, split switching, data resampling, and our comprehensive model analysis kit.

Check out this video by Charlie Wang, lead software engineer on the image classification team at Snorkel, where he demos all the new capabilities of Image Studio.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

Image Studio reference

This page walks through the key components of using Image Studio for your image classification applications:

- Prerequisites to get started
- Exploring your data
- Annotating your data
- Creating labeling functions
- Training models
- Exporting your data

Prerequisite: Enable dedicated resources

Because image data is large and computationally expensive, you must first enable dedicated resources on your application before developing labeling functions (LFs) and models. Snorkel Flow will then allocate compute resources so that you have faster and more performant backend operations on much larger dataset sizes. This allows you to scale up to 500K data points in a split sample. For step-by-step instructions on how to enable dedicated resources on your application, see Using dedicated resources to speed up applications with large datasets.

> NOTE
>
> Snorkel may take time to cache the dataset upon initial load.

> NOTE
>
> When you are finished, be sure to de-allocate dedicated resources to free up memory for others to use.

Explore your data

Image Studio provides three views from which you can explore your data:

- Grid: View multiple images in a gallery view.
- Image: View an individual image and its associated metadata.
- Data: View image data along-side any text-based metadata.

The following sections walk through the available views in more detail.

Grid view

Grid view allows you to view multiple images at a time in an elegant photo gallery view. This provides you with a broad overview of what the images in your dataset look like.

You can also annotate ground truth in **Grid view**. See Annotate your data for more information.

Click the gear icon customize the view settings:

- **Page size**: Specify how many images are viewable per page.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

- **Export Studio dataset**: Export your dataset into a CSV file. You can specify what columns to export as well as which model to include predictions from.
- **Resample data**: Opens a modal with various resampling options:
  - **Sample size**: The approximate number of data points to sample from a split.
  - **Max labeled**: The maximum number of labeled data points to include in the sample.
  - **Min per class**: The minimum number of data points per class to include in the sample.
  - **Random seed**: A random integer seed to use for deterministic sampling.

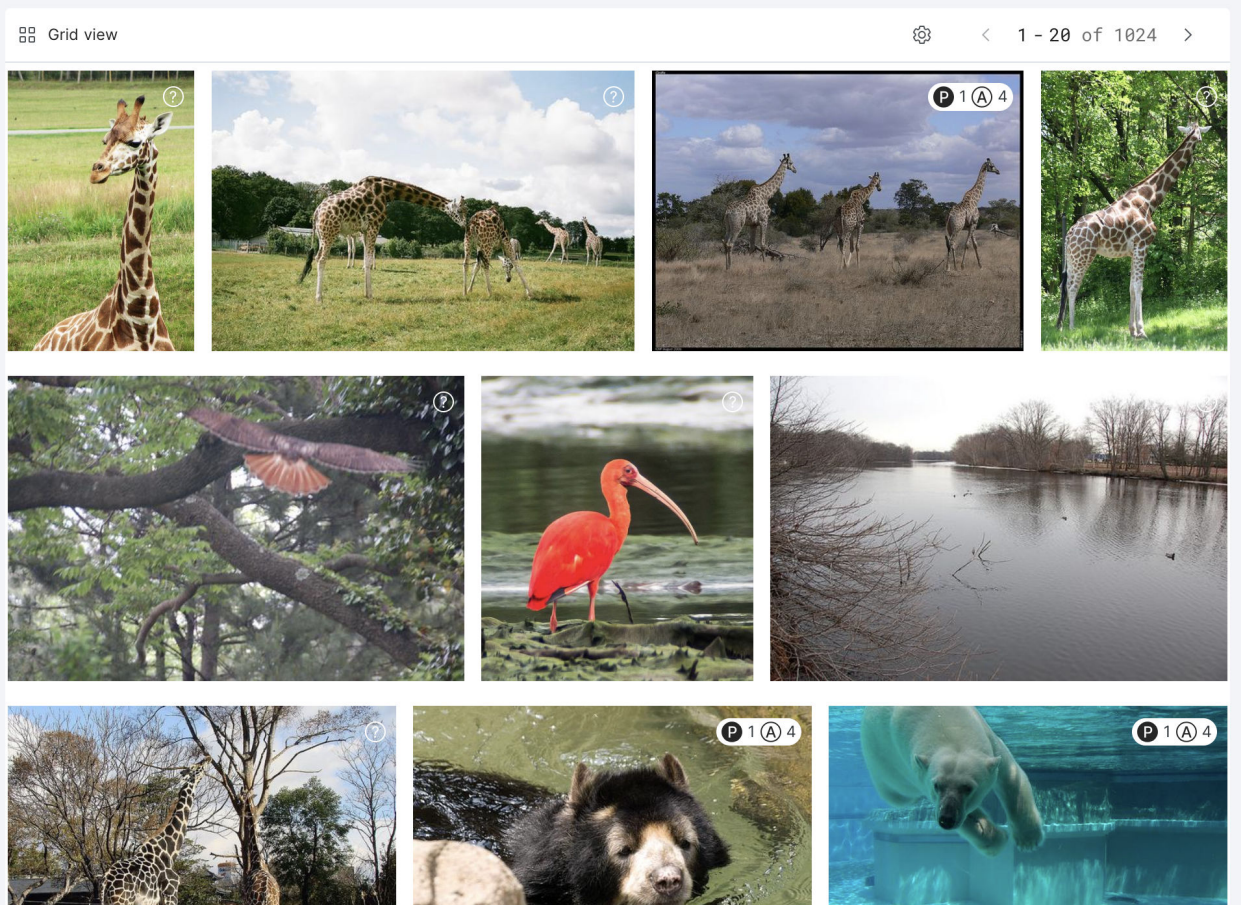


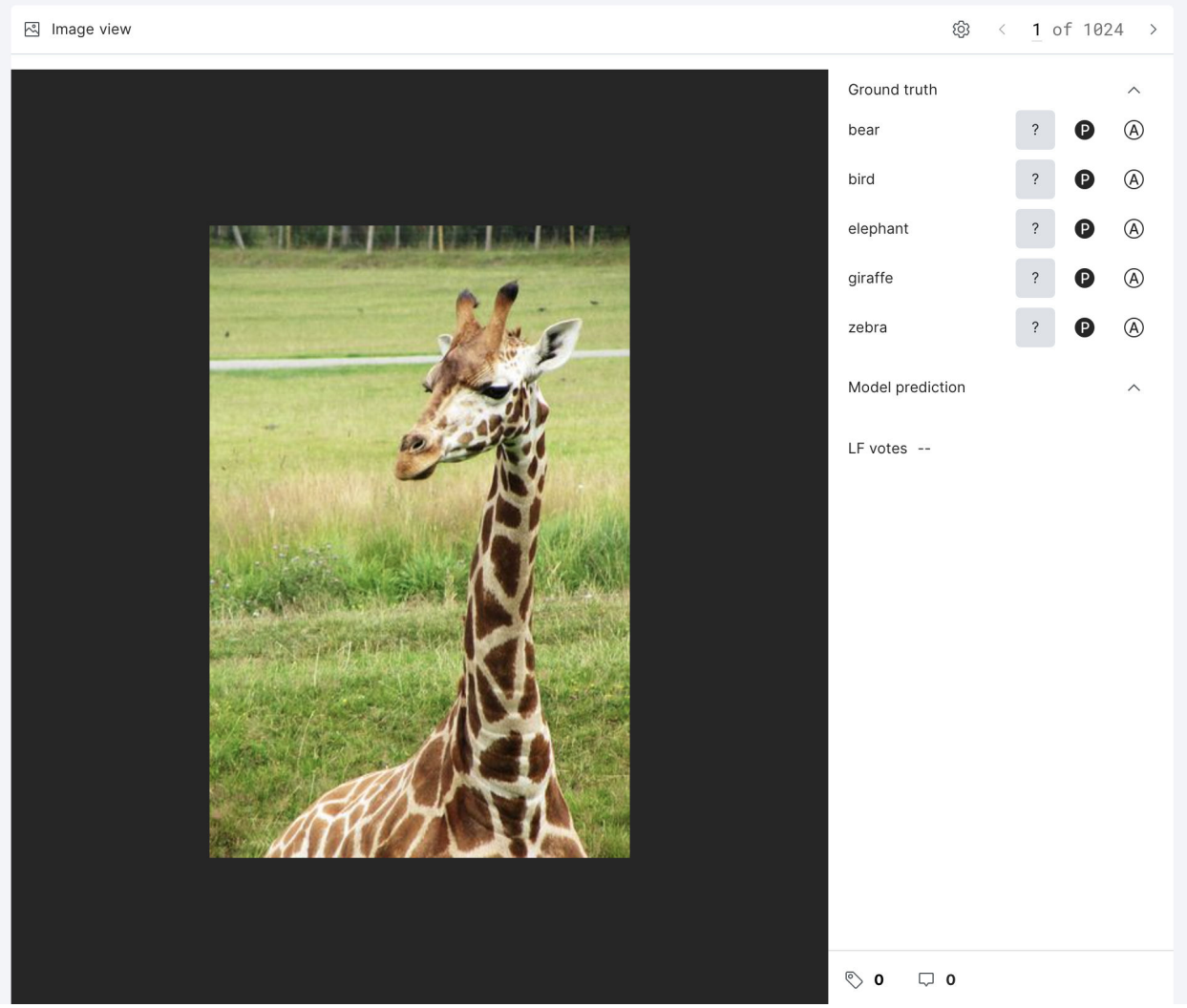
Image view

If you are paging through images in **Grid view**, and want a more detailed view of an individual image, select that image to open it up in **Image view**. Here, you can view an individual image and its associated metadata. You can view an image's ground truth, model predictions for the selected model, LF votes, and have access to slices and comments. In addition, you can update the ground truth of the image.

Click the gear icon customize the view settings:

- **Export Studio dataset**: Export your dataset into a CSV file. You can specify what columns to export as well as which model to include predictions from.
- **Resample data**: Opens a modal with various resampling options:
  - **Sample size**: The approximate number of data points to sample from a split.
  - **Max labeled**: The maximum number of labeled data points to include in the sample.
  - **Min per class**: The minimum number of data points per class to include in the sample.
  - **Random seed**: A random integer seed to use for deterministic sampling.

Copyright © 2025 Snorkel AI, Inc. All rights reserved



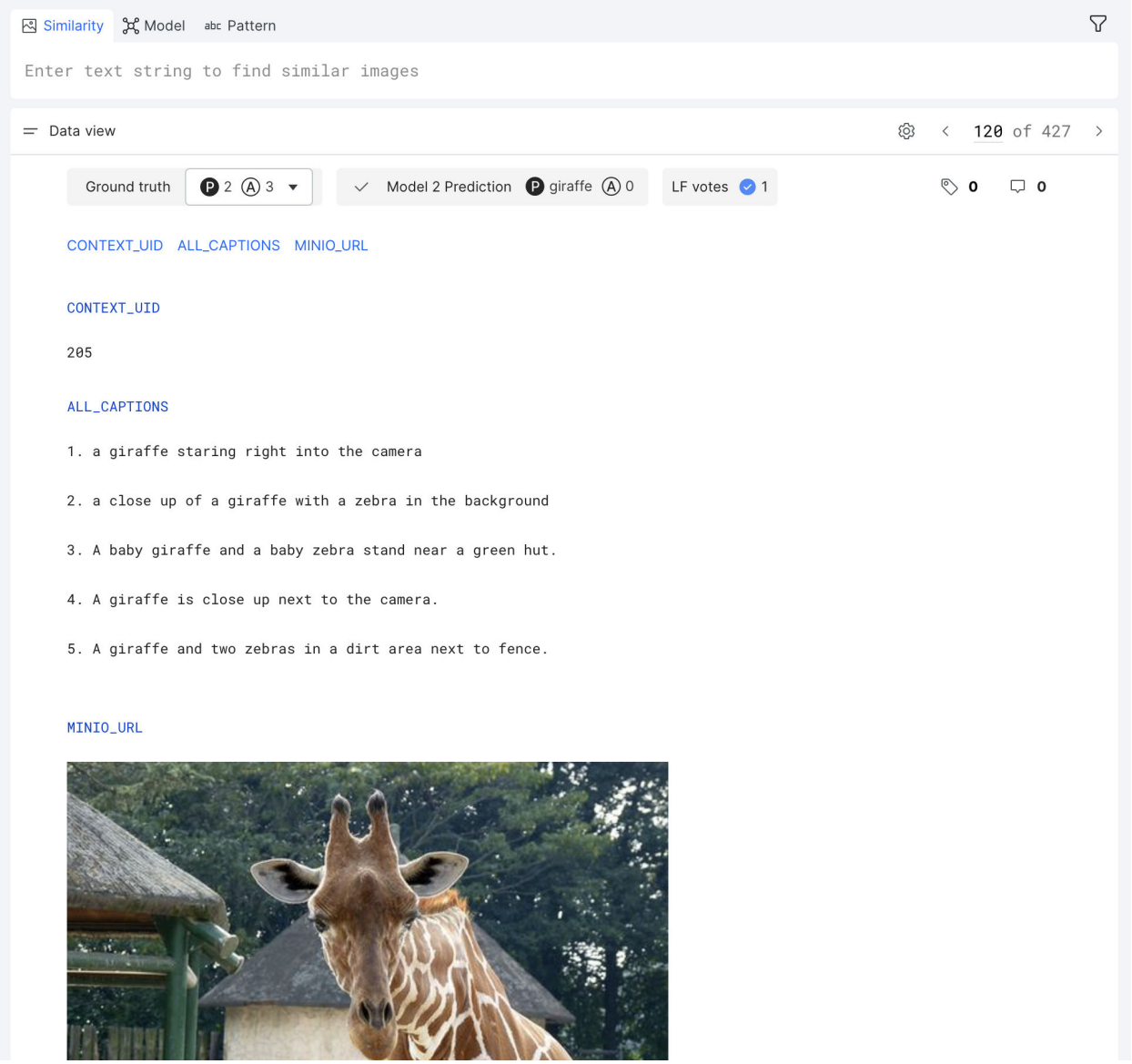
Data view

Data view allows you to view image data alongside any text-based metadata, such as captions. This is particularly useful for writing LFs based on pattern matches against text columns.

Click the gear icon customize the view settings:

- **Select displayed fields**: Customize the data fields that are displayed.
- **Export Studio dataset**: Export your dataset into a CSV file. You can specify what columns to export as well as which model to include predictions from.
- **Resample data**: Opens a modal with various resampling options:
  - **Sample size**: The approximate number of data points to sample from a split.
  - **Max labeled**: The maximum number of labeled data points to include in the sample.
  - **Min per class**: The minimum number of data points per class to include in the sample.
  - **Random seed**: A random integer seed to use for deterministic sampling.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

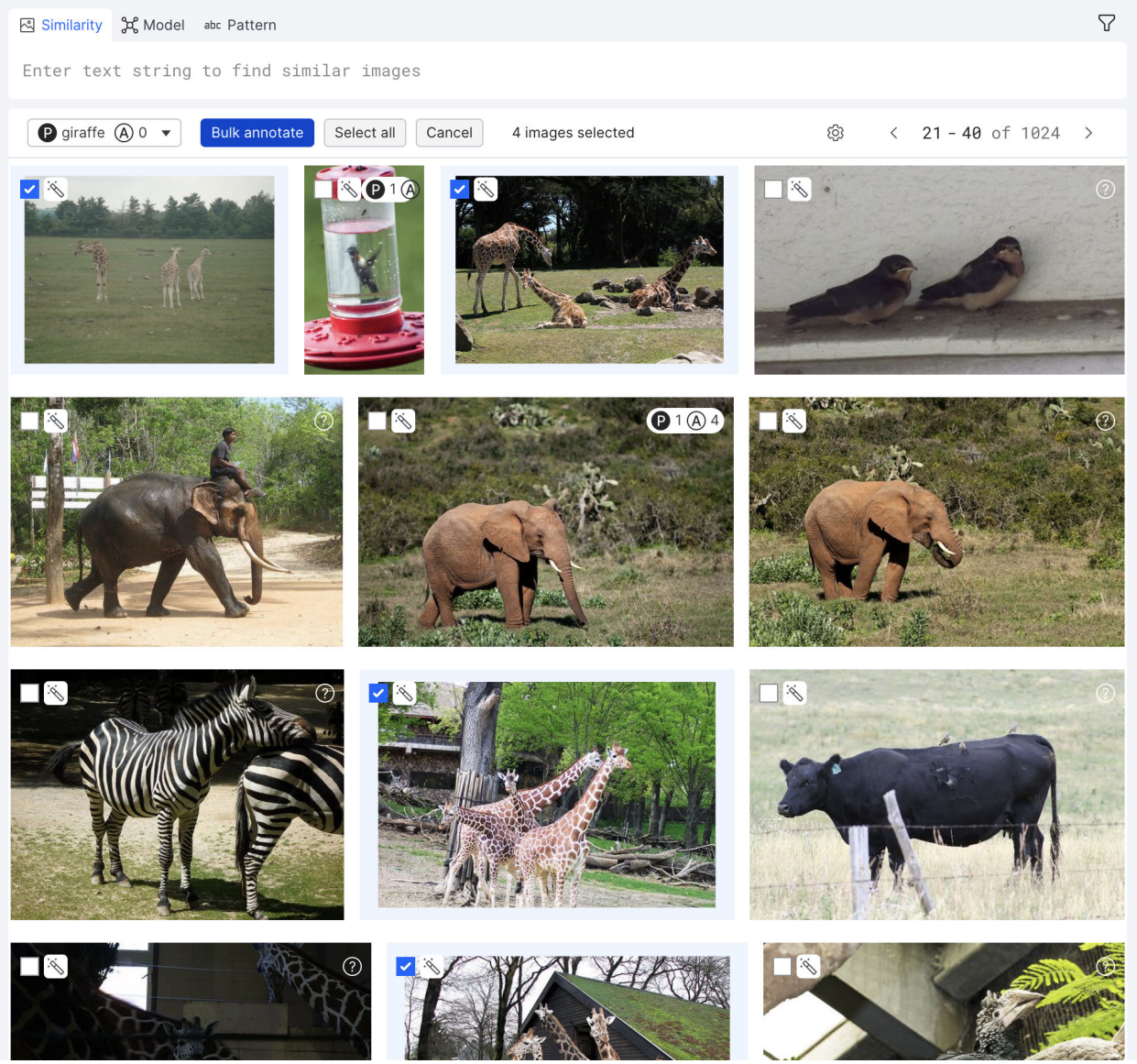


Annotate your data

Image Studio supports simple ground truth annotation workflows to easily annotate your data from any data view. Images can be labeled with multiple classes by three statuses: Present, Absent, or Abstain.

Grid view supports a special bulk annotation feature that allows you to label multiple data points at the same time. Use this to easily apply class labels to batches of image data points.

Copyright © 2025 Snorkel AI, Inc. All rights reserved



Create labeling functions (LFs)

Labeling functions (LFs) are programmatic rules or heuristics that assigns labels to unlabeled data without the need for manual annotations. Each labeling function is defined for a single class such that it votes whether or not the image should be assigned to that class. See Introduction to labeling functions (LFs) for more information.

There are three types of LFs that can be generated in Image Studio:

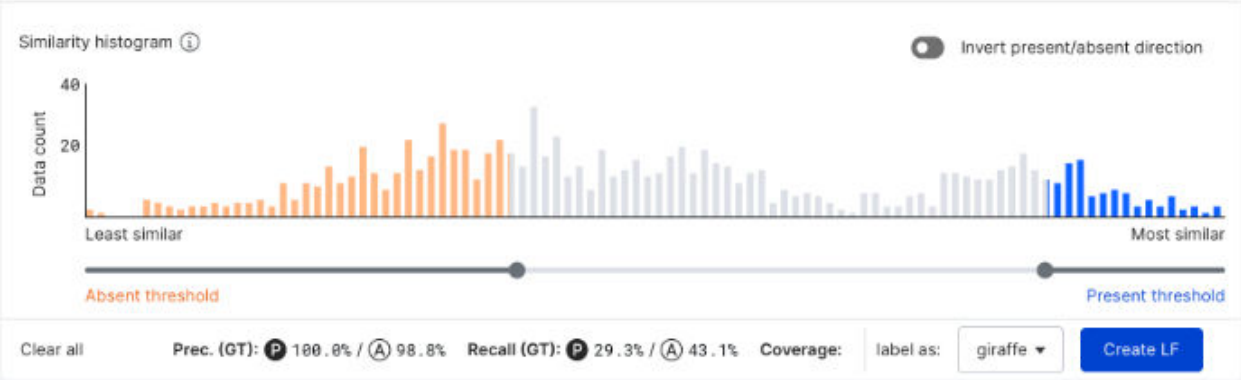
- Similarity: Labels images based on their similarity score to text, other images, or cropped images.
- Model: Labels images based on the model predictions of a pre-trained model.
- Pattern: Labels images based on user-defined rules (e.g., keyword or regex match) against text data that is associated with your images. This is a native feature in Snorkel Flow.

The following sections walk through the available LFs in more detail.

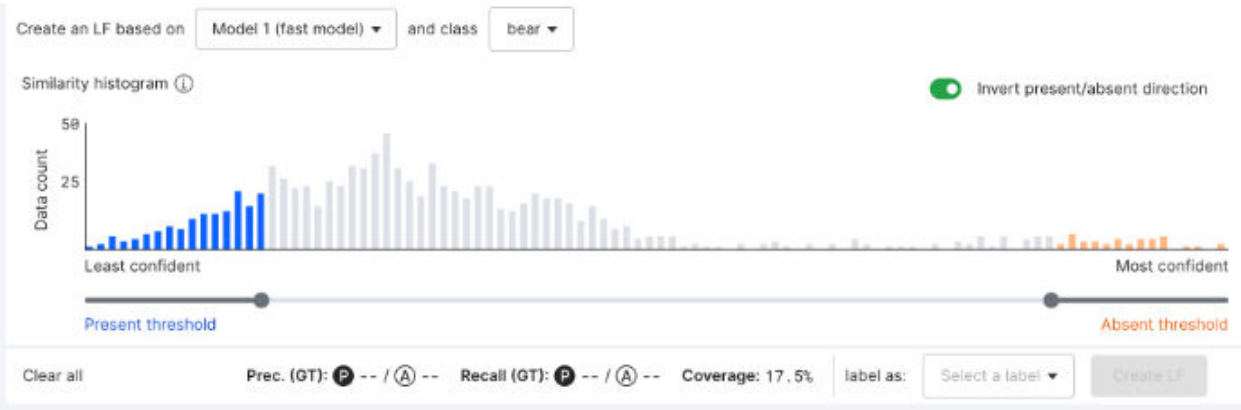
Copyright © 2025 Snorkel AI, Inc. All rights reserved

Similarity histogram

Each LF tool includes a visual similarity histogram that you can use to filter the data points that you would like to label. The histogram ranks your data points from left-to-right as least similar to most similar. That is, the selector on the left selects data points with low similarity scores that will be labeled as absent. Similarly, the selector on the right selects data points with high similarity scores that will labeled as present. All unselected images are not labeled (i.e., are abstained).



You can turn on the **Invert present/absent direction** option to incorporate simple negation into your LFs. When the LF is inverted, data points that are selected as least similar on the left will be voted as present, while data points that are selected as most similar on the right will be voted as absent.



As you adjust the thresholds, the precision and recall values for predicting both present and absent labels are automatically updated for the current split. This is usually the dev or train split rather than the valid split. Because of this, we recommend adding a small sample of ground truth labels for each into in to the dev/train splits, in addition to keeping the majority of ground truth labels within the valid split.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

Similarity LFs

There are three types of similarity LFs that you can create:

LF	Input	Output
Image-to-text comparator	Text query	OpenAI CLIP similarity score
Image-to-image comparator	Dataset image	OpenAI CLIP similarity score
Image-to-image patch comparator	Cropped dataset image	OpenAI CLIP similarity score

Image-to-text comparator

The text-to-image comparator LF tool accepts a text query and outputs image matches ranked by their embedding similarity score. These embeddings are computed using the OpenAI CLIP model.

To create an image-to-text LF:

1. Select the **Similarity** tab on the LF composer toolbar.
2. In the textbox, write a text string for which you would like to find similar images.
3. Click **Preview LF.**
4. Select the class that you want to label in the **label as** drop down.
5. Use the similarity histogram to adjust the thresholds for the images that you would like to label with this LF.
6. Once you are happy with the thresholds that you have selected, click **Create LF**.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

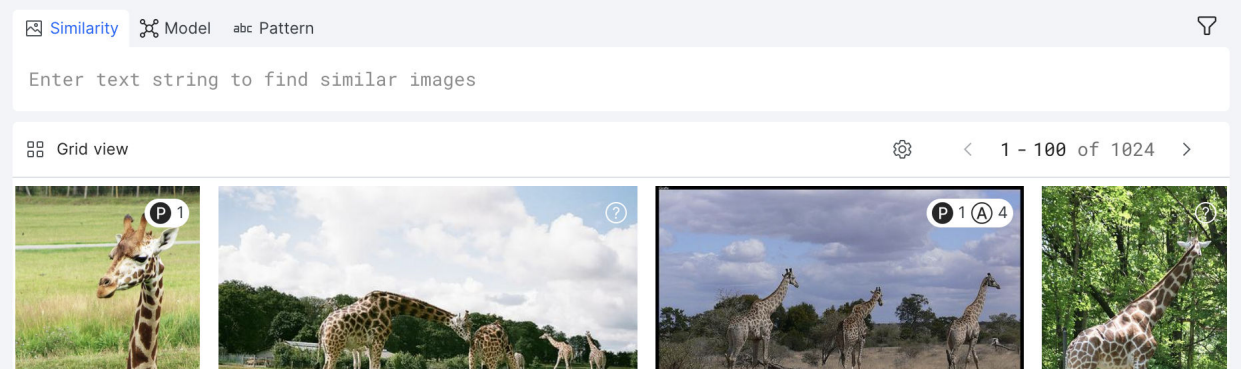


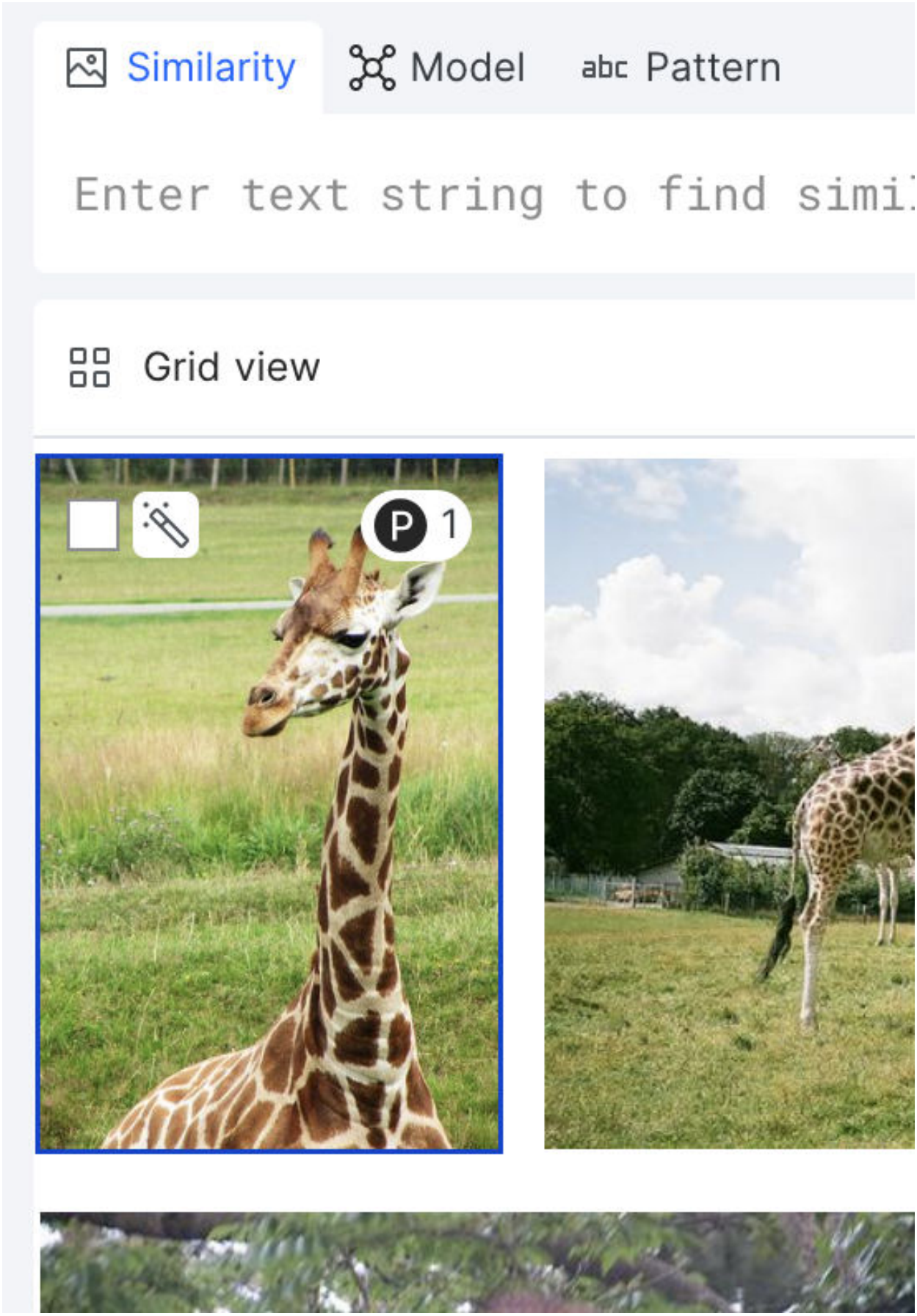
Image-to-image comparator

The image-to-image comparator LF tool allows you to select any image using the magic wand tool and outputs image matches ranked by their embedding similarity score. These embeddings are computed using the OpenAI CLIP model.

To create an image-to-image LF:

1. Select the **Similarity** tab on the LF composer toolbar.
2. Select the image that you want to perform a similarity search.
   - In **Grid view**, hover over the image, and then select the magic wand button.
   - In **Image view**, hover over the image, and then click **Preview image LF**.
3. Select the class that you want to label in the **label as** drop down.
4. Use the similarity histogram to adjust the thresholds for the images that you would like to label with this LF.
5. Once you are happy with the thresholds that you have selected, click **Create LF**.

Copyright © 2025 Snorkel AI, Inc. All rights reserved



Copyright © 2025 Snorkel AI, Inc. All rights reserved

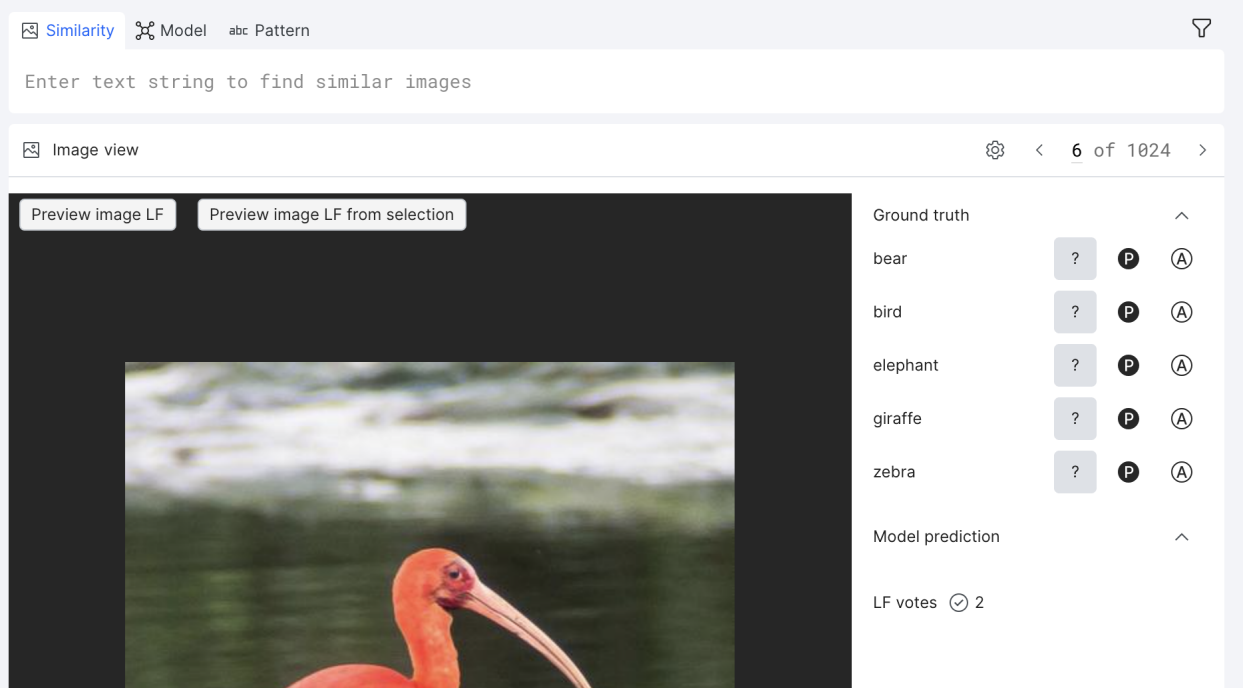


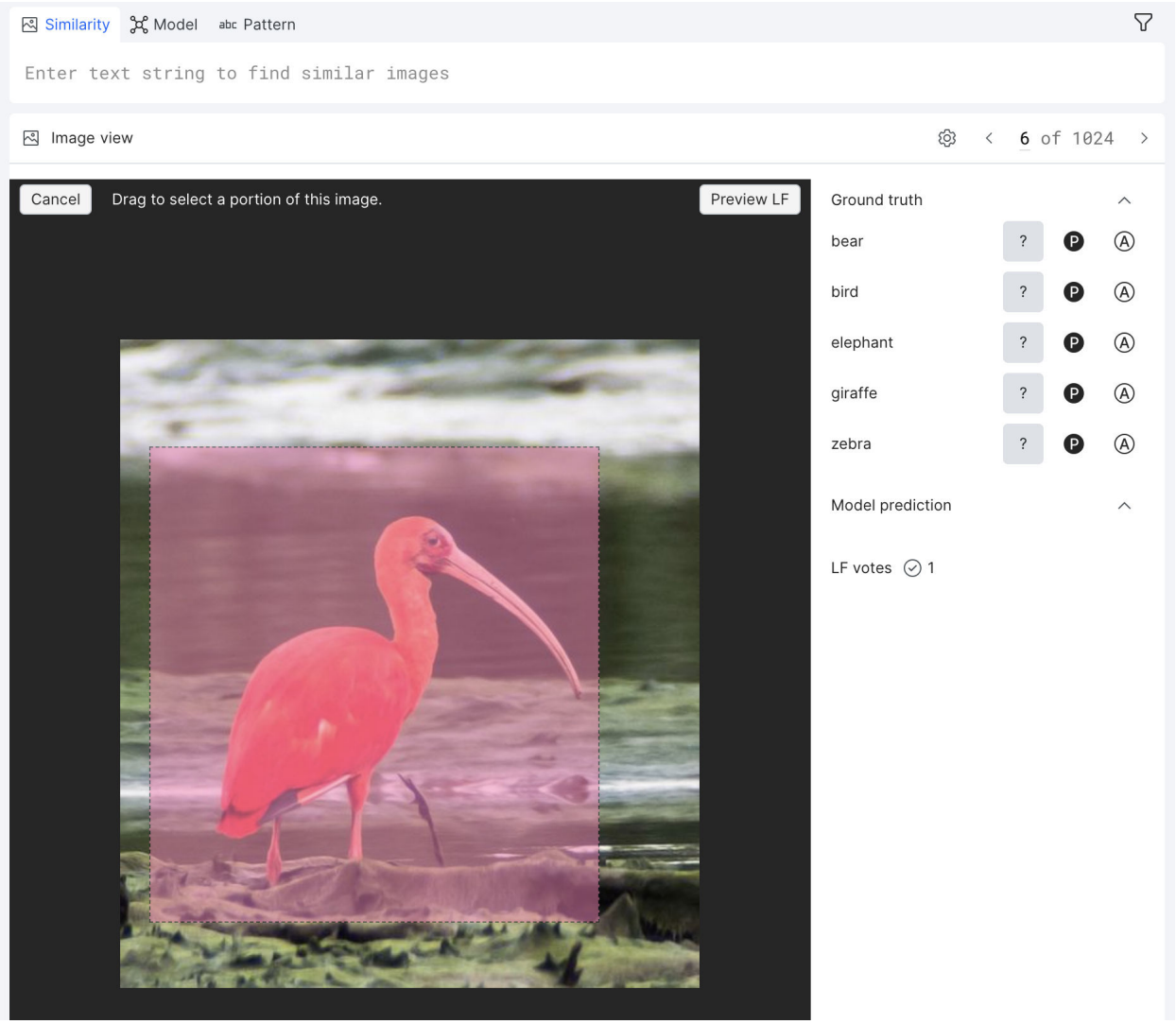
Image-to-image patch comparator

The image-to-image patch comparator LF tool functions the same as the image-to-image comparator LF tool, but with the option to crop the image. You select any image using the magic wand tool, crop the image, and the tool outputs image matches ranked by their embedding similarity score. These embeddings are computed using the OpenAI CLIP model.

To create an image-to-image patch LF:

1. Select the **Similarity** tab on the LF composer toolbar.
2. From **Image view**, hover over the image, and then click **Preview image LF from selection**.
3. Select a cropped section of the image, and then click **Preview image LF**.
4. Select the class that you want to label in the **label as** drop down.
5. Use the similarity histogram to adjust the thresholds for the images that you would like to label with this LF.
6. Once you are happy with the thresholds that you have selected, click **Create LF**.

Copyright © 2025 Snorkel AI, Inc. All rights reserved



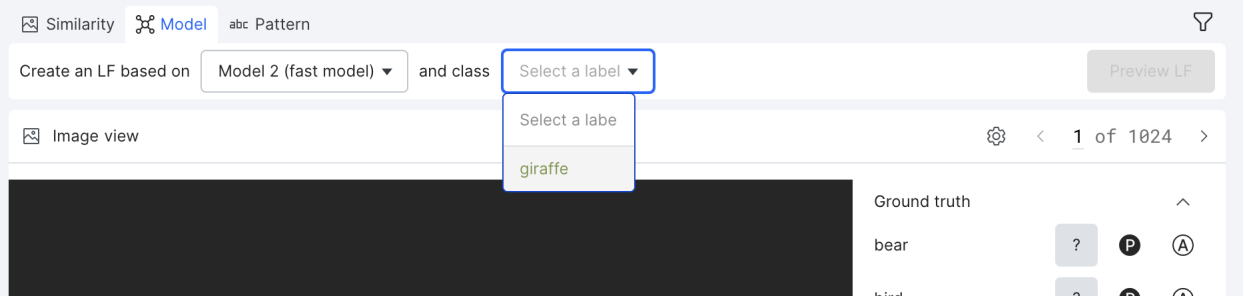
Model LFs

The model-based LF tool allows you to select a pre-trained model and an input class and outputs the model's predictive inference for the class. Results are based on the scikit-learn linear regression decision function score.

To create a model-based LF:

1. Prerequisite: Have a model that is already trained on a class. See Model training for information about how to train a model in Snorkel Flow.
2. Select the **Model** tab on the LF composer toolbar.
3. Select a pre-trained model and the class that the model was trained on.
4. Click **Preview LF**.
5. Use the similarity histogram to adjust the thresholds for the images that you would like to label with this LF.
6. Once you are happy with the thresholds that you have selected, click **Create LF**.

Copyright © 2025 Snorkel AI, Inc. All rights reserved



Pattern LFs

The pattern-based LF tool is a native feature in Snorkel Flow that creates LFs based on user-defined rules (e.g., keyword or regex match) against text data that is associated with your images.

To create a pattern-based LF:

1. Select the **Pattern** tab on the LF composer toolbar.
2. In the textbox, type **/** to view the list of LF builders that are available.
3. Fill out the required fields on the builder that you selected.
4. Click **Preview LF**.
5. Use the similarity histogram to adjust the thresholds for the images that you would like to label with this LF.
6. Once you are happy with the thresholds that you have selected, click **Create LF**.

For more information about creating pattern-based LFs, see Introduction to search based LFs. For more information about the types of pattern-based LFs that are available, Pattern based LF builders.

Model training

Image Studio supports a fast model variant, which supports the following configurations to train a model at a much faster speed than full models:

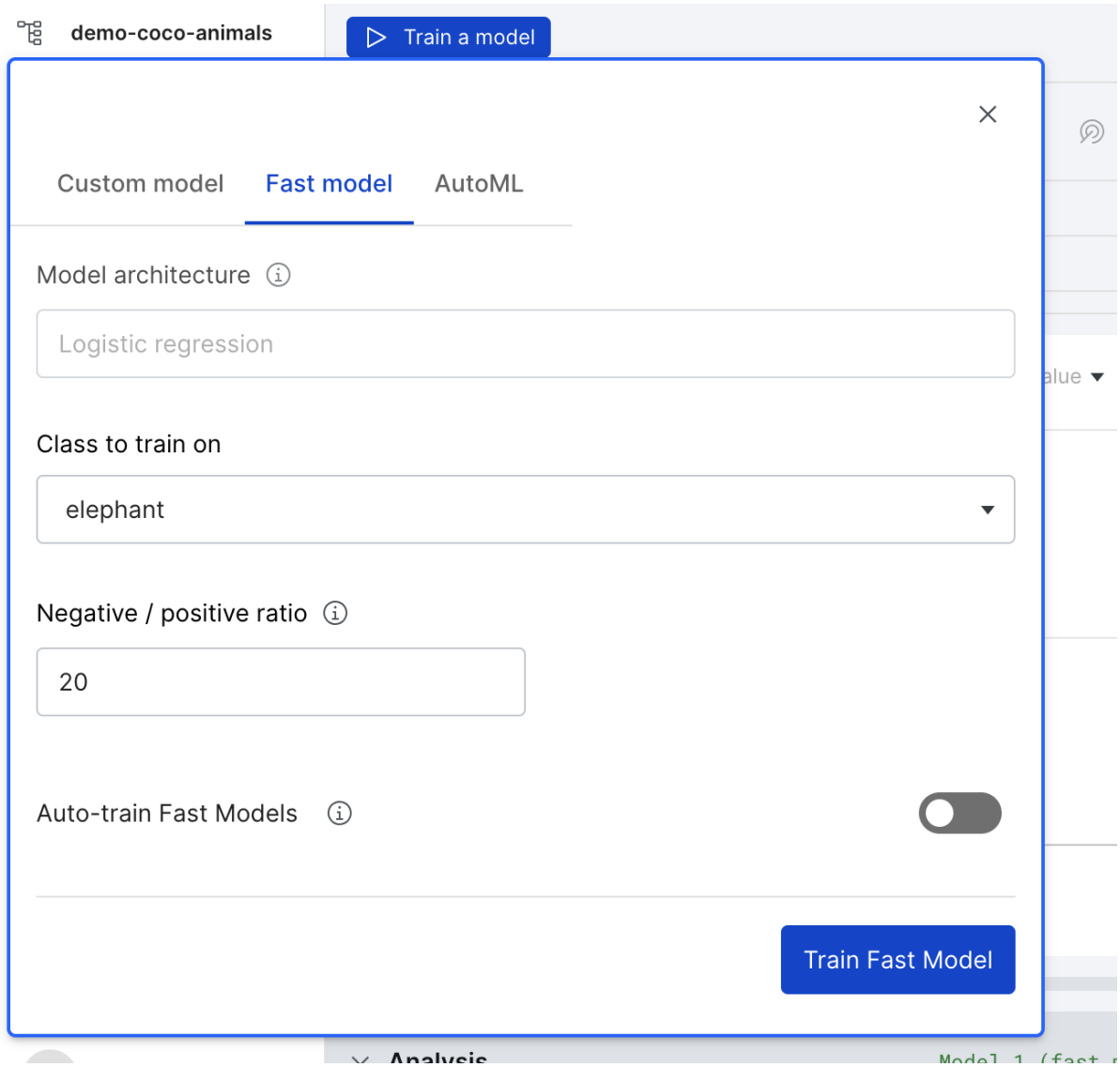
- Training on a specific class.
- Training with a training set sample, configured by a negative-to-positive numeric ratio.

Fast models are currently only supported for the **logistic regression** architecture, and currently only train on the valid data split.

To train a fast model:

1. In the **Models** pane, click **Train a model.**
2. Select the **Fast model** tab.
3. Specify the following options:
   - **Class to train on**: The specific class for which you would like to train your model.
   - **Negative / positive ratio**: The negative to positive ground truth sampling ratio to use in the training dataset.
   - **Auto-train Fast Models**: Turn this on to automatically train a fast model on a sample of your data when your set of active LFs changes.
4. Click **Train Fast Model**.

Copyright © 2025 Snorkel AI, Inc. All rights reserved



While you are waiting for you model to complete, you can continue to explore your data and create LFs.

Once your model is finished running you can view the results in the Models pane and the Analysis pane. Both panes provide charts, metrics and guidance on how to refine your models. For more information about the Models pane, see View and analyze model results. For more information about the Analysis pane, see Analysis: Rinse and repeat.

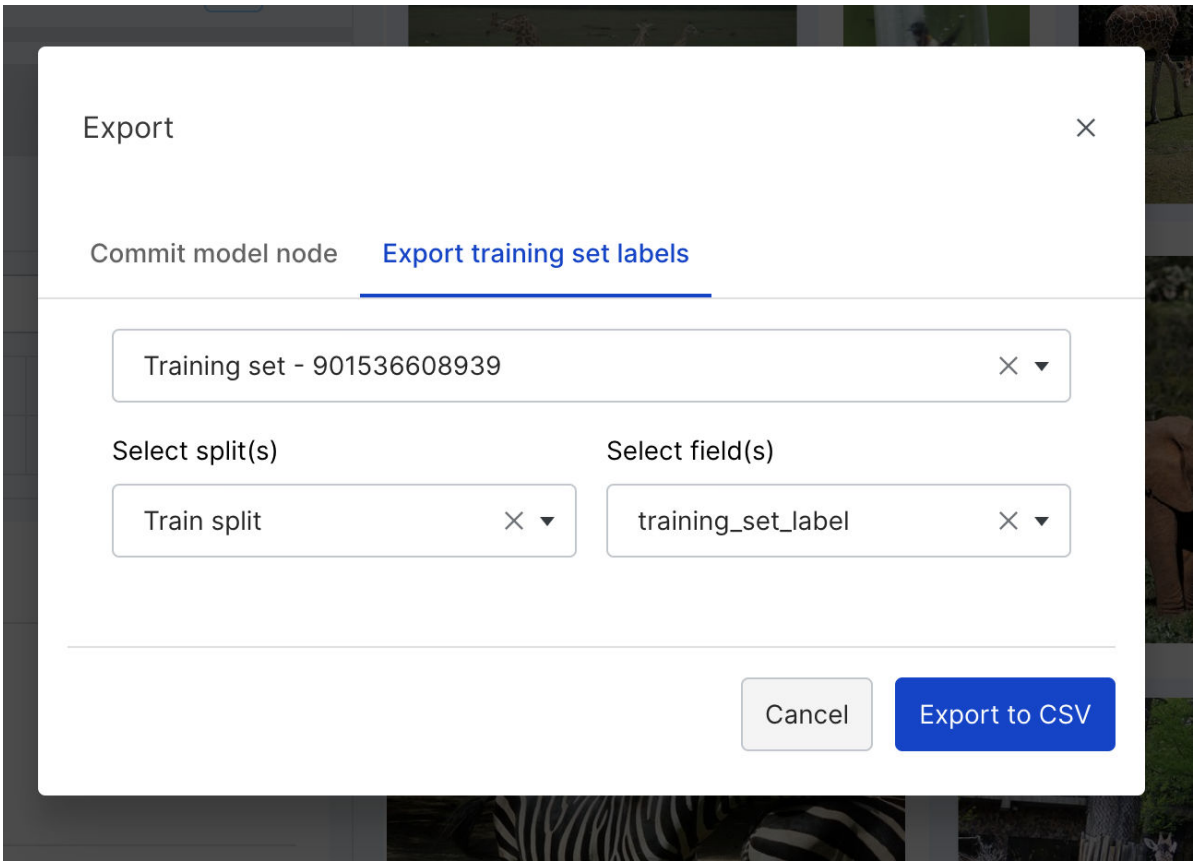
> **NOTE**
> To see model results, you must select the valid split.

Export labels

At any point you have the option to export your labels to a CSV file:

1. In the top right corner of your screen, click **Export**.
2. Click **Export training set labels**.
3. Specify the training set, splits, and fields for which you want to export labels.
4. Click **Export to CSV**.

Copyright © 2025 Snorkel AI, Inc. All rights reserved



Copyright © 2025 Snorkel AI, Inc. All rights reserved

Operators: Transform and process your data

In Snorkel Flow, operators perform transformations over DataFrames. Consider an application where you want to classify emails as SPAM or HAM (not spam). Before you begin your modeling work, you can clean up some of the data to make it easier to work with.

For example, emails might have a lot of whitespace and odd formatting that makes them hard to work with. As a result, you might want to trim whitespace in the main body of each email. With your email data is in a DataFrame, where the body is represented in a column called `text`, you can write labeling functions and train a model. Finally, you can add a post-processor to filter out all SPAM emails and leave only non-spam ones.

Each of these steps can be performed with an operator. Internally, operators act over Dask DataFrames as input, perform transformations, and output a Dask DataFrame.

Snorkel Flow has a series of built-in operators to perform commonly used transformations, and also allows users to write their own user-defined operators. Using Application Studio, you can commit operators to different nodes in the application graph to compose end-to-end applications.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

Applications and operators: Build end-to-end AI applications

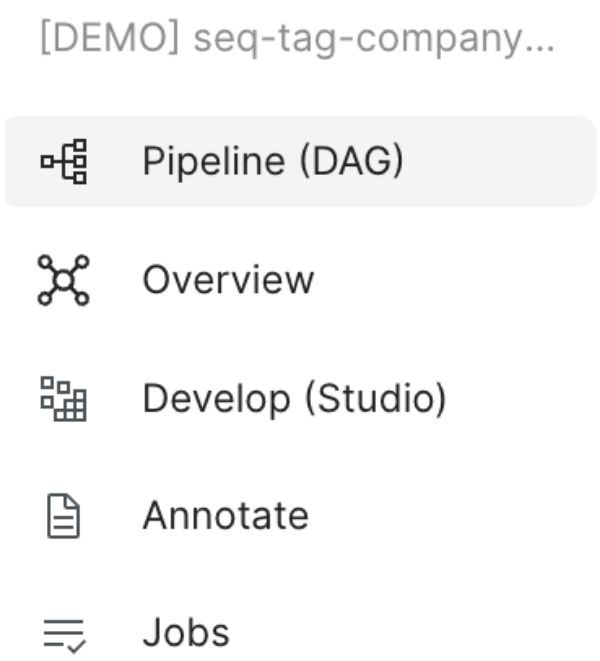
An application is an end-to-end pipeline of data frame transformations implemented as an operator (see Operators) graph. You can use Application Studio to instantiate operator blocks or iterate on individual operators. Once you've finished building your graph, you can package and export the application to make it available for deployment.

This article explains the process for customizing your application.

Application creation

Snorkel recommends creating an application with the guided application creation workflow. Alternatively, you can create an application using templates. This flow is more customizable as described below.

After you create your application, you can customize the application by modifying the Directed Acyclic Graph (DAG) on the **Pipeline (DAG)** page from the left panel.



Instantiating nodes from application templates

If you created your application based on an application template from the home page, you will see a corresponding block of nodes Application Studio.

Committing operators to nodes

Each node in the operator DAG is a "placeholder" with an expected operator type (indicated by the dotted black outline). You may fill a node by *committing* a specific operator to the graph. Once an operator is committed, it will be styled accordingly (see the legend in the top-right of the view).

Copyright © 2025 Snorkel AI, Inc. All rights reserved

Heuristic operators

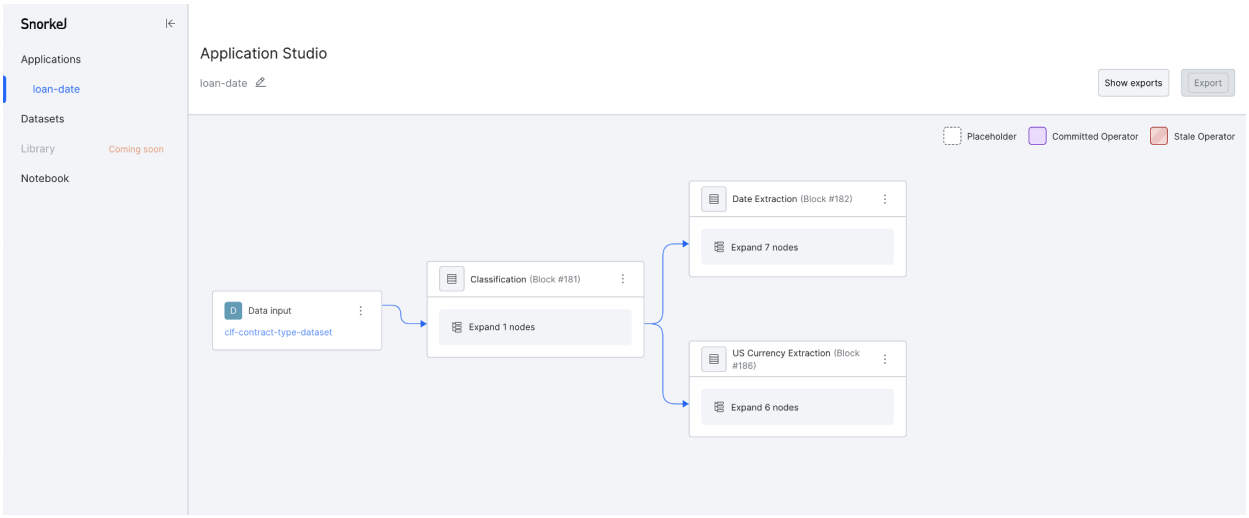
To commit a heuristic (or non-model) operator, click on the corresponding node. You'll see a search box and dropdown listing all operators with a matching operator type. Select the desired operator, and fill out corresponding arguments to commit.

Model operators

Models are *learned* operators — they need to be trained using a Snorkel-generated training dataset. To commit a model operator, click on the desired node with the dotted outline. This will take you to the Overview page, where you'll be able to iteratively develop the model by labeling a dataset, training models, and performing error analysis. Once you're happy with a model that you've built in this ML model development loop, commit the corresponding model from the Train page.

Compose complex operator graphs

If you'd like to compose complex operator graphs in Application Studio, you will need to enable an advanced setting.



To do this: click your user name in the bottom left > "User settings" > "Advanced" > Enable "Application Studio advanced mode". Once this toggle has been enabled, you will be able to build more complex applications using the GUI:

- Add new blocks based on application templates by clicking on the "more options" icon on each block.
- Add featurizers prior to a node.
- Add arbitrary nodes after the current node.

Refreshing stale operators

An operator may become "stale" if you update an upstream operator. Application Studio will detect and indicate stale operators (see the legend in the top right of the view). When an operator is stale, you can refresh its underlying input dataframe by clicking the Refresh icon.

Warning

If you refresh a stale operator, the rows in the dataframe will update in place. You will need to re-tune labeling functions and models correspondingly.

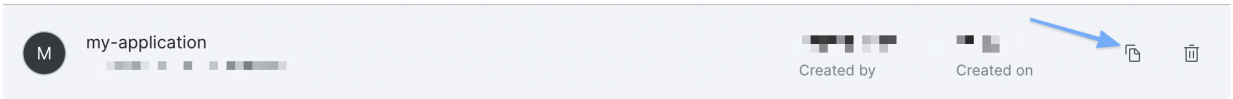
Copyright © 2025 Snorkel AI, Inc. All rights reserved

Deploying completed applications

Once you've finalized your application and committed an operator to all nodes, you can export the application. This will package your application and make it available for deployment. See Deploying Snorkel-built models for more information about how deployments.

Copying blocks

As part of your iterative process, you may need to modify high-level information about your application, such as specifying new labels or new label columns. To do this, create a copy of your application from the **All Applications** screen.



Within Application Studio, you can enable DAG editing by toggling the dropdown menu in the top right from READ to EDIT. You should then see a three dot menu in the top right of each block in the DAG.

Clicking on **Copy block** within that menu will open a dialog which allows you to configure new settings – custom ground truth columns, new labels, and renaming existing labels. Under **Advanced options**, the checkbox **Import LFs/labels/tags** will copy over any existing LFs from the existing block. You'll then be able to interact with the operators within this block as normal.

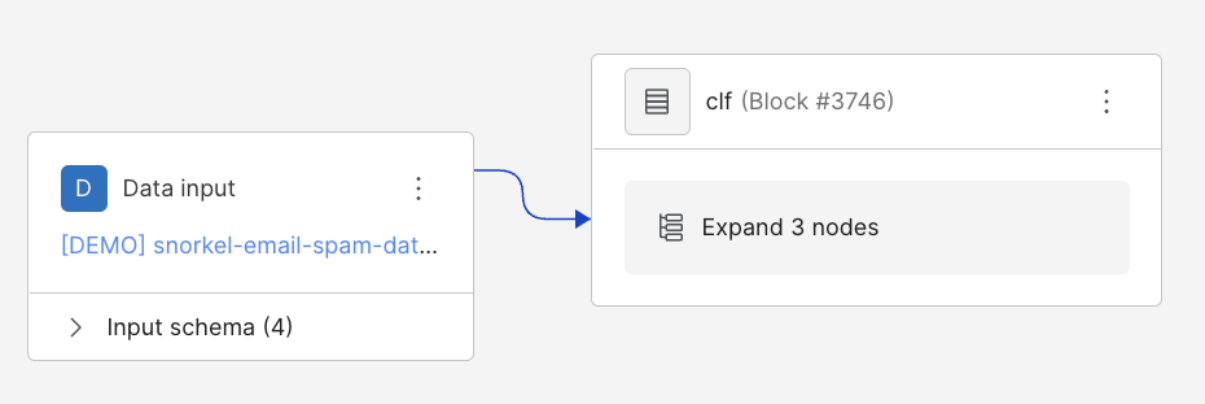
Copyright © 2025 Snorkel AI, Inc. All rights reserved

Adding a Data Table Viewer

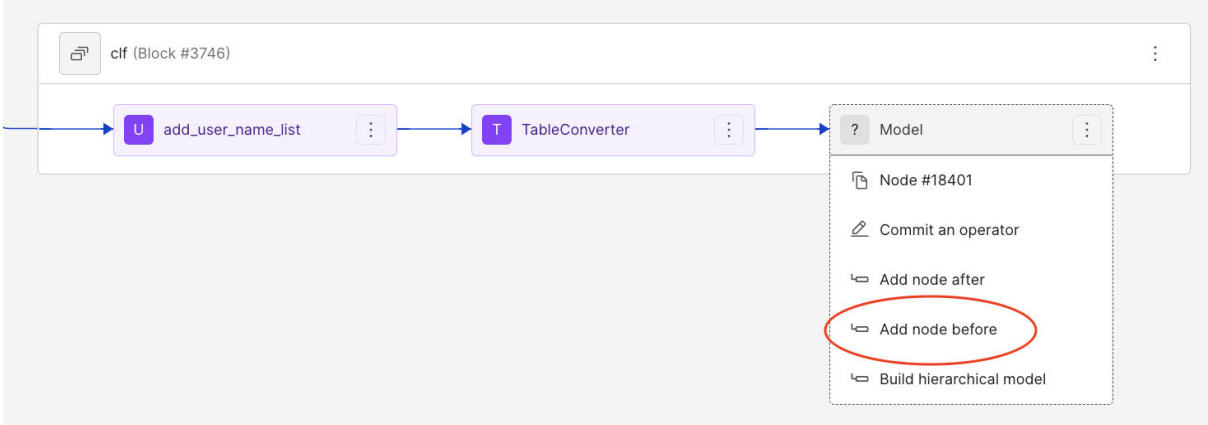
The `TableConverter` operator is a powerful tool in Snorkel Studio that allows you to represent complex, nested data in an easily readable table view. If your dataset includes columns with nested data (like arrays of dictionaries), the `TableConverter` helps you "flatten" this data into a more accessible format. This is especially useful for fields like user interactions, product attributes, or any data that naturally groups multiple related items.

Configuring the `TableConverter` Operator

1. Navigate to the Application Studio and expand the desired node.

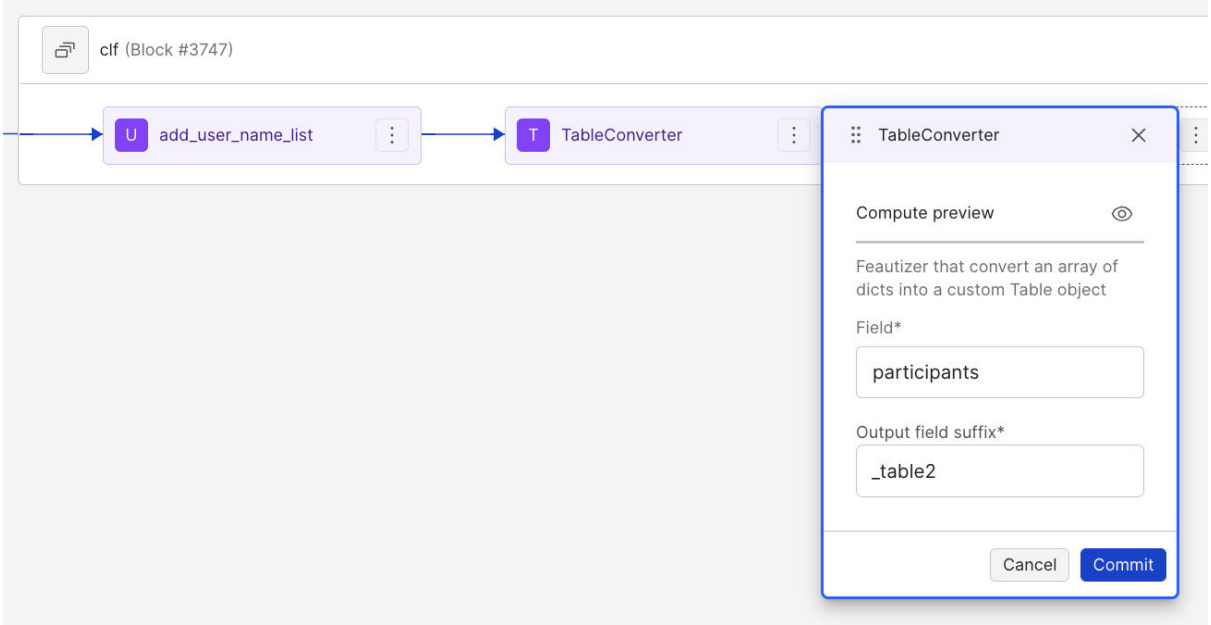


2. Ensure you are in **Edit** mode (use the mode selector in the top right corner of the Application Studio).
3. Add a new node by selecting the three vertical dots on the node labeled **Model**, and then select **Add node before**.
4. When prompted, select `ChangeColumns`.



5. Commit the `TableConverter` operator to the node:
   - Open the node's menu (three dots) and select "commit an operator".
   - Search for and select "TableConverter".
6. Configure the `TableConverter` fields:
   - Select the column containing the array of dictionaries.
   - Specify an output suffix to name the new table column.

Copyright © 2025 Snorkel AI, Inc. All rights reserved



7. Commit the changes and refresh the model node if prompted.

Displaying the Table View

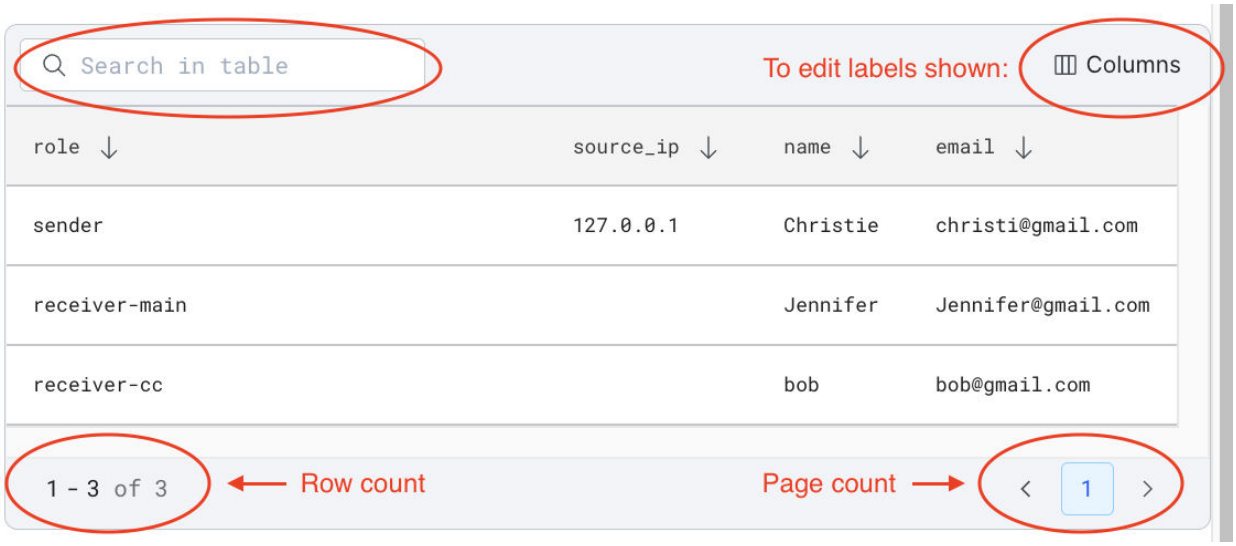
1. Return to the Application Studio.
2. Open the data viewer settings (gear icon).
3. Click "Select displayed columns" and ensure the new table column is checked.
4. Scroll the data viewer to find your newly generated table.

Using the Data Table Viewer

The table view offers several features to enhance data exploration:

Copyright © 2025 Snorkel AI, Inc. All rights reserved

- **Column Selection**: Use the column button in the top right corner to customize displayed columns.
- **Pagination**: Navigate through data pages using arrows in the bottom right corner.
- **Search**: Use the top-left search bar to filter data based on any visible label.
- **Row Count**: View the total number of rows in the bottom right corner.
- **Sorting**: Click on column headers to sort data. Use arrows to toggle ascending/descending order.



Example Use Case

In this documentation, we use an email classification dataset (HAM vs. SPAM) with a `participants` field to demonstrate the `TableConverter`. Your specific use case may vary depending on your dataset structure.

Tips for Effective Use

- Ensure your target column contains an array of dictionaries before applying the `TableConverter`.
- Choose a meaningful output suffix to easily identify your new table column.
- Experiment with different column combinations in the table view to gain insights into your data.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

Built-in operators

In Snorkel Flow, operators perform transformations over DataFrames. Snorkel Flow includes a collection of built-in **operators** to perform commonly used transformations.

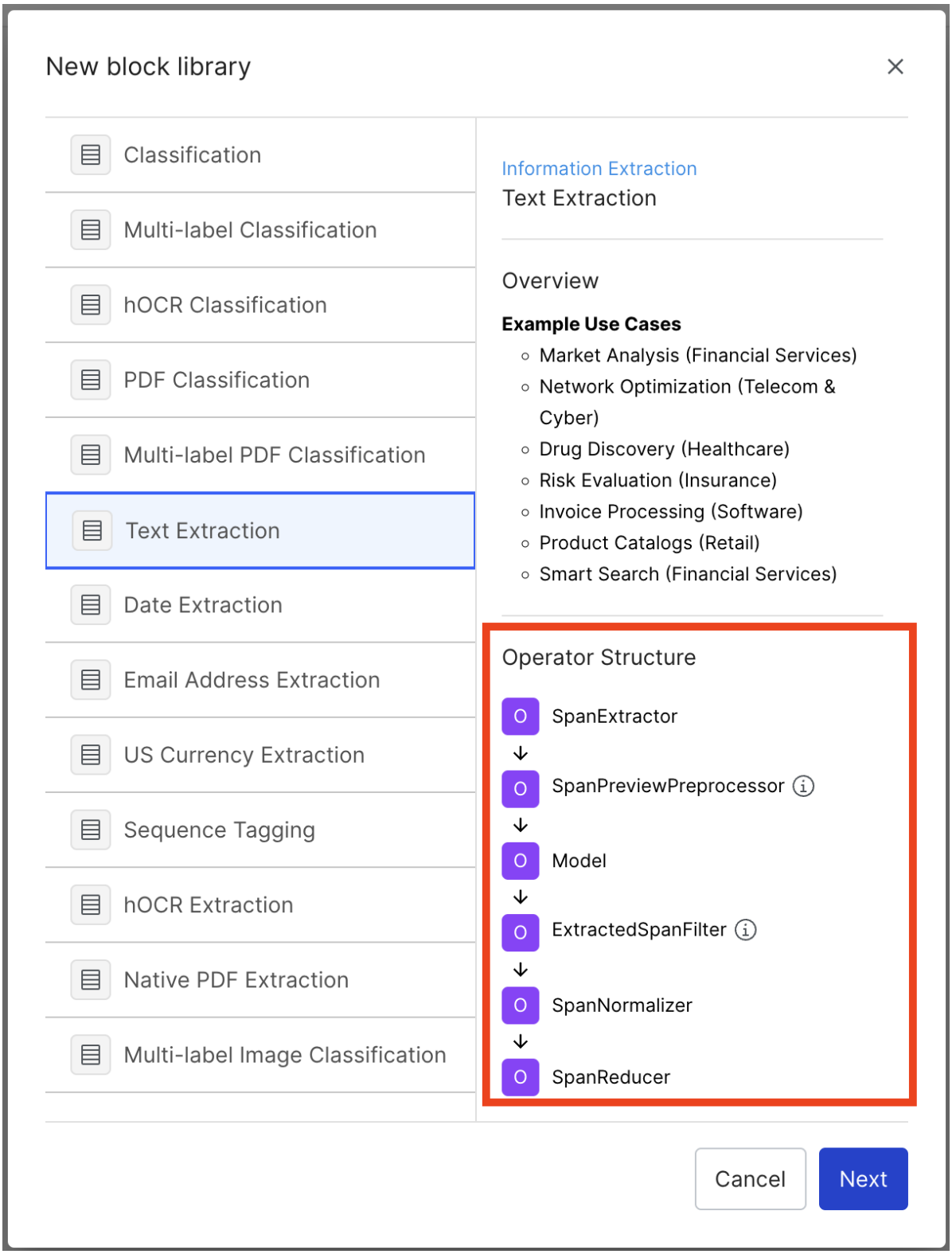
Pipelines that transform DataFrames in Snorkel Flow are constructed via a combination of blocks, where a **block** is a set of pre-defined operations that perform transformations sequentially.

For example, the text extraction block shown below contains 6 operators: **SpanExtractor**, **SpanPreviewPreprocessor**, **Model**, **ExtractedSpanFilter**, **SpanNormalizer**, and **SpanReducer**.

Some operators run automatically while others (e.g., SpanExtractor, Model) require input from the user.

For example, SpanExtractor is a placeholder for selecting a specific span extraction operator for a user's task (e.g. NumericSpanExtractor, DateSpanExtractor).

Copyright © 2025 Snorkel AI, Inc. All rights reserved



See a full list of built-in operators.

Further detail on each Built-in operator below.

Featurizers

Featurizers are operators that add one or more columns, all known as features, to a DataFrame.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

Text-based

- **Truncate**: truncates any field to a maximum token length. By passing the truncated column a name, you can keep the original column as well.
- **Whitespace**: converts all non-standard whitespace characters into whitespace. By default, the following UTF-8 non-standard space characters are used with the regular space: U+00A0, U+2000 to U+200A, U+202F, U+205F, U+3000. These can be overridden using the `to_replace` argument.
- **SpaCy**: applies the spaCy model to a field and produces a new target field containing the jsonified spaCy Doc object. For large datasets of strings, this takes longer to execute.
- **RegisterCustomEmbedding**: lets you register an existing embedding column, which enables access to Cluster View. The RegisterCustomEmbedding featurizer has a few key required inputs:
  - The **Embedding Operator Type** field specifies the type of embedding: EmbeddingFeaturizer for text embeddings, and the EmbeddingCandidateFeaturizer for sequence tagging embeddings.
  - The **Embedding Vector Field** specifies which column contains the embedding vectors. Every row of data should have embeddings and should have the same dimension. For example, all rows should be a list of X floats. If not, an error is shown with remediation steps.
  - The **Embedding Source Field** specifies which column the embeddings were computed over. Snorkel Flow requires a source field to compute certain summaries in **Cluster View**.
  - The **Embedding Source Candidate Field** is optional, specifically for sequence tagging embeddings. This references the context of the sequence tagging task, while the Embedding Source Field references the spans that have been embedded.
    This field is a beta feature for candidate embeddings in sequence tagging tasks. If you ned this feature, contact Snorkel.

The RegisterCustomEmbedding featurizer runs a set of checks on the vector column to ensure it is valid. These checks include that the vectors are all the same dimension, contain a list of floats, and are not null.

> NOTE
>
> The RegisterCustomEmbedding featurizer isn't generating new embeddings. It registers existing embeddings for use within Snorkel Flow.

Model-based

The model and model-based featurizers run predictions over the input data based on the specified feature fields and store those predictions into a prediction column. This featurizer helps to build Model-based Labeling Functions over your data.

In addition, the model-based featurizer can automatically train on any ground truth contained in your data, allowing it to act like a typical supervised model to improve your Label Model.

Model postprocessors

Model postprocessors are useful to post-process or update model predictions. Any postprocessors should be added to the Directed Acyclic Graphs (DAG) after the model node, in the same block as the model node.

- **SpanFilterByLengthPostProcessor**: filters and removes positive spans with length <= the provided `span_length`.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

- **SpanRegexPostProcessor**: labels anything that matches the provided pattern with the provided label.
- **SpanRemoveWhitespacePostProcessor**: removes leading and trailing whitespace in positive spans.
- **SubstringExpansionPostProcessor**: if a prediction boundary fall mid-token, then it expands predictions to the token boundaries.
- **SpanMergeByRegexPatternPostProcessor**: if the negative label text in between the spans matches a regex pattern, merges nearby spans with the same label.
- **SpanMergeByNumberCharacterPostProcessor**: merges nearby spans with the same label, if the negative label text in between the spans is in the lower to upper number of characters (inclusive).

Time-based

TimestampFeatureExtractor: parses a timestamp column (`pandas.Timestamp`), and adds new columns for the following features: day of the year, day of the week, year, month, day, hour, and minute. These features are useful for building labeling functions or for model training.

PDF Operators

PDF Parsers

We use parsing operators to extract text and its corresponding location in the PDF, and store this information using Snorkel's internal RichDoc representation.

There are two kinds of PDFs—native and scanned. Native PDFs are digitally created (for example, exported from Word) and have metadata that can be read. Scanned PDFs are image-based and do not have this metadata. Scanned PDFs require additional preprocessing as discussed in Optical Character Recognition (OCR).

For datasets containing native and scanned documents, we recommend that you select the scanned PDF workflow.

- **PDFToRichDocParser**: parses native PDFs. The input to this operator is the column containing the PDF url. This operator uses a PDF parsing library to extract the text and properties of the text, including the bounding box, paragraph, and page in which the text is present. The input PDF should not have any character or font encoding issues for the operator to function correctly.
- **HocrToRichDocParser**: parses scanned PDFs. This operator relies on the outputs of an OCR model to extract the text and location information from images of documents. The input to this operator is a column containing OCR results in the standard hOCR format.

The parsing operators add the following fields to the dataframe:

- *rich_doc_text*: contains the raw text from the PDF file. This field extracts candidate spans.
- *page_docs*: list of RichDoc objects corresponding to each page.
- *page_char_starts*: list of start character offsets (the index of the first character in the page in the `rich_doc_text`) corresponding to each page.
- *rich_doc_pkl*: serialized object that corresponds to `rich_doc_text`.

The PDF classification and candidate- and word-based PDF extraction workflows support these parsers.

Optical Character Recognition (OCR)

Scanned PDF documents require the use of Optical Character Recognition to convert images to machine-encoded text. Users may run any OCR library outside the Snorkel platform, if the output is in the

Copyright © 2025 Snorkel AI, Inc. All rights reserved

standard hOCR format. Snorkel Flow includes built-in libraries to run OCR. These operators are run before the **HocrToRichDocParser**:

- **TesseractFeaturizer**: uses the open-source tesseract library to extract OCR. The operator takes PDF URLs as input and outputs a hOCR string.
- **AzureFormRecognizerParser**: uses Azure Form Recognizer. The user is required to set up an account and provide their own form recognizer credentials. The operator takes PDF URLs as input and outputs hOCR.

For more information please see the Scanned PDF guide.

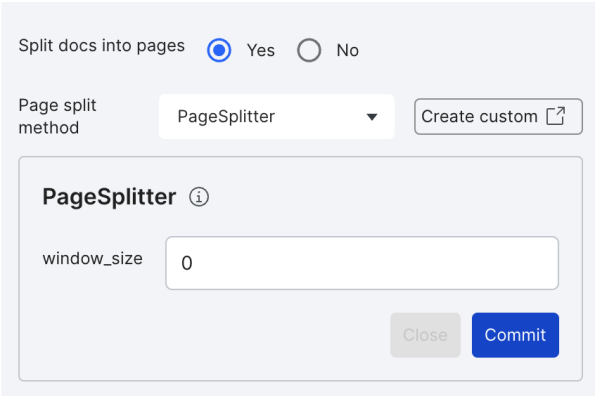
The PDF classification and candidate- and word-based PDF extraction workflows support these parsers.

PageSplitter

The **PageSplitter** operator splits PDF documents into sub-documents or pages. This function allows Snorkel Flow to store data and compute labelling functions more efficiently. However, it restricts the context available when writing the labeling function (LF) to the page or sub-document.

For performance reasons, the PageSplitter is strongly recommended for large documents. Split any documents that are greater than 20 pages. Snorkel Flow filters out documents that are larger than 100 pages.

To use the default PageSplitter, select the PageSplitter when setting up your application:



The window size specifies the number of pages before/after the current page that can be used as context for writing LFs. Use a window size of fewer than 10, and prevent window sizes greater than 50.

Snorkel Flow also supports the use of user-defined custom PageSplitters. For example, you can split documents based on sections. For more information, see the custom operator documentation.

The PDF classification and candidate-based PDF extraction workflows support these parsers. By default, the word-based workflows come with a page-splitter operator.

Candidate Span Extractors

PDF information extraction applications require users to define candidate span extractors. This treats an extraction problem as a classification problem. Span extractors are defined by using a heuristic **and/or** model over the **rich_doc_text** field. This operator converts the dataframe from having a document or page in each row, to having a span in each row. For more information on built-in extractors, see Extractors.

Only candidate-based PDF extraction workflows support these operators.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

PDF Featurizers

The following built-in featurizers add columns to the dataframe that are useful for rendering documents and defining labeling functions in **Develop (Studio)**:

- **RichDocPagePreprocessor**: filters out RichDoc pages that do not contain any candidate spans. This is useful for rendering only the relevant pages in **Develop (Studio)**.
- **RichDocSpanBaseFeaturesPreprocessor**: computes basic RichDoc features for each span, such as the bounding box coordinates, page number, and font size.
- **RichDocSpanRowFeaturesPreprocessor**: extracts row-level features such as the previous row, next row, and the row header (text to the left and top of the span that is closest to the span).

Object Detection Featurizers

Detecting layout features from the PDF image can be useful for defining labeling functions and extracting certain entities. The following built-in object detection operators are available in Snorkel Flow:

Line Detection

- **LinesFeaturizer** extracts all visual horizontal and vertical lines from a specified pdf page. This operator uses morphological operations. The output from this operator can be used to filter pages or combine words between lines with the **TextClusterer** operator.

Checkbox Detection

To enable checkbox detection in PDF documents and use the checkbox-related features when writing label functions, add the **CheckboxFeaturizer** and **CheckboxSpanMapper** operators to the application DAG.

CheckboxFeaturizer detects checkboxes in PDF documents. This operator also detects if the checkboxes are checked or not based on the threshold ratio (1.0 - 0.0). You must specify the minimum and maximum range of the checkboxes to detect. To use the detected checkboxes when writing LFs, you must add the **CheckboxSpanMapper** operator to the DAG.

Based on the provided criteria, **CheckboxSpanMapper** maps the detected checkboxes to the already extracted spans in the PDF document. You must specify the pixel distance and the four directions in which to map them. This operator must be added after the **RichDocSpanBaseFeaturesPreprocessor** operator in the DAG.

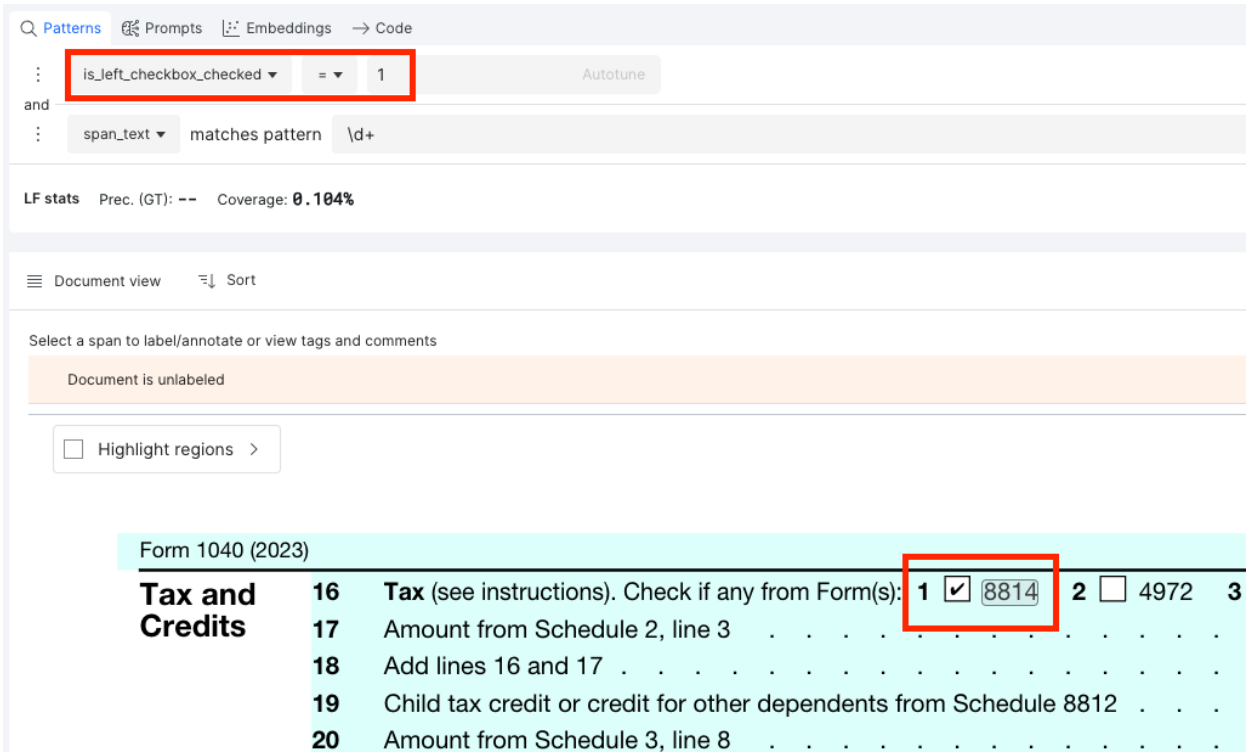
For each of the directions selected, a new column designating the status of the checkbox is added. For example, when left and top checkbox options are selected, the **is_left_checkbox_checked** and **is_top_checkbox_checked** columns are added with integer values. The columns will have one of the following values:

- **1**: The corresponding span has a checkbox detected in the specified direction within the distance. The detected checkbox is checked.
- **0**: The span has a checkbox detected in the specified direction within the distance. The detected checkbox is unchecked.
- **-1**: The span does not have any checkbox detected in the specified direction within the distance.

To define a labelling function using this field, use the Numeric Builder. For example, if the user wants to detect numerical spans with a checkbox checked on the left side of it, the label function can be a combination of the following conditions:

Copyright © 2025 Snorkel AI, Inc. All rights reserved

- **is_left_checkbox_checked = 1**: Check if the span has a checkbox mapped on the left side of it and if the checkbox is checked.
- **span_text matches pattern \d+**: Using regex if the span has only numbers in it. This is an example condition. Other conditions can be used as needed.



Layout Detection

- **DocumentLayoutFeaturizer**: computes layout features in a document using a deep learning model from the Layout Parser library. The Model dropdown can be used to select models from the library. Resolution specifies the input resolution of the image in DPI. Confidence threshold specifies the threshold used to filter the predictions. The **Pages** field optional input can be used to run the deep learning model on a subset of pages.
- **TableRowFilter**: filters out rows without tables. This requires the outputs from the **DocumentLayoutFeaturizer** operator.

Table Detection

The Table Transformer model detects tables and table structures. This featurizer is implemented using the following operators:

- **TableFeaturizer** operator computes the model predictions over PDF pages and saves the outputs in the **tables** column. Add this operator after the PDF parser node in the Application DAG. The operator has the following inputs:
  - *field*: column with the PDF URL path.
  - *model*: pretrained model name. The user can select any pretrained model that is compatible with the TableTransformerForObjectDetection class. Start with `microsoft/table-transformer-detection` or `microsoft/table-transformer-structure-recognition`.
  - *pages field*: runs the model over a subset of pages in the document. If the document has been split using PageSplitter, enter **context_pages** for this field.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

- **TableSpanMapper** operator maps the table predictions to spans. Add this operator before the **Model** node. Add this operator after the **RichDocSpanBaseFeaturesPreprocessor** operator in the DAG. The operator takes the model **confidence threshold** as input. This is set to 0.9 by default. It outputs the following fields to the dataframe:
  - *is_table_span*: an integer value indicating if the span was within a table. It is set to 0 if outside all tables, and 1 if span is within a table.
  - *table_column_id*: an integer value indicating which column the span belongs to.

To define a labelling function using the fields, please use the Numeric Builder. This can be used to encode logic similar to the example below:

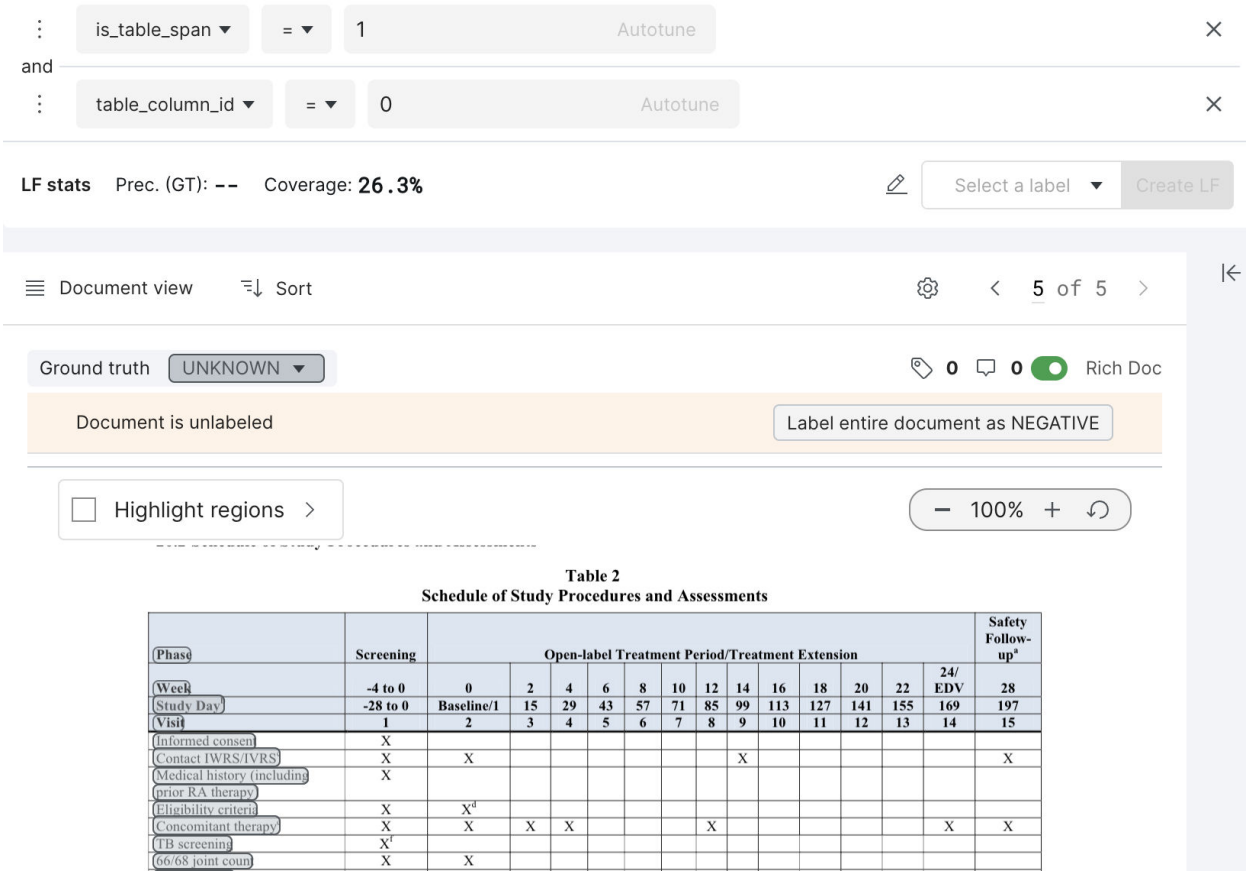


Table 2
Schedule of Study Procedures and Assessments

Phase	Screening	Open-label Treatment Period/Treatment Extension												Safety Follow-up[a]
Week	-4 to 0	0	2	4	6	8	10	12	14	16	18	20	22	24/ EDV	28
Study Day	-28 to 0	Baseline/1	15	29	43	57	71	85	99	113	127	141	155	169	197
Visit	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Informed consent	X														
Contact IWRS/IVRS	X	X							X						X
Medical history (including prior RA therapy)	X														
Eligibility criteria	X	X[d]													
Concomitant therapy	X	X	X	X				X						X	X
TB screening	X[f]														
66/68 joint coun	X	X													

Filters

- **Label**: includes or excludes rows corresponding to specified label strings.
- **Label Int**: includes or excludes rows corresponding to specified label integers.
- **Pandas Query**: includes or excludes rows based on a pandas query.
- **Boolean Column**: includes or excludes rows based on the value of a boolean column.
- **Text Size**: excludes rows with text columns that are larger than specified size (in KB).

For text extraction blocks, select a filter that removes all candidates according to certain criteria:

- **Regex Span**: removes all candidates that match a regular expression.
- **Extracted Span**: filters all span rows with a negative prediction.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

Extractors

Text extraction blocks require at least one extractor over a field:

- **Noun Chunk**: yields all noun phrases according to spaCy.
  - Per spaCy's implementation, the noun chunker iterates over all base noun phrases in a document. A base noun is a noun phrase that does not permit other NPs to be nested within it.
  - Example: For the text `Hi, my name is Dr. Bubbles!`, this extractor returns two spans: `my name` and `Dr. Bubbles`.
- **Regex**: yields all matches for a given regular expression.
  - By default, the entire match is returned. You can specify capture group = 1.
  - For example, to extract only the span that matches what's inside the first group of parentheses. `r"$([0-9]+)"` with `capture group = 1` would extract only the `45` in `$45`.
- **Date**: yields all the dates in documents via a pre-defined regex.
  - The regex pattern is:

```
(?:[0-9]{1,2}(?:st|nd|rd|th)?\s*(?:day)?\s*(?:of)?\s*.?(?:
(?:Jan(?:.|uary)?)|(?:Feb(?:.|ruary)?)|(?:Mar(?:.|ch)?)|
(?:Apr(?:.|il)?)|May|(?:Jun(?:.|e)?)|(?:Jul(?:.|y)?)|(?:Aug(?:.|ust)?)|
(?:Sep(?:.|tember)?)|(?:Oct(?:.|ober)?)|(?:Nov(?:.|ember)?)|
(?:Dec(?:.|ember)?)),?.?\s*[12][0-9]{3})|(?:(?:(?:Jan(?:.|uary)?)|
(?:Feb(?:.|ruary)?)|(?:Mar(?:.|ch)?)|(?:Apr(?:.|il)?)|May|
(?:Jun(?:.|e)?)|(?:Jul(?:.|y)?)|(?:Aug(?:.|ust)?)|(?:Sep(?:.|tember)?)|
(?:Oct(?:.|ober)?)|(?:Nov(?:.|ember)?)|(?:Dec(?:.|ember)?)).?\s*[0-9]
{1,2}(?:st|nd|rd|th)?\s*(?:day)?\s*(?:of)?,?.?\s*[12][0-9]{3})
```

  - Example: for the text `Today is January 1, 2021. And tomorrow is 01/02/2021`, this extractor will only return the `January 1, 2021` span.
  - See the **Spacy NER** section for an alternative extractor for dates.
- **US Currency**: yields all US monetary values via a pre-defined regex.
  - The regex pattern is: `\\$([0-9]{1,3},([0-9]{3},)*[0-9]{3}|[0-9]+)(\\.[0-9][0-9])?`
  - Example: For the text `5 bucks (or $5.00) is actually €4.10.`, this extractor returns the span `$5.00`.
  - See **Spacy NER** for an alternative extractor for currency.
- **Spacy NER**: yields all matches for a given NER tag according to spaCy.
  spaCy's NER models were trained on the OntoNotes 5 corpus and can predict several span labels:
  - `PERSON`: People, including fictional
  - `NORP`: Nationalities or religious or political groups
  - `FAC`: Buildings, airports, highways, bridges, etc.
  - `ORG`: Companies, agencies, institutions, etc.
  - `GPE`: Countries, cities, states

Copyright © 2025 Snorkel AI, Inc. All rights reserved

- `LOCATION`: Non-GPE locations, mountain ranges, bodies of water
- `PRODUCT`: Vehicles, weapons, foods, etc. Excludes services
- `EVENT`: Named hurricanes, battles, wars, sports events, etc.
- `WORK_OF_ART`: Titles of books, songs, etc.
- `LAW`: Named documents made into laws
- `LANGUAGE`: Any named language
- `DATE`: Absolute or relative dates or periods
- `TIME`: Times smaller than a day
- `PERCENT`: Percentage (including `%`)
- `MONEY`: Monetary values, including unit
- `QUANTITY`: Measurements, as of weight or distance
- `ORDINAL`: `first`, `second`
- `CARDINAL`: Numerals that do not fall under another type

- **Token**: yields every token, given a selected tokenization strategy.

  Tokenization options include `spacy` (see implementation) and `whitespace`, which captures tokens by the regex `\S+`.

Entity classification blocks support another pre-processor that serves as an extractor and linker:

- **EntityDictSpanExtractor**: extracts all candidates according to a pre-specified dictionary of entities and links them to a canonical form. The dictionary should map each canonical form to a list of aliases: `{canonical_form: [alias1, alias2, ...], ...}`.
  This extractor includes these options:
  - `ignore_case`: whether to ignore casing when matching entities.
  - `link_entities`: whether to link aliases with their canonical form as the `span_entity`.

- **DocEntityDictSpanExtractor**: extracts all candidates according to a pre-specified dictionary of entities, filter by a dictionary of documents to expected entities, and link them to a canonical form.
  - The `entity_dict_path` specifies a dictionary to map each canonical form to a list of aliases: `{canonical_form: [alias1, alias2, ...], ...}`.
  - The `doc_entity_dict_path` specifies a dictionary to map each document UID to a list of expected span entities: `{context_uid: [entity_1, entity_2, ...], ...}`.

This extractor includes these options:

- - `ignore_case`: whether to ignore casing when matching entities.
  - `link_entities`: whether to link aliases with their canonical form as the `span_entity`.
- **EntityDictRegexSpanExtractor**: extracts all candidates according to a pre-specified dictionary of regexes and links them to a canonical form. The dictionary maps each canonical form to a list of regexes: `{canonical_form: [regex1, regex2, ...], ...}`.
  This extractor includes these options:
  - `ignore_case`: whether to ignore casing when matching entities.
  - `link_entities`: whether to link regex with their canonical form as the `span_entity`.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

- **TextClustererExtractor**: extracts all the horizontally and vertically clustered texts. Texts that are horizontally clustered form a span. The vertically clustered texts are extracted under region_text.

Normalizers and linkers

In extraction-based blocks, select **normalizers** or **linkers** that map the `span_text` to a more canonical `normalized_span` field.

- **Dates**: transforms all dates found in extracted spans into the `YYYY-DD-MM` format.
- **US currency**: transforms all US dollar values into a standard value by removing all special formatting (such as dollar signs, commas, and whitespace) and then casting the numeric value to float.
- **Capitalization**: transforms all string values to lowercase and only capitalize the first letter. This is helpful when working with candidates that refer to cities, names, etc.
- **Numerical**: transforms all numbers into words (e.g., `42` to `forty two`).
- **Ordinal**: transforms all ordinal values to words (e.g., `42nd` to `forty second`).
- **Entity Dict Linker**: maps existing candidate spans to a canonical form. This is similar to the above extractor, except it skips the tagging step.
- **Span-Entity**: assumes that normalized spans are already defined by the `span_entity` field.

Span-based

Under **Advanced Settings**, extraction blocks allow the following load-processors:

- **Span Joiner**: joins extracted candidates with their original contexts.
- **Span Previewer**: adds a field containing a localized preview of an extracted candidate. You can adjust the character window to change how many characters around the extracted span are included in the `span_preview` field.
- **Context Aggregator**: adds a `context_spans` column with aggregated spans for the current `cot_uid`. This can help express notebook LFs related to other spans in the same document.

Filters

The same filters specified in the Featurizers section are available as post-processors. These are useful if you want to filter examples based on the model prediction.

By default, extraction blocks also include an **ExtractedSpanFilter** to remove any negative span predictions.

Reducers

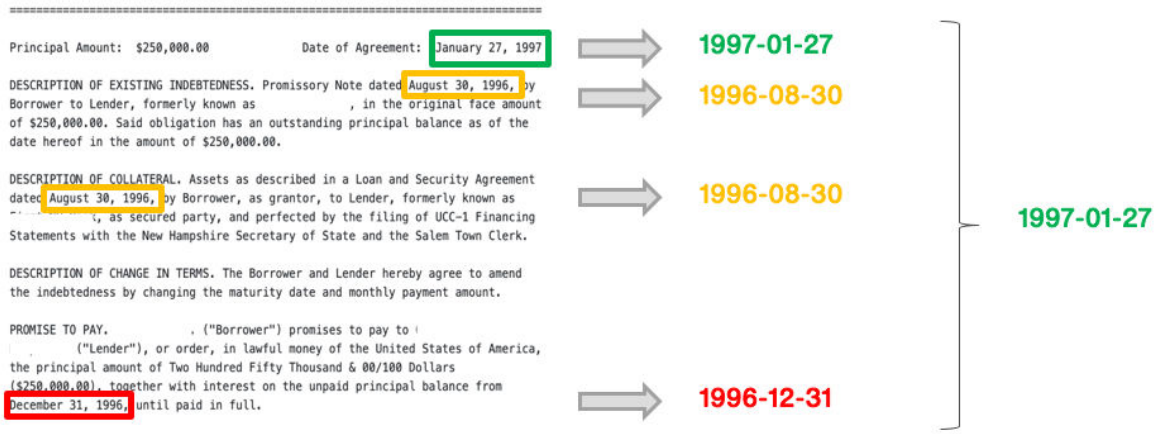
Reducers heuristically aggregate, select, or convert candidate-level predictions to become document or entity-level extractions. For instance, you might know for your text extraction block, you may expect one extraction per document. In this case, you apply a reducer that selects the most-confident span per document.

Similarly, in a company sentiment classification block over news articles, you may expect a single classification per document-entity pair. You can apply a most-common reducer to find the most common sentiment for a particular company in a document.

Snorkel Flow currently supports the following reducers in document/entity-level varieties:

Copyright © 2025 Snorkel AI, Inc. All rights reserved

- **Most Confident**: selects the most confident span-level candidate based on confidence measured by model probabilities.
- **First**: selects the first positive occurrence of an extraction in a particular document.
- **Last**: selects the last positive occurrence of an extraction in a particular document.
- **Most Common**: selects the most common candidate as the extraction for a particular document.



Miscellaneous

- **ColumnDropper**: drops one or more columns from your DataFrame.
- **ColumnRenamer**: renames one or more columns in your DataFrame. For example, `{"old_column_name": "new_column_name"}`.

Post-processors in extraction blocks

Filters, normalizers, and reducers are examples of post-processors that operate over model predictions. After applying post-processors in extraction blocks, you'll be able to see document or entity-level metrics. For an end-to-end example of this, see Information extraction: Extracting execution dates from contracts.

Embedded data table viewer

Enhance data exploration of your tabular data by utilizing the embedded table data viewer. This functionality visualizes any columns that have nested information.

Data format prerequisite

To be compatible with the `TableConverter` operator, the format for each row in the column to be converted should be a list of elements or dictionaries. All dictionaries also need to have the same key names and number of keys.

If converting from a pandas dataframe, use the sample code below. This code creates the list of elements for one row in your dataset:

```
filtered_df_per_row.to_dict(orient='records')
```

Copyright © 2025 Snorkel AI, Inc. All rights reserved

Example of a properly formatted list to be converted:

```
[{"name":"Christie","email":"christi@gmail.com","role":"sender",
"source_ip":"127.0.0.1"},
{"name":"Jennifer","email":"Jennifer@gmail.com","role":"receiver-
main","source_ip":""},
{"name":"bob","email":"bob@gmail.com","role":"receiver-cc","source_ip":""}]
```

Copyright © 2025 Snorkel AI, Inc. All rights reserved

User-defined operators

The Snorkel Flow Python SDK provides utilities to help you define operators using Python code. These user-defined operators be registered with Snorkel Flow and used to process your datasets through the web interface.

You can commit custom operators by selecting on a node in the application graph and searching by name.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

Custom operators (SDK)

All information relating to building custom operators can be found in the SDK documentation:

- Developing and registering custom operators
- Developing and registering custom extractors
- Developing and registering custom operator classes
- Using custom data points

You can find additional details about custom operators in the `snorkelflow.client.operators` documentation.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

Using dedicated resources to speed up applications with large datasets

What will I learn?

You will learn how enable dedicated resources in Snorkel to make your application faster. If you feel that your dataset is particularly large and is slowing down your app, consider using the *dedicated resources* feature to speed it up.

Dedicated resources caches your dataset in memory where it can be fetched and operated on quickly. To do this, Snorkel requires sufficient resources to cache your dataset in memory.

When would I need this?

Consider using dedicated resources if your app is has the following dataset size and characteristics:

- IE (sequence tagging) app, 2000 documents with 1000 tokens each
- PDF app, 50 documents or 15000 spans
- Multi-Label 2000 documents with more than 30 classes
- Classification app, more than 10000 data points

If your app feels sluggish or slow, consider using dedicated resources for a performance boost.

Who is eligible for this?

This feature is currently available for Snorkel Hosted customers.

If you are an on-prem customer with a *standard-sized* deployment, you can still attempt to use dedicated resources by following the guide below.

- The `studio-ray-worker` pod needs to have at least 8 CPU and 32 GB RAM, which is the default on a *standard-sized* deployment. If you do not have a *standard-sized* deployment, please contact your Snorkel representative to ask if you can use this feature.

If we cannot gather enough Dedicated Resources to enable the feature, Snorkel will display an error message that will tell you how many additional resources you need.

Please contact your Snorkel representative if you run into this message.

How do I use dedicated resources?

Step 1: Click the **Shared resources** button at the top left of your Data Studio.

Step 2: Click **Allocate dedicated resources**.

Step 3: Continue using the app normally while you see the spinner that says **Allocating resources...**.

Step 4: After at a minute, you'll see a green dot and **Dedicated resources** on the button.

Step 5: When you are done using the app, you can explicitly release resources by clicking **Release dedicated resources**.

After an hour of application inactivity, dedicated resources will release themselves back to the general pool.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

Troubleshooting

Because this is a Beta feature that is intended for Snorkel hosted customers, please reach out to your Snorkel representative if you run into challenges with using dedicated resources.

What if the text-box still says "Shared resources" after "Allocate dedicated Resources"?

You may be encountering this error because you don't have enough resources on your deployment. If other people on your team are actively using dedicated resources, please ask them to click the **Release dedicated resources** button if they are no longer using them.

Check if you see an error message that tells you how much additional memory and CPU is needed to provision dedicated resources, and how much memory and CPU is remaining.

Provide this information to your Snorkel representative for faster triage

Copyright © 2025 Snorkel AI, Inc. All rights reserved

Annotation Studio overview

Data annotation is the process of assigning labels or classes to specific data points for training datasets. For example, for a classification problem, you could assign banking contract documents to one of the following classes:

- employment
- loan
- services
- stock

Ground truth (GT) refers to the set of labeled and accurately annotated data that serves as a reference or benchmark for training and evaluating machine learning models. During the evaluation phase, use ground truth data to assess the model's performance. By comparing the model's predictions with the actual labels in the ground truth, you can calculate metrics such as accuracy, precision, recall, and F1 score.

Annotation Studio is divided into four sections:

- **Overview page:** Provides aggregate metrics on the number of completed annotations, the distribution of labels annotator agreement, and a view into recent annotator activity. See Overview page: View aggregate metrics for more information.
- **Review page:** Provides a full list of all annotations and their corresponding documents to easily review all annotations in one place. See Walkthrough for reviewers for more information.
- **Batches page:** Provides access to and information about all batches that have been created. You can create and manage batches and commit annotations to ground truth. See Create batches and Manage batches and commit ground truth for more information.
- **Within a batch:** Provides a canvas where annotators can view and label documents. See Walkthrough for annotators for more information about how to annotate in Annotation Studio.

Snorkel Flow provides multiple user roles that administrators can assign, each with different levels of data access and permissions. See Roles and user permissions for more information.

Why do we need to do manual annotation in Snorkel Flow?

While Snorkel Flow programmatically generates labels for your training data sets, you need to define your initial ground truth to develop labeling functions (LFs) and models. In addition, manual annotation allows for an iterative process of model development. Annotators can review model predictions, identify errors, and refine the labeled dataset. These manual annotations lead to improved model performance. With Snorkel, you can pinpoint where manual annotation is needed, which significantly reduces the time and money spent on manual annotation.

Typically, you'll want multiple annotators to review each data point for these reasons:

- **Reduce bias:** Manually labeling data is prone to bias because the person labeling the data may have preconceptions that can influence the labels they assign.
- **Handle ambiguity:** Some documents might be inherently ambiguous or open to interpretation. In these cases, having multiple annotators allows for capturing different viewpoints and addressing the inherent uncertainty in the data.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

- **Enhance robustness:** By aggregating annotations from multiple annotators, the labeled dataset becomes more robust and less dependent on the idiosyncrasies of any single annotator. This robustness is particularly important when dealing with diverse datasets or complex tasks.
- **Refine annotation guidelines:** Comparing annotations from multiple annotators identifies areas where guidelines need clarification. It also provides an opportunity for continuous training to improve consistency and understanding among annotators.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

Guideline goal

Annotation guidelines allow users to describe a phenomenon or concept as generally and precisely as possible. Good annotation guidelines are helpful for subject matter experts to annotate the concept at hand in any text without running into problems or ambiguity issues.

An iterative process

Developing annotation guidelines is an iterative process. Once a pilot annotation, or first draft, is created, its shortcomings still need to be identified and fixed. For instance, issues such as unclear label definitions can be identified in the first draft, and fixed for the second draft. The second draft is also looked over for further shortcomings, which repeats the revision process and leads to the most polished version of the pilot annotation.

Note

The most important idea about the iterative nature of the annotation process is that in each round, **multiple annotators independently annotate the same text**.

Pilot annotation

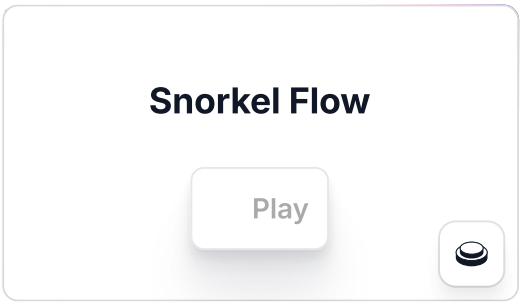
The first round of annotations is best done by annotators who are familiar with the data and context around the data used. As with subsequent annotation rounds, please annotate in parallel and discuss afterwards. It's not necessary to spend a lot of time on preparation. Specifying a list of references or theoretical works, or agreeing on a single text should be sufficient as a starting point.

Your time is spent best on discussing annotation disagreements. In particular in the very first round, many parameters are still undecided and likely to cause disagreement.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

Walkthrough for annotators

This page walks through the process of manually annotating documents in Annotation Studio. This walkthrough is designed for users with the Annotator role, who need to access Snorkel Flow to annotate the documents that are assigned to them. Below, you'll find an interactive demo showing how to annotate your data.



The instructions below further outline what you saw in the interactive demo.

Access your batches

After you log in to Snorkel Flow, you need to access your batches to start annotating. Batches are a collection of documents that are assigned to you to manually annotate. A batch is typically a subset of the total documents. With Snorkel Flow, the number of documents that need to be manually annotated is significantly smaller.

To access your batches, click **My work** on the left-side menu.

The **My work** page shows a list of batches for you to annotate:

Batch	Status	Annotators	Annotated	Size	Created
batch-2-1-3	In progress	ronnie.reviewer, alex.a...	593	978	01/23/2024
batch-2-1-4	In progress	ronnie.reviewer, alex.a...	597	820	01/23/2024
batch-2-1-2	In progress	ronnie.reviewer, alex.a...	592	815	01/23/2024
batch-1-1	In progress	ronnie.reviewer, alex.a...	1463	1952	01/23/2024

The following information about each batch is available:

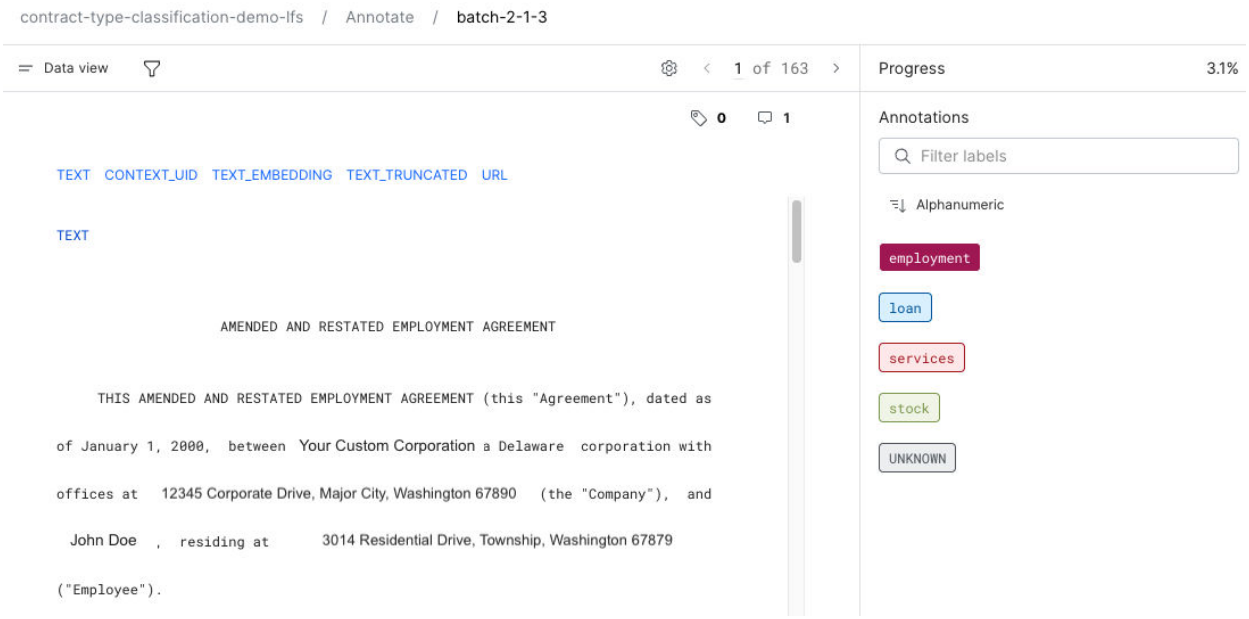
- **Batch**: The name of the batch.
- **Status**: Specifies whether annotations on the batch have not be started, are in progress, or have been completed.
- **Annotators**: The list of annotators that have been assigned to annotate the batch.
- **Annotated**: The number of documents in the batch that have been annotated.
- **Size**: The total number of documents that are in the batch.
- **Created**: The date that the batch was created.

Now, you can start annotating your documents!

Copyright © 2025 Snorkel AI, Inc. All rights reserved

Annotate your documents

Once you click **Annotate**, your screen will look similar to the image below:



The main canvas shows the document text. For classification and extraction applications, the right side-pane shows all classes that you can label the document.

Here is some information about the various buttons that you can see on your screen:

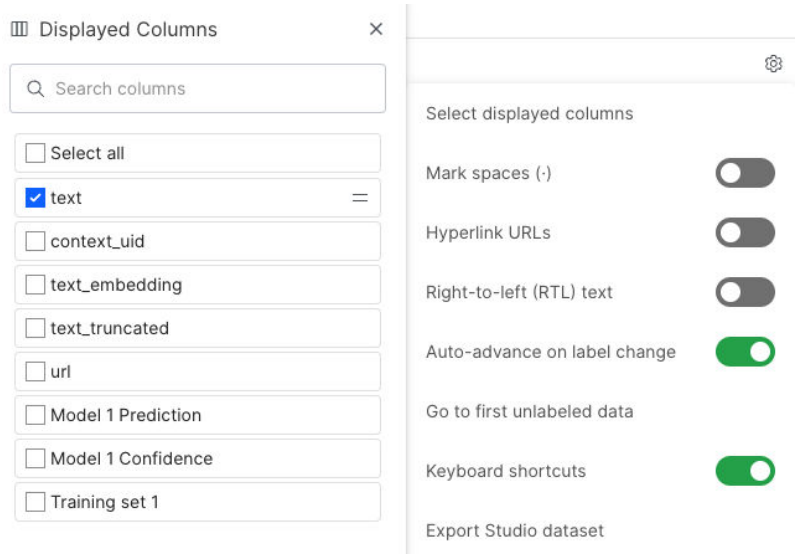
- Click **Data view** (or **Document view** for extraction applications) to change the appearance of the documents in the main canvas. The views that are available differ based on the application type.
- Click the filter icon to filter the documents that are shown to you.
- Click the gear icon to adjust various settings. We recommend that you adjust a few settings before you get started annotating.
- Click the arrow icons to page through the documents.
- Click the slice icon to add slices to the document.
- Click the comment icon to add comments to the document. For example, you can write a comment to explain your reasoning behind a selecting a particular label.

Adjust settings

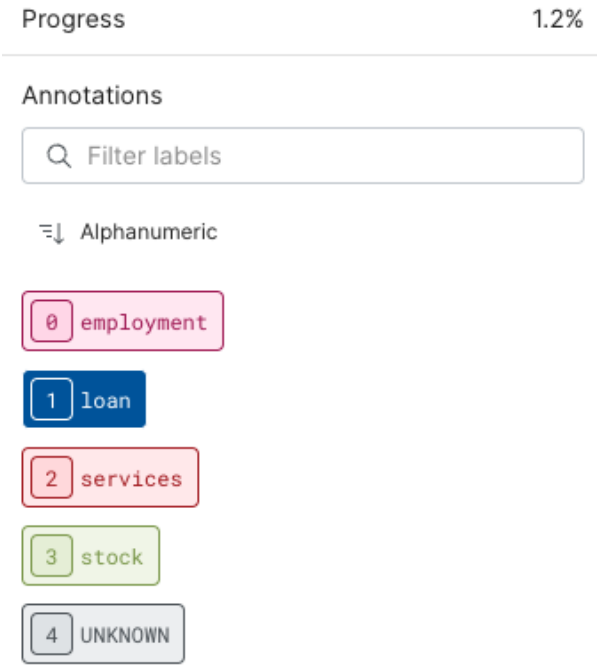
After accessing your documents, we recommend that you first adjust some settings to get some quality of life improvements while annotating! To change your settings, click the **gear** icon at the top-right corner of your screen. We recommend changing the following settings:

- **Select display columns**: This setting lets you select and display only the data columns that are necessary for marking your annotations (e.g., the text column). This removes clutter from the main canvas when you are reviewing documents.

Copyright © 2025 Snorkel AI, Inc. All rights reserved



- **Auto-advance on label change**: With this setting enabled, after you label a document, you automatically go to the next document. This prevents you from having to manually click the **arrow** > icons to page through the documents every time to view new, unlabeled documents. This setting is only available for single-label and text extraction applications.
- **Keyboard shortcuts**: With this setting enabled, keys are assigned to each label class, which makes it quicker to assign a label. In the example below, if you want to label a document "employment," then you just have to press 0 on your keyboard.



Here are descriptions of the rest of the settings that are available when you click the **gear** icon:

- **Mark spaces (·)**: Replaces all spaces in your document with a dot (·).
- **Hyperlink URLs**: Hyperlinks any URL in your document. This makes it easier to see links in your documents and allows you to click the hyperlink to go directly to the website.
- **Right-to-left (RTL) text**: Right justifies your document text. By default, the document text is left justified.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

- **Go to first unlabeled data**: Brings you to the first document in the batch that has not yet been annotated. This option is helpful any time you re-enter a batch to continue annotating.
- **Export Studio dataset**: Export your dataset into a CSV file.

How to Annotate

The following sections walk through how to annotate documents for the four different task types:

- single-label classification
- multi-label classification
- text extraction
- sequence tagging

Single-label classification applications

In single-label classification applications, your goal is to assign a single class to each document. For example, assigning banking contract documents to one of the following classes: "employment," "loan," "services," or "stock." For more information about this setting, see Set the default label for multi-label annotation applications.

On the right-side pane, you'll see all possible classes that you can label your document. To label a document, click the class in the right-side menu. Then click the arrow button to move on to the next document. If you adjusted the recommended settings, then you use can use the keyboard shortcut to assign a class, and then you will automatically be brought to the next document.

Multi-label classification applications

In multi-label classification applications, individual documents can have multiple label values. For example, let's say you are looking at movie review documents. You can label the movie as "Short Film," "Black and White," "Japanese Movies," or "World Cinema." Given these labels, you can see that a single movie can fall into multiple categories. In this case, for a given document, you can label each possible class as present, absent, or abstain from voting.

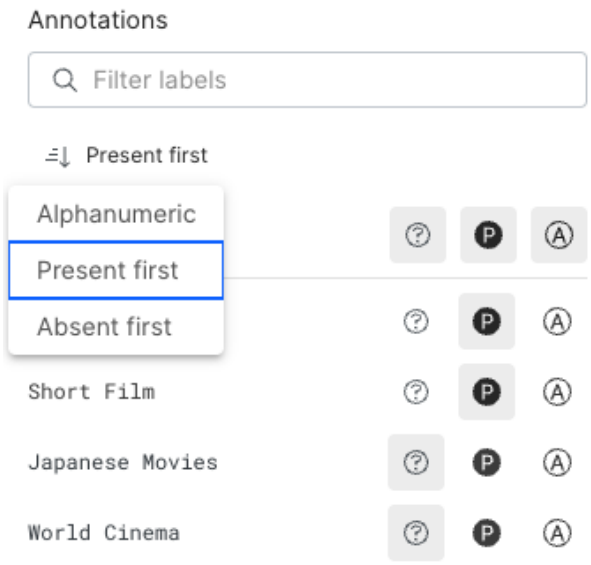
On the right-side pane, you'll see all possible classes. For each class, you can click:

- ? to label the class as Abstain.
- P to label the class as Present.
- A to label the class as Absent.

By default, all classes are initially labeled as Abstain. If enabled for your application, you can set the default label for each class to either Present, Absent, or Abstain.

You can also sort the classes, which is particularly helpful for applications with a large number of classes. You can sort classes marked as **Present first**, classes marked as **Absent first**, or in **Alphanumerical** order.

Copyright © 2025 Snorkel AI, Inc. All rights reserved



(Beta) Set the default label for multi-label annotation applications

> NOTE
> This is a beta feature available to customers using a Snorkel-hosted instance of Snorkel Flow. Beta features may have known gaps or bugs, but are functional workflows and eligible for Snorkel Support. To access beta features, contact Snorkel Support to enable the feature flag for your Snorkel-hosted instance.

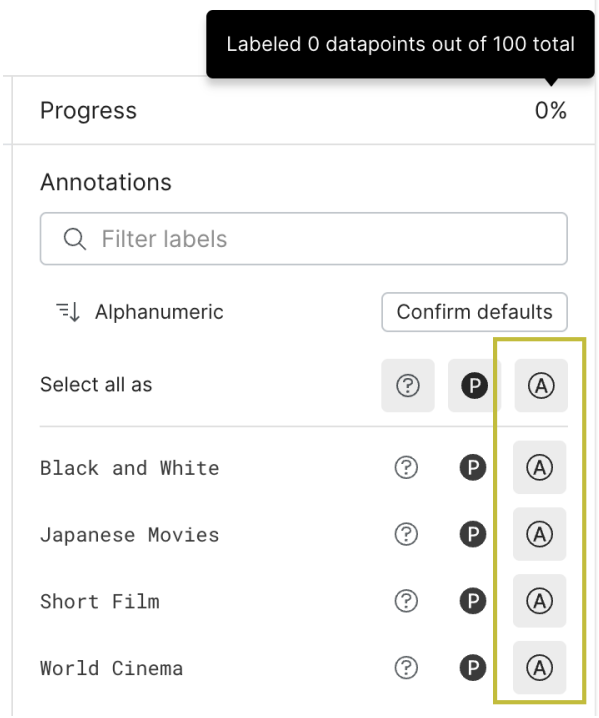
In multi-label classification applications, individual documents can have multiple label values. For a given document, you can label each possible class as present, absent, or abstain from voting. From the **Overview** page, you have the option to set the default label value for your application. This is helpful in situations where different teams in your organization want their default label to be different based on their particular use case. In addition, this setting is application specific so it will only affect your current working application.

> NOTE
> This functionality is a beta launch that is only made available on a request basis. Please contact your Snorkel representative if you are interested in getting access.

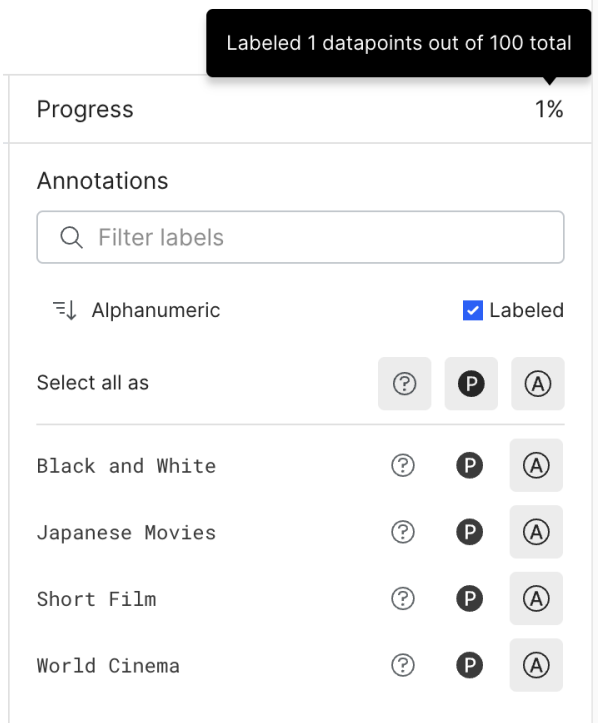
Follow the steps below to set up your default label:

1. Click the menu in the top right corner of your screen.
2. In the modal, select a default label of **Present**, **Absent**, or **Abstain** from the dropdown.
3. Click the **Save** button to save your changes.
4. Open a batch and then you'll see that for unlabeled documents, the default label is set as your selection.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

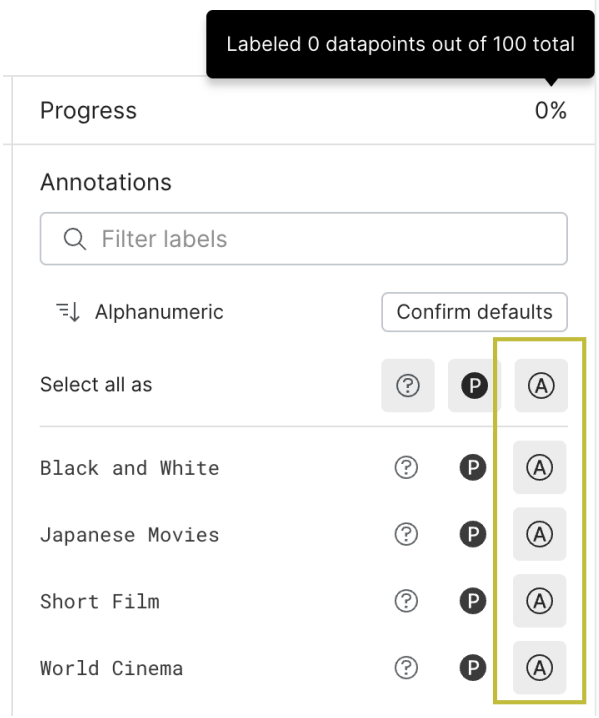


5. You can now take one of the following actions when reviewing a document:
   a. You can select labels however you like.
   b. You can go with the default labels by clicking the **Confirm defaults** button.
   You can see the progress has gone up after you've completed either of the actions listed above and the state has changed to **Labeled**.



6. If you want to revert your changes and go back to the unlabeled state, click **Labeled**. The modal asks for your confirmation as this action cannot be undone.
7. After you confirm, the document returns to an unlabeled state. You'll also see the **Confirm defaults** button again along with the default labels and progress.

Copyright © 2025 Snorkel AI, Inc. All rights reserved



> **TIP**
> You can also change your default label while annotating your batches. This action will not affect any of the labeled documents and only reflect the default label for unlabeled ones.

Text extraction applications

In text extraction applications, the goal is to extract key information from a document. For example, let's say that we want to extract all dates from a document. In this case, a date is considered a span. Your goal in this example is to review all highlighted spans in a document and identify whether the highlighted spans are in fact dates. You can label each highlighted span as "NEGATIVE," "POSITIVE," or "UNKNOWN" in the right-side pane. In this example, "POSITIVE" means that the highlighted span is a date, and "NEGATIVE" means that the highlighted span is not a date.

If you adjusted the recommended settings, then you can use the keyboard shortcuts to label a span, and then automatically to be brought to the next highlighted span in the document.

By default, you will be viewing the spans in Document view. We recommend annotating in either Document view or Span view:

- **Document view** shows the entire document on the main canvas. You can use the up and down arrows on your keyboard to navigate between spans of text to annotate. This view provides you with all context surrounding a span.
- **Span view** shows a single span per page. With this view, you don't have all the context surrounding a span. However, the simplified view can enable you to more quickly annotate things like dates that are easy to identify with less context.

Sequence tagging applications

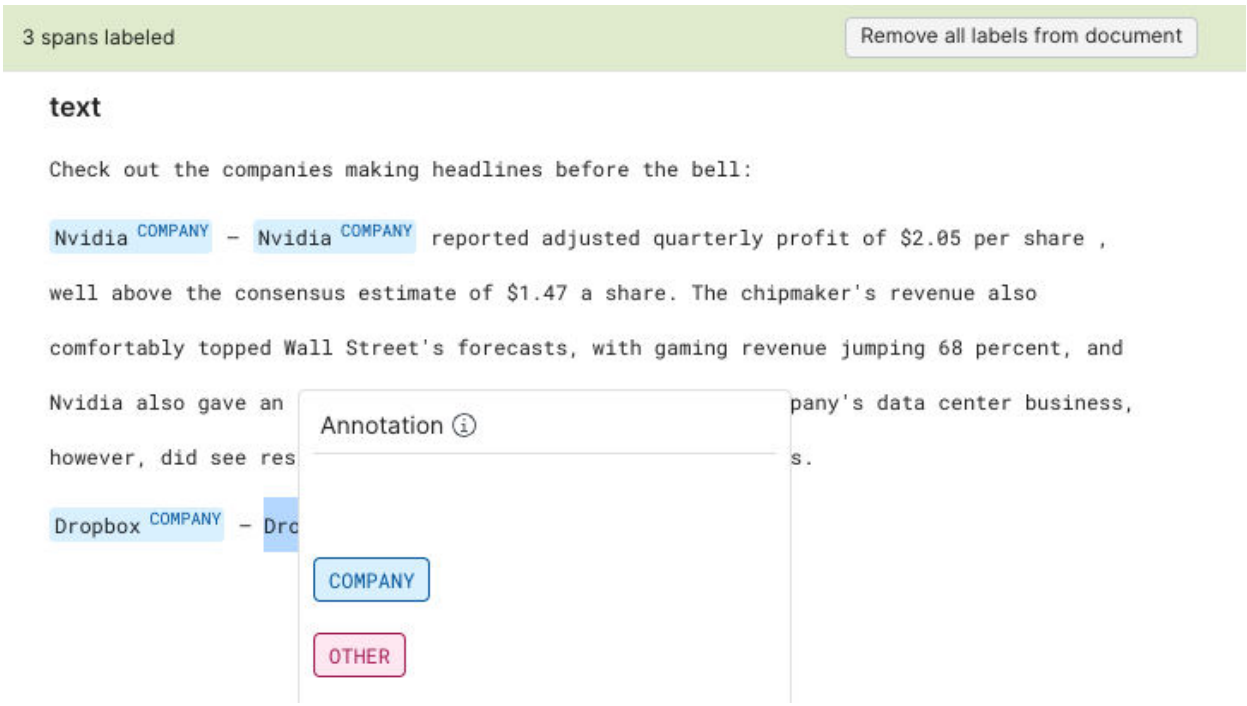
In sequence tagging applications, your goal is to highlight and label spans throughout a document. Spans are key pieces of information that you want to extract from a document. For example, let's say we want

Copyright © 2025 Snorkel AI, Inc. All rights reserved

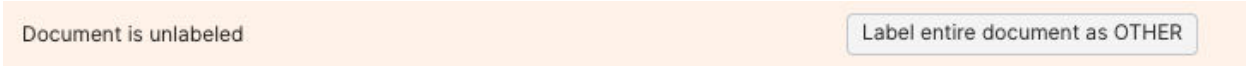
to identify all mentions of company names in a document. In this case, a company name is considered a span. Your goal is then to highlight and label all company names that you find while reading through the document.

To label spans in Annotation Studio:

1. Highlight a span.
2. Select the label that you want for that span in the Annotation modal.



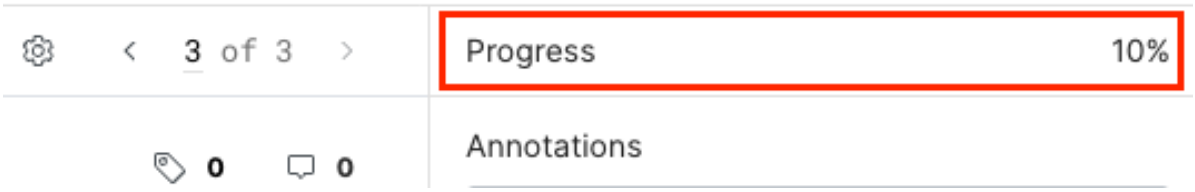
If you don't find any spans to label in the document, you can indicate this by clicking the **Label entire document as OTHER** button above the document text.



At any point, you can reset and clear all spans that you have labeled in the document by clicking the **Remove all labels from document** button above the document text.

Check progress

While you are annotating documents, you can see your progress percentage tick up in the right-hand corner of your screen.



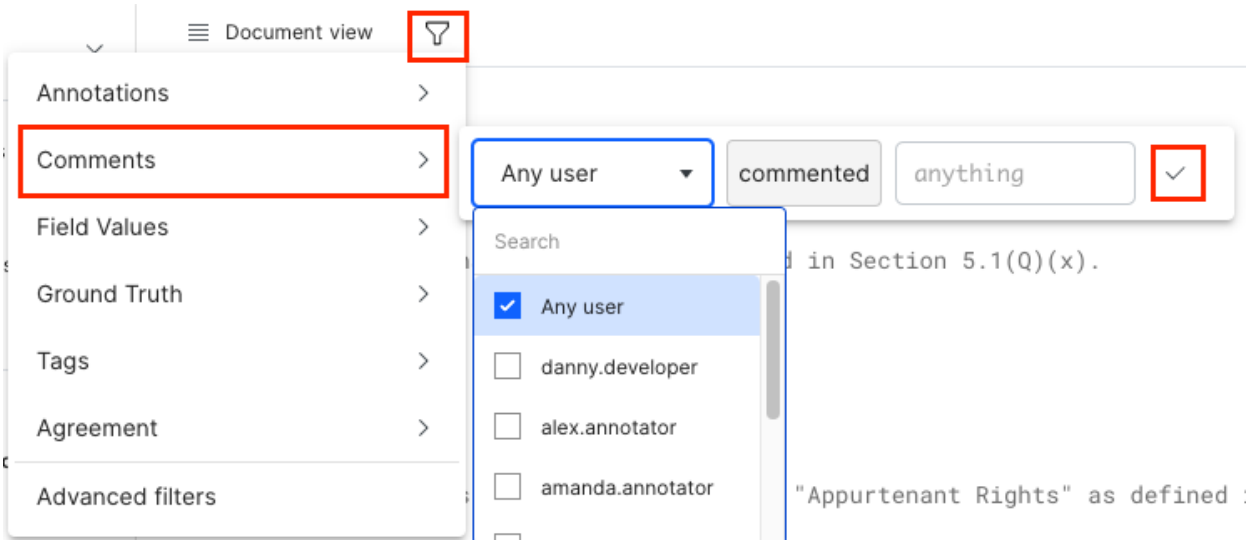
Once you are finished, your progress bar will be labeled 100%, and the status will be updated to **Complete** on the Batches page.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

Review comments and slices

Occasionally a reviewer will leave comments on documents. For example, if you left a comment asking a clarifying question, a reviewer may respond to your comment. To see all documents with comments:

1. Click the filter icon.
2. Click **Comments**.
3. Under **User**, select the person that you want to see comments from. Alternatively, you can select **Any user**.
4. Click the checkmark ✓ to save the filter.



Now, just the documents with comments are shown, making it easier for you to review any comments.

Learn more about slices in Using data slices.

You can follow similar steps to filter documents based on slices:

1. Click the filter icon.
2. Click **Slice**.
3. Under **Slice**, select a slice option.
4. Under **Operator**, select **is** if you want to see all documents with that slice or **is not** if you want all documents with that slice removed from view.
5. Click the checkmark ✓ to save the filter.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

Walkthrough for reviewers

This page walks through the different ways that users with the Reviewer role can review annotations from all annotators.

As a reviewer, you will likely be tasked to manually annotate documents, in addition to reviewing annotations from others. Follow the Walkthrough for annotators to get step-by-step instructions annotating documents for the different task types.

You can review annotations in two places:

- Within a batch: View all annotations made on a single document side-by-side.
- On the Review page: View a full list of all annotations and their corresponding documents.

In addition, you have some additional permissions that allow you to manage batches.

Review documents within a batch

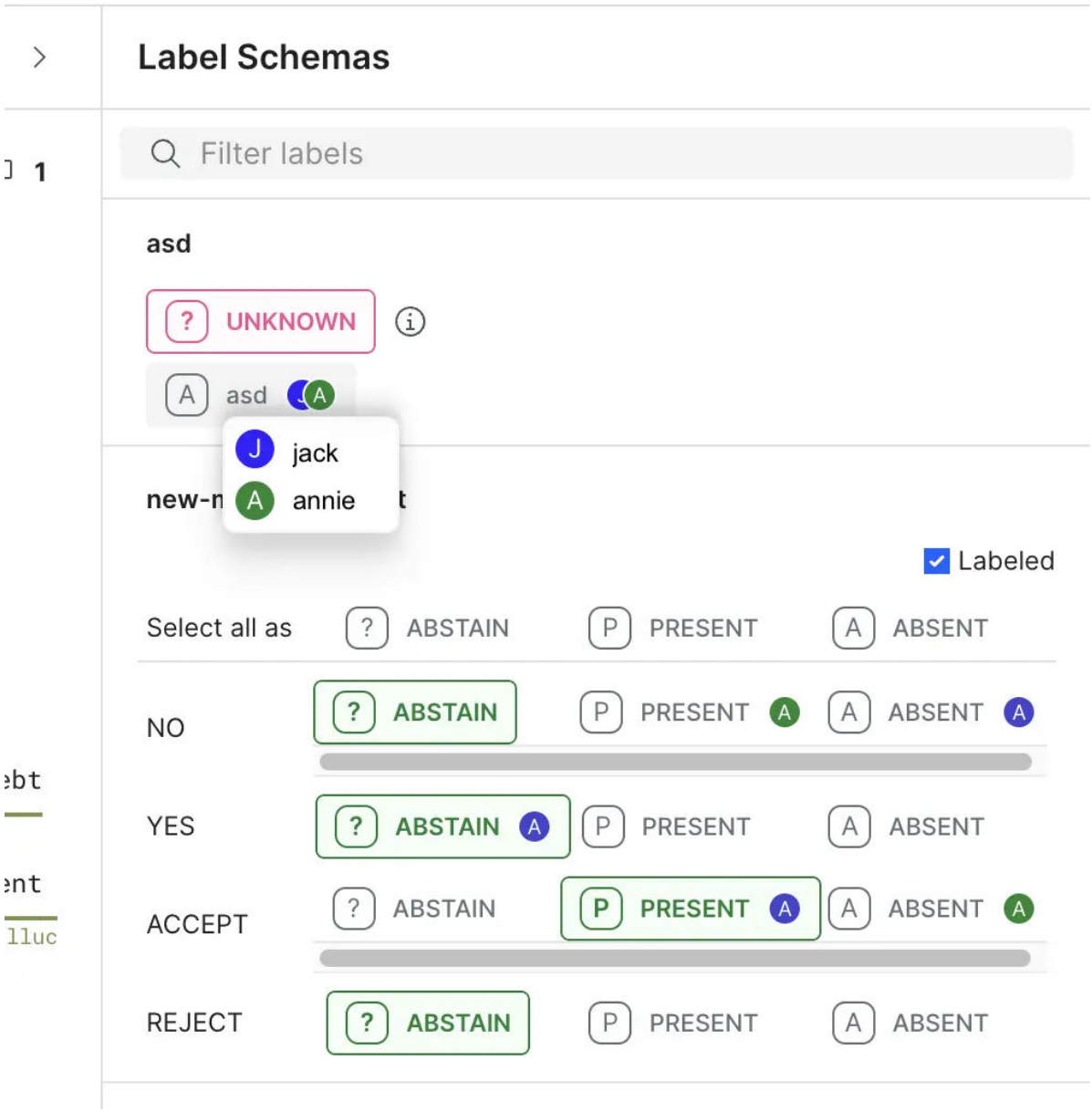
While inside a batch that you are assigned, being a reviewer allows you to review other users' annotations. With multi-schema annotation, reviewing becomes a superset of annotating. In this view, you can see all annotations by toggling the **Reviewer** mode on and off.

How you review the annotations differs based on the task type. For all task types, you can click the comment icon to add an explanation for why you agree or disagree with other annotations. In addition, you can click the slice icon to add a slice to the document. For example, you may want to create a slice called **contains-disagreements** to slice documents that have significant disagreement among the annotators.

Review classification and extraction annotations

For classification and extraction applications, you can see all annotations made by other annotators. You can also see the annotator's name and the annotation label that they selected.

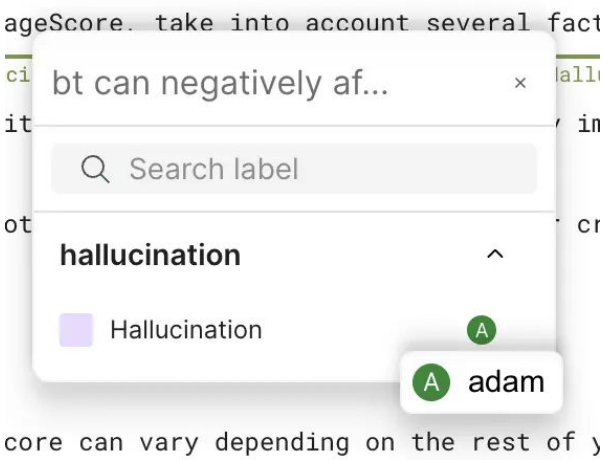
Copyright © 2025 Snorkel AI, Inc. All rights reserved



Review sequence tagging annotations

In sequence tagging applications, you can see annotations in the label pop-up when you select an annotated span.

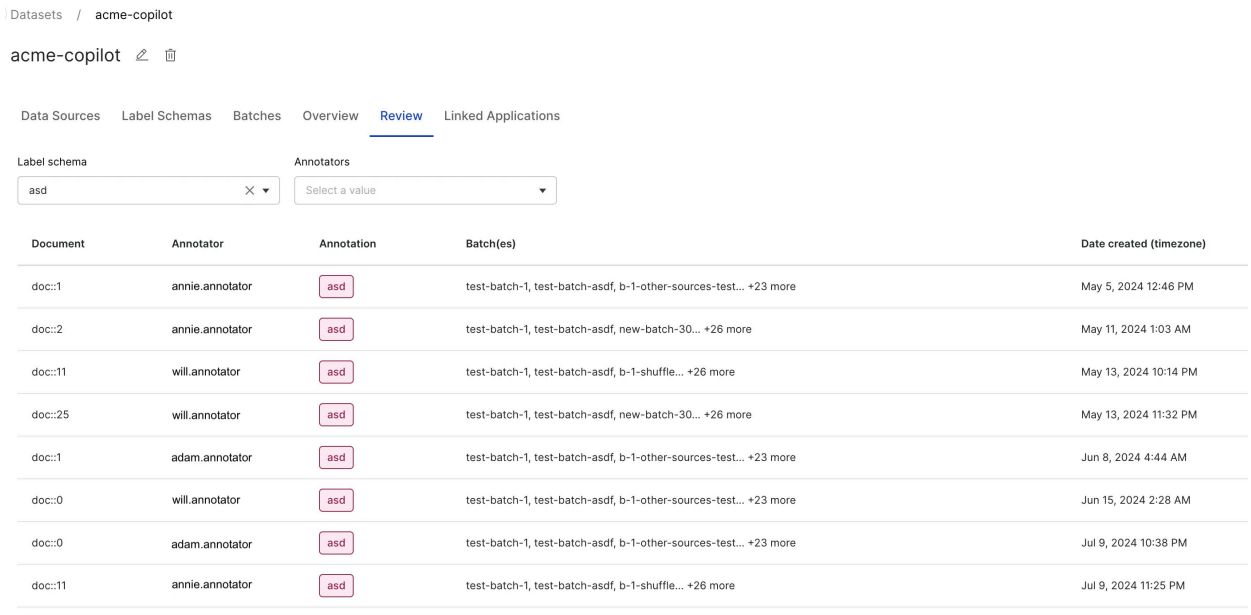
Copyright © 2025 Snorkel AI, Inc. All rights reserved



Review all annotations on the review page

The Review page provides a full list of all annotations and their corresponding documents to easily review all annotations in one place. To access the Review page:

1. Click **Datasets** in the left-side menu.
2. Click the name of the dataset that you want to review.
3. Click the **Review** tab.



Select an annotation to pull up the full document text. Here, you have the option to leave comments and add slices to the document. In addition, you can select the filters icon on the top-right corner of your screen to filter the table by a particular annotator.

Manage batches

As a reviewer, you have additional permissions that allow you to manage batches:

- Rename and delete batches.
- Aggregate annotations in a batch.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

- Set an annotator as an expert for that batch to view the agreement rate for each annotator relative to the expert.

Slices

Learn more about slices in Using data slices.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

Create batches

This page walks through how you can create batches of documents for manual annotation when the dataset uses multi-schema annotations.

> **NOTE**
>
> To create batches and commit annotations to ground truth, you must have the Developer role. For more information about access and permissions, see Roles and user permissions.

What are batches?

Before documents can be annotated in Annotation Studio, they must be assigned to batches. A batch is simply an arbitrary collection of data points—it can be as large as an entire split or data source or as small as a few examples.

Typically, you'll want documents manually annotated at a couple different points during the data development process:

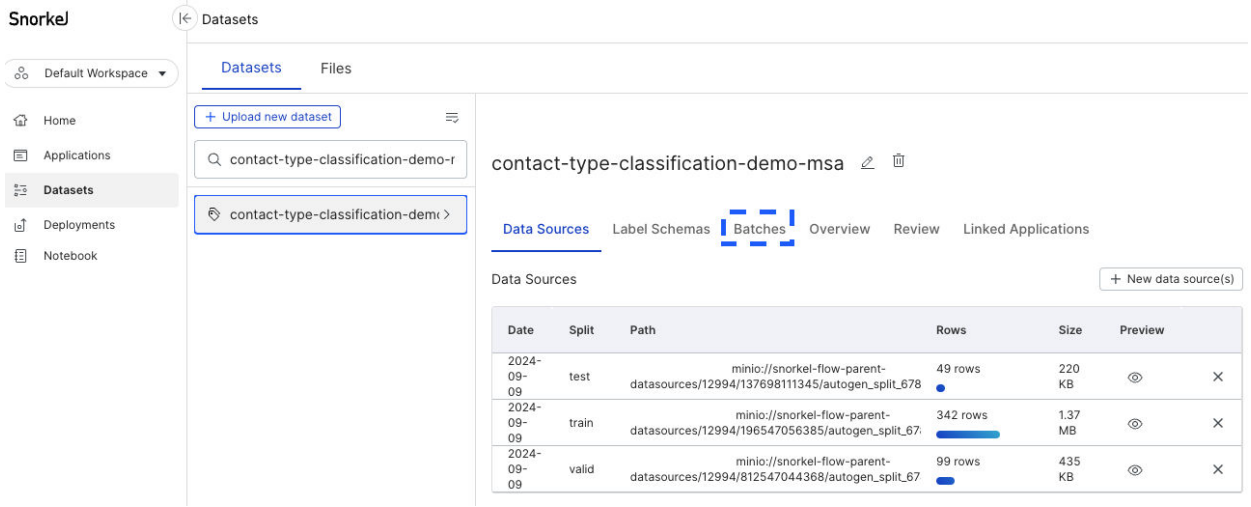
- At the beginning of a project so that you can get some initial ground truth to begin development.
- During labeling function (LF) and model development. The development cycle is an iterative process and you may want to introduce more datapoints with ground truth. The reasons may be to improve the inbalances in the dataset, or simply to add more training data.

How to create batches

There are two ways that you can create batches in Snorkel Flow:

- **Dataset page:** Best method to use if you want to quickly create batches for an entire split or data source.
- **In-platform notebook:** Best method to use if you have a specific list of `x_uid`s that you want to create batches from.

All batches that you create can be seen and managed in the **Batches** tab. This page can be accessed by going to the dataset page, then clicking **Batches** as shown below.

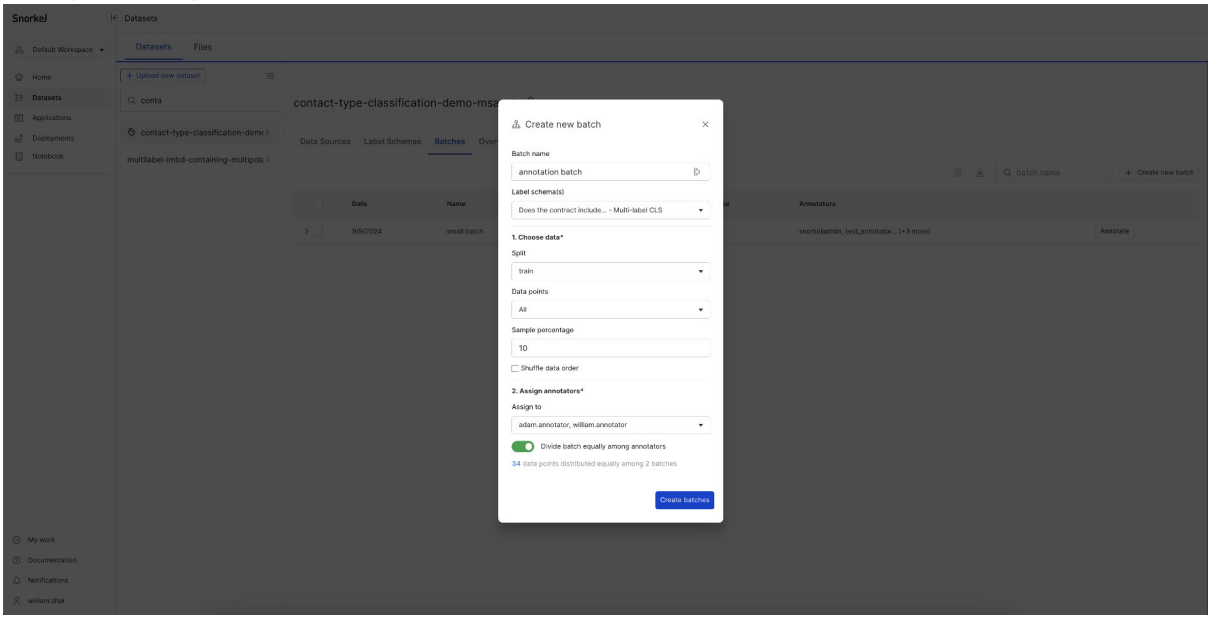


Copyright © 2025 Snorkel AI, Inc. All rights reserved

Create batches in the Dataset page

The easiest way to create batches is in the **Batches** tab from the selected dataset page.

1. Go to the **Batches** tab.
2. Select **+ Create a new batch** to bring up the **Create new batch** modal.
3. Specify the following options:
   - **Batch name:** The name for the batch. If you choose to create multiple batches by enabling **Divide batch equally among annotators**, then each batch is named the specified name with an appended numerical index to differentiate the batches.
   - **Label schema(s):** Specifies the label schema(s) to use for the batch.
   - **Select a split:** Specifies which split to use to create the batch. Each batch comes from a single source—a split, a data source, or an existing batch.
   - **Datapoints:** Specifies the data points to include in the batch. You can select all data points or data points that are not assigned to any batches.
   - **Sample percentage:** Specifies the percentage of data points to use for the batch between 1-100%.
   - **Shuffle data order:** Shuffles the order of the data points in your data source. This option enables you to randomly sample which data points go into each batch.
   - **Assign annotators:** Specifies the annotators to assign the batch to.
   - **Divide batch equally among annotators:** If enabled, datapoints are divided equally to *n* batches where *n* is the number of annotators specified in **Assign annotators**.
4. Once you finish your selections, click **Create Batch**.



The new batch or batches can now be seen on the **Batches** tab.

Create batches with the in-platform notebook

You can also create batches using the SDK in the in-platform notebook. This method is convenient if you have a specific list of `x_uid`s that you want to create batches from.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

You can create batches in the SDK using the `Dataset.create_batches` function. Some examples of how to use the function can be seen below. You can search `Dataset.create_batches` in the SDK documentation for more information about the parameters.

```
ds = Dataset(name="dataset_name", uid=0, mta_enabled=True)

# Creating batches with default parameters.
ds.create_batches(name="notebook_batch", assignees=[5,6], split='train')

# Creating a batch with specific data points using the index. Data points must
be from the same split.
ds.create_batches(name="notebook_batch", assignees=[5,6], x_uids=["span::5",
"span::7"])

# Creating batches with a fixed number of data points.
sf.create_batches(node, username="assigner", assignees=["user 1", "user 2"],
batch_size=10)
```

The created batches can be seen and managed in the Batches page at the dataset page.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

Manage batches and commit ground truth

This page walks through how to manage your batches and commit annotations to your ground truth to be used for development in Studio.

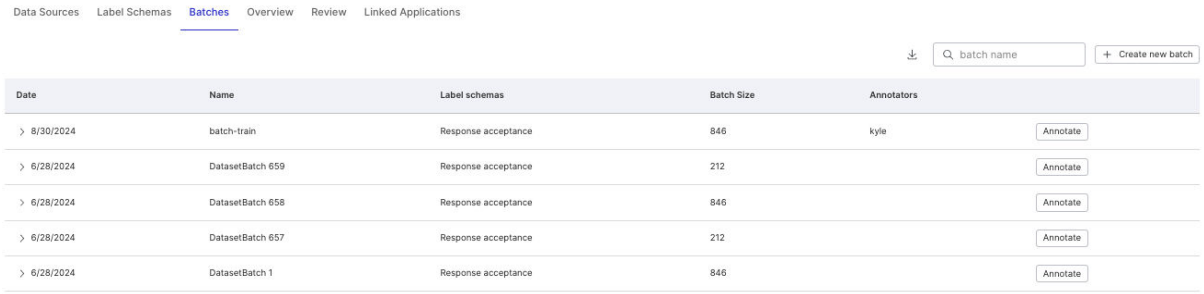
Ground truth refers to the set of labeled and accurately annotated data that serves as a reference or benchmark for training and evaluating machine learning models. During the evaluation phase, ground truth data is used to assess the model's performance. By comparing the model's predictions with the actual labels in the ground truth, metrics such as accuracy, precision, recall, and F1 score can be calculated.

Snorkel Flow allows you to programmatically generate labels for your training split. Snorkel recommends using ground truth (GT) labels for data points in other splits:

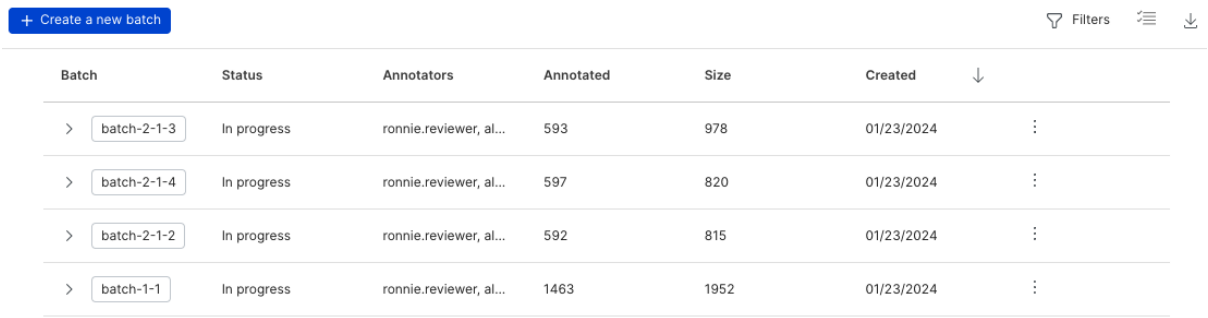
- In the dev split, GT-labeled examples assist with discovering and iterating on labeling functions (LFs).
- In the valid and test splits, use GT-labeled examples to evaluate model performance and facilitate error analysis.

Manage your batches

1. Once create your batches, you can manage the batches in Dataset view from the Batches tab.



2. On the top bar, select the action you want to take:
   - Select the + Create a new batch button to create a new batch of data points for annotation.
   - Select the Filters button to filter the batches by annotator or status.
   - Select the Toggle bulk options icon to show the option to bulk delete batches.
   - Select the Export dataset across all batches icon to export all batches to a CSV file.



3. For individual batches, the following information is available:
   - Date: The date the batch was created.
   - Name: The name of the batch.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

- **Label schemas:** The label schema(s) associated with the batch.
- **Batch Size:** The total number of data points in the batch.
- **Annotators:** The list of annotators that have been assigned to annotate the batch.
- **Annotate:** The button to start annotating the batch.

On the top bar, select the action you want to take:

- **Create new batch:** Select the **+ Create a new batch** button to create a new batch of data points for annotation.
- **Search batches:** Use the search bar to search for a specific batch by name.
- **Export data:** Select the **Export dataset across all batches** icon to export all batches to a CSV file.

4. Enter the following options in the **Export annotation batches** modal:
   - **Include columns:** Specify which columns to include in the export. This option enables you to select a focused slice of your data, or you can speed up export time by excluding large columns in a dataset.
   - **Include options:** Specify additional information about individual data points to include such as annotations, comments, slices, filters applied, and model predictions.
   - **Optional settings:** Specify additional options to customize your export. These options include the ability to start an export at a specific index in the dataset, setting a maximum number of data points to export, and options to configure delimiters, quote characters, and escape characters.

Manage individual batches

Select the arrow > icon next to the batch name to view additional information about an individual batch. This menu give you additional actions you can take on an individual batch:

- **Aggregate:** Aggregate annotations from multiple annotators.
- **Commit:** Commit annotations to the ground truth.
- **Set as expert:** Mark selected annotator(s) as an expert for the batch. To mark an individual annotator as an expert for the batch, select the annotator, and then click **Set as expert.** A badge icon shows up next to the annotator's name to indicate the expert status.
- **Remove expert:** To remove the expert designation for an annotator, select **Remove expert**.
- **Edit batch details:** Batch name and annotators list can be updated by clicking on **Edit** icon for a batch.
- **Export batch data:** Export the batch data to a CSV file.
- **Delete batch:** Deletes the batch.

Aggregate annotations

Typically, you'll have more than one annotator reviewing and labeling documents. You can only commit a single vote to ground truth. You can aggregate annotations instead of committing the annotations from a single person. This aggregation helps eliminate potential bias from any particular annotator.

To aggregate annotations, select the annotators that you want to aggregate, then select **Aggregate**.

A majority vote is the only supported aggregation strategy. This strategy takes the majority label for each data point if one exists, and leaves an `UNKNOWN` label where no annotations exist. These results create a new set of annotations with a single vote for each data point. The aggregated annotations can be seen in the expanded view of batch on the **Batches** page, under **Other annotations**.

The majority vote aggregation algorithm works differently for each of these task types.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

Single-label

Snorkel Flow uses a simple majority algorithm as the aggregation strategy for single-label applications. Based on the number of votes for each label, the label with the most votes is applied to the ground truth as the final decision.

For example, consider a single-label application with ten annotators and two classes. Six annotators label a data point with class A, and four annotators label a data point with class B. Because class A now represents the majority vote at aggregation time, the aggregated result will label this data point as class A.

If there is a tie between two or more majority labels, Snorkel Flow uses a pre-determined random seed to perform a random choice selection.

Multi-label

Snorkel Flow uses a simple union algorithm as the aggregation strategy for multi-label applications. Based on the number of present/absent votes for each class, Snorkel Flow applies the label (`present` or `absent`) with the most votes for each class as the final decision.

For example, consider a multi-label application with ten annotators and two classes. For class A, six annotators vote `present` and four annotators vote `absent`. For class B, four annotators vote `present` and six annotators vote `absent`. By taking the majority vote of each class, the aggregated label for the data point applies `present` for class A and `absent` for class B.

If there is a tie between `present` and `absent` labels, the resulting label has an equal chance of being marked as `present` or `unknown`.

Sequence tagging

Snorkel Flow uses a simple majority algorithm as the aggregation strategy for each character in a span in sequence tagging applications. Based on the number of votes for each label in any interval where annotators disagree, Snorkel Flow selects the label with the most votes as the final decision.

If there is a tie between two or more majority labels, the resulting label is a negative class to reduce the risk of false positives, such as trailing spaces or mistaken tokens. If the tie is between only positive labels, the resulting label is a pre-determined random seed to perform a random choice selection.

Commit annotations

Once you have a set of annotations that are accurate, commit the annotations as the ground truth for development in Studio.

> **NOTE**
>
> Only users with the **Developer** or **Administrator** role can commit ground truth.

1. Select the desired annotation set. You can select annotations from either an individual annotator or from an aggregated set of annotations.
2. Select **Commit**.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

> ⚠ WARNING
>
> Once you commit annotations, the new annotations overwrite the existing ground truth labels in the data source.

Format for ground truth interaction in the SDK

This section shows example formats for each label space for usage in our SDK for a given datapoint.

Multi-label spaces ground truth is represented as a dictionary where there is a mapping from each label to one of `PRESENT`, `ABSENT`, or `ABSTAIN`:

```
label: {"Japanese Movies": "PRESENT", "World cinema": "ABSTAIN", "Black-and-
White": "ABSENT", "Short Film": "ABSTAIN"}
```

Sequence tagging ground truth for a document is a list of spans, where each span is a triple of (`char_start`, `char_end`, `label`). The spans cannot be empty (`char_start` must be smaller than `char_end`). Overlapping or duplicating spans are not allowed. The sets of char offsets (`char_start`, `char_end`) must be sorted:

```
label: \[ \[0, 29, 'OTHER'\], \[29, 40, 'COMPANY'\], \[40, 228, 'OTHER'\], \
[228, 239, 'COMPANY'\], \[239, 395, 'OTHER'\], \]
```

Single label space ground truth format is represented by their class label

```
label: "loan"
```

Copyright © 2025 Snorkel AI, Inc. All rights reserved

How-to: Create dataset views for custom annotation user experiences

Dataset views allow you to customize the presentation of your data for annotators. This makes it easier for annotators to work on a focused set of labeling requirements for a particular annotation batch.

Two types of dataset views are available:

- **Single:** Use this view to display one record at a time to the annotator.
- **Ranking:** Use this view to display multiple records for ranking.

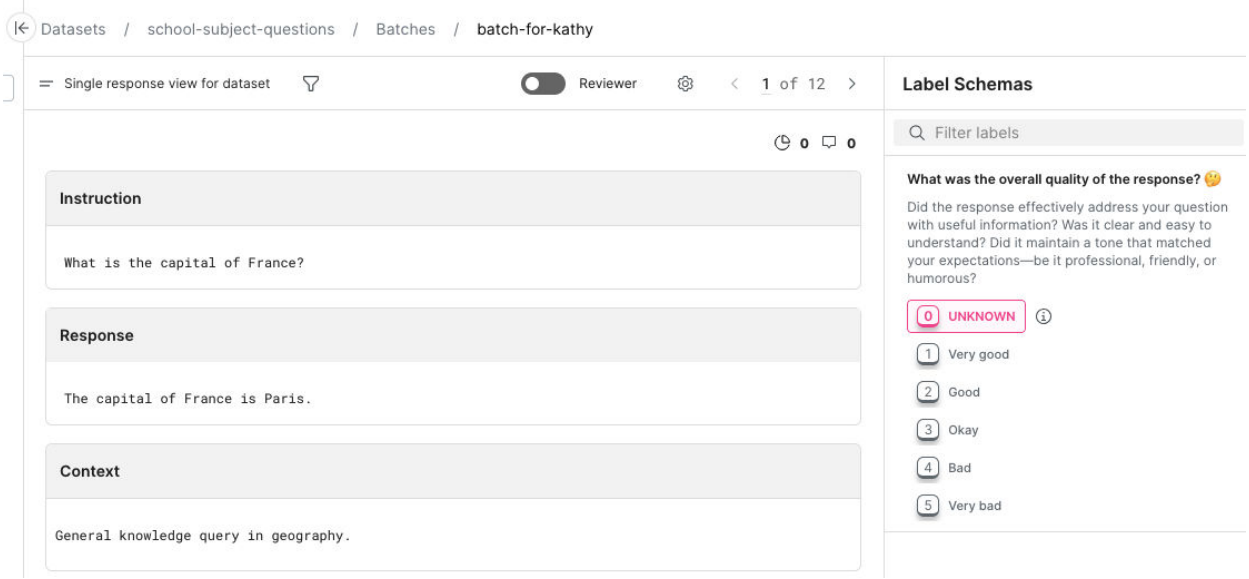
This guide introduces the concepts and the SDK commands for creating dataset views, and also includes two full end-to-end examples and a notebook.

- SDK: Create dataset view
- Example: Create a single response view
- Example: Create a ranking view
- Notebook: dataset-view-creation.ipynb

Single response view

Create a single response data viewer to show the annotator one LLM-generated response at a time. Alongside the response, display one or more label fields for the annotator to fill out. These fields can be multiple choice from a predetermined set of labels, or free text entry.

The single dataset view looks like this:

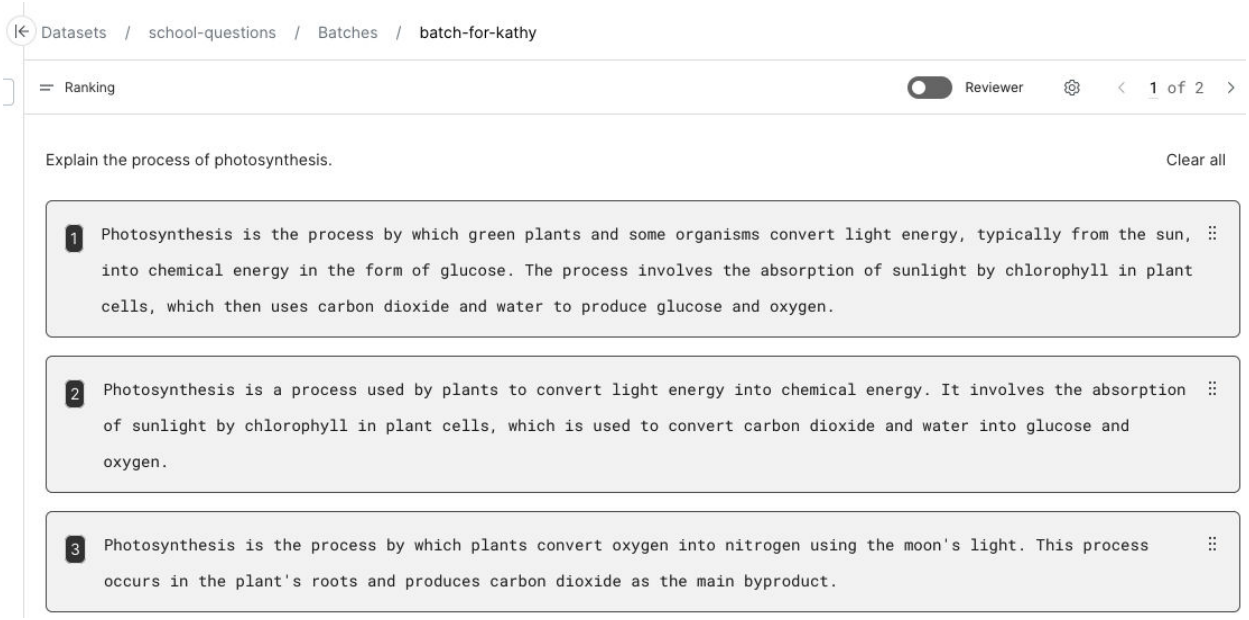


Ranking responses view

Create a ranking data viewer to show the annotator multiple LLM-generated responses to a single prompt. The annotator can drag and drop the responses to rank them from best to worst. Once the annotator orders the responses, each is labeled with the rank number.

The ranking dataset view looks like this:

Copyright © 2025 Snorkel AI, Inc. All rights reserved



Prerequisites

- A dataset for which you want to create a view
- A label schema for which you want to create a view. Note that only classification-type label schemas are supported for dataset views.

SDK: Create dataset view

Dataset views require using the Snorkel SDK.

> **NOTE**
>
> The `CONTEXT` column supports a formatted view to display JSON data.

```
# Map the column names in your dataset to the view
column_mapping = {
  FineTuningColumnType.INSTRUCTION: "YOUR_COLUMN_NAME", # Input required, use the column name that has the LLM prompts
  FineTuningColumnType.RESPONSE: "YOUR_COLUMN_NAME", # Input required, use the column name that has the LLM responses
  FineTuningColumnType.CONTEXT: "YOUR_COLUMN_NAME", # Optional, use for notes about the record(s)
  FineTuningColumnType.PROMPT_PREFIX: "YOUR_COLUMN_NAME" # Optional, use the column name that has the system prompt(s)
}
```

Copyright © 2025 Snorkel AI, Inc. All rights reserved

```
snorkelflow.create_dataset_view(
  dataset="YOUR_DATASET_NAME", # Input required, enter the existing dataset name
  name="NEW_VIEW_NAME", # Input required, specify a unique view name for this dataset view
  view_type="VIEW_TYPE", # Input required, choose "DatasetViewTypes.SINGLE_LLM_RESPONSE_VIEW" or "DatasetViewTypes.RANKING_LLM_RESPONSES_VIEW"
  column_mapping=column_mapping,
  label_schema_uids=[] # Input required, use the label schema UID you want to use with the view
)
```

Download this code as a notebook.

Example: Create a single response view

Follow this end-to-end tutorial to upload a new dataset, create a label schema, and then create a single response view with an associated batch. This example uses Arcades to illustrate the major steps. Arcades are lightly interactive annotated screenshots that you can click through to see the steps in action.

Upload a dataset

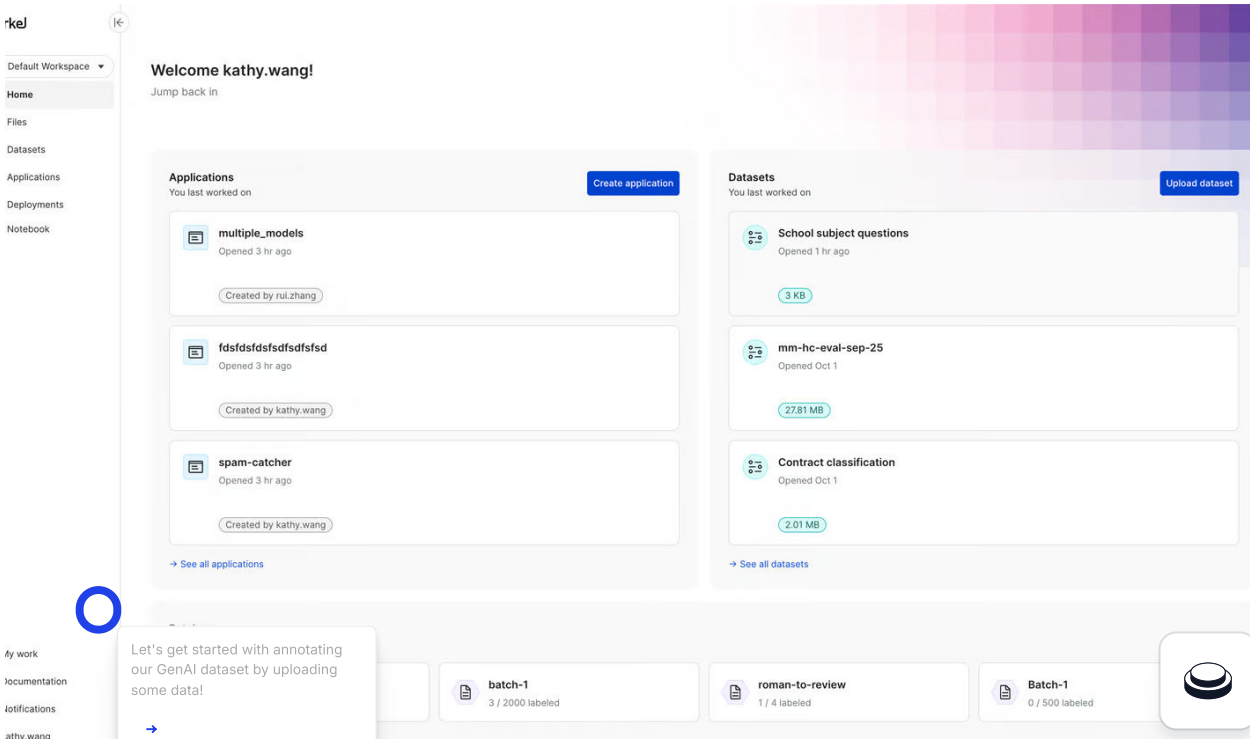
1. Download the school-subject-questions.csv dataset.
2. In Snorkel Flow, select **Upload dataset**.
   - **Dataset name:** `school-subject-questions`.
   - **Enable multi-schema annotations:** Select this checkbox.
   - **File Upload:** Select this option.
   - **Choose File:** Choose the `school-subject-questions.csv` file.
   - **Split:** Select the `train` split.
3. Select **Verify data source(s)**.
   - **UID Column:** Select `UID` from the dropdown.
   - **Data type:** Select **Raw text**.
   - **Text type:** Select **Classification**.
   - **Primary text field:** Select `Response`.
4. Select **Add data source(s)**.

Wait for the dataset to upload. You should see **Data ingestion complete** and the name of the dataset.

Click through the Arcade below to see these steps in action.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

[GenAI annotation docs] Uplo...

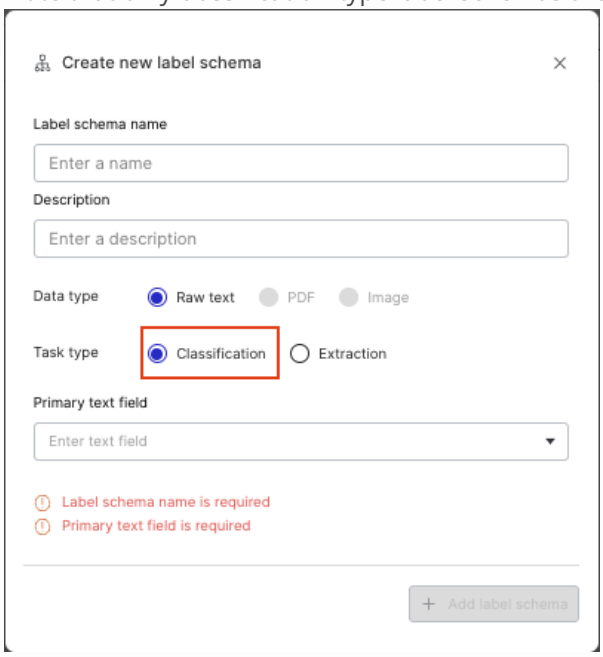


Create a label schema

1. From the dataset page, select **Label Schemas**.
2. Select **Create new label schema**.
   - **Label schema name:** Enter a name for the label schema.
   - **Description:** Enter a description for the label schema.
   - **Data type:** Select **Raw text**.
   - **Task type:** Select **Classification**.
   - **Primary text field:** Select `Response`.
3. Select **+ Add new label**. Enter as many labels as you want to use for the label schema.
4. Select **Add label schema**.

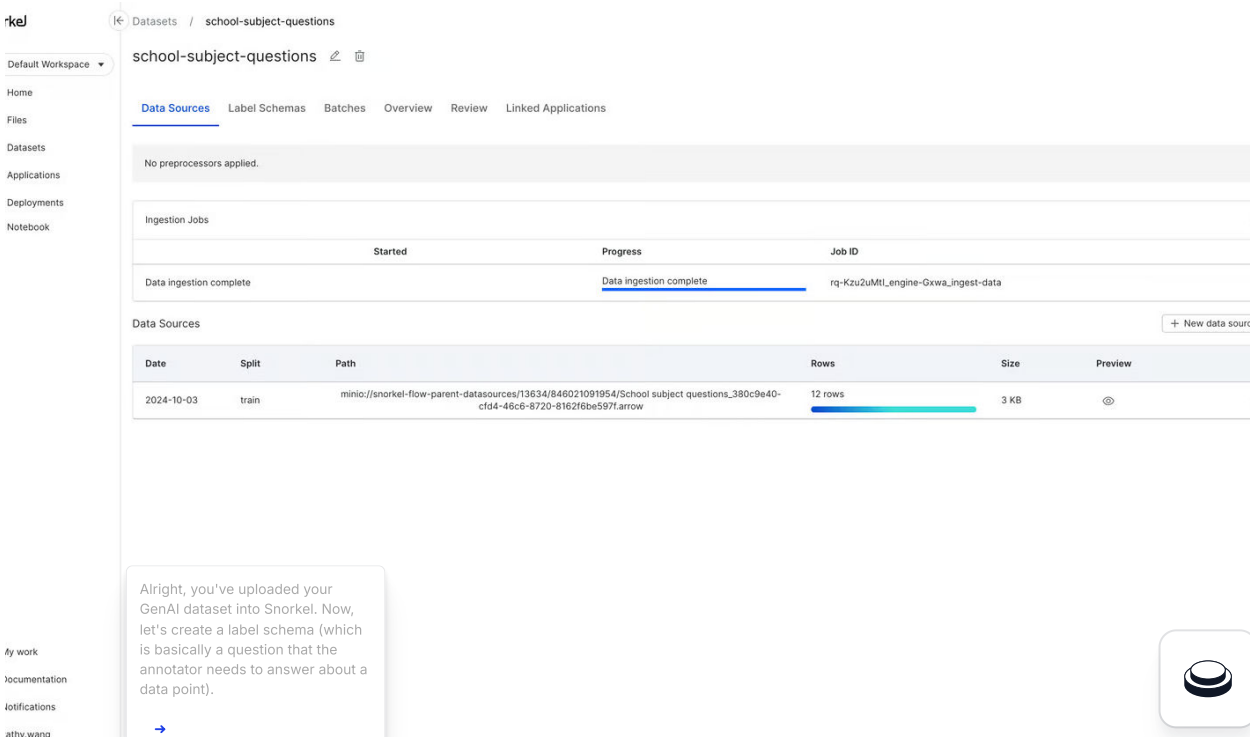
Copyright © 2025 Snorkel AI, Inc. All rights reserved

> **NOTE**
>
> Note that only classification-type label schemas are supported for dataset views.



Click through the Arcade below to see these steps in action.

Create a label schema



Create a single view using the SDK

1. Download the dataset-view-creation.ipynb notebook.
2. In Snorkel Flow, select **Notebook**.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

3. Select **Upload**.
4. Upload the notebook.
5. Select `dataset-view-creation.ipynb` to open it.
6. Run the notebook to create a single response view. Many cells require user input. Make note of the following:

- `dataset_name = "school-subject-questions"`
- `view_name = "Single Response View"`: You can create any name you want.
- `view_type= "DatasetViewTypes.SINGLE_LLM_RESPONSE_VIEW"`
- `FineTuningColumnType.INSTRUCTION: "Instruction"`
- `FineTuningColumnType.RESPONSE: "Response"`
- `FineTuningColumnType.CONTEXT: "Context"`
- `FineTuningColumnType.PROMPT_PREFIX: ""`: This line is optional.
- `label_schema_uids=[XXXX]`: Your UID will vary.

7. When you have finished running the notebook, return to Snorkel Flow.

Next, create a batch that uses the view.

Click through the Arcade below to see these steps in action. Note that the notebook may not match the Arcade exactly, and you must create the batch before you can annotate.

Create a batch for annotation

1. From the dataset page, select **Batches**.
2. Select **Create new batch**.
   - **Batch name:** Enter a name for the batch.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

- **Label schemas:** Select the label schema you created.
- **Split:** Select `train`.
- **Assign to:** Select an annotator.

3. Select **Create batch**.

Click through the Arcade below to see these steps in action.

Annotate using the single dataset view

1. From the batch page, use the two-line menu in the top left of the batch, and choose the view that matches the `view_name` that you created earlier in the notebook.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

2. Now you can explore the dataset view from the perspective of the annotator. Make adjustments to the view as needed.

Click through the Arcade below to see these steps in action.

Example: Create a ranking view

Follow this end-to-end tutorial to upload a new dataset, create a label schema, and then create a ranking view with an associated batch. This example uses Arcades to illustrate the major steps. Arcades are lightly interactive annotated screenshots that you can click through to see the steps in action.

Upload a dataset

1. Download the school-subject-questions.csv dataset.
2. In Snorkel Flow, select **Upload dataset**.
   - **Dataset name:** `school-subject-questions`.
   - **Enable multi-schema annotations:** Select this checkbox.
   - **File Upload:** Select this option.
   - **Choose File:** Choose the `school-subject-questions.csv` file.
   - **Split:** Select the `train` split.
3. Select **Verify data source(s)**.
   - **UID Column:** Select `UID` from the dropdown.
   - **Data type:** Select **Raw text**.
   - **Text type:** Select **Classification**.
   - **Primary text field:** Select `Response`.
4. Select **Add data source(s)**.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

Wait for the dataset to upload. You should see **Data ingestion complete** and the name of the dataset.

Click through the Arcade below to see these steps in action.

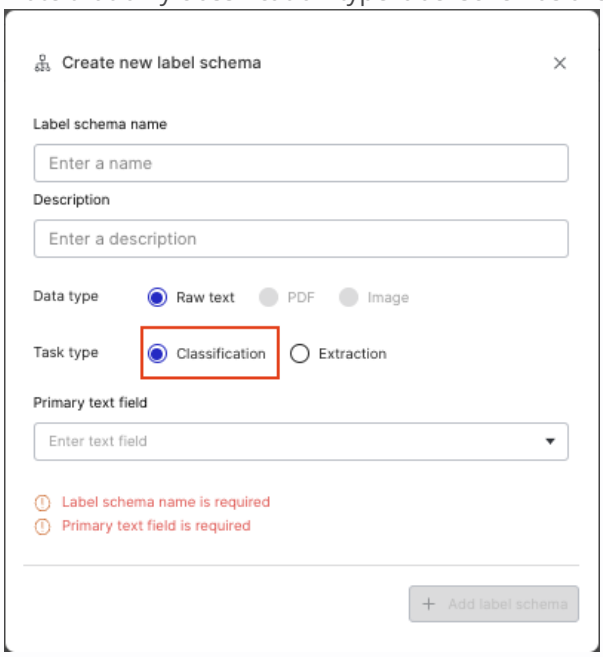
Create a label schema

1. From the dataset page, select **Label Schemas**.
2. Select **Create new label schema**.
   - **Label schema name:** Enter a name for the label schema.
   - **Description:** Enter a description for the label schema.
   - **Data type:** Select **Raw text**.
   - **Task type:** Select **Classification**.
   - **Primary text field:** Select `Response`.
3. Select **+ Add new label**. Enter numeric labels from 1 to 3.
4. Select **Add label schema**.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

NOTE

Note that only classification-type label schemas are supported for dataset views.



Click through the Arcade below to see these steps in action.

Create a ranking view using the SDK

1. Download the dataset-view-creation.ipynb notebook.
2. In Snorkel Flow, select **Notebook**.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

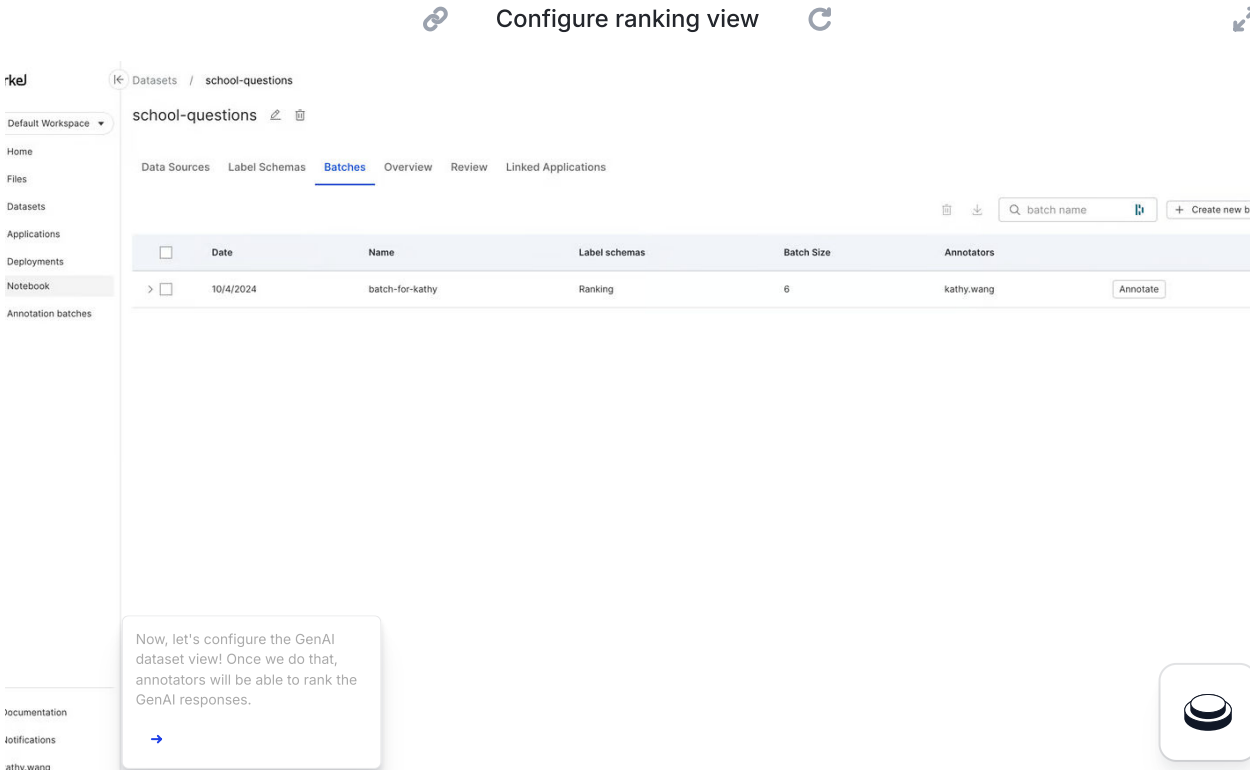
3. Select **Upload**.
4. Upload the notebook.
5. Select `dataset-view-creation.ipynb` to open it.
6. Run the notebook to create a ranking response view. Many cells require user input. Make note of the following:

- `dataset_name = "school-subject-questions"`
- `view_name = "Ranking View"`: You can create any name you want.
- `view_type= "DatasetViewTypes.RANKING_LLM_RESPONSES_VIEW"`
- `FineTuningColumnType.INSTRUCTION: "Instruction"`
- `FineTuningColumnType.RESPONSE: "Response"`
- `FineTuningColumnType.CONTEXT: "Context"`
- `FineTuningColumnType.PROMPT_PREFIX: ""`: This line is optional.
- `label_schema_uids=[XXXX]`: Your UID will vary.

7. When you have finished running the notebook, return to Snorkel Flow.

Next, create a batch that uses the view.

Click through the Arcade below to see these steps in action. Note that the notebook may not match the Arcade exactly, and you must create the batch before you can annotate.



Create a batch for annotation

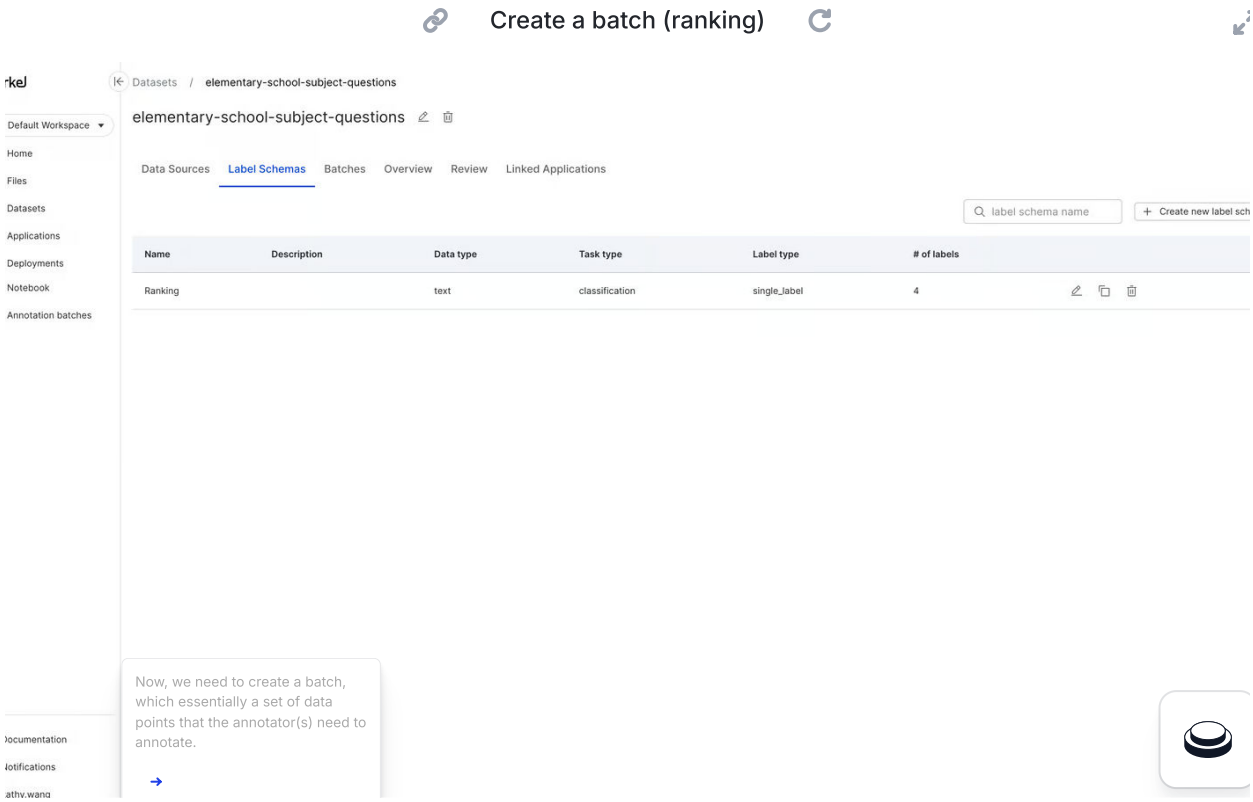
1. From the dataset page, select **Batches**.
2. Select **Create new batch**.
   - **Batch name:** Enter a name for the batch.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

- **Label schemas:** Select the label schema you created.
- **Split:** Select `train`.
- **Assign to:** Select an annotator.

3. Select **Create batch**.

Click through the Arcade below to see these steps in action.



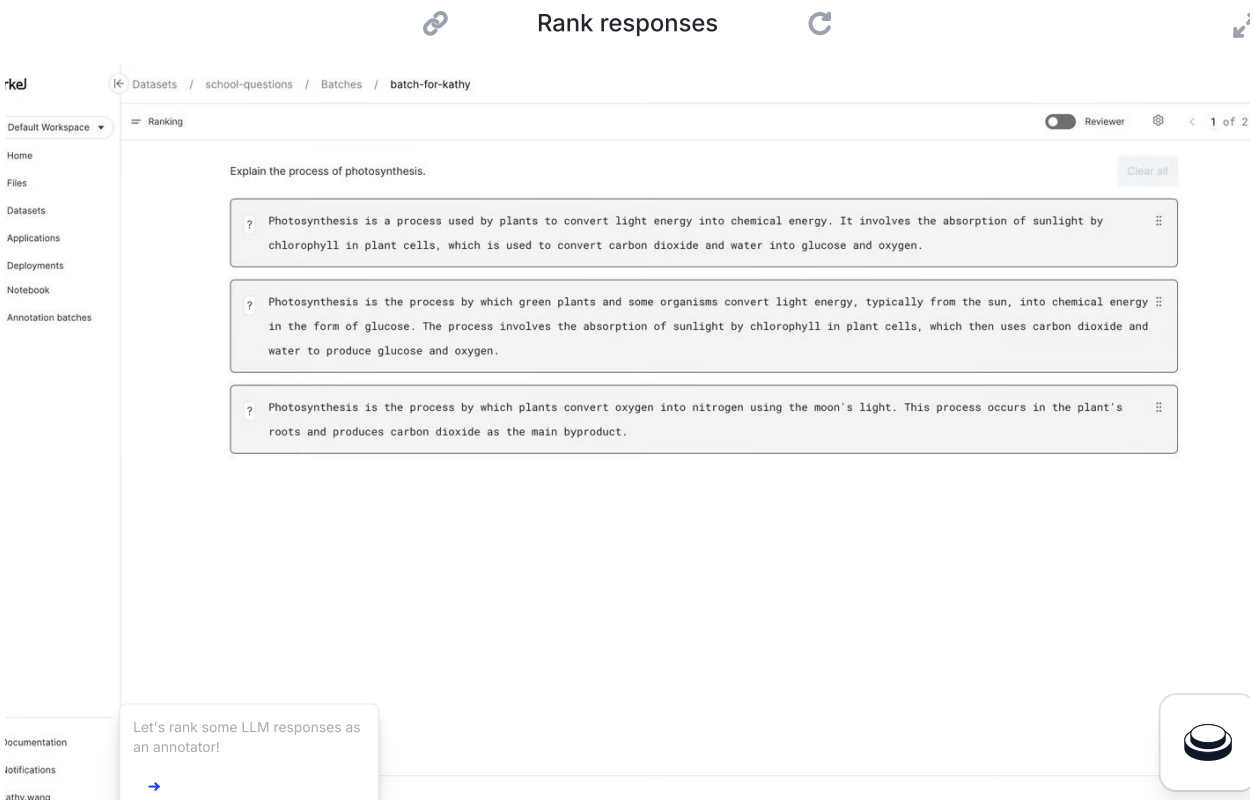
Annotate using the ranking dataset view

1. From the batch page, use the two-line menu in the top left of the batch, and choose the view that matches the `view_name` that you created earlier in the notebook.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

2. Now you can explore the dataset view from the perspective of the annotator. Make adjustments to the view as needed.

Click through the Arcade below to see these steps in action.



Notebook: dataset-view-creation.ipynb

View and download the dataset-view-creation.ipynb notebook:

Copyright © 2025 Snorkel AI, Inc. All rights reserved

Introduction

This notebook creates a new dataset view for an existing dataset and label schema. This makes it easier for annotators to work on a focused set of labeling requirements for a particular annotation batch.

Two types of dataset views are available:

- **Single:** Use this view to display one record at a time to the annotator.
- **Ranking:** Use this view to display multiple records for ranking.

For more context about dataset views, read:

- How-to: Create dataset views for custom annotation user experiences (https://docs.snorkel.ai/docs/user-guide/annotation/using-dataset-views-for-generative-ai)

Prerequisites

- A Snorkel Flow instance
- A dataset for which you want to create a view
- A label schema for which you want to create a view

If you would like to use the `school_subject_questions.csv` example dataset, please download it here (https://snorkel-docs-downloads.s3.amazonaws.com/viewer/school-subject-questions.csv).

Look for "Input required" comments

This notebook requires user input for many code blocks. All fields that require user input have an `# Input required` comment.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

Using multi-schema annotations

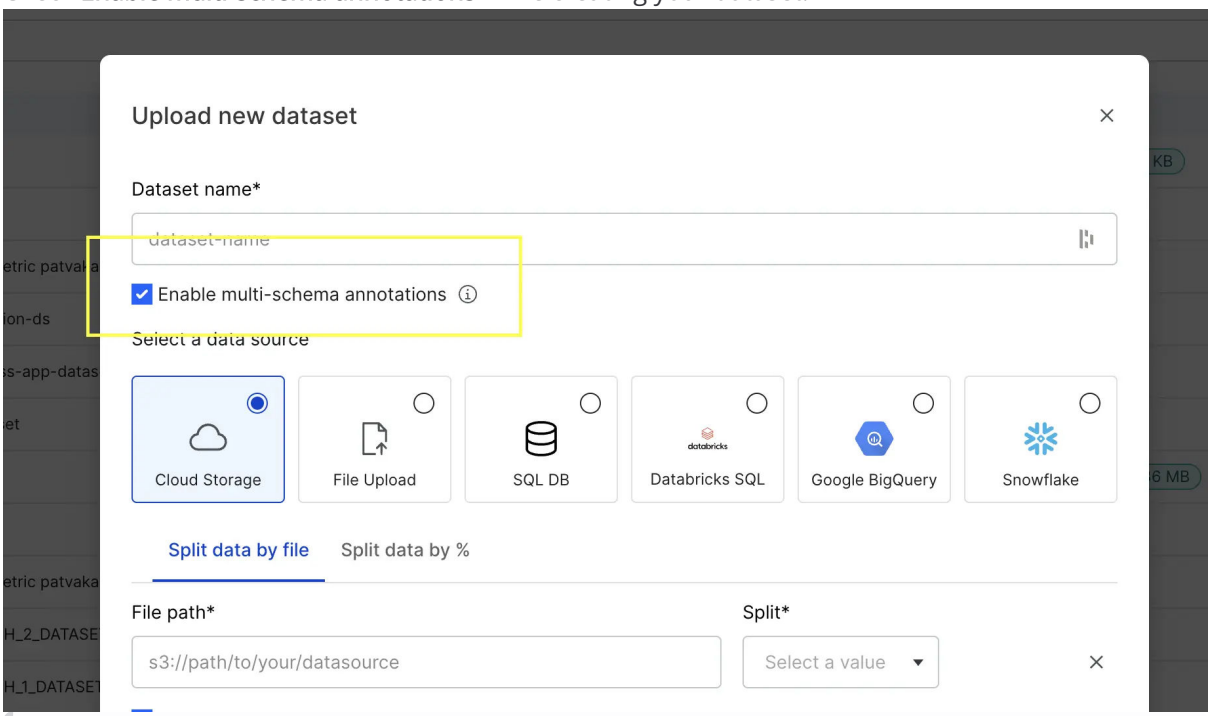
This article explains how to use multi-schema annotations, including uploading a multi-schema annotation dataset, annotating multiple schemas, and reviewing the annotations and progress.

Multi-schema annotations empowers subject matter experts to work more efficiently. This feature lets you collect annotations across multiple schemas at one time, unlocking complex workflows. With multi-schema annotations, datasets become the new home for all of your annotations and ground truth (GT). The GT is stored in a label schema for a dataset, which can be used by all of the downstream model nodes.

Multi-schema annotations are available for text-only datasets. Multi-schema annotations are not supported for PDF or image datasets.

Upload a multi-schema annotation dataset

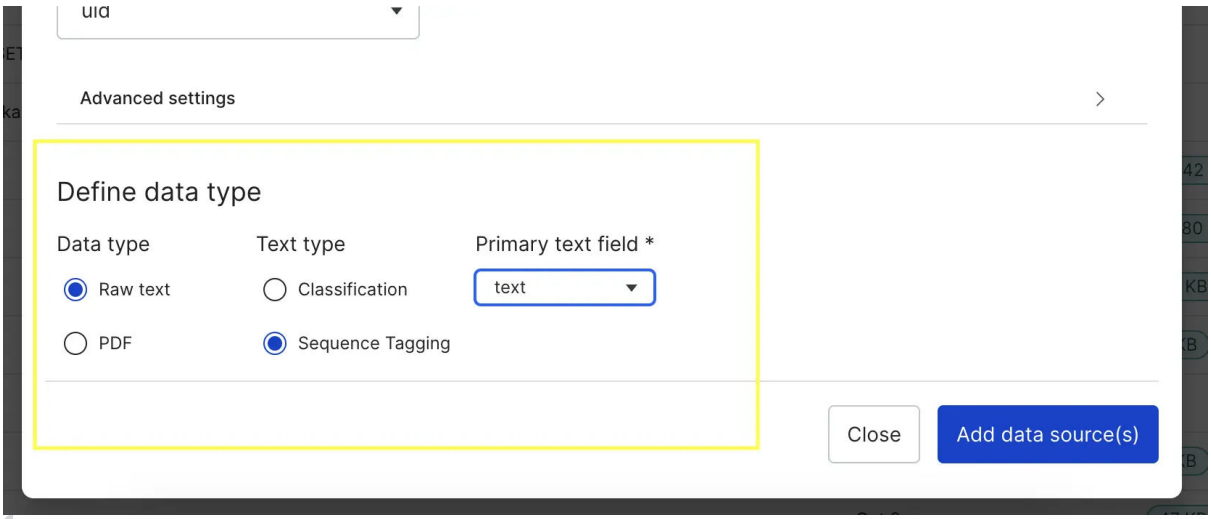
1. To create a new dataset, select **Datasets > Upload new dataset**.
2. Check **Enable multi-schema annotations** while creating your dataset.



> **NOTE**
> You cannot enable multi-schema annotations later. If you do not opt in to enabling multi-schema annotations when creating your dataset, you must create a new dataset to enable it for that dataset.

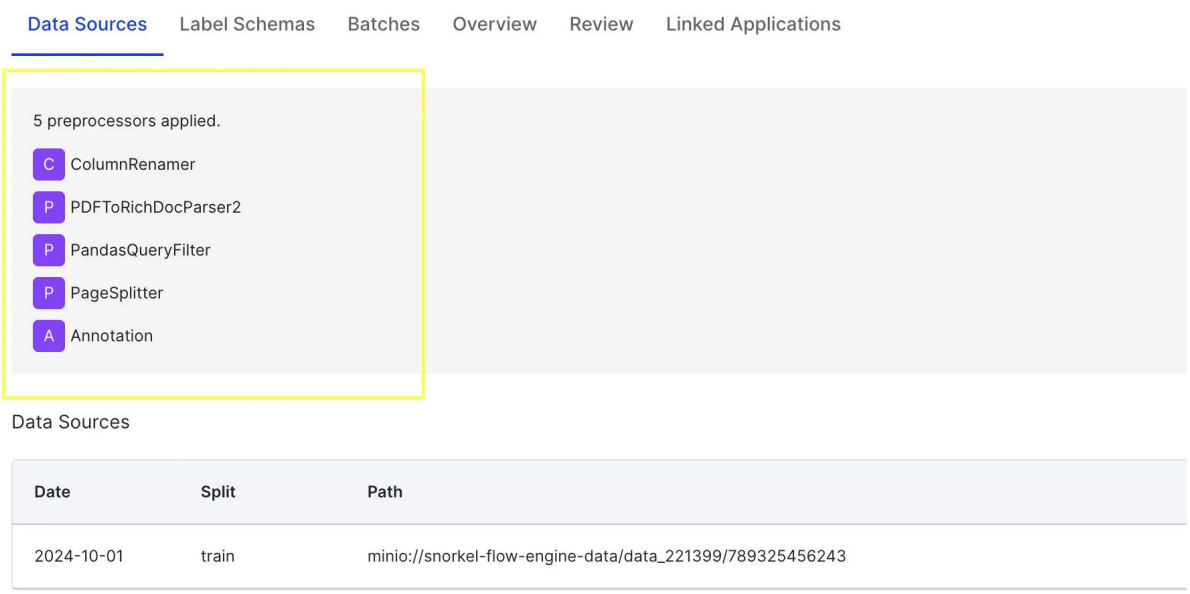
3. Enter the required information for creating your dataset. For more, see Uploading a dataset.
4. Select **Verify data source(s)**.
5. Select UID column, data type, task type and primary field within Define Schema section.

Copyright © 2025 Snorkel AI, Inc. All rights reserved



> **NOTE**
>
> Based on the data type you select, the options for task type and primary field may change. For supported data types, we'll automatically pre-process your data to make it easier for you to work with during annotation.

6. Once the data sources are uploaded, you'll see the applied pre-processors in the Data sources tab.



7. Within **Datasets > "your dataset name" > Label Schemas**, select **+ Create new label schema**.
8. Enter a name, description, data type, task type, and additional fields for each task type.
   - Classification tasks:
     1. Select **Text** as the **Primary text field**.
     2. Select Single label, Multi-label, or **Text label**. **Text label** allows for free text in your labels instead of a defined label or labels for the other options.
   - Extraction tasks:
     1. Select Sequence tagging.
     2. Enter the label for sequence tagging with a defined primary text field.

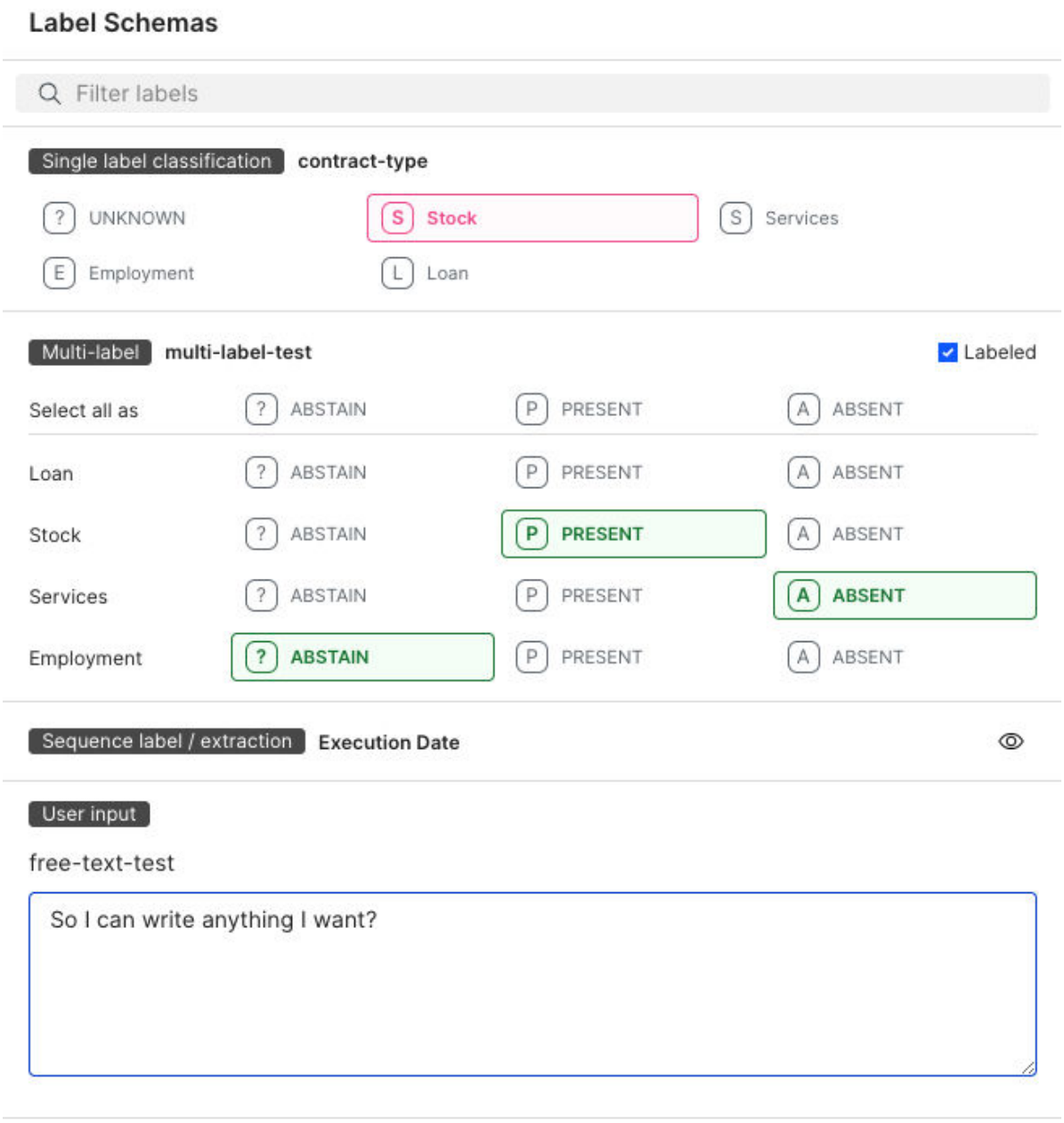
Copyright © 2025 Snorkel AI, Inc. All rights reserved

9. Select **+ Add label schema**.
10. In **Batches**,select **+ Create new batch.**
11. Enter the batch name.
12. Select your split and your label schema.
13. Enter your batch numbers and batch sizes.
14. (Optional) Assign users to annotate the batch.
15. Select **Create batch**.

Annotate multiple schemas

Snorkel Flow applies annotations to the data points across any batch in which the annotations are used.

1. In the **Batches** tab, select **Annotate** beside the batch you want to annotate.
2. Select the labels that apply to the data point.

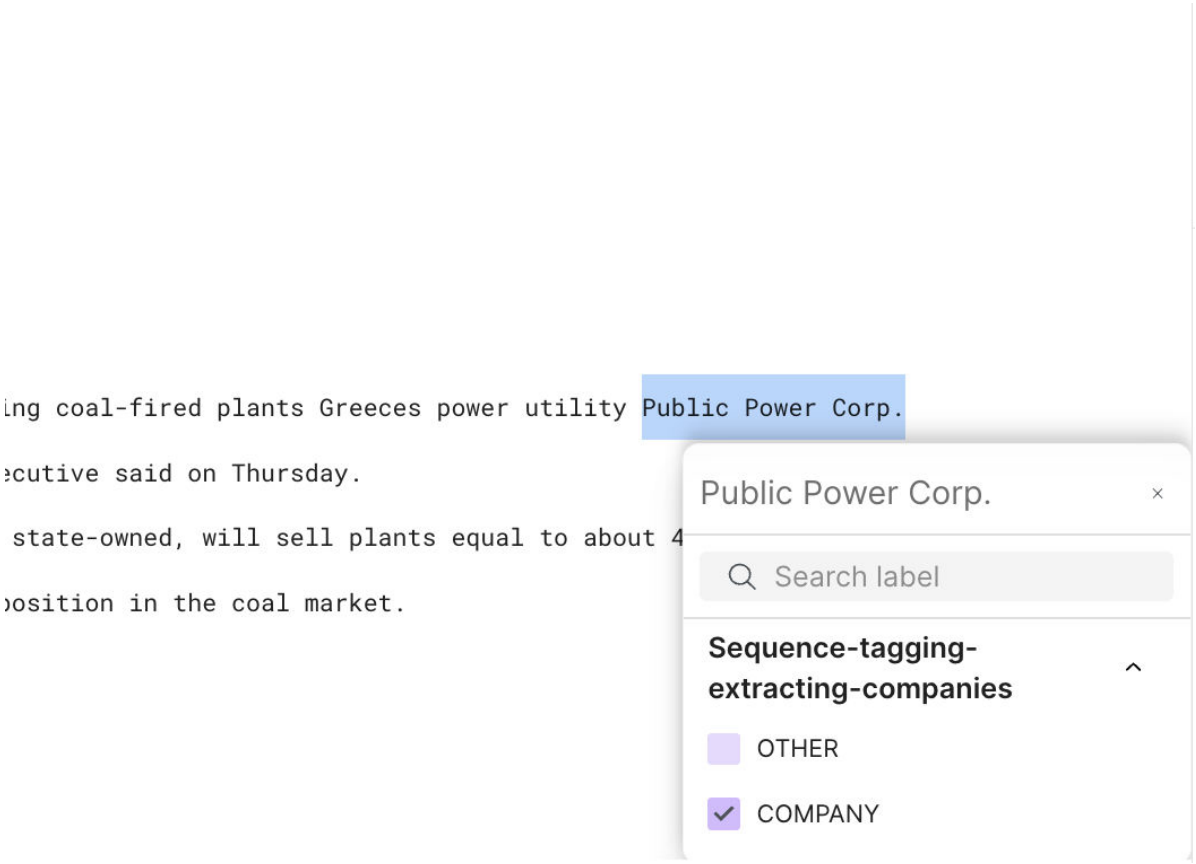


Copyright © 2025 Snorkel AI, Inc. All rights reserved

3. Select the previous and next arrows to move between data points.
4. Continue annotating each data point until you have completed your annotations.

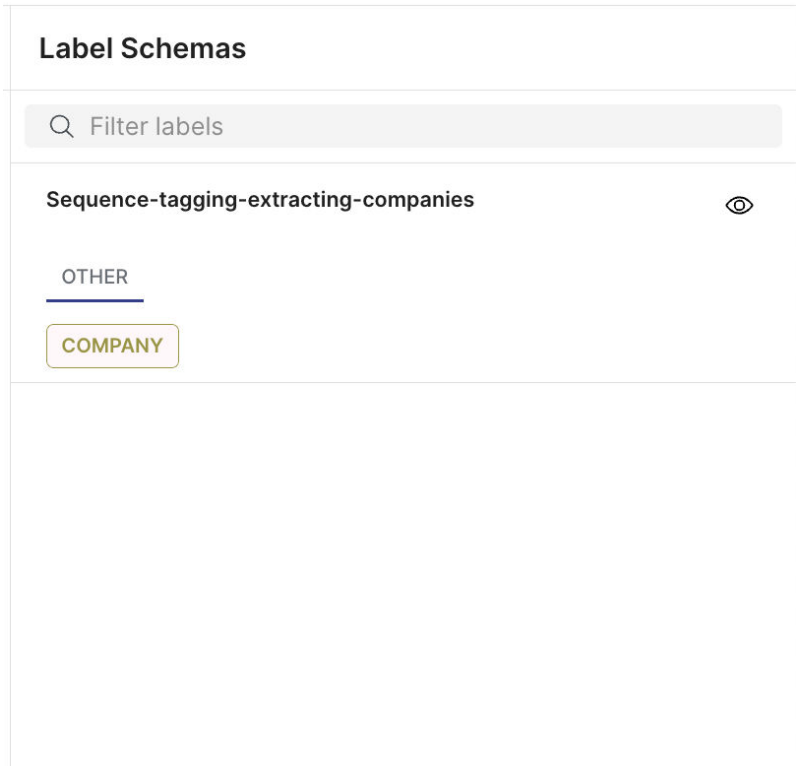
Annotate sequence tagging

Snorkel Flow supports sequence tagging for extraction tasks. Spans are key pieces of information that you want to extract from a document. To label spans in the document, you can highlight a section of text and select the span label from the pop-up menu.



If you want to apply the same label to a series of spans, first select the label from the right-side menu, and then highlight all of relevant the text segments in the document.

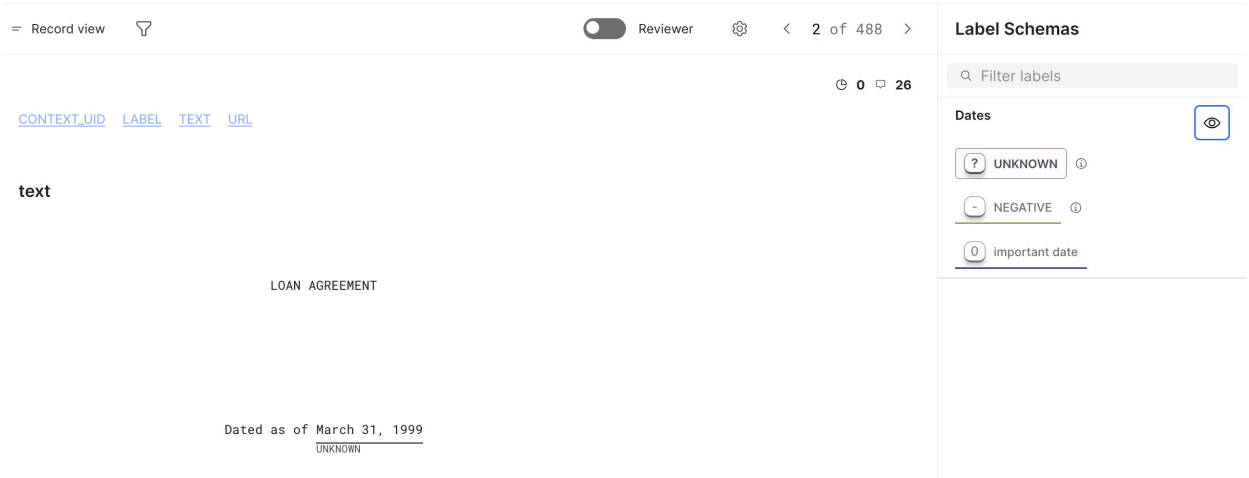
Copyright © 2025 Snorkel AI, Inc. All rights reserved



Annotate candiate extraction

Similar to sequence tagging, candidate extraction (CE) involves annotating spans in the text. The key difference is that CE schemas are pre-defined based on an extractor, such as a regex. To label these spans, click on a highlighted schema and use the sidebar to choose the label.

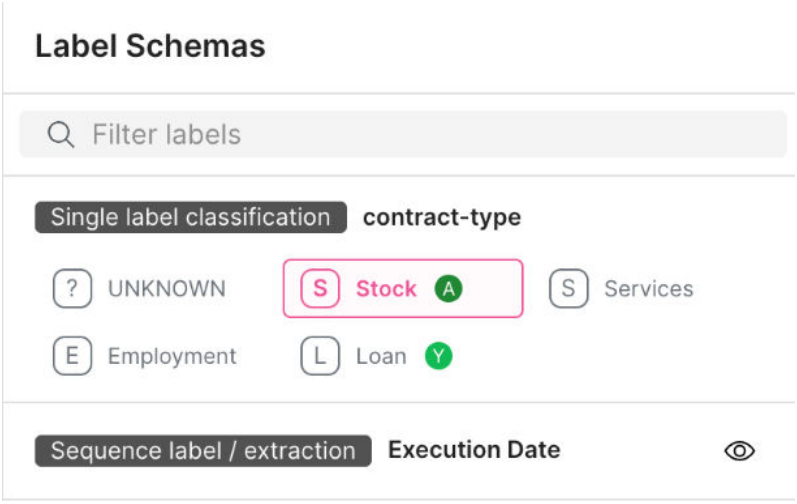
You can also use keyboard shortcuts to navigate through the spans. Right and left arrows change the documents while the top and bottom arrows move through the spans within a document. Keyboard shortcuts corresponding to the labels in the sidebar, which can be used to label a selected span.



Review annotations

1. To see the annotations from other annotators, enable **Reviewer** mode with the toggle to see all of the annotations for a dataset. You can see the annotations each annotator made.

Copyright © 2025 Snorkel AI, Inc. All rights reserved



2. Continue reviewing each data point until you have determined the correct annotations.
3. After all of the annotations are complete for a dataset and ready to be used for the ground truth, select **Datasets (sidebar nav) > Select a dataset > Select Batches tan > Expand a batch > Select an annotator/aggregated source > Commit**.

> NOTE
> You can only commit annotations from a single source, which is from a single annotator or an aggregated source. Snorkel Flow doesn't support committing annotations from multiple sources.
> Every commit overwrites the existing ground truth in your dataset.

Configure your annotation display

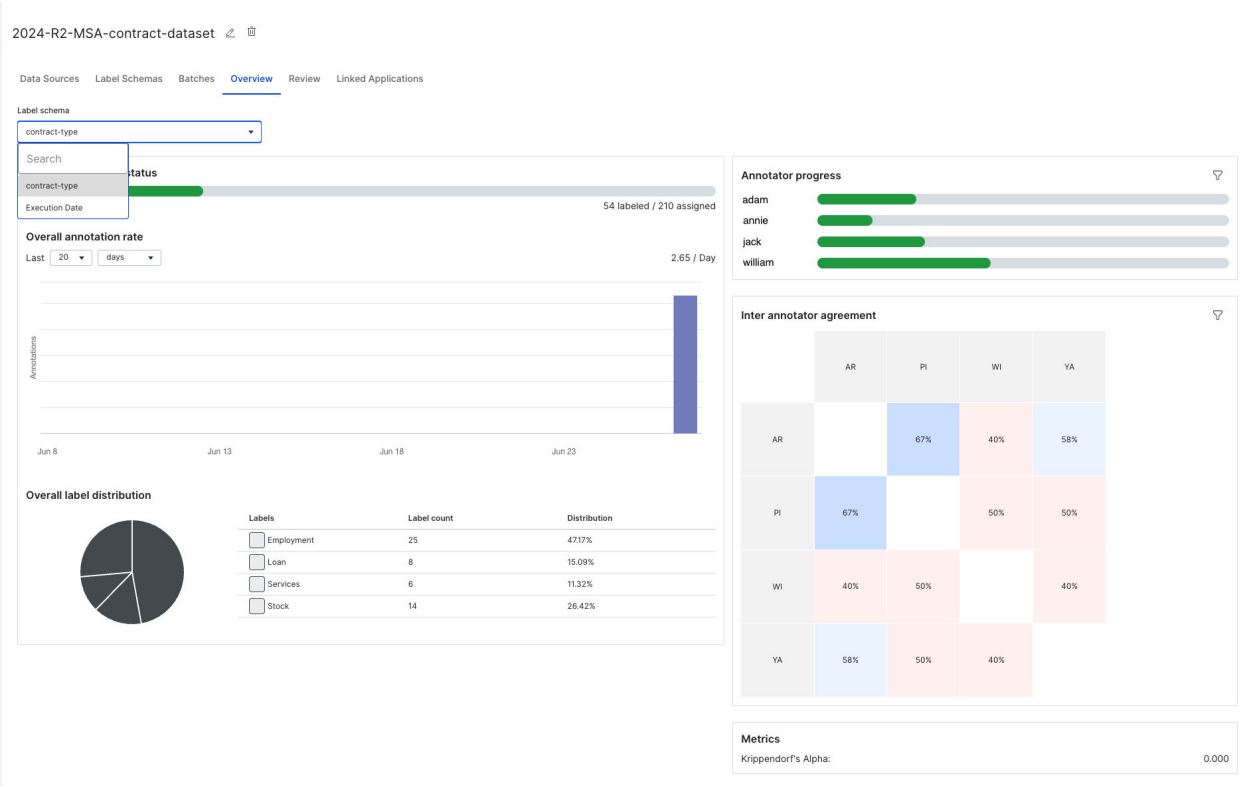
1. In the **Batches** tab, select **Annotate** beside the batch you want to annotate.
2. If you want to filter the data points to annotate, select your filtering options with fields and operators.
3. To change your display settings, select the gear icon. You can change the displayed columns, column order, and text direction. You can also prioritize unlabeled documents and set a default multi-label class.

View annotation progress

In the **Overview**, you can select your **Label schema** from the dropdown menu to see the current status and how much each annotator has finished.

You can select filters for **Annotator progress** to see the progress for specific annotators.

Copyright © 2025 Snorkel AI, Inc. All rights reserved



In the **Review** tab, you can see can see that annotators, annotations, and batches.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

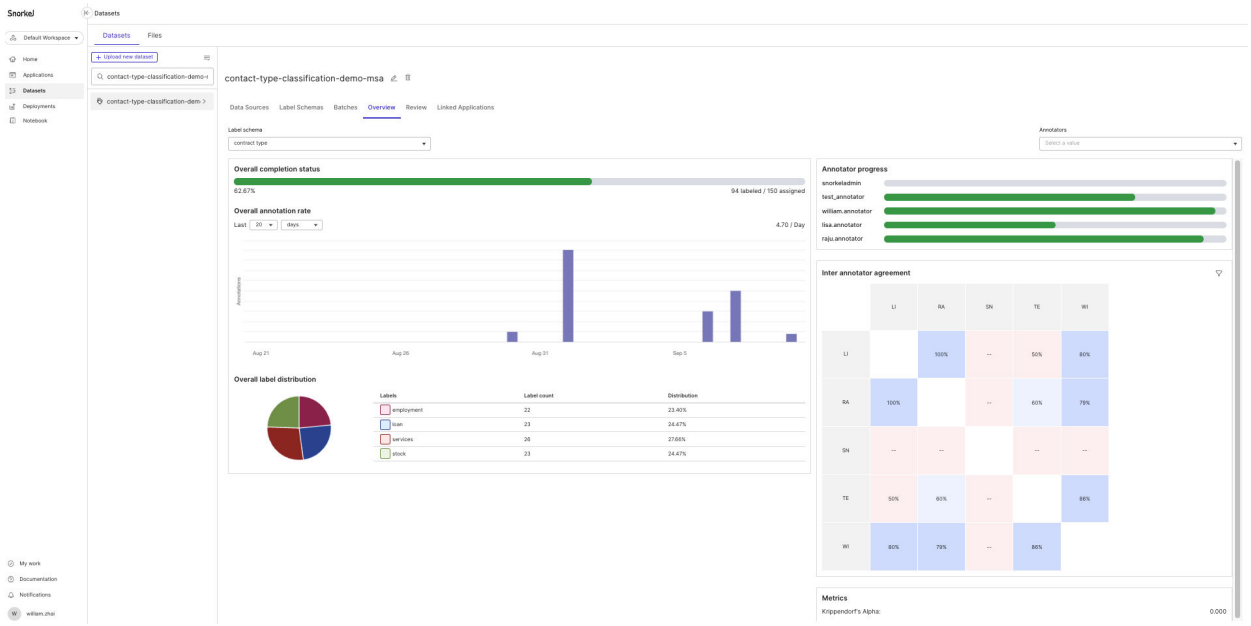
Overview page: View aggregate annotation metrics

This page walks through the summary metrics and settings that are found on the Datasets **Overview** page. This page is typically used by those with Developer and Reviewer roles that want a broad view of the annotations that have been completed as well as insight into the level of agreement among annotators.

Overview page

The **Overview** page shows you various aggregate metrics on the number of annotations that have been completed, the distribution of labels, annotator agreement, and a view into recent annotator activity.

You can select the **Filter** button on the top-left corner of the page to filter data by label schema, allowing you to view metrics specific to a particular schema. Additionally, you can use the **Filter** button on the top-right corner of the screen to filter charts by specific annotators.



Overall completion status

The **Overall completion status** chart displays the percentage of assigned datapoints that have been annotated. The progress bar gives a good visual representation of how much annotation progress has been made.

Overall annotation rate

The **Overall annotation rate** bar chart displays the aggregate number of annotations that have been completed at different granularities that you can specify (e.g., last 20 days, last 4 months).

Calendar boundaries are used when referencing days (starting at 12:00am), weeks (starting Monday), and months (starting on the 1st day of the month). For example, viewing annotations from the last 2 weeks on a Wednesday will show all annotations that have been completed since two Mondays ago, not 14 days ago. Similarly, the last X months view will show all annotations that occurred within each corresponding month.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

All annotations are displayed relative to your timezone, which is displayed in the chart. To change your timezone, select your user name in the bottom left corner of your screen. Click **User settings**, then click **Advanced**. Note that the SDK retrieves annotation timestamps in the UTC timezone, which is how timestamps are stored in the database.

Overall label distribution

The **Overall label distribution** chart displays the distribution of labels that have been annotated. This chart enables you to quickly see if there are particular classes that are underrepresented in the annotations that have been completed so far. Users with developer access can take this information and create new batches to get a more even distribution of labels. See Create batches for more information on how to create new batches.

Annotator progress

The **Annotator progress** chart displays the percentage of assigned datapoints that have been annotated, split up by each annotator. Hover over the progress bars for each annotator to see exactly how many datapoints that they have annotated compared to the number of datapoints that have been assigned to them.

Inter-annotator agreement

The **Inter-annotator agreement** chart displays the agreement rate between annotators where they have labeled the same data points. For example, if annotators A and B have annotated 10 of the same data points, and on 8 of those they have the same labels, then the shared grid will show 80% in the corresponding cell.

You can filter the chart by label class and or by batch. To do so, click the **Filters** button at the top right corner of the chart, then select your desired filters.

The **Inter-annotator agreement** chart will only show up to 10 annotators at a time to ensure a consistent, readable experience. To view the chart for all annotators, click the chart to render a modal of the full chart that is not truncated.

Below the chart you can find the value for Krippendorff's Alpha. This metric measures disagreement among annotators, corrected for disagreement that is expected from random chance. Ideally you want to see a value closer to 1, which indicates better agreement among annotators.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

(Beta) Using word-based annotations

> NOTE
> This is a beta feature available to customers using a Snorkel-hosted instance of Snorkel Flow. Beta features may have known gaps or bugs, but are functional workflows and eligible for Snorkel Support. To access beta features, contact Snorkel Support to enable the feature flag for your Snorkel-hosted instance.

Snorkel Flow supports word-based annotations within PDFs, specifically for named entity recognition (NER). You can annotate a single word or a group of words based on your annotation task, such as a person, location, or organization, etc.

Follow the steps outlined below to add and remove word-based annotations in Annotation Studio.

Annotating a single word

Follow these steps to annotate a single word in your PDF.

Annotate a single word

1. Navigate to the word you want to annotate.
2. Double-click on the word to select the word and open the annotation menu.
3. Choose the correct label from the annotation menu.

The word is now annotated with the label you selected.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

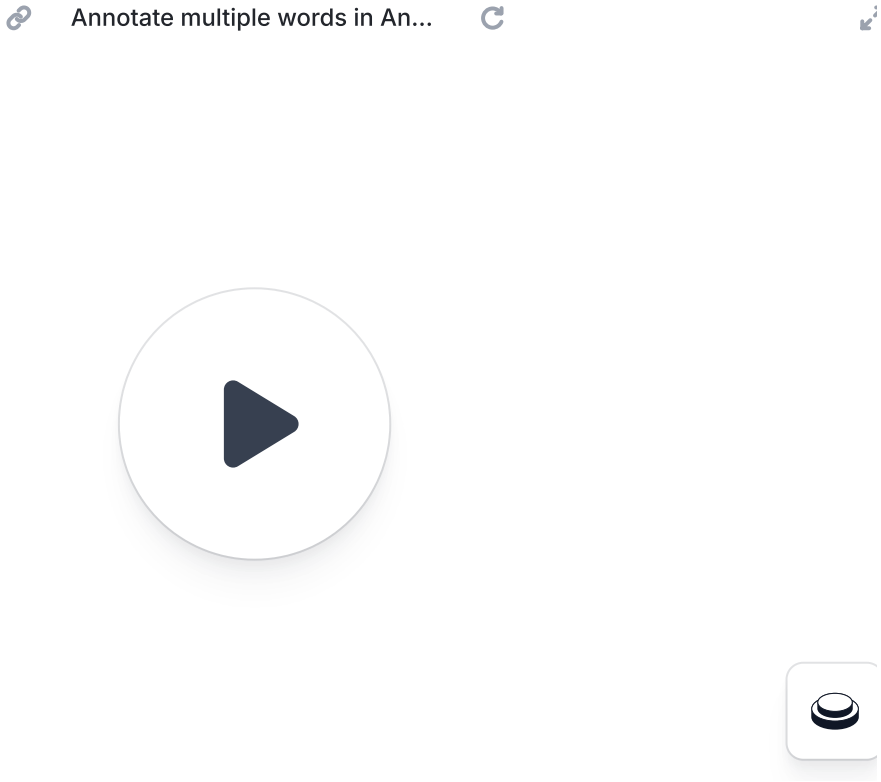
Remove annotation from a single word

1. Navigate to the annotation you want to remove.
2. Single- or double-click on the annotated word to select the word and open the annotation menu.
3. Hover over the selected label and click on the "x" icon to remove it.

The annotation is now removed from the word.

Annotating multiple words

Follow these steps to annotate multiple words in your PDF.



Annotate multiple words

1. Navigate to the section containing the group of words you want to annotate.
2. Click and hold your mouse at the starting point of the desired text, then drag to create a bounding box around the group of words.
3. Release the mouse button to highlight the selected text and open the annotation menu.
4. Choose the correct label for the selected text from the annotation menu.

The group of words is now annotated with the label you selected.

Remove annotation from multiple words

1. Navigate to the section containing the group of words from which you want to remove the annotation.
2. Click and hold your mouse at the starting point of the desired text, then drag to create a bounding box around the group of words.
3. Release the mouse button to highlight the selected text and open the annotation menu.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

4. Hover over the selected label and click on the "x" icon to remove it.

The annotation is now removed from the group of words.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

Guideline goal

Annotation guidelines allow users to describe a phenomenon or concept as generally and precisely as possible. Good annotation guidelines are helpful for subject matter experts to annotate the concept at hand in any text without running into problems or ambiguity issues.

An iterative process

Developing annotation guidelines is an iterative process. Once a pilot annotation, or first draft, is created, its shortcomings still need to be identified and fixed. For instance, issues such as unclear label definitions can be identified in the first draft, and fixed for the second draft. The second draft is also looked over for further shortcomings, which repeats the revision process and leads to the most polished version of the pilot annotation.

Note

The most important idea about the iterative nature of the annotation process is that in each round, **multiple annotators independently annotate the same text**.

Pilot annotation

The first round of annotations is best done by annotators who are familiar with the data and context around the data used. As with subsequent annotation rounds, please annotate in parallel and discuss afterwards. It's not necessary to spend a lot of time on preparation. Specifying a list of references or theoretical works, or agreeing on a single text should be sufficient as a starting point.

Your time is spent best on discussing annotation disagreements. In particular in the very first round, many parameters are still undecided and likely to cause disagreement.

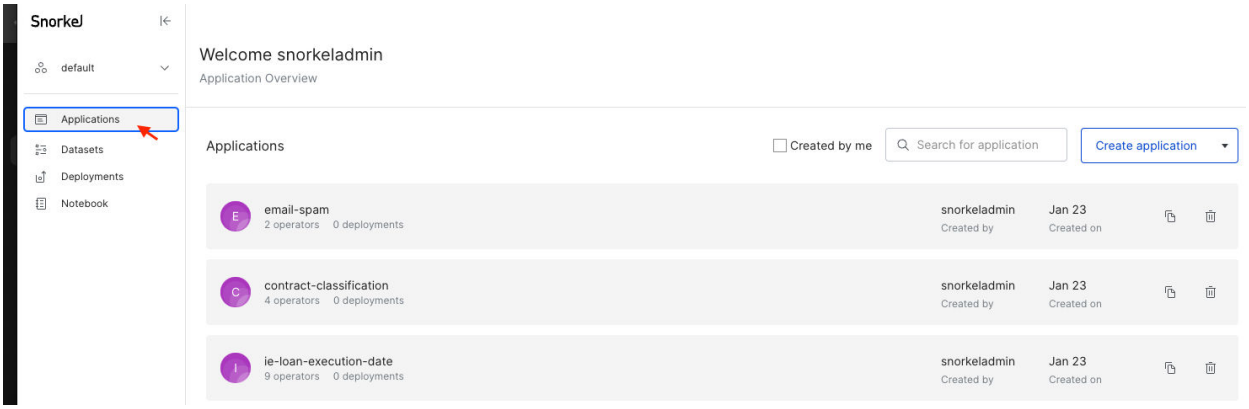
Copyright © 2025 Snorkel AI, Inc. All rights reserved

Walkthrough for annotators

This page walks through the process of manually annotating documents in Annotation Studio. This walkthrough is designed for users with the Annotator role, who need to access Snorkel Flow to annotate the documents that are assigned to them.

Access Annotation Studio

Typically when you log in to Snorkel Flow, you will land on the Application Overview page. If this is not the case, or you are elsewhere and need to get back into the application screen, click **Applications** on the left-side menu. From there, you can select the application from which you will annotate documents from. If you do not have this information, ask someone from your team.

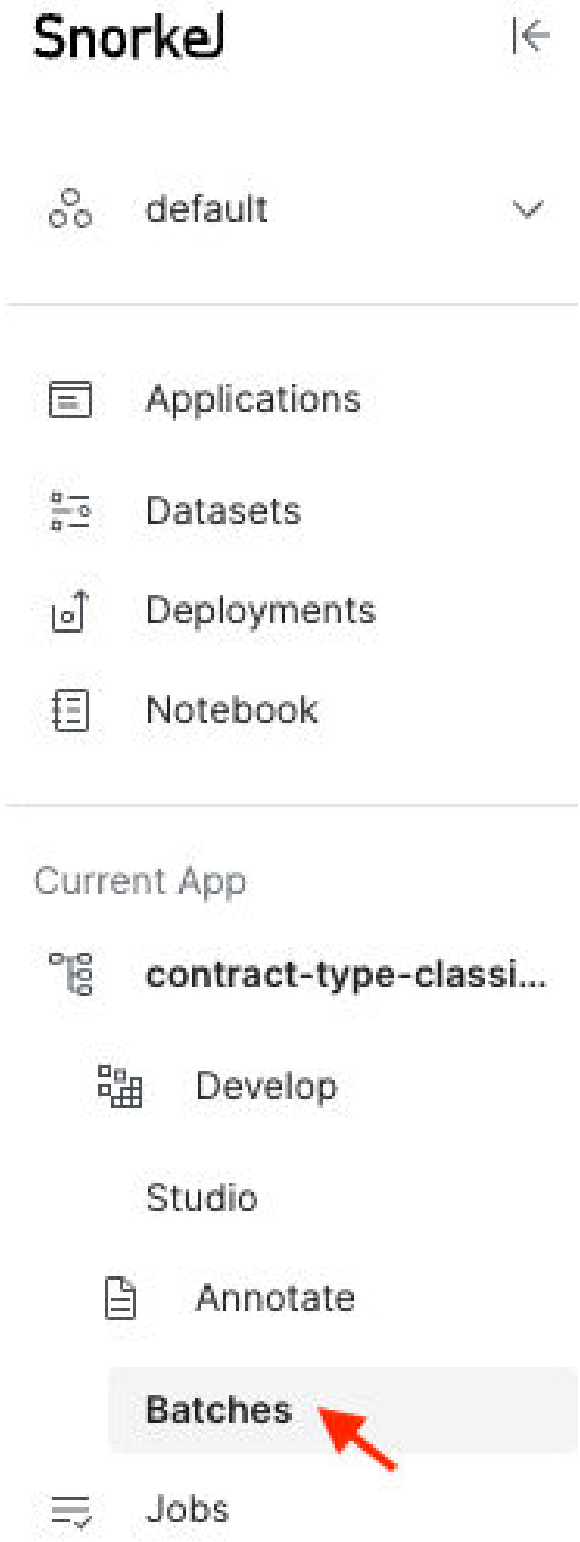


Access your batches

Once you are in an application, the next step is to access your batches so that you can start annotating. Batches are simply a collection of documents that are assigned to you to manually annotate. A batch is typically a subset of the total amount of documents. With Snorkel, the amount of documents that you will have to manually annotate is significantly smaller!

To access your batches, click **Batches** on the left-side menu.

Copyright © 2025 Snorkel AI, Inc. All rights reserved



The **Batches** page shows a list of all batches that have been created:

Copyright © 2025 Snorkel AI, Inc. All rights reserved

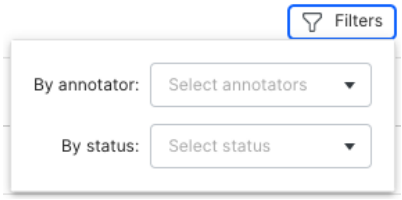
Batch	Status	Annotators	Annotated	Size	Created
batch-2-1-3	In progress	ronnie.reviewer, alex.a...	593	978	01/23/2024
batch-2-1-4	In progress	ronnie.reviewer, alex.a...	597	820	01/23/2024
batch-2-1-2	In progress	ronnie.reviewer, alex.a...	592	815	01/23/2024
batch-1-1	In progress	ronnie.reviewer, alex.a...	1463	1952	01/23/2024

The following information about each batch is available:

- **Batch**: The name of the batch.
- **Status**: Specifies whether annotations on the batch have not be started, are in progress, or have been completed.
- **Annotators**: The list of annotators that have been assigned to annotate the batch.
- **Annotated**: The number of documents in the batch that have been annotated.
- **Size**: The total number of documents that are in the batch.
- **Created**: The date that the batch was created.

To view your batches:

1. Find the batch that you want to annotate. There are two ways to find the batches that are assigned to you:
   - Look for your name under the **Annotators** column.
   - Click the **Filters** icon at the top right corner of your screen. Here, you can filter to only the batches that are assigned to you and that have not yet been completed (where **status** is either **In progress** or **Not started**).



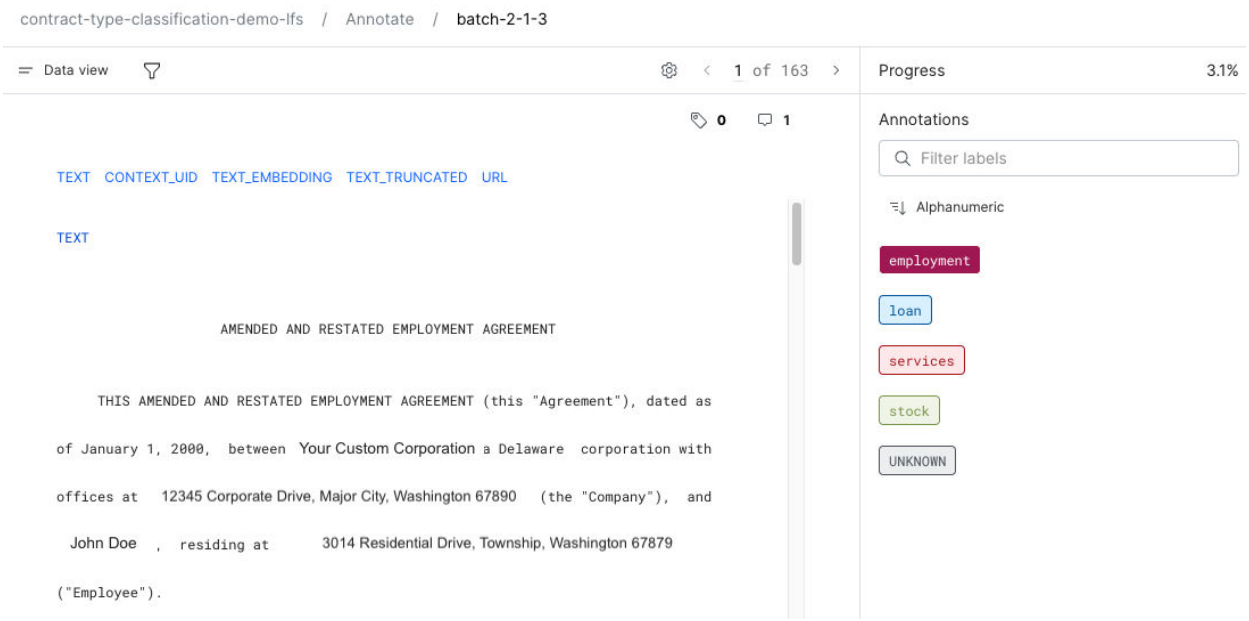
2. Click the button with the batch name under the **Batch** column.

Now, you can start annotating your documents!

Annotate your documents

Once you enter a batch, your screen will look similar to the image below:

Copyright © 2025 Snorkel AI, Inc. All rights reserved



The main canvas shows the document text. For classification and extraction applications, the right side-pane shows all classes that you can label the document.

Here is some information about the various buttons that you can see on your screen:

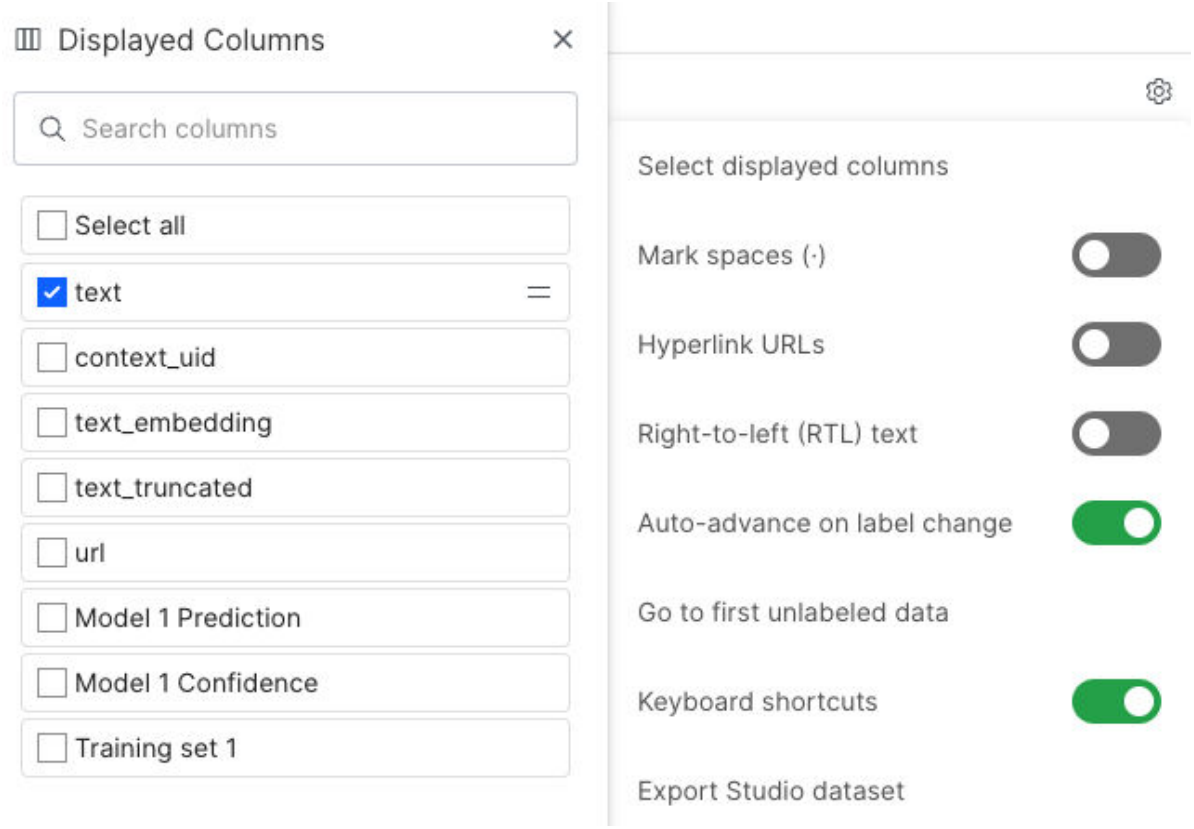
- Click **Data view** (or **Document view** for extraction applications) to change the appearance of the documents in the main canvas. The views that are available differ based on the application type.
- Click the **Filter** icon to filter the documents that are shown to you. For example, you can use the filter to just show the documents that have comments on them.
- Click the **gear** icon to adjust various settings. We recommend that you adjust a few settings before you get started annotating.
- Click the **arrow** icons to page through the documents.
- Click the **slice** icon to add slices to the document.
- Click the **comment** icon to add comments to the document. For example, you can write a comment to explain your reasoning behind a selecting a particular label.

Adjust settings

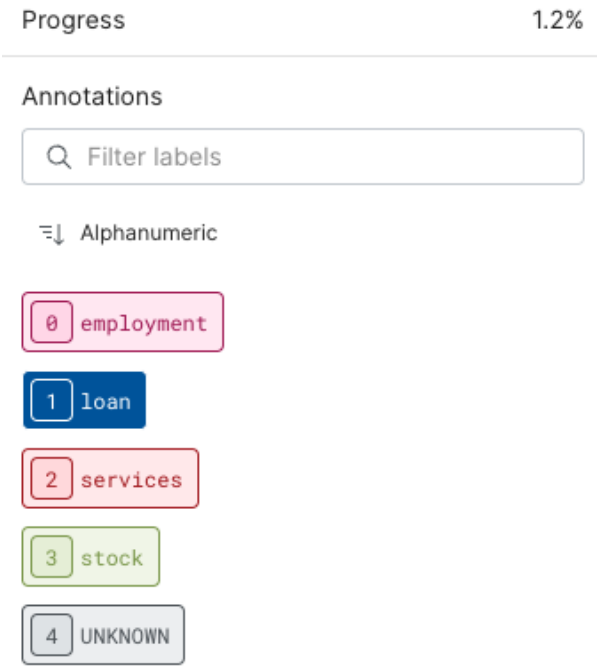
After accessing your documents, we recommend that you first adjust some settings to get some quality of life improvements while annotating! To change your settings, click the **gear** icon at the top-right corner of your screen. We recommend changing the following settings:

- **Select display columns**: This setting lets you select and display only the data columns that are necessary for marking your annotations (e.g., the text column). This removes clutter from the main canvas when you are reviewing documents.

Copyright © 2025 Snorkel AI, Inc. All rights reserved



- **Auto-advance on label change**: With this setting enabled, after you label a document, you automatically go to the next document. This prevents you from having to manually click the arrows > every time to view new, unlabeled documents. This setting is only available for single-label and text extraction applications.
- **Keyboard shortcuts**: With this setting enabled, keys are assigned to each label class, which makes it quicker to assign a label. In the example below, if you want to label a document "employment," then you just have to press 0 on your keyboard.



Copyright © 2025 Snorkel AI, Inc. All rights reserved

Here are descriptions of the rest of the settings that are available when you click the **gear** icon:

- **Mark spaces (·)**: Replaces all spaces in your document with a dot (·).
- **Hyperlink URLs**: Hyperlinks any URL in your document. This makes it easier to see links in your documents and allows you to click the hyperlink to go directly to the website.
- **Right-to-left (RTL) text**: Right justifies your document text. By default, the document text is left justified.
- **Go to first unlabeled data**: Brings you to the first document in the batch that has not yet been annotated. This option is helpful any time you re-enter a batch to continue annotating.
- **Export Studio dataset**: Export your dataset into a CSV file.

How to Annotate

The following sections walk through how to annotate documents for the four different task types:

- single-label classification
- multi-label classification
- text extraction
- sequence tagging

Single-label classification applications

In single-label classification applications, your goal is to assign a single class to each document. For example, assigning banking contract documents to one of the following classes: "employment," "loan," "services," or "stock."

On the right-side pane, you'll see all possible classes that you can label your document. To label a document, click the class in the right-side menu. Then click the arrow button to move on to the next document. If you adjusted the recommended settings, then you use can use the keyboard shortcut to assign a class, and then you will automatically be brought to the next document.

Multi-label classification applications

In multi-label classification applications, individual documents can have multiple label values. For example, let's say you are looking at movie review documents. You can label the movie as "Short Film," "Black and White," "Japanese Movies," or "World Cinema." Given these labels, you can see that a single movie can fall into multiple categories. In this case, for a given document, you can label each possible class as present, absent, or abstain from voting.

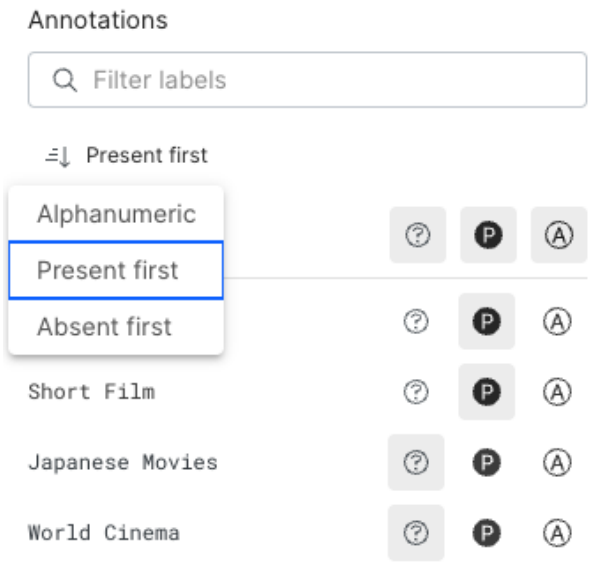
On the right-side pane, you'll see all possible classes. For each class, you can click:

- ? to label the class as Abstain.
- P to label the class as Present.
- A to label the class as Absent.

By default, all classes are initially labeled as Abstain. If enabled for your application, you can set the default label for each class to either Present, Absent, or Abstain.

You can also sort the classes, which is particularly helpful for applications with a large number of classes. You can sort classes marked as **Present first**, classes marked as **Absent first**, or in **Alphanumerical** order.

Copyright © 2025 Snorkel AI, Inc. All rights reserved



Text extraction applications

In text extraction applications, the goal is to extract key information from a document. For example, let's say that we want to extract all dates from a document. In this case, a date is considered a span. Your goal in this example is to review all highlighted spans in a document and identify whether the highlighted spans are in fact dates. You can label each highlighted span as "NEGATIVE," "POSITIVE," or "UNKNOWN" in the right-side pane. In this example, "POSITIVE" means that the highlighted span is a date, and "NEGATIVE" means that the highlighted span is not a date.

If you adjusted the recommended settings, then you can use the keyboard shortcuts to label a span, and then automatically to be brought to the next highlighted span in the document.

By default, you will be viewing the spans in Document view. We recommend annotating in either Document view or Span view:

- **Document view** shows the entire document on the main canvas. You can use the up and down arrows on your keyboard to navigate between spans of text to annotate. This view provides you with all context surrounding a span.
- **Span view** shows a single span per page. With this view, you don't have all the context surrounding a span. However, the simplified view can enable you to more quickly annotate things like dates that are easy to identify with less context.

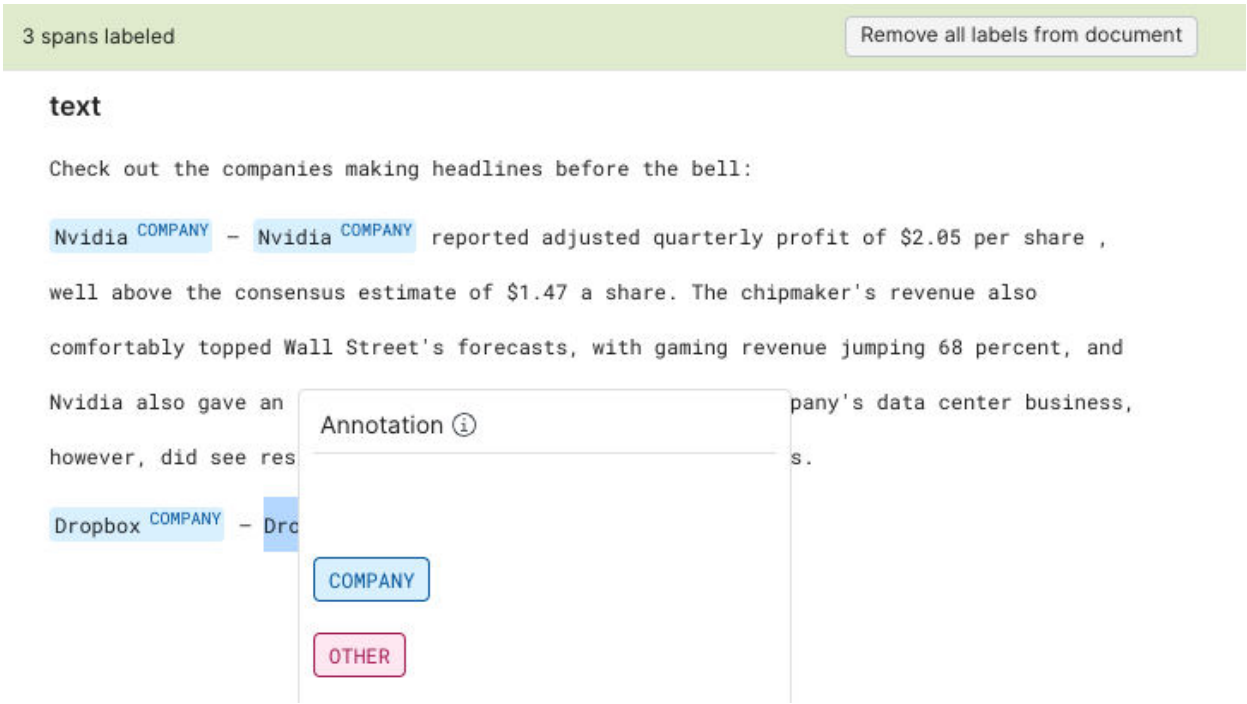
Sequence tagging applications

In sequence tagging applications, your goal is to highlight and label spans throughout a document. Spans are key pieces of information that you want to extract from a document. For example, let's say we want to identify all mentions of company names in a document. In this case, a company name is considered a span. Your goal is then to highlight and label all company names that you find while reading through the document.

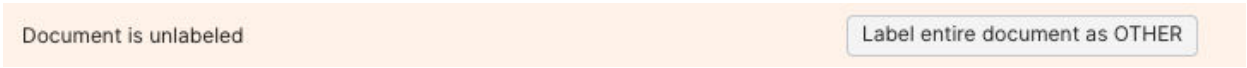
To label spans in Annotation Studio:

1. Highlight a span.
2. Select the label that you want for that span in the Annotation modal.

Copyright © 2025 Snorkel AI, Inc. All rights reserved



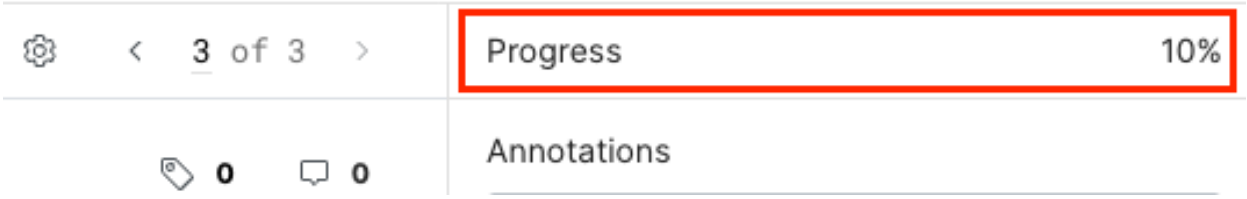
If you don't find any spans to label in the document, you can indicate this by clicking the **Label entire document as OTHER** button above the document text.



At any point, you can reset and clear all spans that you have labeled in the document by clicking the **Remove all labels from document** button above the document text.

Check progress

While you are annotating documents, you can see your progress percentage tick up in the right-hand corner of your screen.



Once you are finished, your progress bar will be labeled 100%, and the status will be updated to **Complete** on the Batches page.

Learn more about slices in Using data slices.

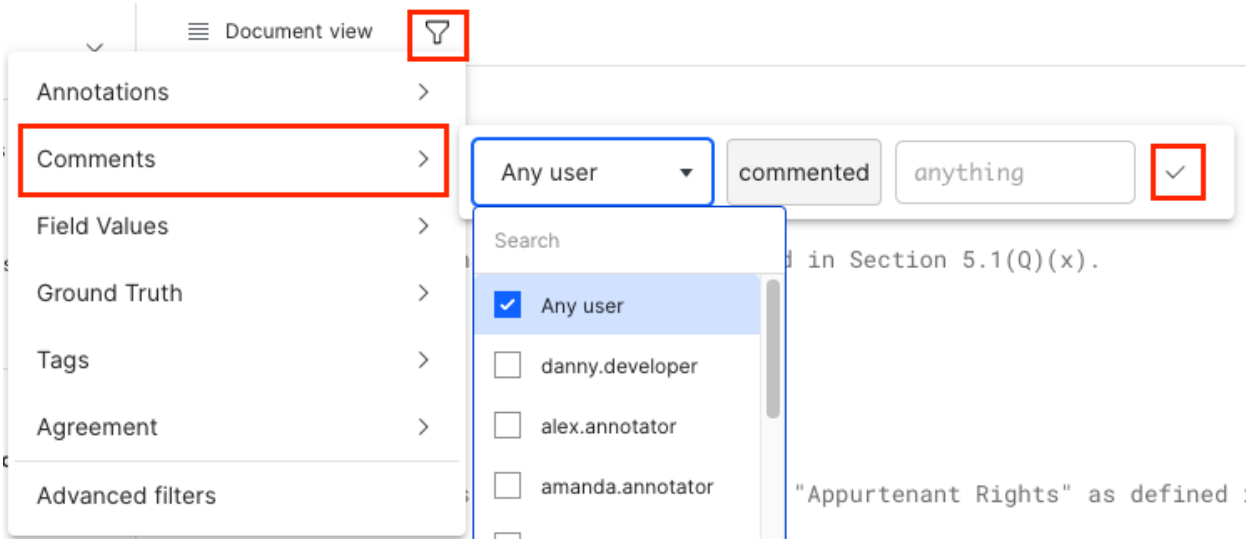
Review comments and slices

Occasionally a reviewer will leave comments on documents. For example, if you left a comment asking a clarifying question, a reviewer may respond to your comment. To see all documents with comments:

1. Click the filter icon .
2. Click **Comments**.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

3. Under **User**, select the person that you want to see comments from. Alternatively, you can select **Any user**.
4. Click the checkmark ✓ to save the filter.



Now, just the documents with comments are shown, making it easier for you to review any comments.

You can follow similar steps to filter documents based on slices:

1. Click the filter icon .
2. Click **Slices**.
3. Under **Slice**, select a slice.
4. Under **Operator**, select **is** if you want to see all documents with that slice or **is not** if you want all documents with that slice removed from view.
5. Click the checkmark ✓ to save the filter.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

Walkthrough for reviewers

This page walks through the different ways that users with the Reviewer role can review annotations from all annotators.

As a reviewer, you will likely be tasked to manually annotate documents, in addition to reviewing annotations from others. Follow the Walkthrough for annotators to get step-by-step instructions annotating documents for the different task types.

You can review annotations in two places:

- Within a batch: View all annotations made on a single document side-by-side.
- On the Review page: View a full list of all annotations and their corresponding documents.

In addition, you have some additional permissions that allow you to manage batches.

Review documents within a batch

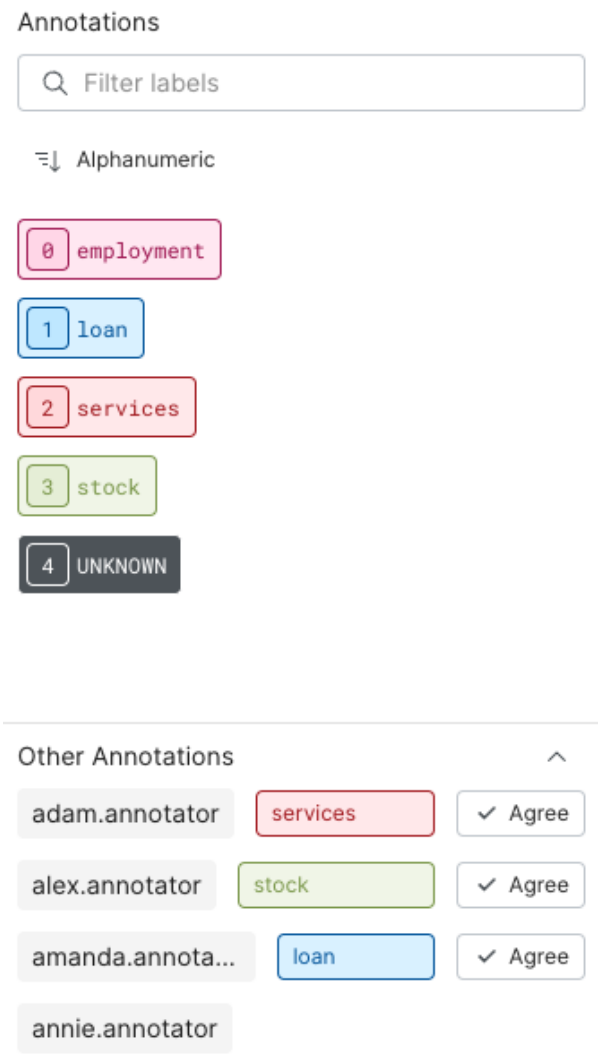
While inside a batch that you are assigned, being a reviewer allows you to review other users' annotations and mark them with agreement or disagreement. In this view, you can see all annotations made on a single document side-by-side.

How you review the annotations differs based on the task type. For all task types, you can click the **comment** icon to add an explanation for why you agree or disagree with other annotations. In addition, you can click the **slice** icon to add a slice to the document. For example, you may want to create a slice called **contains-disagreements** to slice documents that have significant disagreement among the annotators.

Review classification and extraction annotations

For classification and extraction applications, click **Other Annotations** at the bottom of the right pane to view all annotations for the document. If you agree with an annotation, you can click the **Agree** button to update your annotation to match.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

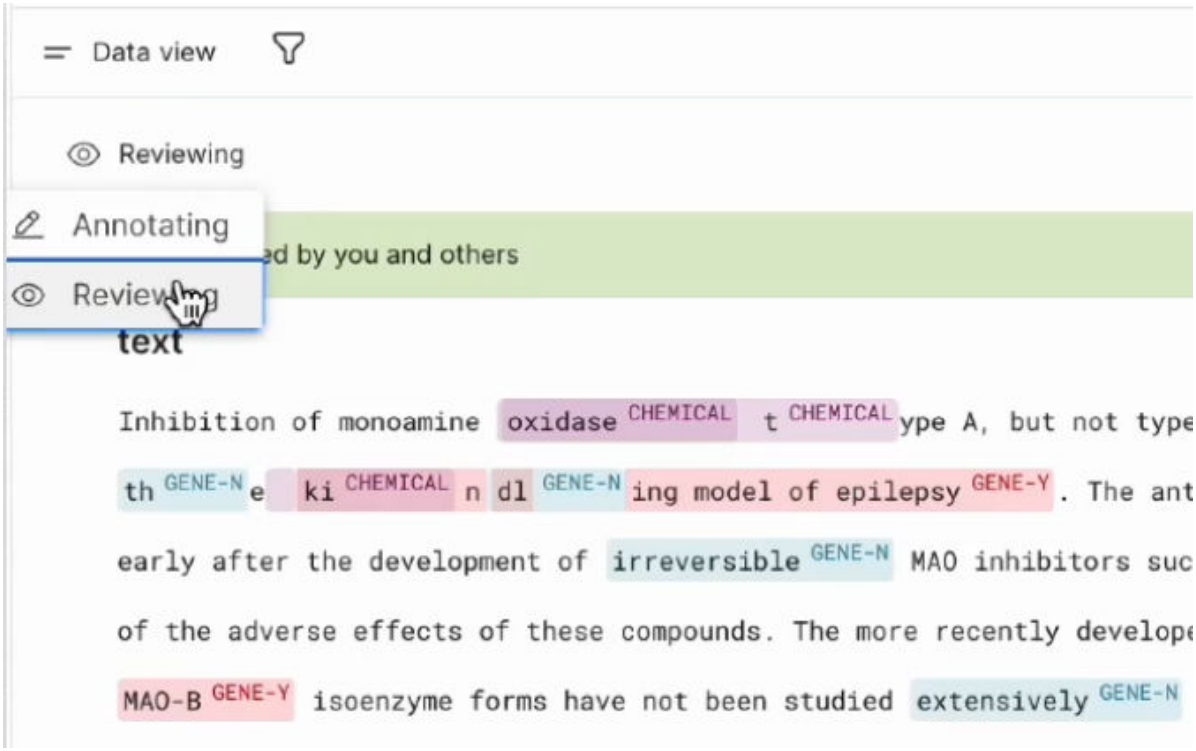


Review sequence tagging annotations

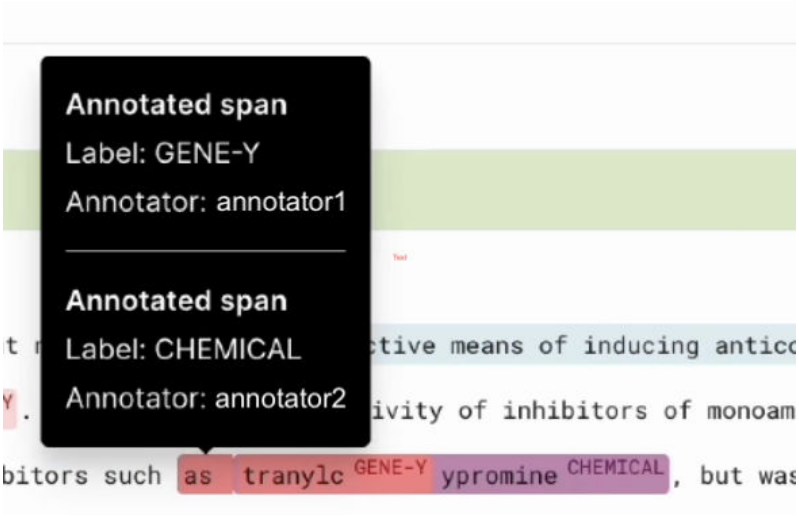
In sequence tagging applications, you can toggle between two modes: **Annotating** and **Reviewing**.

- In **Annotating** mode, you can label spans as usual and see your own annotations.
- In **Reviewing** mode, you can see the highlighted spans from other annotators.

Copyright © 2025 Snorkel AI, Inc. All rights reserved



You can visualize annotator conflict by inspecting where multiple overlapping labels exist. Click the label highlight to get more information about the span and annotator. You can use the up and down arrows on your keyboard to quickly look through the data associated with each highlight.

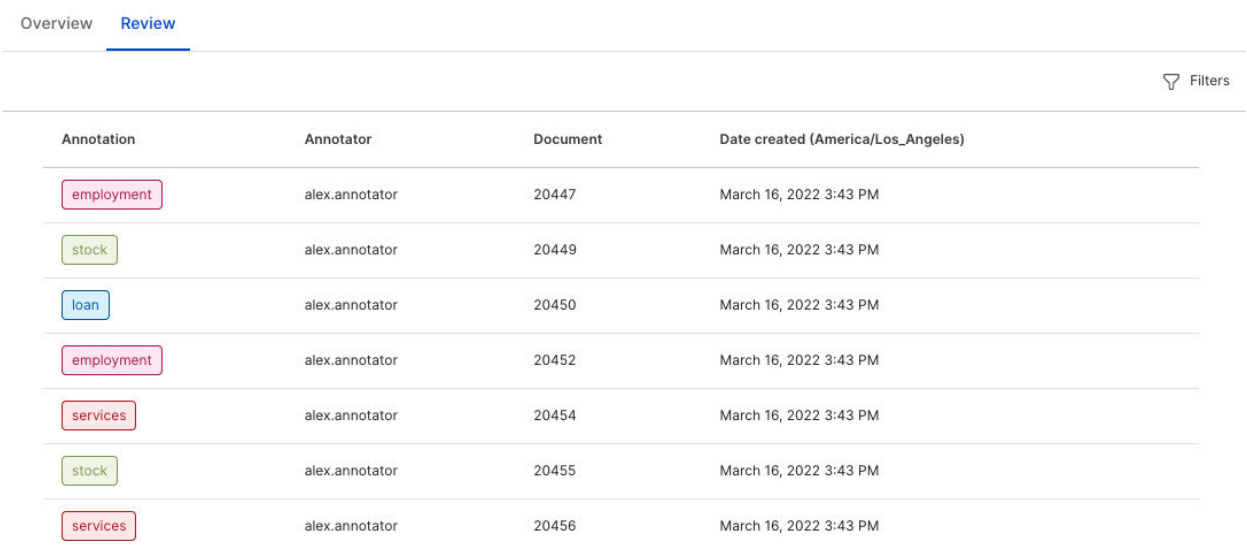


Review all annotations on the review page

The Review page provides a full list of all annotations and their corresponding documents to easily review all annotations in one place. To access the Review page:

1. Click **Annotate** in the left-side menu.
2. Click **Review** in the top-left corner of your screen.

Copyright © 2025 Snorkel AI, Inc. All rights reserved



Select an annotation to pull up the full document text. Here, you have the option to leave comments and add slices to the document. In addition, You can select the **Filters** icon on the top-right corner of your screen to filter the table by a particular annotator.

Manage batches

As a reviewer, you have additional permissions that allow you to manage batches:

- Rename and delete batches.
- Aggregate annotations in a batch.
- Set an annotator as an expert for that batch to view the agreement rate for each annotator relative to the expert.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

Create batches

This page walks through how you can create batches of documents for manual annotation when the dataset does not use multi-schema annotations.

> NOTE
>
> To create batches and commit annotations to ground truth, you must have the Developer role. For more information about access and permissions, see Roles and user permissions.

What are batches?

Before documents can be annotated in Annotation Studio, they must be assigned to batches. A batch is simply an arbitrary collection of data points—it can be as large as an entire split or data source or as small as a few examples.

Typically, you'll want documents manually annotated at a couple different points during the data development process:

- At the beginning of a project so that you can get some initial ground truth to begin development.
- During labeling function (LF) and model development. The development cycle is an iterative process and you may find that your models are struggling to predict certain chunks of documents well. For example, if you are classifying documents into one of ten different classes, the model may predict some classes better than others. In this case, you'll want some additional ground truth on those classes that the model is struggling to predict.

How to create batches

There are three ways that you can create batches in Snorkel Flow:

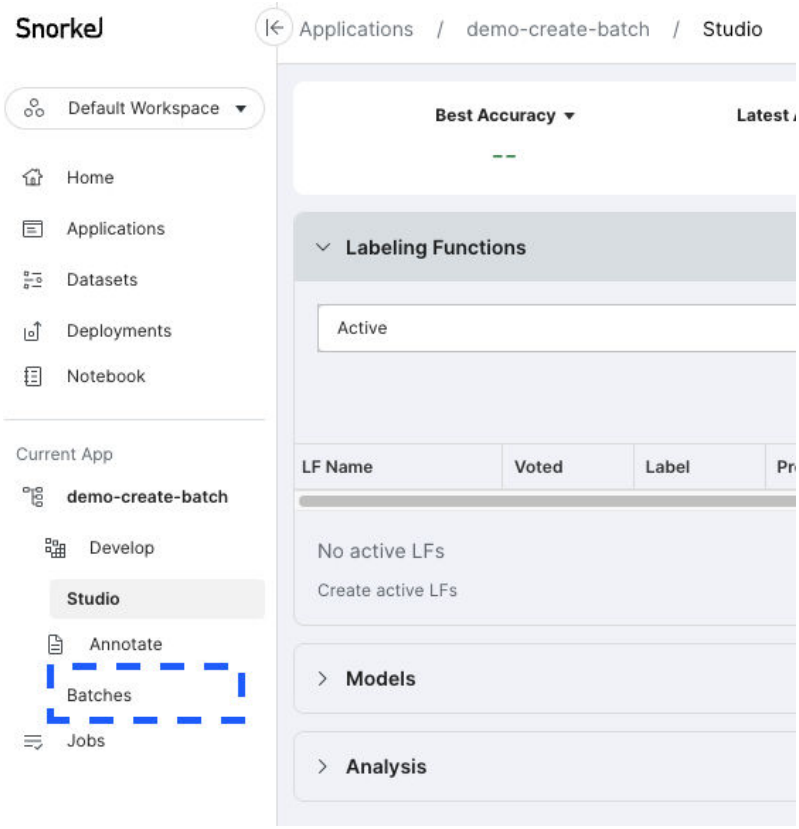
- Annotation Studio: Best method to use at the beginning of a project to get some initial ground truth.
- Develop (Studio): Best method to use during the LF and model development process were you can pinpoint chunks of data points that need more ground truth.
- In-platform notebook: Best method to use if you have a specific list of x_uids that you want to create batches from. All batches that you create can be seen and managed in the Batches page. This page can be accessed by opening an application, then clicking **Batches** in the left-side menu.

Create batches in Annotation Studio

The easiest way to create batches is in Annotation Studio from the Batches page. This method is convenient at the beginning of a project when you need some initial ground truth to begin development.

To access the Batches page, Click **Batches** on the left-side menu.

Copyright © 2025 Snorkel AI, Inc. All rights reserved



Then click **+ Create a new batch** to bring up the **Create new batch** modal. Specify the following options:

- **Batch name**: A name for the batch. If you choose to create multiple batches (by setting the **Number of batches** option to a value greater than 1), then each batch will be named the specified name with an appended numerical index to differentiate the batches.
- **Select a value**: Specifies the data source from where to create the batch. Each batch comes from a single source—either a split, a data source, or an existing batch.
- **Shuffle data order**: Shuffles the order of the data points in your data source. Batches are created by selecting data points in order up to the **Batch size** that you specify. This option enables you to randomly sample which data points go into each batch. If you select the dev split as the data source, then you can choose data selection strategy for sampling your data in the **Data selection strategy** option.
- **Data selection strategy**: If you specify the **dev split** as the data source in **Select a value** option, then you can select a strategy for data selection. You can choose from the following options:
  - **Uniform random sampling**: Randomly sample which data points go into each batch.
  - Active Learning - **Model confidence**: If you have already developed labeling functions and built a model, then set this option to create a batch from the data points with the lowest model confidence.
  - **Active Learning - Model entropy**: If you have already developed labeling functions and built a model, then set this option to create a batch from the data points with the highest model entropy.
- **Number of batches**: Specifies how many batches to divide the data source into. The default value is 1 but you can specify a value up to 100.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

- **Batch size**: Specifies the number of data points in each batch. The maximum batch size that you can specify is 10,000 data points. If no value is provided, it will divide the data source as evenly as possible among the specified **Number of batches**.

  > NOTE
  >
  > For **extraction** applications, the **Batch size** represents the number of spans. This means that if each document has a large number of spans, then the total number of documents in your batch will be significantly smaller than the **Batch size** that you specify.
  > For all other applications, the **Batch size** represents the number of documents.

- **Assign to (optional)**: Optionally select one or more annotators to assign to the batch(es). You will have the option to add and remove annotators once the batch is created.

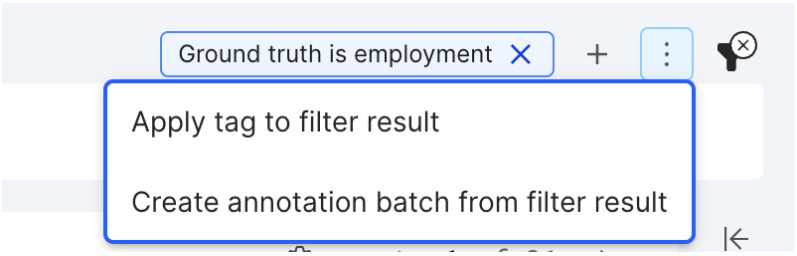
Once you are happy with your selections, click **Create Batch**. The new batch or batches can now be seen on the Batches page.

Create batches in **Develop (Studio)**

You can also create batches in **Develop (Studio)** during LF and model development. This method is convenient when you have certain chunks of data that your model is struggling to predict well. For example, if you are classifying documents into one of ten different classes, the model may predict some classes better than others. In this case, you'll want some additional ground truth on those classes that the model is struggling to predict.

To create a batch in **Develop (Studio)**:

1. Filter you data to just the types of documents that you want manually annotated. See Dataviewer: Filter data for more information about the types of filters that are available.
2. Click the three dot overflow menu, then click **Create annotation batch from filter result**.



3. The **Create new annotation batch** modal will pop up and tell you how many filters have been applied to your data. Specify the following options:
   - **Batch name**: A name for the batch.
   - **Assign to**: Select one or more annotators to assign to the batch. You will have the option to add and remove annotators once the batch is created.
   - **Sampling rate**: The percentage of documents from the filtered data set to sample in the new batch. Specify a decimal up to 1. If you specify 1, then the batch will contain all data points from the filtered set.
   - **Document overlap**: If more than one annotators is assigned to the batch, then this option specify the percentage of in documents that will overlap
   - **Split set equally**: If more than one annotators is assigned to the batch, then this option specify if the documents should be split equally among the annotators.
4. Click **Create batch**.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

5. In the **Batches created** modal, you can see information about the batch that you just created. Click the icon to copy a link that takes you to the batch in Annotation Studio.
6. Click **Cancel** to return to Studio.

To manage the batches that you create in **Develop (Studio)**, click **Batches** in the left-side menu to get to the Batches page.

Create batches with the in-platform notebook

You can also create batches using the SDK in the in-platform notebook. This method is convenient if you have a specific list of x_uids that you want to create batches from.

You can create batches in the SDK using the `create_batches` function. Some examples of how to use the function can be seen below. You can search `create_batches` in the SDK documentation for more information about the parameters.

```
# Creating batches with default parameters.
sf.create_batches(node, username="assigner", assignees=["user 1", "user 2"])

# Creating a batch with specific data points using the index. Data points must
be from the same split.
sf.create_batches(node, username="assigner", assignees=["user 1"], x_uids=
["span::5", "span::7"])

# Creating batches with a fixed number of data points.
sf.create_batches(node, username="assigner", assignees=["user 1", "user 2"],
batch_size=10)

# Creating batches with a fixed number of documents.
# In candidate-based extraction applications the data points are spans, but
users would like to create batches of documents.
sf.create_batches(node, username="assigner", assignees=["user 1", "user 2"],
batch_size=5, sample_by_docs=True)
```

To manage the batches that you have created with the SDK, open your application in Snorkel Flow, then click **Batches** in the left-side menu to get to the Batches page.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

Legacy ground truth annotations

In this article, you will learn about ground truth annotations and how to add these annotations for multiple use cases.

Ground truth refers to the set of labeled and accurately annotated data that serves as a reference or benchmark for training and evaluating machine learning models. During the evaluation phase, ground truth data is used to assess the model's performance. By comparing the model's predictions with the actual labels in the ground truth, metrics such as accuracy, precision, recall, and F1 score can be calculated.

Snorkel Flow allows you to programmatically generate labels for your training split. Snorkel recommends using ground truth (GT) labels for data points in other splits:

- In the dev split, GT-labeled examples assist with discovering and iterating on labeling functions (LFs).
- In the valid and test splits, use GT-labeled examples to evaluate model performance and facilitate error analysis.

Annotation options

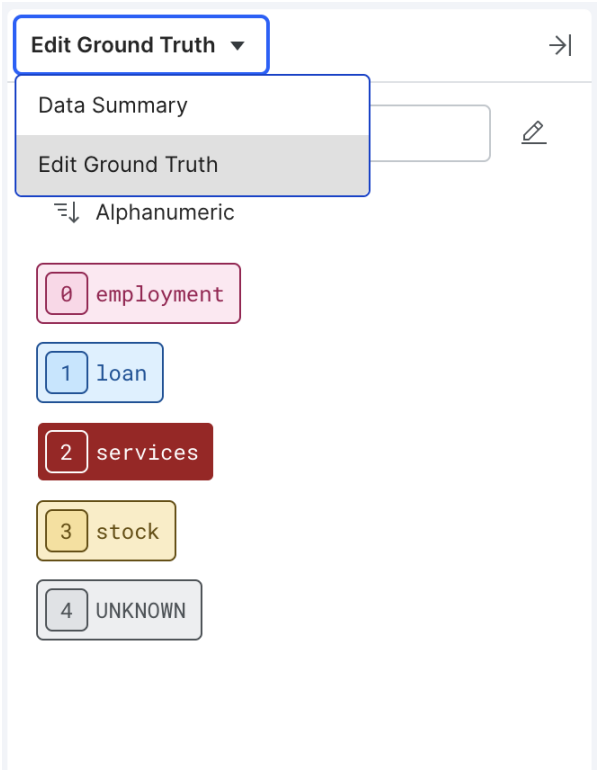
There are two ways to update GT labels in Snorkel Flow:

- Updating GT labels on the **Label** page
- Creating and committing annotations in **Annotation Studio** to GT labels

Updating GT labels directly on the **Label** page

1. On the Label page, update GT via the top bar or the Annotation sidebar.
2. To enter **Annotation Mode** on the **Label** page, select the collapse icon on the right side of the dataviewer.
3. Ensure **Edit ground truth** is selected in the dropdown menu.

Copyright © 2025 Snorkel AI, Inc. All rights reserved



4. To exit **Annotation Mode**, select the collapse icon on the top-right corner of the pane.

Annotation Studio

Annotation Studio consists of two pages: **Batches** and **Annotate**.

The **Annotate** page is a bird's-eye view of all data points to review annotations. You can see different metrics such as the number of completed annotations, number of annotators, label distribution, and more.

The **Batches** page acts as a workspace to create and manage batches of data points to annotate.

Follow these steps to apply GT labels in the Annotation Studio:

1. Via the **Batches** page, create and assign batches of data points to specific users to annotate.
2. Annotators view the data points in a batch and propose GT labels for each.
3. (Optional) Aggregate annotations from multiple sources via majority vote on the **Batches** page to form a new set of annotations.
4. Select a specific annotation set from a batch to commit as GT for those data points.

For more information about **Annotation Studio**, see Annotation Studio overview.

Adding annotations

The steps for adding a GT label in the **Label** page or proposed GT label in the **Annotation** page are the same, but vary by application type.

In addition to GT labels, you can add **notes** for free-form comments or **slices** for arbitrary grouping.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

Classification: Adding ground truth

In **Record View**, modify the GT label of an individual data point using the dropdown at the top of the dataviewer.

In **Annotation Mode**, select on the appropriate label in the **Annotation** pane or use the corresponding shortcut key to the left of the label.

Extraction: Adding span ground truth

In extraction tasks, GT labels are collected for individual spans. For a description of how these spans are created, see the Information extraction: Extracting execution dates from contracts tutorial.

Once GT extractions exist, you can propagate these labels to individual spans using the **auto-fill** functionality in the **Annotation** pane. Auto-fill performs a string comparison between document level extractions and all spans found in their corresponding documents. If a span matches a specified GT extraction, it receives a Positive label. All others in that document receive a Negative label.

Entity classification: Adding entity ground truth

For entity classification tasks, you can assign a GT label per entity in each document using the **Annotation** pane. You can explicitly update the GT label for a given span in **Record View** by selecting it and updating the GT label dropdown. This change updates the corresponding entity label, too.

Auto-fill works similarly to Extraction tasks, except labels are propagated based on entity assignments, rather than string matches.

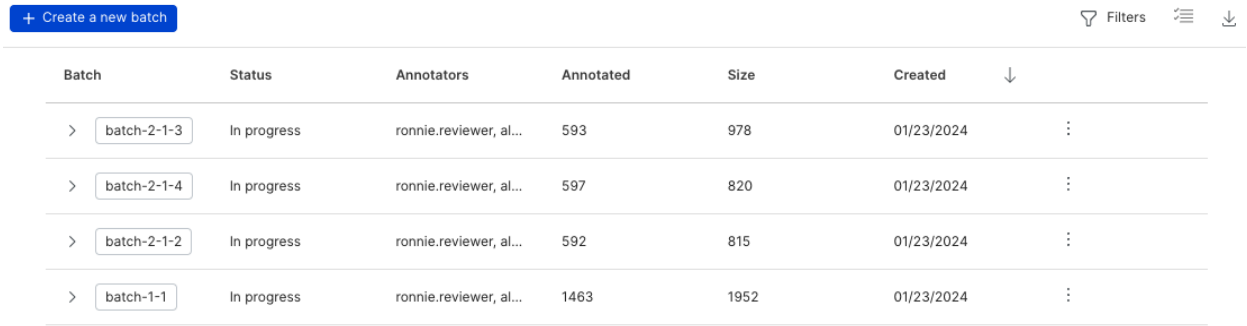
Copyright © 2025 Snorkel AI, Inc. All rights reserved

Manage batches and commit ground truth

This page walks through how to manage your batches and commit annotations to your ground truth to be used for development in **Develop (Studio)**.

Manage your batches

Once you have create batches, you can manage the batches in Annotation Studio from the **Batches** page. Select **Batches** in the left-side menu to get to the **Batches** page. It will look alike following screenshot-



For individual batches, the following information is available:

- **Batch:** The name of the batch. Select the batch name to access the batch.
- **Status:** The status of whether annotations on the batch have not been started, are in progress, or have been completed.
- **Annotators:** The list of annotators that have been assigned to annotate the batch.
- **Annotated:** The number of data points in the batch that have been annotated.
- **Size:** The total number of data points in the batch.
- **Created:** The date the batch was created.
- **Overflow menu:** The ⋮ icon used to rename or delete the batch.

> NOTE
>
> You can select any of the columns to change the sort order of the table.

On the top bar, select the action you want to take:

- Create new batch: Select the **+ Create a new batch** button to create a new batch of data points for annotation.
- Filter batches: Select the **Filters** button to filter the batches by annotator or status.
- Delete batches: Select the **Toggle bulk options** icon to show the option to bulk delete batches.
- Export data: Select the **Export dataset across all batches** icon to export all batches to a CSV file.

More on Export data

It might be helpful to export the current dataset and annotations for further analysis or reporting outside of Snorkel Flow. You can export data across all batches or at the batch level on the **Batches** page.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

Export all data

1. To export a dataset that includes all batches, select the Export dataset across all batches icon in the top right corner of your screen.
2. Fill out the following options in the **Export annotation batches** modal:
   - **Include columns:** Specify which columns to include in the export. This option enables you to select a focused slice of your data, or you can speed up export time by excluding large columns in a dataset.
   - **Include options:** Specify additional information about individual data points to include such as annotations, comments, slices, filters applied, and model predictions.
   - **Optional settings:** Specify additional options to customize your export. These options include the ability to start an export at a specific index in the dataset, setting a maximum number of data points to export, and options to configure delimiters, quote characters, and escape characters.

Export data from single batch

1. To export a dataset for an individual batch, open the desired batch by clicking on it's name.
2. Inside the batch annotation, Click on "Export" button at the top right corner of your screen.
3. Fill out the **Include columns**, **Include options**, and **Optional settings:** options. You can also include predictions from a specified model.

Manage individual batches

Coming back to batches page again, select the arrow > next to the batch name to view additional information about an individual batch. This menu give you additional actions you can take on an individual batch.

Assign/update annotators to batches

1. Click the **+ Add Annotator** / **Update Annotators** button.
2. In the **Assign** modal, select one or multiple annotators from the drop-down list to assign to the batch.
3. Select **Update annotators** to add the selected annotators.

Set annotators as experts

You can mark an annotator as an expert, which populates the **Agreement** column for every annotator on that batch with an agreement rate relative to the expert. To view the agreement rate across all batches and annotators, see the **Inter-annotator agreement** chart on the **Annotate** page.

To mark an individual annotator as an expert for the batch, select the annotator, and then click **Set as expert.** A badge icon shows up next to the annotator's name to indicate the expert status.

To remove the expert designation for an annotator, select **Remove expert status**.

Aggregate annotations

Typically you'll have more than one annotator reviewing and labeling documents. You can only commit a single vote to ground truth. You can aggregate annotations instead of committing the annotations from a single person. This aggregation helps eliminate potential bias from any particular annotator.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

To aggregate annotations, select the annotators that you want to aggregate, then select **Aggregate**.

A majority vote is the only supported aggregation strategy. This strategy takes the majority label for each data point if one exists, and leaves an `UNKNOWN` label where no annotations exist. These results create a new set of annotations with a single vote for each data point. The aggregated annotations can be seen in the expanded view of batch on the **Batches** page, under **Other annotations**.To know more about aggregation strategies for different task types, refer to Aggregate annotations.

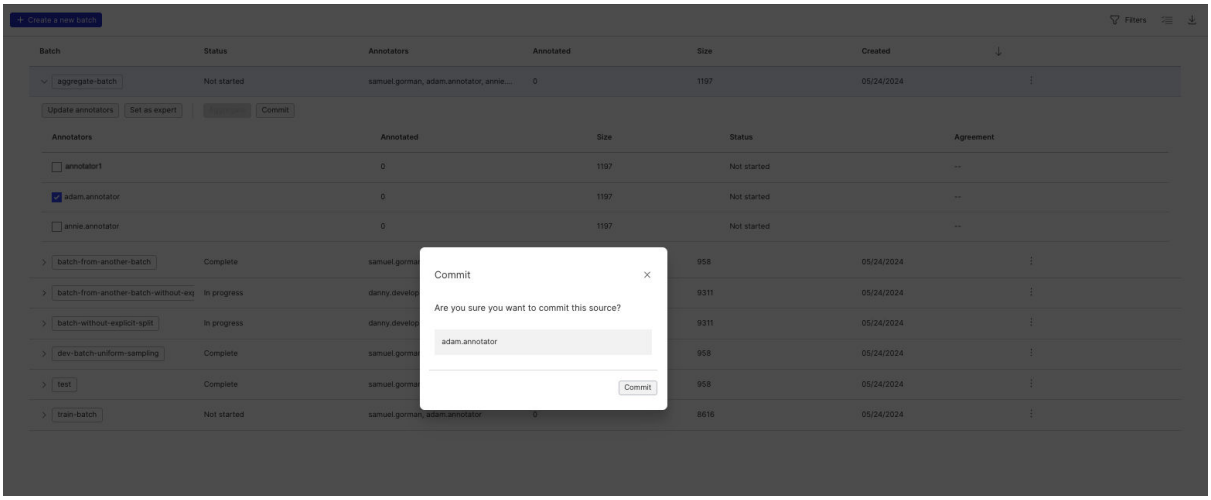
Commit annotations

Once you have a set of annotations that are accurate, commit the annotations as the ground truth for development in **Develop (Studio)**.

> **NOTE**
> Only users with the **Developer** or **Administrator** role can commit ground truth.

1. Select the desired annotation set. You can select annotations from either an individual annotator or from an aggregated set of annotations.
2. Select **Commit**.

   > **WARNING**
   > Once you commit annotations, the new annotations overwrite the existing ground truth labels in the data source.



Copyright © 2025 Snorkel AI, Inc. All rights reserved

Using dataset views for generative AI

Dataset views power the generative AI annotation data viewers. Snorkel Flow offers dataset view options with various benefits:

- **Single response review:** To view and annotate LLM responses one by one.
- **Response ranking:** To rank multiple LLM responses.

To create a dataset view

You need to create the dataset view once per dataset. You can create the dataset view using `snorkelflow.client.dataset_views` in the SDK.

Single response view

```
dataset_name = "your-dataset"
view_name = "Single Response View"
view_type="single_llm_response_view"
column_mapping={
"instruction": "questions",
"response": "responses",
"context": "rc_title_text",
"prompt_prefix": "prompt_prefix"}

sf.create_dataset_view(
  dataset=dataset_name,
  name=view_name,
  view_type=view_type,
  column_mapping=column_mapping)
```

Ranking view

The ranking view can scale to as many labels as desired. Once Snorkel Flow assigns a rank to a response, the response changes from white to gray. When Snorkel Flow assigns a rank, it rearranges the responses in the user interface by ascending order. The responses are grouped by the prompt and retrieved content.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

```
dataset_name = "your-dataset"
view_name = "Ranking View"
view_type="ranking_llm_responses_view"
column_mapping={
"**instruction**": "questions",
"**response**": "responses",
"**context**": "rc_title_text",
"**prompt_prefix**": "prompt_prefix"}

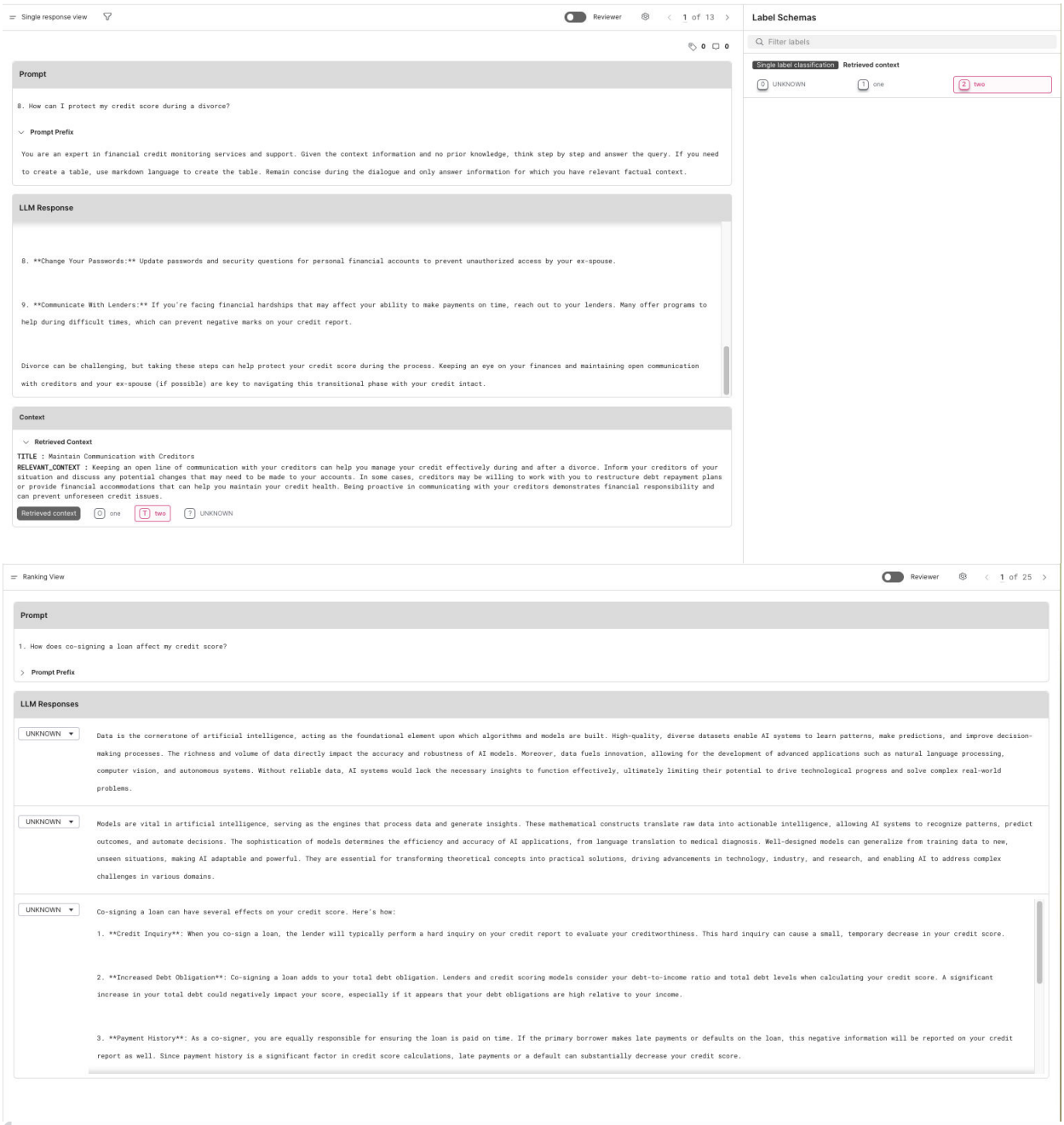
# get ranking schema UID for create_dataset_view
dataset = Dataset.get(dataset_name)
for label_schema in dataset.label_schemas:
print(label_schema.name, label_schema.uid)

sf.create_dataset_view(
**dataset**=dataset_name,
**name**=view_name,
**view_type**=view_type,
**column_mapping**=column_mapping,
**label_schema_uids**=[<ranking schema id>])
```

To view a dataset

From within a batch, select the dataset view from the dropdown:

Copyright © 2025 Snorkel AI, Inc. All rights reserved



NOTE

To annotate on retrieved context in the view, the data for a column has to be in a valid JSON format. For example:

```
[
    {
        "TITLE": "Raise Credit Score Quickly",
        "RELEVANT_CONTEXT": "Once you've paid …”
    }
]
```

The keys (`title`, `relevant_context`) are in correctly formatted in quotes.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

Copyright © 2025 Snorkel AI, Inc. All rights reserved

Using multi-schema annotations

Multi-schema annotations unlock complex workflows by letting you collect annotations across multiple schemas at one time. This feature empowers subject matter experts to work more efficiently.

With multi-schema annotations, datasets become the new home for all of your annotations and ground truth (GT). The GT is stored in a label schema for a dataset, which can be used by all of the downstream model nodes.

Multi-schema annotations are available for text-only datasets. Multi-schema annotations are not supported for PDF or image datasets.

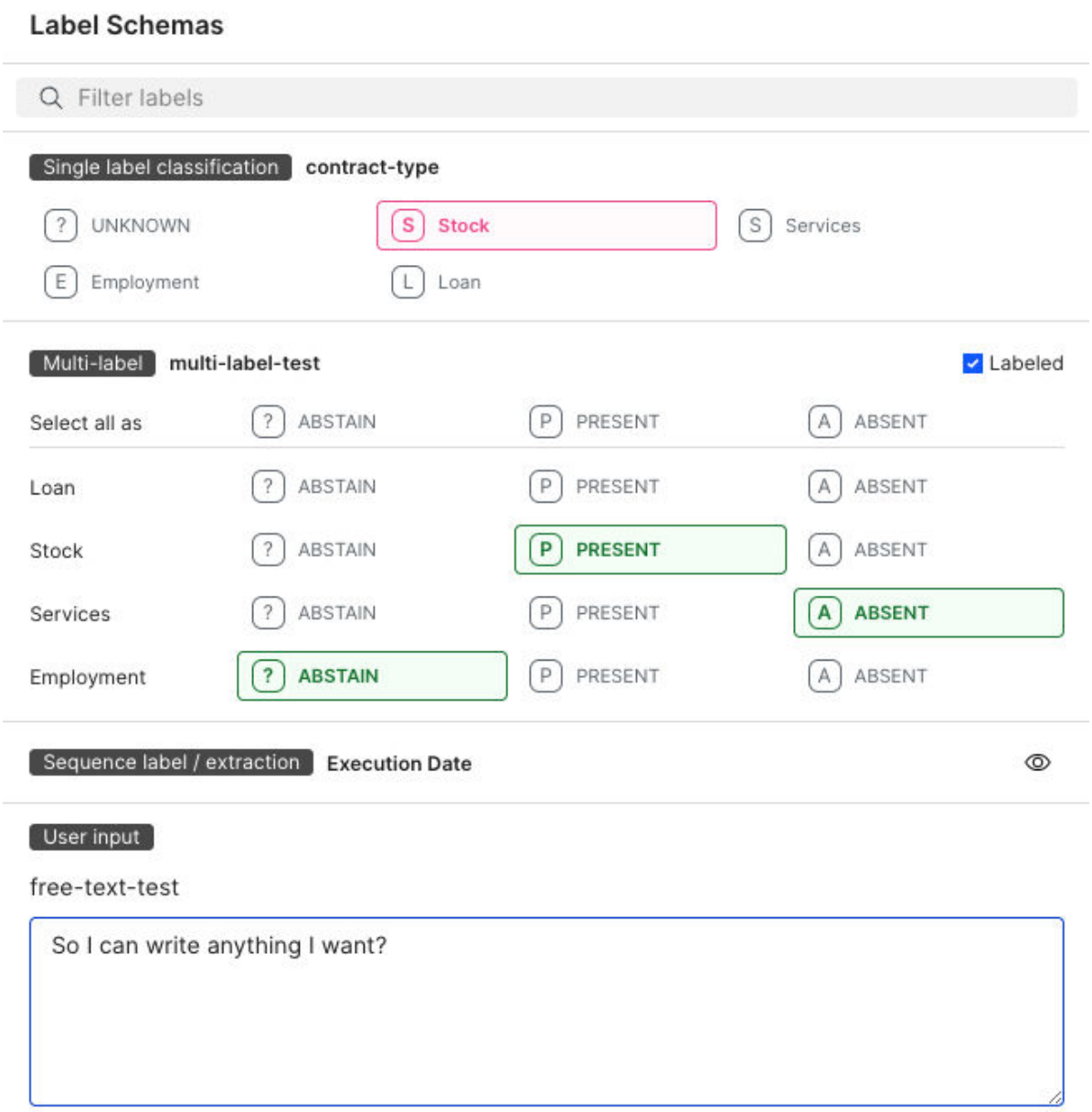
To upload a multi-schema annotation dataset

1. To create a new dataset, select **Datasets > Upload new dataset**.
2. Enter the required information for creating your dataset. For more, see Uploading a dataset.
3. Check **Enable multi-schema annotations** while creating your dataset.
   NOTE: You cannot enable multi-schema annotations later. If you do not opt in to enabling multi-schema annotations when creating your dataset, you must create a new dataset to enable it for that dataset.
4. Select **Verify data source(s)**.
5. Within **Datasets > Label Schemas**, select **+ Create new label schema**.
6. Enter a name, description, data type, task type, and additional fields for each task type.
   - Classification tasks: Select **Text** as the **Primary text field**. Select **Single label**, **Multi-label**, or **Text label**. **Text label** allows for free text in your labels instead of a defined label or labels for the other options.
   - Extraction tasks: Select **Sequence tagging**. Enter the label for sequence tagging with a defined primary text field.
7. Select **+ Add label schema**.
8. In **Batches**,select **+ Create new batch.**
9. Enter the batch name. Select your split and your label schema. Enter your batch numbers and batch sizes. If you want, you can assign users to annotate the batch. Select **Create batch**.

To annotate multiple schemas

1. In the **Batches** tab, select **Annotate** beside the batch you want to annotate.
2. Select the labels that apply to the datapoint.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

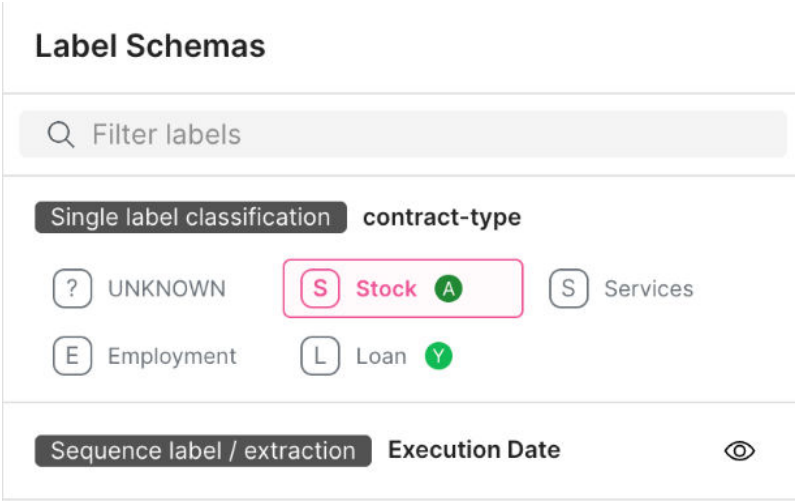


3. Select the previous and next arrows to move between data points.
4. Continue annotating each data point until you have completed your annotations. These annotations are applied to the data points across any batch in which they are used.

To review annotations

1. To see the annotations from other annotators, enable **Reviewer** mode with the toggle to see all of the annotations for a dataset.

Copyright © 2025 Snorkel AI, Inc. All rights reserved



You can see the annotations each annotator made.

2. Continue reviewing each data point until you have determined the correct annotations. These annotations are applied to the data points across any batch in which they are used.
3. After all of the annotations are complete for a dataset and ready to be used for the ground truth, select **Batches > Commit** for the dataset.

To configure your annotation display

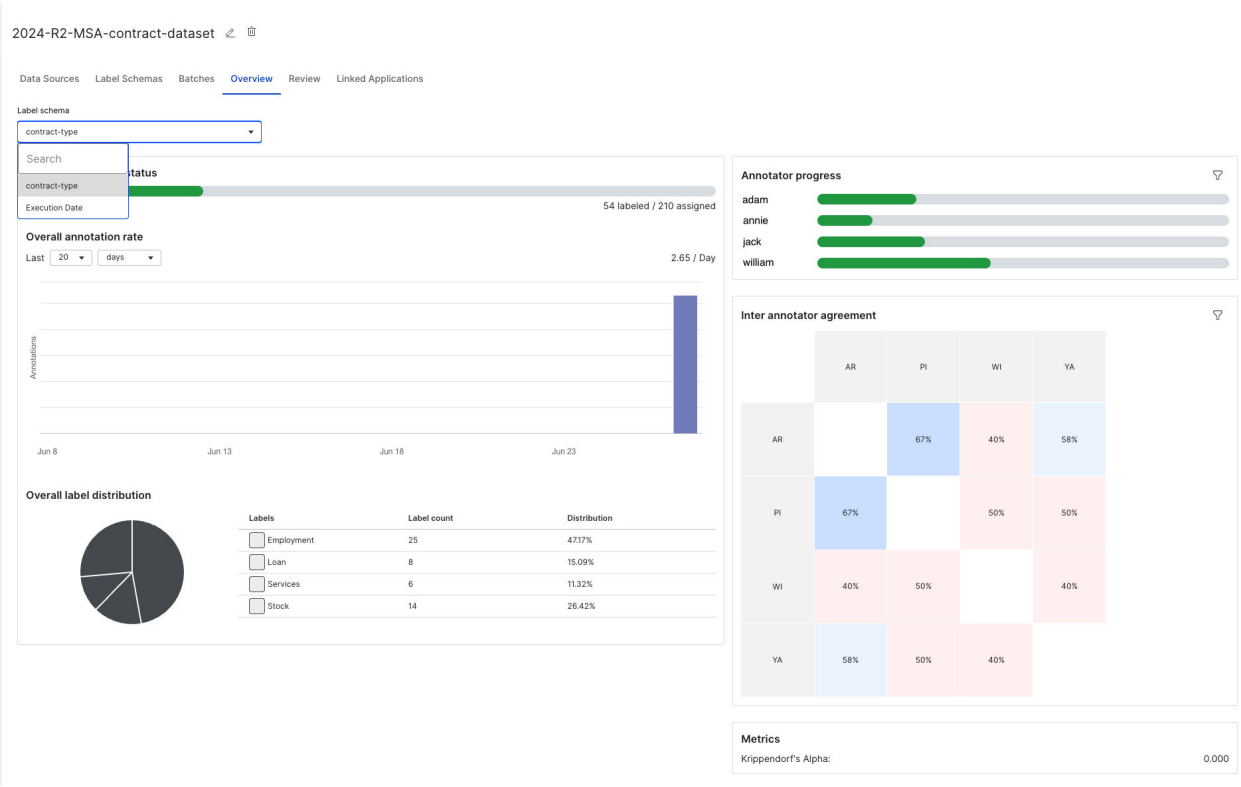
1. In the **Batches** tab, select **Annotate** beside the batch you want to annotate.
2. If you want to filter the data points to annotate, you can select your filtering options with fields and operators.
3. To change your display settings, select the gear icon. You can change the displayed columns and order, text direction, prioritize unlabeled documents, and set a default multi-label class.

To see annotation progress

In the **Overview**, you can select your **Label schema** from the dropdown menu to see the current status and how much each annotator has finished.

You can select filters for **Annotator progress** to see the progress for specific annotators.

Copyright © 2025 Snorkel AI, Inc. All rights reserved



In the **Review** tab, you can see can see that annotators, annotations, and batches.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

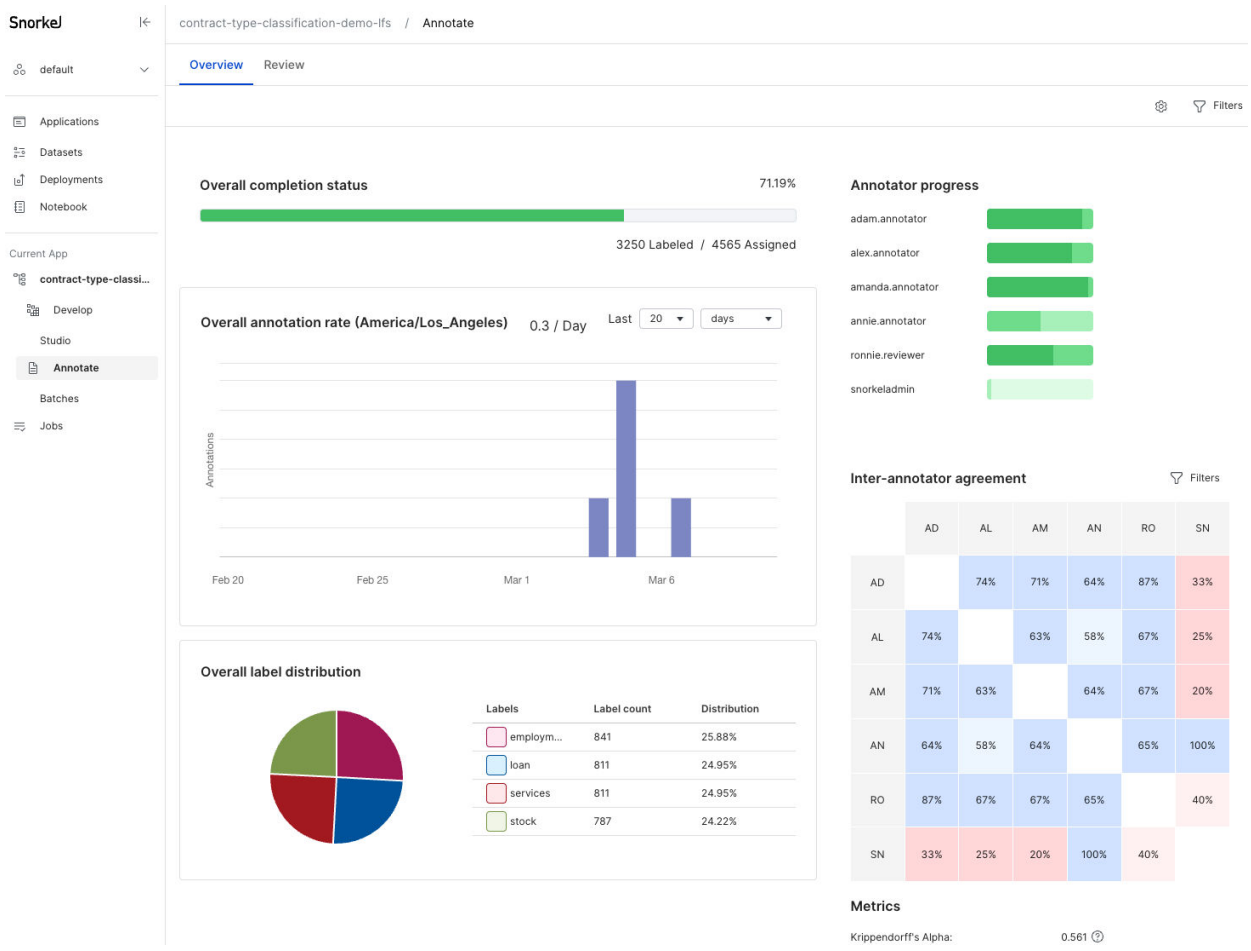
Overview page: View aggregate annotation metrics

This page walks through the summary metrics and settings that are found on the Annotation Studio **Overview** page. This page is typically used by those with Developer and Reviewer roles that want a broad view of the annotations that have been completed as well as insight into the level of agreement among annotators.

Overview page

The **Overview** page shows you various aggregate metrics on the number of annotations that have been completed, the distribution of labels, annotator agreement, and a view into recent annotator activity.

You can select the **Filter** button on the top-right corner of your screen to filter the charts by a particular annotator.



Overall completion status

The **Overall completion status** chart displays the percentage of assigned datapoints that have been annotated. The progress bar gives a good visual representation of how much annotation progress has been made.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

Overall annotation rate

The **Overall annotation rate** bar chart displays the aggregate number of annotations that have been completed at different granularities that you can specify (e.g., last 20 days, last 4 months).

Calendar boundaries are used when referencing days (starting at 12:00am), weeks (starting Monday), and months (starting on the 1st day of the month). For example, viewing annotations from the last 2 weeks on a Wednesday will show all annotations that have been completed since two Mondays ago, not 14 days ago. Similarly, the last X months view will show all annotations that occurred within each corresponding month.

All annotations are displayed relative to your timezone, which is displayed in the chart. To change your timezone, select your user name in the bottom left corner of your screen. Click **User settings**, then click **Advanced**. Note that the SDK retrieves annotation timestamps in the UTC timezone, which is how timestamps are stored in the database.

Overall label distribution

The **Overall label distribution** chart displays the distribution of labels that have been annotated. This chart enables you to quickly see if there are particular classes that are underrepresented in the annotations that have been completed so far. Users with developer access can take this information and create new batches to get a more even distribution of labels. See Create batches for more information on how to create new batches.

Annotator progress

The **Annotator progress** chart displays the percentage of assigned datapoints that have been annotated, split up by each annotator. Hover over the progress bars for each annotator to see exactly how many datapoints that they have annotated compared to the number of datapoints that have been assigned to them.

Inter-annotator agreement

The **Inter-annotator agreement** chart displays the agreement rate between annotators where they have labeled the same data points. For example, if annotators A and B have annotated 10 of the same data points, and on 8 of those they have the same labels, then the shared grid will show 80% in the corresponding cell.

You can filter the chart by label class and or by batch. To do so, click the **Filters** button at the top right corner of the chart, then select your desired filters.

The **Inter-annotator agreement** chart will only show up to 10 annotators at a time to ensure a consistent, readable experience. To view the chart for all annotators, click the chart to render a modal of the full chart that is not truncated.

Below the chart you can find the value for Krippendorff's Alpha. This metric measures disagreement among annotators, corrected for disagreement that is expected from random chance. Ideally you want to see a value closer to 1, which indicates better agreement among annotators.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

Updating the label schema

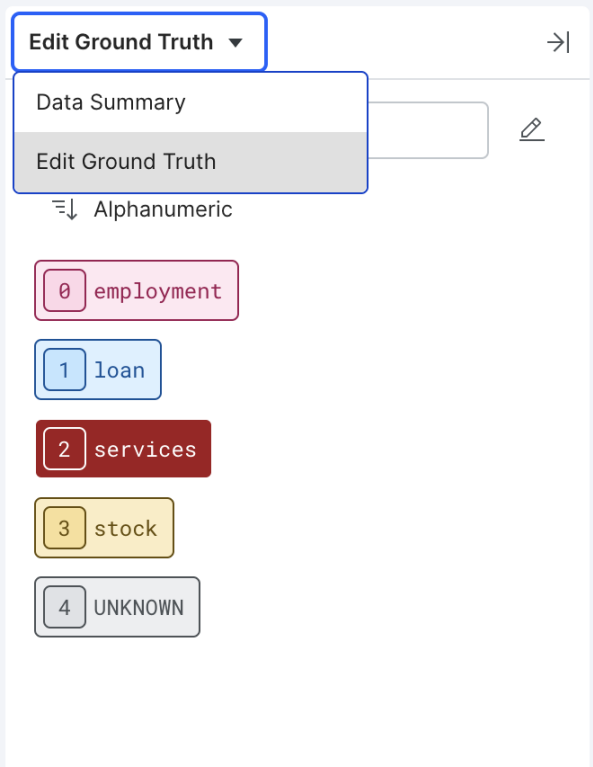
In this guide we will demonstrate how to update the label schema (list of classes) of your application. There are two ways to update your label schema: from the Label Studio or from the Application Studio.

> NOTE
>
> Updating the label schema creates a new model node with the updated label schema and also transfers assets such as ground truth labels and labeling functions (LFs). Transferring LFs to the new model node will fail if the original model node has multi-polar LFs (note that warm start LFs are multi-polar). Use the SDK functions `sf.delete_lf_package` and `sf.delete_lf` to delete any multi-polar LFs before updating the label schema.

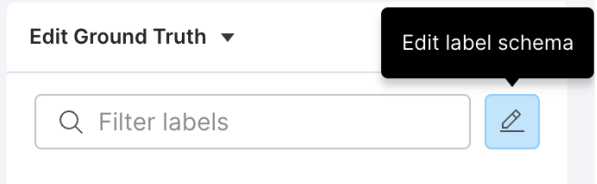
From the label studio

Accessing the editor

From Studio, click the collapse icon on the right side of the dataviewer (this pane may already be visible when you enter Studio). From there, select **Edit ground truth** from the dropdown.



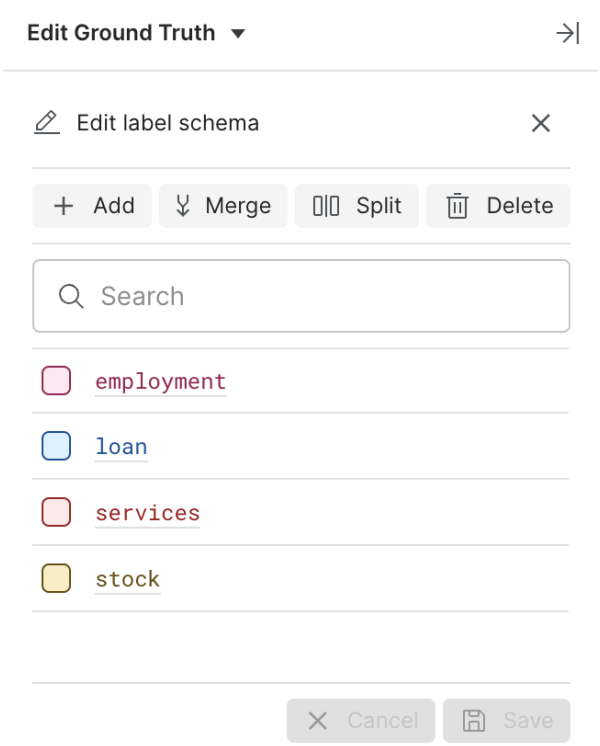
Next, click the pencil icon to access the label schema editor.



Copyright © 2025 Snorkel AI, Inc. All rights reserved

Using the editor

You can rename, add, merge, split, and delete your labels using the buttons near the top of the editor. Upon clicking **Save**, Snorkel will automatically update your existing ground truth and labeling functions to reflect your changes.



Merging labels for multi label text classification is somewhat complex to understand. If any of the to be merged labels is marked PRESENT, the target label is marked PRESENT, else if any of the to be merged labels is marked ABSENT, the target label is marked ABSENT, otherwise ABSTAIN. For example,

```
{
  "class_1" : "PRESENT",
  "class_2" : "ABSENT",
  "class_3" : "ABSENT",
  "class_4" : "PRESENT",
}
```

merging class_1, class_2, and class_3 to class_2 in the above example, will result in the label to be modified as

```
{
  "class_2" : "PRESENT",
  "class_4" : "PRESENT",
}
```

and merging class_2, and class_3 to class_2 in the prior example, will result in the label to be modified as

Copyright © 2025 Snorkel AI, Inc. All rights reserved

```
{
  "class_1" : "PRESENT",
  "class_2" : "ABSENT",
  "class_4" : "PRESENT",
}
```

Reverting your changes

Every change to the label schema is saved as a block. To access previous versions of your label schema you can visit the Application Studio, locate the block containing your preferred version, and then enter the Label Studio from there.

From the application studio

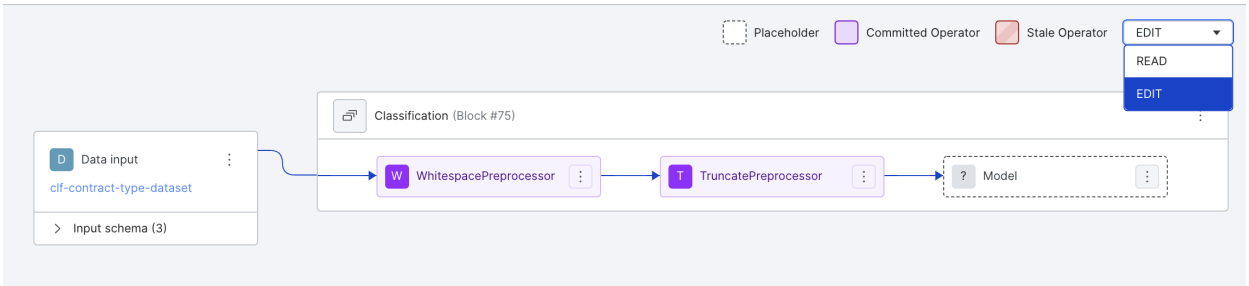
While it's not possible to edit the label schema of an existing block, you can accomplish the same goal by copying the block and updating the label schema during that process.

The three basic steps are:

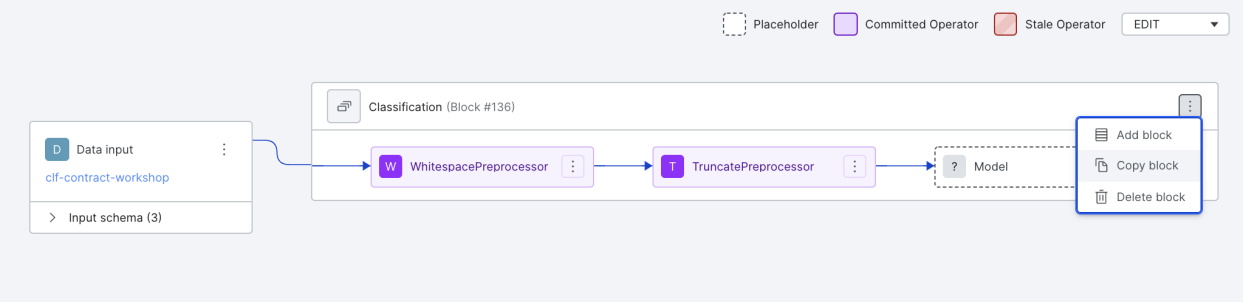
1. Make a copy of the existing block with the current label schema
2. Update the label schema for the new block
3. Delete the original block

Copying a block

You can update the label schema by making a copy of the relevant block. To copy a block, first go to Application Studio and make sure you select "EDIT" mode in the dropdown on the upper-right.

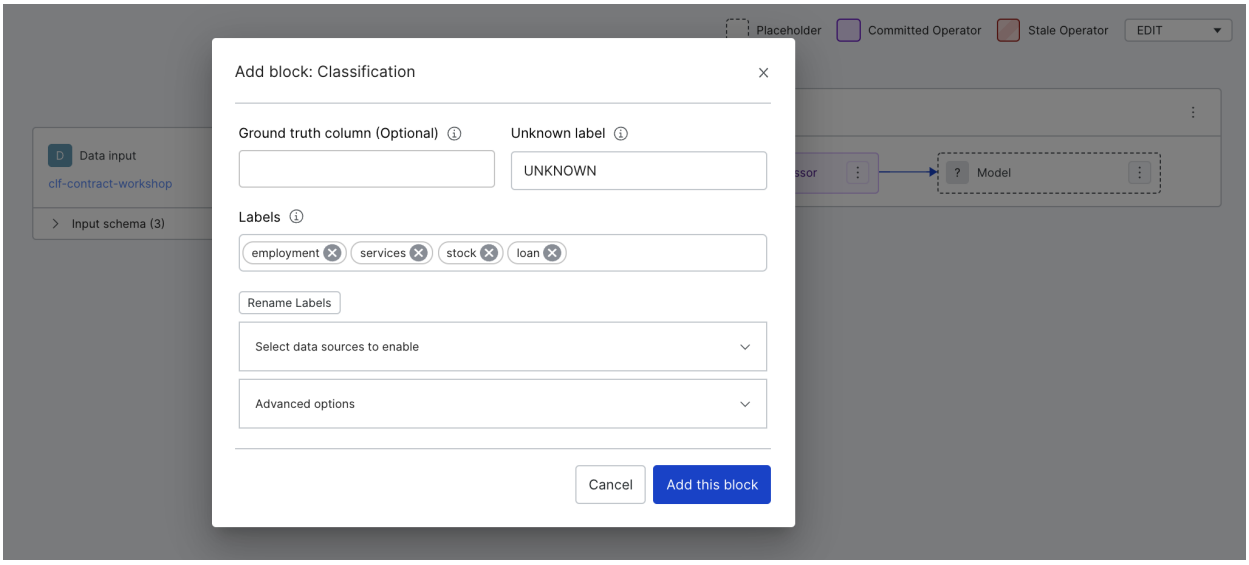


Then, navigate to the block whose label schema you would like to update and click on the menu icon at the upper-right of that block and select "Copy block".



You will now see an "Add block" modal.

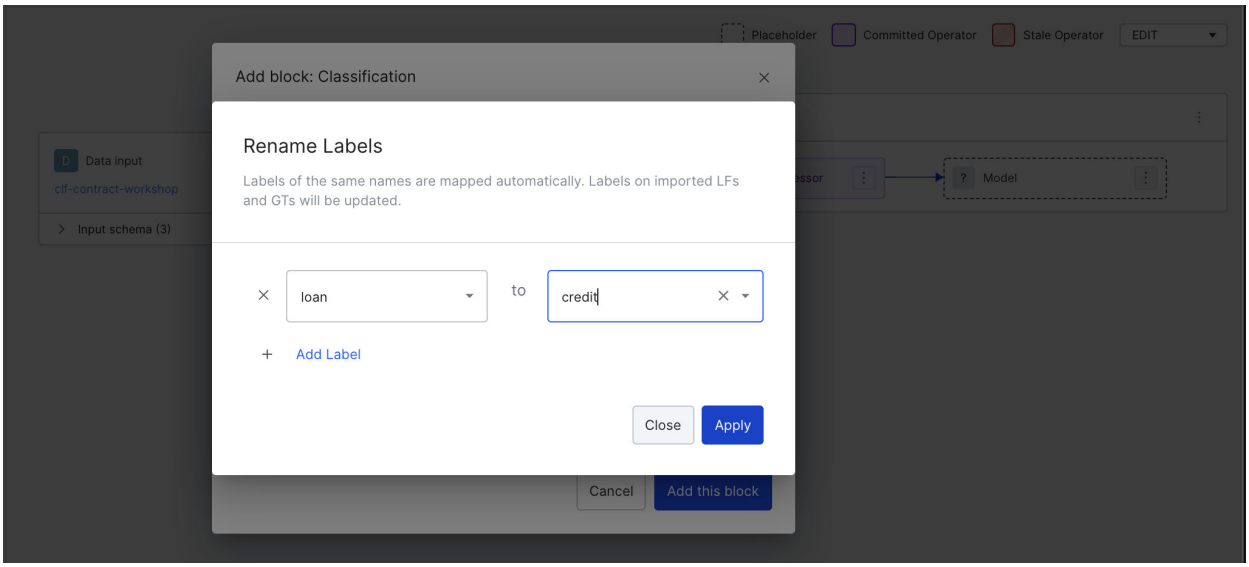
Copyright © 2025 Snorkel AI, Inc. All rights reserved



Adding, removing, and renaming labels

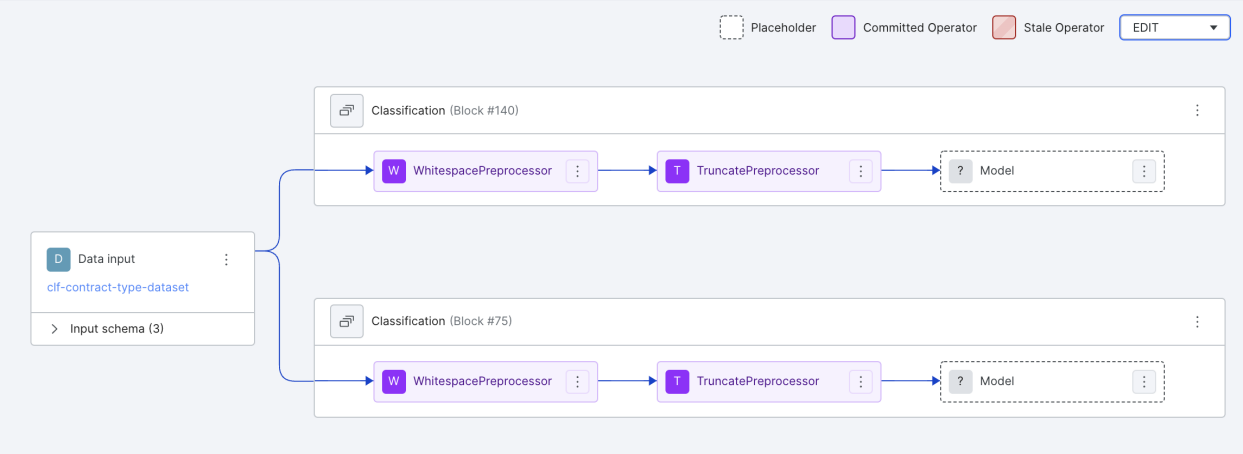
In the “Labels” box of the “Add block” modal, you’ll be able to see the current label schema. You can add labels by typing them in and remove labels by clicking on the “X” next to each label.

To rename labels, click on the “Rename Labels” button after updating the Labels box above. You’ll now be able to create mapping(s) from an old label to one in the new label schema by clicking on “Add Label”. Note that copied LFs and GT will be updated automatically with respect to this renaming.



Once you update the label schema, click on “Add this block” and a new block will be created with the new label schema.

Copyright © 2025 Snorkel AI, Inc. All rights reserved



Deleting the original block

You can now delete the old block from the Application Studio view. Use the menu icon in the upper-right of the block and select “Delete block”.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

Labeling function builders index

This page is an index of all labeling function (LF) builders that are available in Snorkel Flow. See Introduction to labeling functions for more information about LFs and how they are used in Snorkel Flow.

Category	Subcategories
Foundation model based	Warm start Prompt builder
Embeddings based	
Search based	Additional Timestamp IP address SQL query
Pattern based	Regex Full text regex Keyword Fuzzy keyword Keyword count Keyword location
Numerical	Field length Numeric Numeric comparator Two attribute numeric Bounding polygon Bounding rectangle
External resource	Dictionary External model Crowdworker
Span based	Span content Span context Span location Span regex
Rich document	Expression Bounding box Span regex proximity Span regex alignment Span regex row Span regex position

Copyright © 2025 Snorkel AI, Inc. All rights reserved

Category	Subcategories
	Span page Span font size
Sequence	Context Keyword Fuzzy keyword Substring expansion Spacy prop Named entity recognition (NER) Entity dictionary Word vector Letter case Regex
Other LFs	Multi-polar Custom Code Suggested

Copyright © 2025 Snorkel AI, Inc. All rights reserved

Foundation model suite

Overview

The foundation model suite is a collection of foundation model-based features incorporated into the end-to-end Snorkel workflow. These features distill, adapt, and fine-tune foundation models using the data-centric development workflow and train specialized, enterprise-ready production models.

What are foundation models?

Foundation models (FMs), also known as large language models (LLMs), are extremely large models trained on massive amounts of data, forming a general foundation for use in more specific tasks.

Snorkel Flow provides the bridge for these powerful generic models to be applied to real-world enterprise AI use cases.

For more about foundation models, see Foundation models: a guide.

Use cases

The FM suite focuses on predictive AI use cases. Predictive AI is critical to successfully derive value from AI in enterprise software, especially when it comes to automating mission-critical processes, such as underwriting, know your customer (KYC), and document intelligence.

Because most real-world use cases in enterprise software are complex and performance-critical, generalist foundation models struggle to drive value out of the box due to the lack of domain-specific knowledge. However, the data-centric FM Suite helps you use modern foundation models to accelerate the development of deployable specialist models tailored to the specific use case at hand, all within Snorkel Flow.

What is in the FM suite?

The FM Suite contains these main features:

- **Prompt Builder:** Explore and label data through natural language prompts using FM knowledge and translate it into labels that are ready for your weakly supervised learning use cases. See Prompt builder for more information.
- **Warm Start:** Auto-label training data with the power of foundation models plus state-of-the-art zero/few-shot learning techniques during onboarding. This approach helps you get to a powerful baseline first pass with minimal human effort. See Warm start for more information.

Infrastructure requirements

Snorkel-hosted deployments

Feature	Release v0.94
Prompt Builder	Does not require a GPU because it can run on external infrastructure. Requires a valid account for the infrastructure (Hugging Face, OpenAI, Vertex AI, Azure ML, Bedrock Claude, Azure OpenAI, or Amazon SageMaker).

Copyright © 2025 Snorkel AI, Inc. All rights reserved

Feature	Release v0.94
	If external connections are not possible, contact the Snorkel team to explore alternatives.
Warm Start	After upgrading to 0.94, models are downloaded and readily available for use in Snorkel Flow. Infrastructure: - 1 GPU - 16 GB Memory If GPUs are unavailable, contact Snorkel to assess alternatives and trade-offs. Warm start is also available using external models (Hugging Face, OpenAI, Vertex AI, Azure ML, Azure OpenAI, or Amazon SageMaker).
Fine-tuning	Hugging Face models are accessible in the Model Zoo in Snorkel Flow. Infrastructure: - Recommended: 1 GPU - Possible to run on CPU with significantly longer run times.

Customer-hosted deployments (On-premise & private cloud)

Feature	Release v0.94
Prompt Builder	FM inference is widely supported for connections outside of the Snorkel platform, including Hugging Face, OpenAI, Vertex AI, Azure ML, Azure OpenAI, and Amazon SageMaker. Requires internet access. If an internet connection is unavailable, contact Snorkel for alternatives.
Warm Start	Requires internet access to download models for the first use. If an internet connection is unavailable, contact Snorkel support. Infrastructure: - 1 GPU - 16 GB Memory If GPUs are unavailable, contact Snorkel to assess alternatives and trade-offs.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

Feature	Release v0.94
Fine-tuning	Hugging Face models are accessible in Snorkel. Infrastructure: - Recommended: 1 GPU - Possible to run on CPU with significantly longer run times.

Next steps

You can begin to leverage Snorkel Flow's foundation model suite by configuring external models across a wide array of foundation model integrations.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

About external models

Many foundation model suite workflows can use third-party external services to run inference over your data. For more information about how to use these models, see Using external models.

Available external providers and models

Predictive use case support

- Prompt builder
- Warm start (SDK): `zsl_prompt_remote`

Provider	Model Type(s)
Amazon SageMaker	Text2Text
Azure Machine Learning	Text2Text
Azure OpenAI	Text2Text
Bedrock Claude	Text2Text
Hugging Face	Text2Text, QA, DocVQA
OpenAI	Text2Text
Vertex AI	Text2Text
Amazon Bedrock	Text2Text
Custom inference (OpenAI API specification)	Text2Text

Generative use case support

- Inference (generative)
- Fine-tuning (generative)

Provider	Model Type(s)	Management interface
Amazon SageMaker	Text2Text	SDK only

Supported Model Types

We support prompting for specific model types across external model providers to support a variety of use cases in Snorkel Flow. This list includes only models that the Snorkel team has tested. Endpoints with compatible specifications to these providers could be swapped in as well.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

Model Type	Supported Models	Supported Applications
Text2Text	- Hugging Face Text2Text and Text Generation models - OpenAI chat models - Azure OpenAI chat models - Azure Machine Learning Text2Text and Text Generation models - Amazon SageMaker Text2Text and Text Generation models - Bedrock Claude chat models - Vertex AI Palm and Gemini models - Custom inference service Text2Text models	- Text Classification - Sequence Tagging - PDF Extraction
QA	- Hugging Face Question Answering models	- Sequence Tagging
DocVQA	- Hugging Face Document Question Answering models	- PDF Extraction

What's next?

You can now use an external model by linking external services and configuring an external model endpoint.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

Using external models

Many foundation model suite workflows can use third-party external services to run inference over your data. To configure these third-party models, you need to create an account with the third-party service. Once an account has been established, you can configure an instance-wide connection for your selected service.

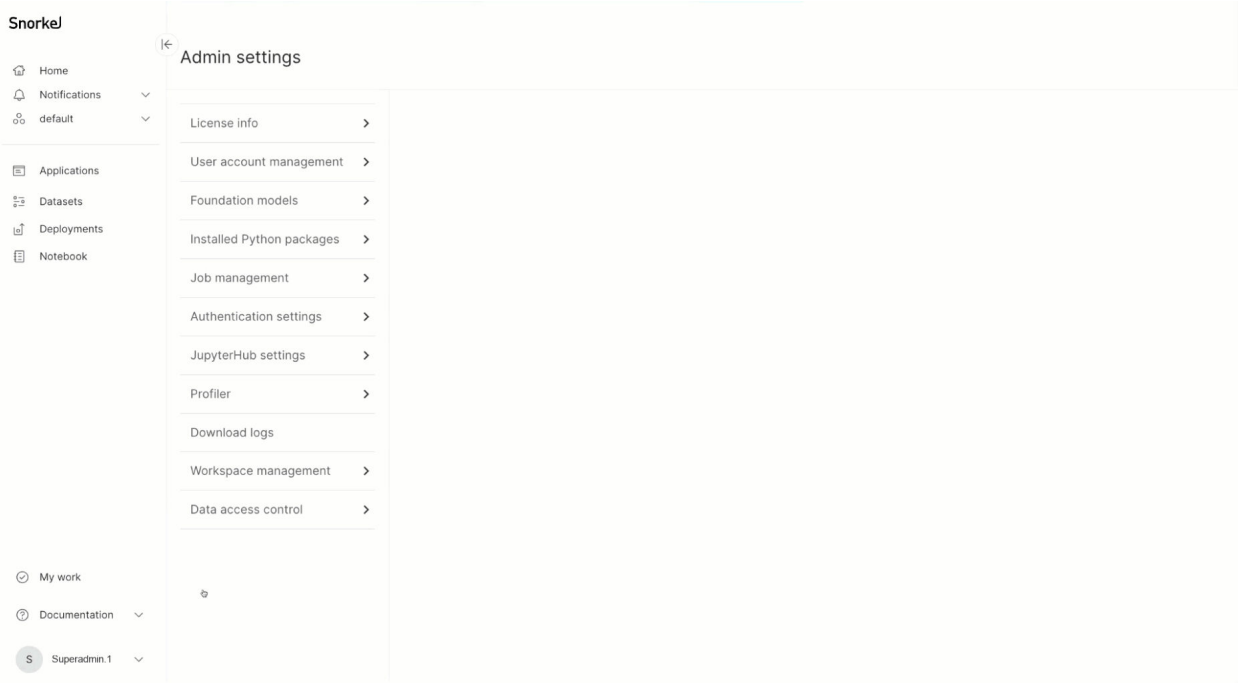
To link external services

You can link external integrations to your Snorkel Flow instance. Snorkel Flow secrets service securely manages your API keys and secrets.

Using Snorkel Flow Admin Settings

1. Superadmin roles can add new integrations through Snorkel Flow **Admin Settings**. Select your profile, and then navigate to **Admin Settings** > **Foundation Models.**
2. Select **Connect** for the integration you want to add.
3. Enter the required details from your external integration provider and select **Save.**
4. Configure available models for the provider you added.
5. (Optional) To make changes, select the **Edit** or **Delete** button for your integration.

> ⚠ WARNING
> Deleting an integration also deletes all models associated with that integration.



Copyright © 2025 Snorkel AI, Inc. All rights reserved

Using the SDK

1. To list the integrations that are currently connected, use this SDK command:

```
sf.list_integrations()
```

2. To add a connection to a new integration, use of the `set_secret` command for your supported services:

- Hugging Face

```
sf.set_secret("huggingface::inference::api_token", "**YOUR API KEY**")
```

For more, see Hugging Face token settings.

- OpenAI

```
sf.set_secret("openai_api_key", "**YOUR API KEY**")
```

For more, see OpenAI API keys.

- Azure OpenAI

```
sf.set_secret("azure_openai_api_key", "**YOUR API KEY**")
```

- Azure Machine Learning

```
sf.set_secret("azure::ml::api_key", "**YOUR API KEY**")
```

- Vertex AI

```
sf.set_secret("vertexai_lm_location", "**YOUR PROJECT LOCATION**")
sf.set_secret("vertexai_lm_project_id", "**YOUR PROJECT ID**")
sf.set_secret("vertexai_lm_credentials_json", "**YOUR CREDENTIALS
JSON**")
```

- Amazon SageMaker

```
sf.set_secret("aws::finetuning::region", "**YOUR REGION**")
sf.set_secret("aws::finetuning::access_key_id", "**YOUR ACCESS KEY ID**")
sf.set_secret("aws::finetuning::secret_access_key", "**YOUR SECRET ACCESS
KEY**")
sf.set_secret("aws::finetuning::sagemaker_execution_role", "**YOUR
EXECUTION ROLE**")
```

- Amazon Bedrock

Copyright © 2025 Snorkel AI, Inc. All rights reserved

```
sf.set_secret("aws::bedrock::region", "**YOUR REGION**")
sf.set_secret("aws::bedrock::access_key_id", "**YOUR ACCESS KEY ID**")
sf.set_secret("aws::bedrock::secret_access_key", "**YOUR SECRET ACCESS
KEY**")
sf.set_secret("aws::bedrock::bedrock_execution_role", "**YOUR EXECUTION
ROLE**")
```

- Custom inference service (conforms to the OpenAI API specification)

```
sf.set_secret("custom_inference_api_key", "**YOUR API KEY**")
```

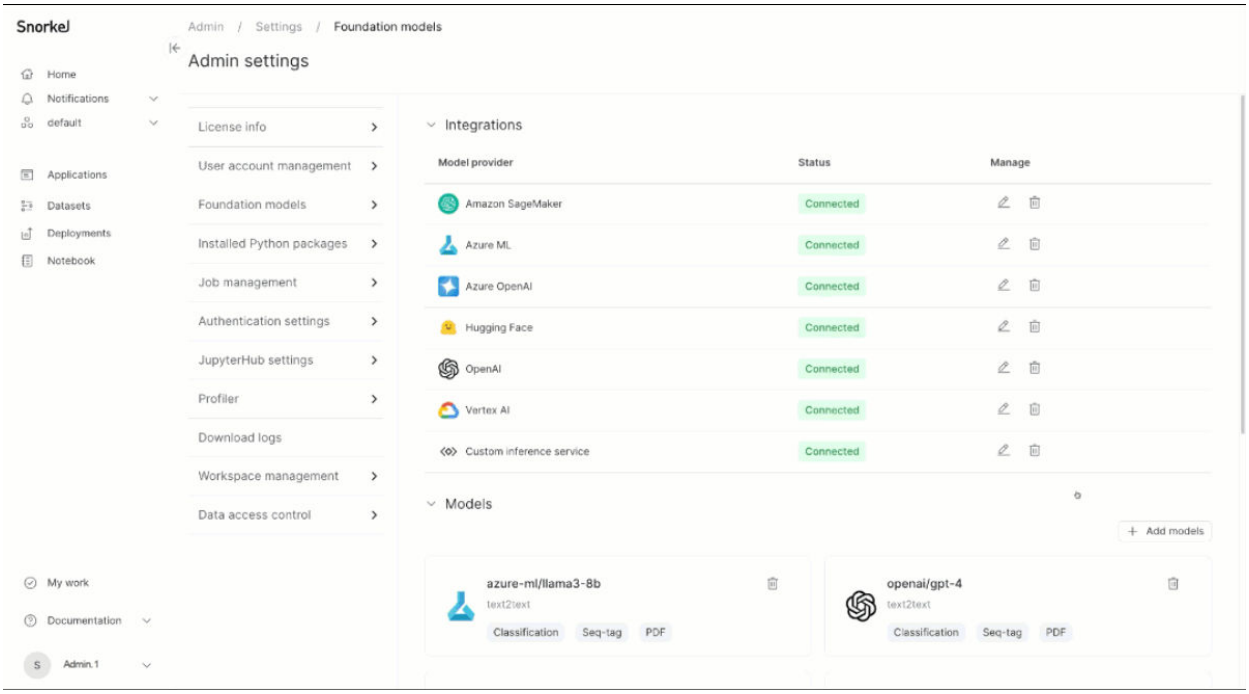
To configure available models

Once you link a service, you can add models that are served within those services.

Using Snorkel Flow Admin Settings

1. In the **Admin Settings** > **Foundation Models** page, select the **Add models** button.
2. From the dropdown, choose the model provider you set up and enter the details for **Model type**, **Model name**, and the **Endpoint URL**.

The model you added shows up under the **Models** section and is available for use in Snorkel Flow.



Using SDK

To view the models that are currently connected, use this SDK command:

```
sf.get_external_model_endpoints()
```

You can view detailed information about configured models using the detail parameter:

Copyright © 2025 Snorkel AI, Inc. All rights reserved

```
sf.get_external_model_endpoints(detail=True)
```

You can inspect configuration information for a particular model using the model_name parameter:

```
sf.get_external_model_endpoints(model_name="openai/gpt-4o", detail=True)
```

To add models to Snorkel Flow, include the following information for each model:

- **Model name:** The name of the provider followed by the name of the model. For example, `openai/gpt-4`.
- **Endpoint URL:** The endpoint to perform inference requests.
- **Model provider:** The provider serving the model. For example, Hugging Face, OpenAI, Vertex AI, or a custom inference service.
- **FM type:** The associated task type for the model. For more information on which model type to select when configuring a model, see Supported Model Types.

  This example shows how to add the `openai/gpt-4o` model to Snorkel Flow:

  ```
  from snorkelflow.client_v3.tdm.models import ExternalLLMProvider, FMType

  sf.set_external_model_endpoint(
      model_name="openai/gpt-4o",  # Compatible chat completions model
      endpoint="https://api.openai.com/v1/chat/completions",  # Chat
  completions endpoint
      model_provider=ExternalLLMProvider.OPENAI,  # Model provider
      fm_type=FMType.TEXT2TEXT,  # The task type of the model
  )
  ```

To delete a configured model, use this SDK command:

```
sf.delete_external_model_endpoint("openai/gpt-4o")
```

Hugging Face 🤗 inference endpoints

To add a Hugging Face model, launch your chosen Hugging Face model on their servers using Hugging Face endpoints. You will be given an endpoint URL for that model.

This example shows how to set up a Hugging Face **Text2Text** or **TextGeneration** model for Classification, Sequence Tagging, and PDF Extraction applications:

Copyright © 2025 Snorkel AI, Inc. All rights reserved

```
sf.set_external_model_endpoint(
    model_name="**MODEL NAME**",  # Set this to the model you have created the endpoint for
    endpoint="**ENDPOINT URL**",  # Set this to the URL for the model found in the Inference Endpoints Dashboard
    model_provider=ExternalLLMProvider.HUGGINGFACE, # Model provider
    fm_type=FMType.TEXT2TEXT, # The task type of the model; in this case, text2text
)
```

This example shows how to set up a Hugging Face **Question Answering** (QA) model for Sequence Tagging applications:

```
sf.set_external_model_endpoint(
    model_name="**MODEL NAME**",  # Set this to the model you have created the endpoint for
    endpoint="**ENDPOINT URL**",  # Set this to the URL for the model found in the Inference Endpoints Dashboard
    model_provider=ExternalLLMProvider.HUGGINGFACE, # Model provider
    fm_type=FMType.QA, # The task type of the model; in this case, question answering
)
```

This example shows how to set up a Hugging Face **Document Question Answering** (DocVQA) model for PDF Extraction applications:

```
sf.set_external_model_endpoint(
    model_name="**MODEL NAME**",  # Set this to the model you have created the endpoint for
    endpoint="**ENDPOINT URL**",  # Set this to the URL for the model found in the Inference Endpoints Dashboard
    model_provider=ExternalLLMProvider.HUGGINGFACE, # Model provider
    fm_type=FMType.DOCVQA, # The task type of the model; in this case, DocVQA
)
```

OpenAI API

To add an Open AI model, set up your API token and ensure that you are using the appropriate API endpoint. Specify a chat completions endpoint for chat models or a completions endpoint for language models. For more information about what API endpoint a model belongs, see OpenAI's Text generation models documentation.

This example shows how to configure an OpenAI model with the chat completions API endpoint:

Copyright © 2025 Snorkel AI, Inc. All rights reserved

```
sf.set_external_model_endpoint(
    model_name="openai/o1-mini", # Compatible chat completions model
    endpoint="https://api.openai.com/v1/chat/completions", # Chat completions
endpoint
    model_provider=ExternalLLMProvider.OPENAI, # Model provider
    fm_type=FMType.TEXT2TEXT, # The task type of the model
)
```

This example shows how to configure an OpenAI model with the legacy completions API endpoint:

```
sf.set_external_model_endpoint(
    model_name="openai/gpt-3.5-turbo-instruct", # Compatible completions model
    endpoint="https://api.openai.com/v1/completions", # Completions endpoint
    model_provider=ExternalLLMProvider.OPENAI, # Model provider
    fm_type=FMType.TEXT2TEXT, # The task type of the model
)
```

Azure OpenAI API

To add an Azure OpenAI model, set up your API token and ensure that you are using the appropriate API endpoint.

This example shows how to configure a supported Azure OpenAI chat model.

```
sf.set_external_model_endpoint(
    model_name="azure_openai/your-deployment-name", # Compatible chat
completions model
    endpoint="https://your-instance-name.openai.azure.com/chat/completions", #
Chat completions endpoint
    model_provider=ExternalLLMProvider.AZURE_OPENAI, # Model provider
    fm_type=FMType.TEXT2TEXT, # The task type of the model
)
```

Azure Machine Learning API

To add an Azure Machine Learning AI model, set up your API token and ensure that you are using the appropriate API endpoint.

This example shows how to configure a supported Azure Machine Learning model.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

```
from snorkelflow.models.prompts.prompts_services.azure import
AzureDataInferenceInterfaceTypes

sf.set_external_model_endpoint(
    model_name="your-model-name", # Name of your deployed Azure ML model
    endpoint="https://<deployment-name>.westus2.inference.ml.azure.com/score",
# Chat completions endpoint
    model_provider=ExternalLLMProvider.AZURE_ML, # Model provider
    fm_type=FMType.TEXT2TEXT, # The task type of the model
    azure_task_type=AzureDataInferenceInterfaceTypes.Llama.value,
)
```

Bedrock Claude API

To add a Bedrock Claude model endpoint:

```
sf.set_external_model_endpoint(
    model_name="bedrock/anthropic.claude-3-5-sonnet-20241022-v2:0",  # Model
name
    endpoint="bedrock-runtime.us-west-2.amazonaws.com",  # Endpoint
    model_provider=ExternalLLMProvider.Bedrock,  # Model provider
    fm_type=FMType.TEXT2TEXT, # The task type of the model
)
```

Vertex AI language models API

To add a Vertex AI model, set up your Vertex AI location, project ID, and credentials JSON.

This example shows how to configure a Vertex AI model:

```
sf.set_external_model_endpoint(
    model_name="vertexai_lm/gemini-1.5-pro-002",  # Model name
    endpoint="https://cloud.google.com/vertex-ai",  # Endpoint
    model_provider=ExternalLLMProvider.VERTEXAI_LM,  # Model provider
    fm_type=FMType.TEXT2TEXT, # The task type of the model
)
```

Amazon SageMaker API

To add an Amazon Sagemaker model endpoint for predictive use cases:

Copyright © 2025 Snorkel AI, Inc. All rights reserved

```
sf.set_external_model_endpoint(
    model_name="sagemaker/jumpstart-dft-meta-textgeneration-llama-3-8b-instruct", # Model name
    endpoint="https://runtime.sagemaker.<region-name>.amazonaws.com/endpoints/jumpstart-dft-meta-textgeneration-llama-3-8b-instruct/invocations", # Endpoint
    model_provider=ExternalLLMProvider.SAGEMAKER, # Model provider
    fm_type=FMType.TEXT2TEXT, # The task type of the model
)
```

To add an Amazon SageMaker model for generative use cases, see the LLM fine-tuning and alignment tutorial.

Custom Inference Service API

The custom inference service enables users to configure custom endpoints and alternative FM providers that adhere to the OpenAI API specification. Some examples of FM providers that are supported by the custom inference service includes Together AI and Groq.

After setting up your custom inference API key, add a model with custom inference service that conforms to the OpenAI API specification. Similar to the OpenAI setup, specify a chat completions endpoint for chat models or a completions endpoint for language models. This inference service is also extensible to other foundation model providers that conform to OpenAI's API specification, such as Together AI.

This example shows how to set up a supported model with the chat completions API endpoint:

```
sf.set_external_model_endpoint(
    model_name="meta-llama/Llama-3.2-3B-Instruct-Turbo", # Chat model
    endpoint="https://api.together.xyz/v1/chat/completions", # Inference service chat endpoint
    model_provider=ExternalLLMProvider.CUSTOM_INFERENCE_SERVICE, # Model provider
    fm_type=FMType.TEXT2TEXT, # The task type of the model
)
```

This example shows how to set up a supported model with the completions API endpoint:

```
sf.set_external_model_endpoint(
    model_name="meta-llama/Meta-Llama-3-70B", # Language model
    endpoint="https://api.together.xyz/v1/completions", # Inference service completions endpoint
    model_provider=ExternalLLMProvider.CUSTOM_INFERENCE_SERVICE, # Model provider
    fm_type=FMType.TEXT2TEXT, # The task type of the model
)
```

Copyright © 2025 Snorkel AI, Inc. All rights reserved

Specifying model hyper-parameters

We support the setting of arbitrary model hyper-parameters when you configure a model in Snorkel Flow. Some example parameters that this include are:

- `temperature`
- `top_p`
- `max_input_length`
- `max_output_length`

When setting a model hyper-parameter, please ensure it appears exactly as documented by the model's provider. For example, when configuring the temperature hyper-parameter, some models leverage the keyword `temp`, others use `t`, and some require the full word `temperature`.

Once you have confirmed the hyper-parameter names and values you'd like to set, provide them as keyword arguments to `sf.set_external_model_endpoint`. For example, to set `temperature` and `max_tokens` on `openai/gpt-4o`, you'd run:

```
sf.set_external_model_endpoint(
    model_name="openai/gpt-4o", # Compatible chat completions model
    endpoint="https://api.openai.com/v1/chat/completions", # Chat completions
endpoint
    model_provider=ExternalLLMProvider.OPENAI, # Model provider
    fm_type=FMType.TEXT2TEXT, # The task type of the model
    temperature=0.5,
    max_tokens=500,
)
```

Configuring rate limits

Snorkel Flow supports more efficient interactions for select providers by maximizing the utilization of your organization's quota while previewing and creating prompt labeling functions (LFs). Currently, the OpenAI, Azure OpenAI, and custom inference service integrations support custom rate limits.

To enable this feature, use the SDK function `sf.set_external_model_endpoint()` to configure the `requests_per_sec` and `tokens_per_sec` parameters. Snorkel Flow consumes as much as the provided quota when computing the preview of prompt LFs across your instance.

When setting these values, get the correct values from your organization's usage tier and provide the values in per-second level. If you provide incorrect values, Snorkel Flow might hit rate limits for OpenAI or underutilize the provided quota.

To determine OpenAI usage limits, visit OpenAI organization limits page. Find the **token limits** and **request limits** per minute for your desired model and divide it by 60 to compute the per-second value. For example, when the model `gpt-4o-mini` shows 10,000 RPM for request and other limits and 30,000,000 TPM for token limits, provide `requests_per_sec=166` and `tokens_per_sec=500000` as extra keyword arguments in the `set_external_model_endpoint()` function.

Here is an example for configuring `gpt-4o-mini` with rate limits:

Copyright © 2025 Snorkel AI, Inc. All rights reserved

```
# delete an existing model endpoint, only if already registered
sf.delete_external_model_endpoint("openai/gpt-4o-mini")
sf.set_external_model_endpoint(
    model_name="openai/gpt-4o-mini", # Compatible chat completions model
    endpoint="https://api.openai.com/v1/chat/completions", # Chat completions
endpoint
    model_provider=ExternalLLMProvider.OPENAI, # Model provider
    fm_type=FMType.TEXT2TEXT, # The task type of the model
    requests_per_sec=166, # request limit per second
    tokens_per_sec=500000, # token limit per second
)
```

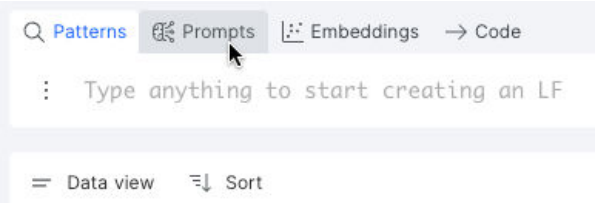
Copyright © 2025 Snorkel AI, Inc. All rights reserved

Prompt Builder

Prompt Builder offers the ability to precisely inject foundation model expertise into your application through an interactive workflow.

To open the Prompt Builder, click the **Prompts** icon and text within the LF composer toolbar as shown below. If you do not see the icon, ensure that the application type is supported. Application types that are currently supported include:

- Single-label text classification
- Sequence tagging
- PDF candidate-based extraction



As a user, you should feel empowered to try out any prompt engineering technique and observe how well it performs on your data. Once you are happy with the prompt, click **Create prompt** to capture this signal inside a Snorkel labeling function (LF). This enables you to go further by building with additional signal to surpass the original prompt scores.

Configuring Available Models

Prompt Builder workflows rely on third-party external services to run inference over your data. In order to configure these, you will need to use an account from your organization. Once an account has been established, you can configure the instance-wide settings for the Prompt Builder according to the instructions in Using external models. If your Snorkel Flow instance has GPUs available, you can also contact a Snorkel representative to run foundation models locally.

Prompting Workflows

Across all workflows, you can select your desired configured foundation model. Different types of foundation models are available for each workflow. For more information on which foundation models are supported for your application, refer to About external models. For information on configuring supported models, refer to Using external models.

In all workflows, start by drafting a prompt to send to the model. Click **Preview prompt** to analyze the live-streaming results as the model processes a sample of the current data split.

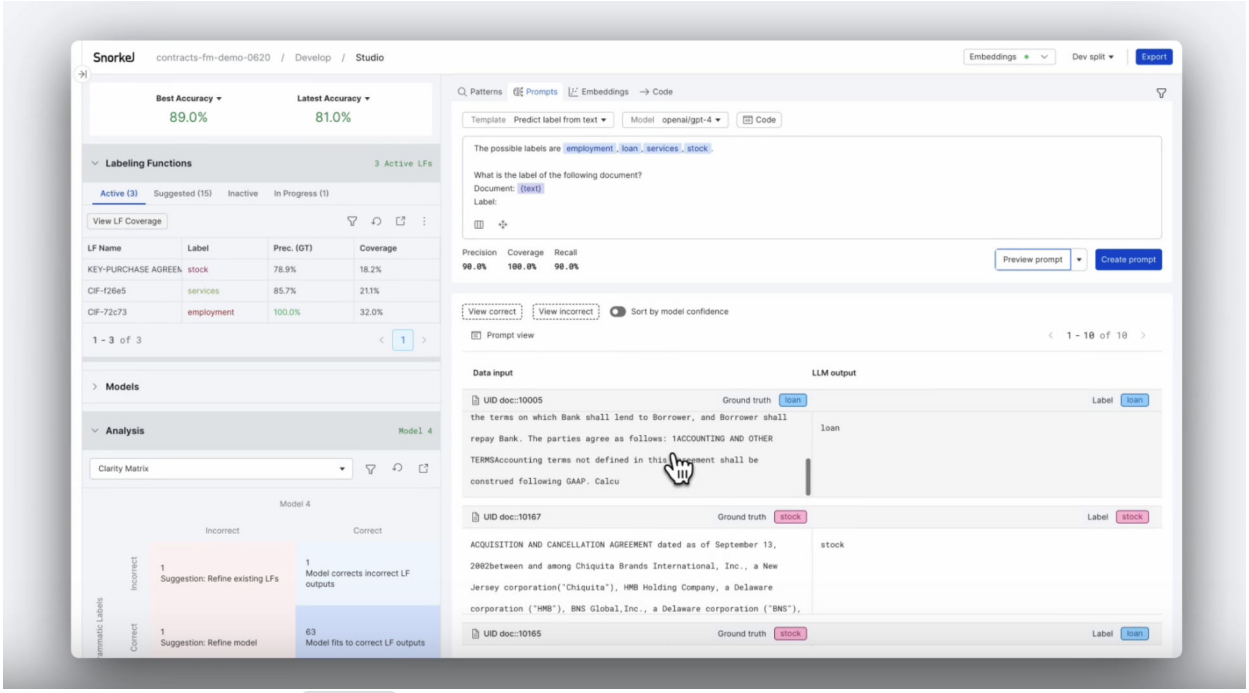
Copyright © 2025 Snorkel AI, Inc. All rights reserved

At any point, you can cancel the running job, and progress will be saved. Clicking **Resume** will begin sending data to the model again for processing. Once you are satisfied with the LF performance on your current data split, click **Create prompt** to run the prompt over all of your active data sources.

Text Classification Prompt Workflow

The classification workflow provides a flexible way to prompt engineer your data where any instruction can be passed to the foundation model. There are some template examples that are pre-populated for you, but you should feel empowered to explore a variety of prompt engineering techniques.

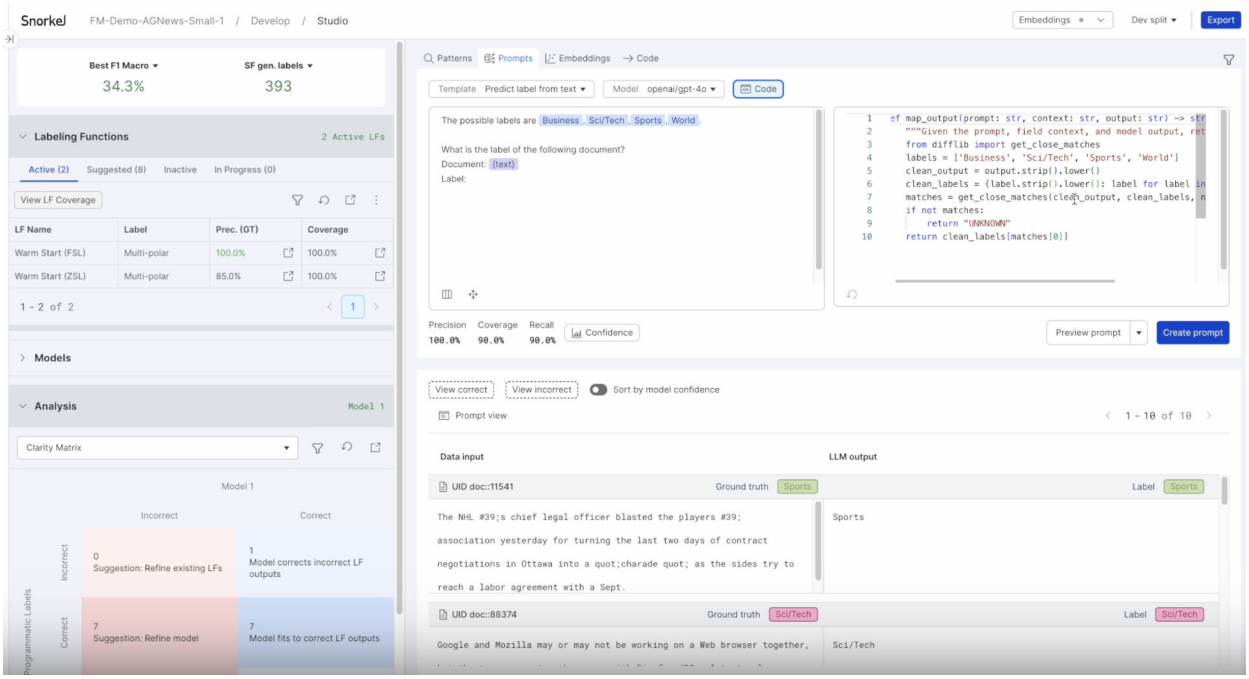
In the workflow, you can fully specify both the prompt that you want to send to the model, including how the context of each example appears, along with the code to map the foundation model output to your classes.



In the screenshot above, `{text}` refers to a specified text field within your dataset. In each request to the model, any field name appearing between braces will be replaced with the value for that field in that example. This allows you to specify multiple fields in a single prompt to the model if necessary. In this example, we are asking `openai/gpt-4` to label a document as one of four classes (employment, loan, services, or stock) based on the document text.

In addition, the text classification prompt workflow allows you to customize the mapping from model output to label. To do this, click the **Code** icon. There are several templates with pre-defined prompts and code that you can edit to output label mappings. Upon selecting any of these templates, you can modify each area as you see fit. This gives you significant low-level control over the input and output to the model.

Copyright © 2025 Snorkel AI, Inc. All rights reserved



Code Requirements:

- All logic must be contained within the body of a single function.
- The single defined function must include `output` as its only parameter.
- The defined function should return the exact label name when voting for that label. Any other return value (including `None` and `UNKNOWN`) will be treated as abstentions for the LF.

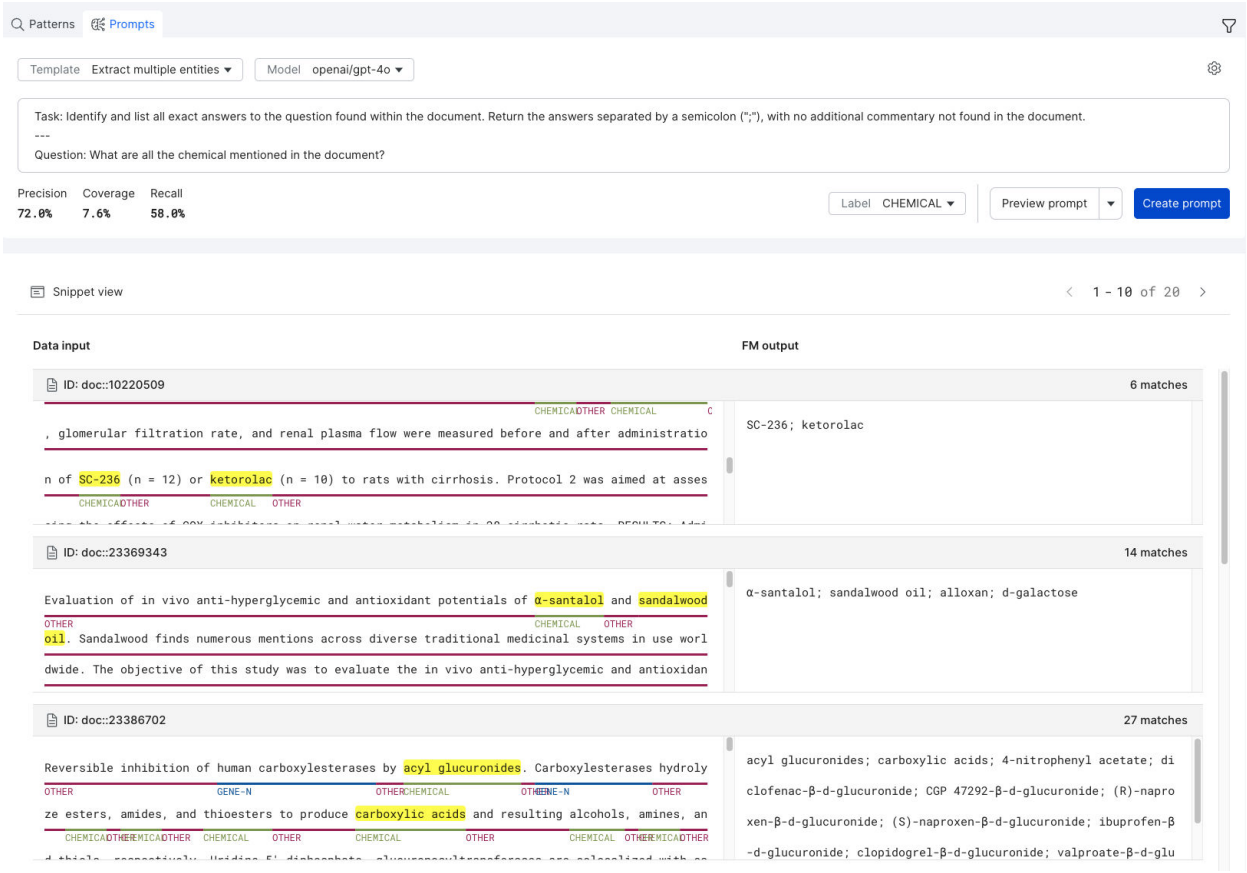
Once you make changes to the code, the refresh icon in the bottom left corner of the editor will become enabled. Clicking it will update your metrics and the **Code Mapped Label** column to reflect the latest code in the editor.

Sequence Tagging Prompt Workflow

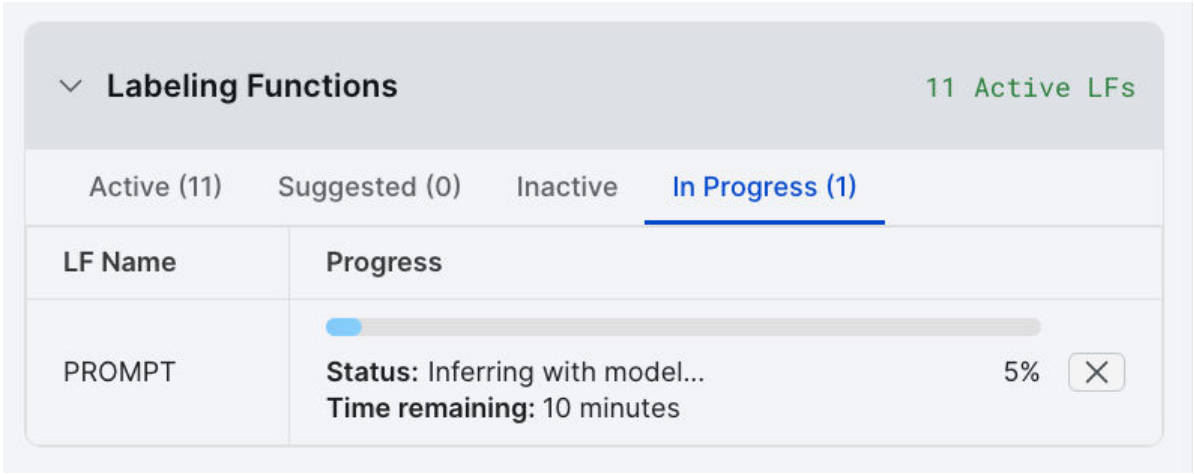
The sequence tagging workflow provides an intuitive interface for extracting information from your documents. Similar to the classification workflow, you should feel empowered to explore a variety of prompt engineering techniques to see how well they perform on your data. This workflow involves asking a question about your document. The model then extracts the spans from the text that it thinks best answers your question. The sequence tagging prompt workflow supports Text2Text and Question-Answering (QA) models.

For Text2Text models, you can extract multiple entities for each document. To do this, prompt your foundation model to separate each entity to extract using the semi-colon (";") delimiter. All relevant matches for each identified entity will be extracted.

Copyright © 2025 Snorkel AI, Inc. All rights reserved



After previewing your result, select **Create prompt** to create a prompt LF. You can view the progress of the LF in the **In Progress** tab of the Labeling Functions table. Once prompt inference has completed, the LF will be available from the **Active** tab of the Labeling Functions table.

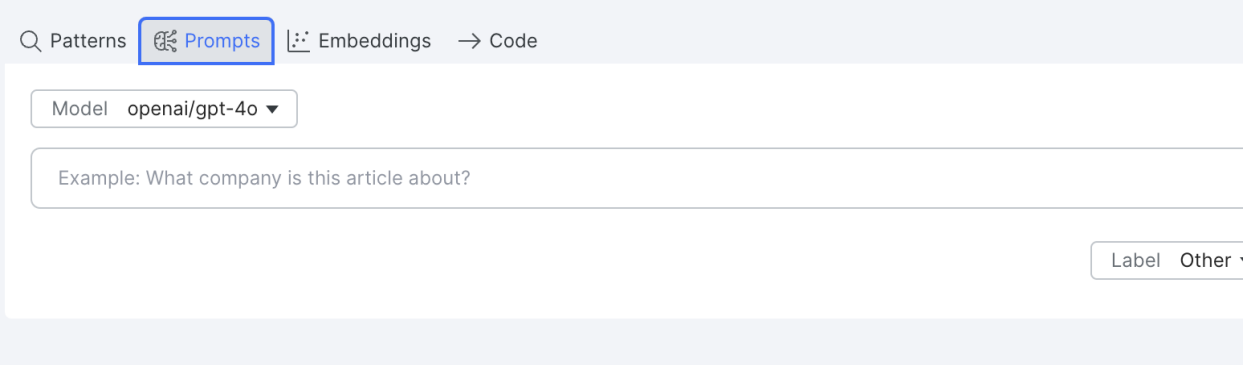


PDF Extraction Prompt Workflow

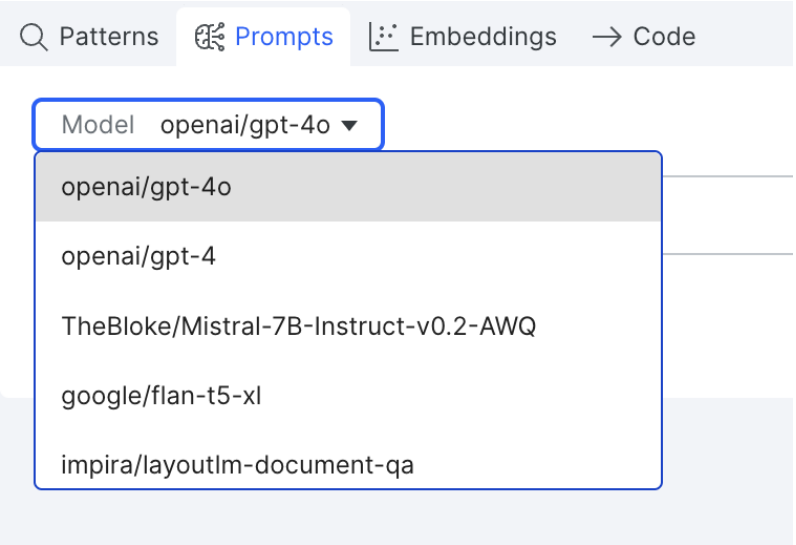
The PDF extraction prompt workflow allows you to extract entities from PDF documents in a PDF candidate-based extraction application. The PDF extraction prompt workflow supports Text2Text (e.g. `openai/gpt-4o`) and Document Visual Question Answering (DocVQA) (e.g. `impira/layoutlm-document-qa`) models.

This prompting workflow can be accessed through the **Prompts** icon in **Develop (Studio)** for PDF extraction applications.

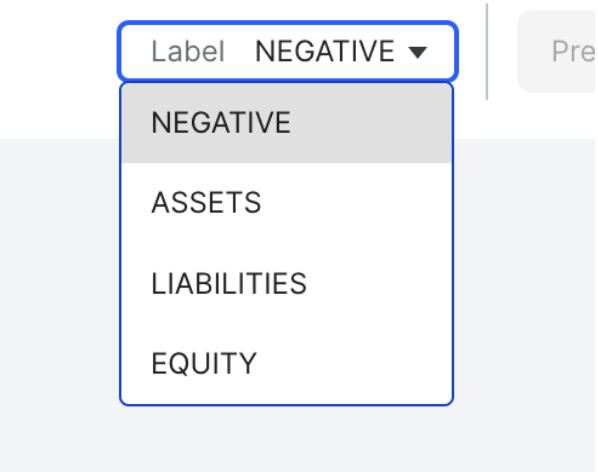
Copyright © 2025 Snorkel AI, Inc. All rights reserved



Select the desired model to run the prompt against. All configured Text2Text and DocVQA models are available for selection.

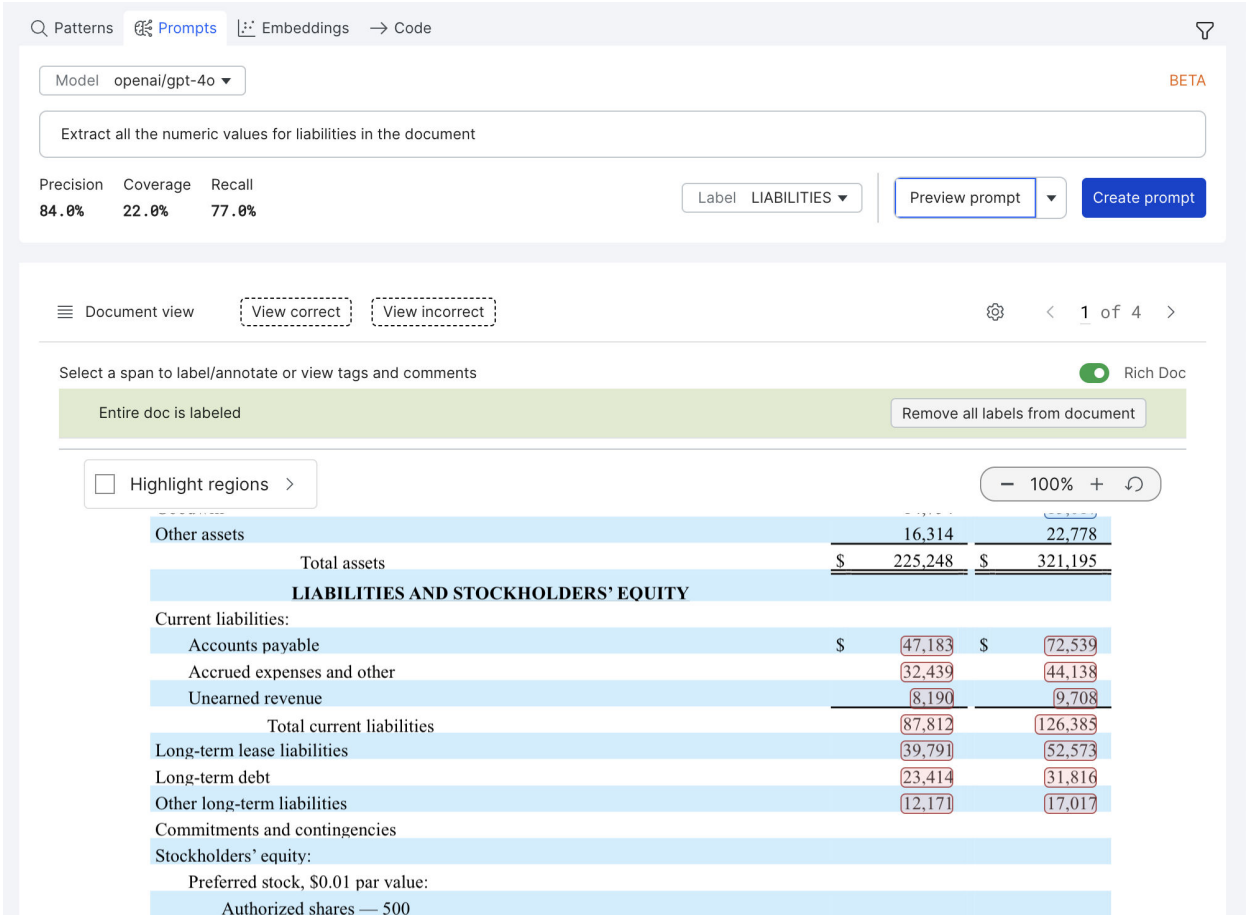


Select the label from the dropdown menu. The label should correspond to the entity to be extracted from the PDF documents.



Write a prompt to extract entities from the PDF documents; the prompt will be run against PDF documents. Preview the results by clicking on **Preview prompt**. Review the extracted spans and the metrics for the previewed sample.

Copyright © 2025 Snorkel AI, Inc. All rights reserved



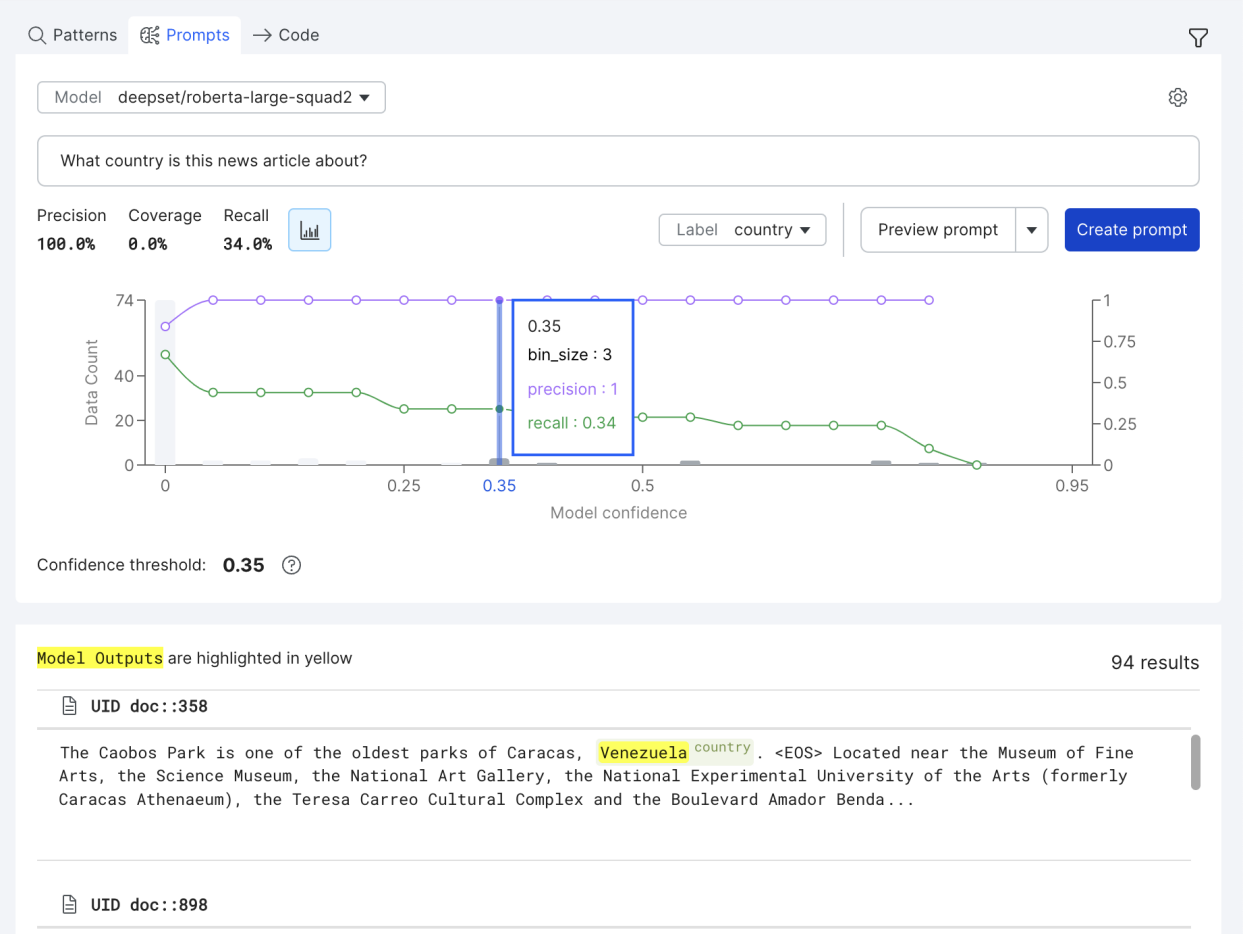
Select **Create prompt** to create a prompt LF to run across all splits in the dataset.

Advanced Features

Thresholding Model Confidence

Model confidence is a number between 0 and 1 for each datapoint/row that indicates the foundation model's certainty that its prediction is correct. Extractive models provide a confidence score with each prediction. In some cases, thresholding this confidence provides a useful tradeoff between precision and recall. To visualize this, click the **Graph** icon to open a threshold metrics view.

Copyright © 2025 Snorkel AI, Inc. All rights reserved



In this view, the purple line shows the precision of the LF at different thresholds, while the green line shows the recall of the LF at different thresholds. Additionally, histogram bars represent the number of examples that fall into each threshold bucket.

You can control the threshold used by selecting different operating points on the graph.

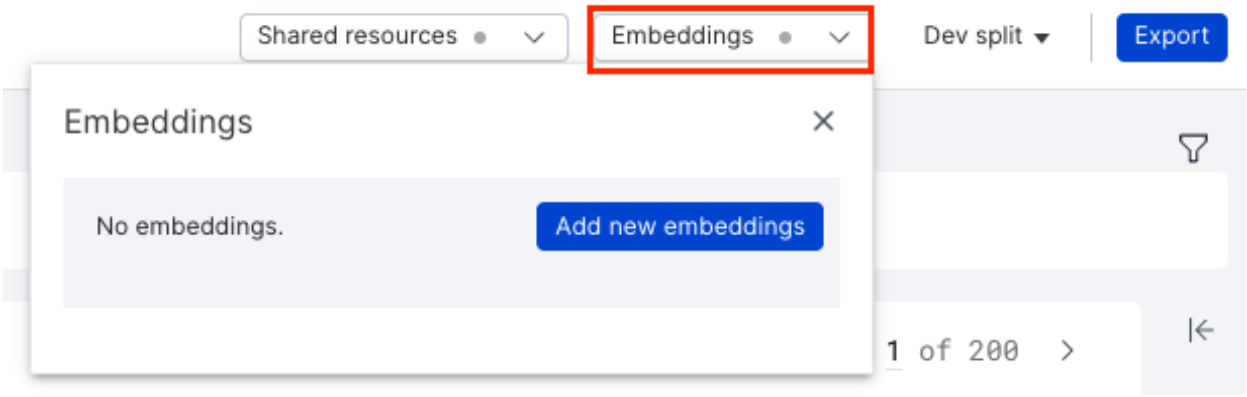
Prompting with Document Chunking (RAG)

Document chunking is a retrieval-augmented generation (RAG) technique available within the sequence tagging prompt workflow. This aims to improve prompt results, particularly on longer documents. Leveraging RAG unlocks prompting on long documents that may fall outside of the maximum context length of the foundation model. RAG can also help circumvent the “Lost in the Middle" phenomenon by including only the most relevant document information in the input context.

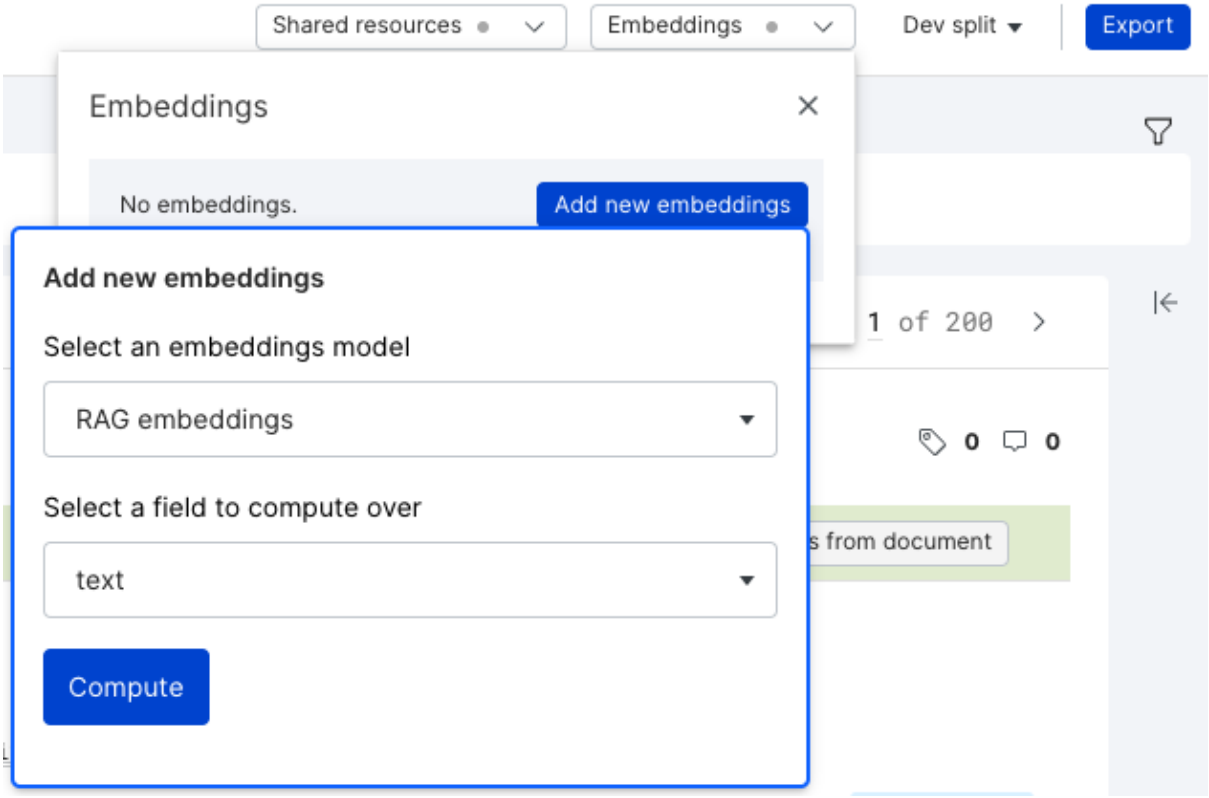
Follow these steps to enable document chunking with RAG on your application:

1. Compute embeddings for RAG. Click **Embeddings** in the application data pane (top-right corner of your screen). If no RAG embeddings exist, select **Add new embeddings**.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

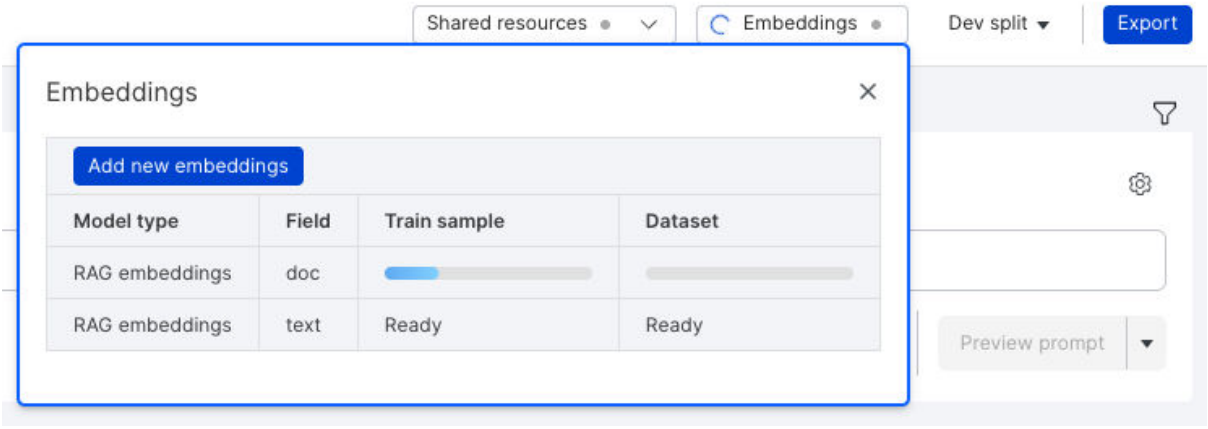


2. Once the **Add new embeddings** modal pops up, select **RAG embeddings** as the embeddings model to use, and then select the primary text field of the application.

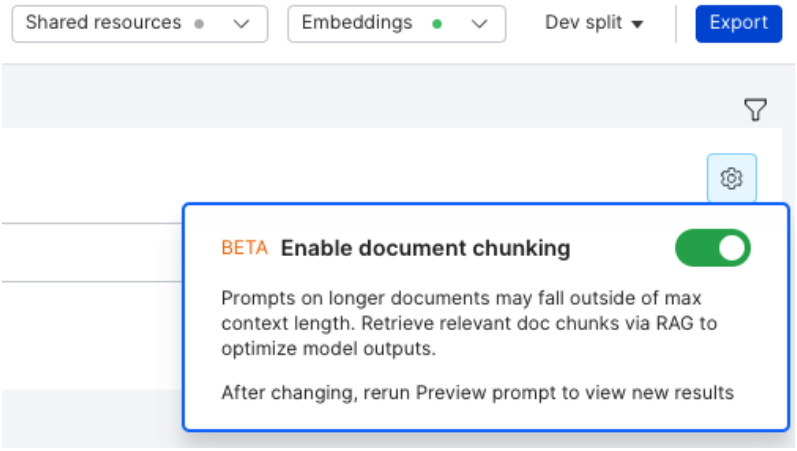


3. Kick off embeddings by selecting **Compute**, which will trigger a success notification.
4. View the progress of the embeddings by hovering over the **Train sample** and **Dataset** columns in the embedding home.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

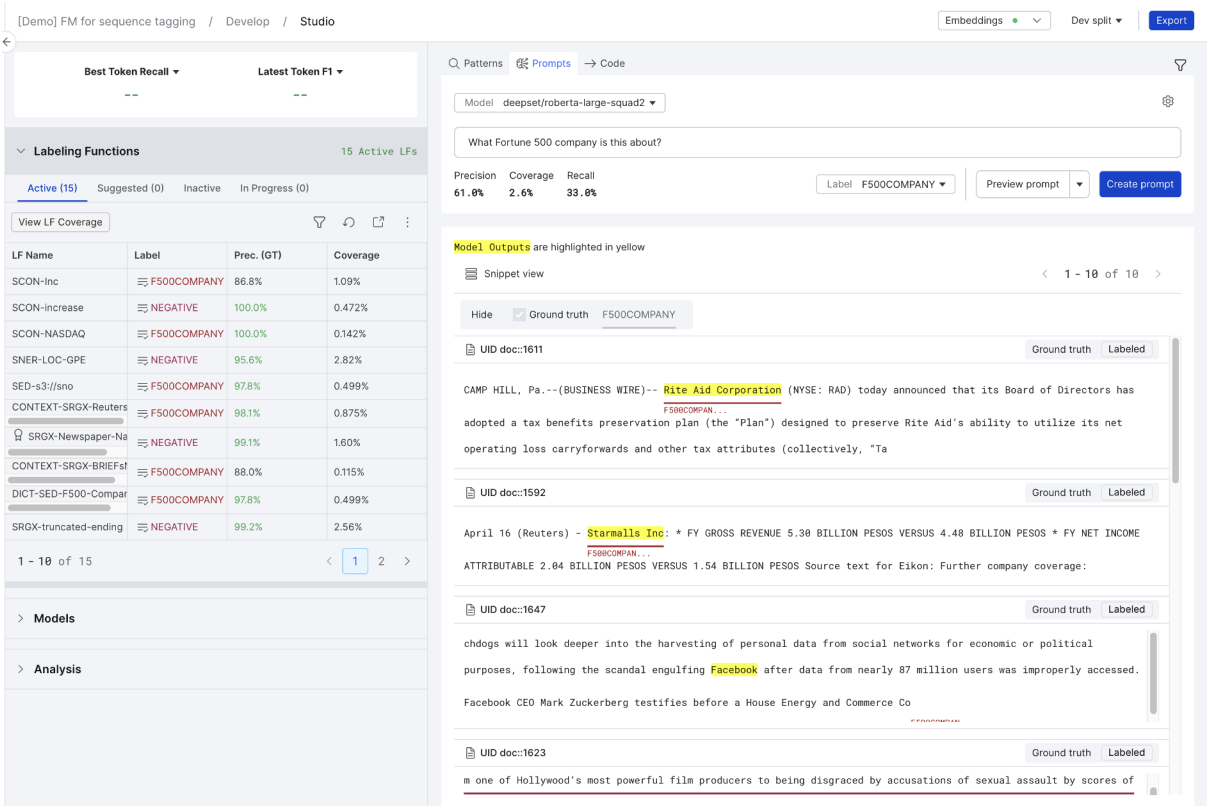


5. Once RAG embeddings have been generated, enable document chunking by clicking the settings icon next to the prompt toolbar. Toggle document chunking on and then click **Preview prompt**.



6. After clicking **Preview prompt** with document chunking enabled, you can observe the previewed results in real-time. Finally, click **Create prompt** to create a prompt labeling function for use in the broader Snorkel data-centric workflow.

Copyright © 2025 Snorkel AI, Inc. All rights reserved



Copyright © 2025 Snorkel AI, Inc. All rights reserved

Prompt builder (SDK)

Prompt Builder offers the ability to precisely inject Foundation Model expertise into your application via an interactive workflow.

Typically, you will use the UI to see how a prompt is performing and create new labeling functions (LFs) from the prompt. Learn more about the prompt builder here. However, prompt LFs can be previewed and created through the notebook as well.

> NOTE
>
> Prompt builder supports the following application types: text classification (single-label and multi-label), sequence tagging, and PDF candidate-based extraction.

Previewing a prompt LF

Import the Snorkel Flow SDK and retrieve the model node for your application:

```
import snorkelflow.client_v3 as sf
ctx = sf.SnorkelFlowContext.from_kwargs()  # Import context to our current
application

APP_NAME = "APPLICATION_NAME"  # Name of your application
NODE_UID = sf.get_model_node(APP_NAME)  # Model node used in the application
```

The following example previews a prompt LF over 5 examples in the dev split using a Text2Text model on a multi-label classification application.

You can preview a prompt LF with the following SDK command:

```
sf.preview_prompt_lf(
    NODE_UID,
    model_name="google/flan-t5-xxl",
    split="dev",
    num_examples=5,
    model_type="text2text_multi_label",
    prompt_text="""The possible labels are anova, bayesian, mathematical-
statistics, R.
    What is the most likely label for the following document: {Body_q}
{Body_a}\n\nLabel: """,
)
```

This will show the result of running that prompt LF using 5 examples in the dev split:

Copyright © 2025 Snorkel AI, Inc. All rights reserved

x_uid	output	confidence	label	ground_truth
doc::106217	R	0.942892	{'_default': 'ABSENT', 'r': 'PRESENT'}	{'r': 'PRESENT', '_default': 'ABSENT'}
doc::111005	mathematical-statistics	0.945986	{'_default': 'ABSENT', 'mathematical-statistic...	{'time-series': 'PRESENT', 'r': 'PRESENT', '_d...
doc::113804	mathematical-statistics	0.943889	{'_default': 'ABSENT', 'mathematical-statistic...	{'time-series': 'PRESENT', 'regression': 'PRES...
doc::108647	R	0.860703	{'_default': 'ABSENT', 'r': 'PRESENT'}	{'r': 'PRESENT', '_default': 'ABSENT'}
doc::111001	mathematical-statistics	0.933181	{'_default': 'ABSENT', 'mathematical-statistic...	{'machine-learning': 'PRESENT', 'regression': ...

- output: The output of the LLM used, in this case, `google/flan-t5-xxl`.
- label: The in-Snorkel label that is assigned by the code mapper based on the LLM output. A default code mapper is provided for each supported application type (basic keyword mapping), but a custom one can be passed using the `output_code` parameter.
- ground truth: The actual label.

In the next example, we will preview a prompt LF over 10 examples in the dev split using a Text2Text model on a sequence tagging application. Note that the `primary_text_field` and `label` parameters are required for sequence tagging applications.

```
sf.preview_prompt_lf(
    NODE_UID,
    model_name="openai/gpt-4o",
    split="dev",
    num_examples=10,
    model_type="text2text_qa",
    primary_text_field="Description" # Required for sequence tagging
applications
    label="Color", # Required for sequence tagging applications
    prompt_text="""List all occurrences of the exact answers to the question.
Return answers separated by the delimiter ";". Make sure each answer is
separated exactly by this delimiter without any additional spaces or
characters.
    Question: What is the color here?""",
)
```

This shows the result of running that prompt LF using 10 examples in the dev split:

Copyright © 2025 Snorkel AI, Inc. All rights reserved

x_uid	context_uid	output	confidence	label	ground_truth	snippet_type	text
doc::99002	doc::99002	Blue/Orange	0.999355	[[25, 36, Color]]	[[50, 58, Material]]	contextual_snippet	Miami Throwback Football Blue/Orange Area Rug....
doc::99005	doc::99005	Blue	0.998568	[[15, 19, Color]]	[[120, 125, Pattern]]	contextual_snippet	Wichita Kansas Blue Area Rug.This area rug is ...
doc::99004	doc::99004	Red/Brown/Black	0.997253	[[432, 447, Color]]	[[4, 12, Material]]	contextual_snippet	The chenille rug creates a luxurious soft area...
doc::99001	doc::99001	Blue/White	0.995916	[[27, 37, Color]]	[[97, 102, Pattern]]	contextual_snippet	Seattle Throwback Football Blue/White Area Rug...
doc::98998	doc::98998	Orange/Ivory	0.990480	[[32, 44, Color]]	[[7, 16, Pattern], [27, 31, Material], [206, 2...	contextual_snippet	Bangor Geometric Handwoven Wool Orange/Ivory A...
doc::99003	doc::99003	Red	0.980685	[[10, 13, Color], [23, 26, Color]]	[[40, 48, Material]]	contextual_snippet	Tennessee Red Football Red Area Rug.The chenil...
doc::98999	doc::98999	Black	0.961353	[[1207, 1212, Color]]	[[1041, 1047, Rug Shape], [1193, 1199, Rug Sha...	contextual_snippet	Makeover your rooms with just a single touch! ...
doc::99000	doc::99000	Orange	0.954024	[[29, 35, Color]]	[[95, 100, Pattern]]	contextual_snippet	Cleveland Throwback Football Orange Area Rug.S...
doc::98996	doc::98996	Black	0.799078	[[605, 610, Color]]	[[90, 96, Location], [255, 270, Pattern], [285...	contextual_snippet	Contemporary style and functionality come toge...

To learn more information regarding the parameters of `sf.preview_prompt_lf()`, run:

```
sf.preview_prompt_lf?
```

Creating a Prompt LF

Once you are satisfied with the preview results of the prompt LF, you can create the prompt LF by calling `sf.create_prompt_lf()`.

Here's how you can create the above prompt LF for a multi-label application:

```
sf.create_prompt_lf(
    NODE_UID,
    model_name="google/flan-t5-xxl",
    lf_name="my multilabel prompt LF",
    model_type="text2text_multi_label",
    prompt_text="""The possible labels are anova, bayesian, mathematical-
statistics, R.
    What is the most likely label for the following document: {Body_q}
{Body_a}\n\nLabel: """,
)
```

Here's how you can create the above prompt LF for a sequence tagging application:

Copyright © 2025 Snorkel AI, Inc. All rights reserved

```
sf.create_prompt_lf(
    NODE_UID,
    model_name="openai/gpt-4o",
    lf_name="my sequence tagging prompt LF",
    model_type="text2text_multi_label",
    primary_text_field="Description" # Required for sequence tagging
applications
    label="Color", # Required for sequence tagging applications
    prompt_text="""List all occurrences of the exact answers to the question.
Return answers separated by the delimiter ";". Make sure each answer is
separated exactly by this delimiter without any additional spaces or
characters.
    Question: What is the color here?""",
)
```

Note that the parameters from `sf.preview_prompt_lf()` stay the same for `sf.create_prompt_lf()`, with the exception of `lf_name` which is required to create the prompt LF. In addition, the `split` and `num_examples` are no longer needed as the prompt LF votes on all splits.

To learn more information regarding the parameters of `sf.create_prompt_lf()`, run:

```
sf.create_prompt_lf?
```

If you go to **Develop (Studio)** for the same node, you should now be able to see the newly created LF under **Labeling Functions** from the **In Progress** or **Active** tab.

Selecting Model Type

Refer to the following table to select the appropriate `model_type` value based on your application type, label space, and the task type of your selected foundation model.

model_type	Application	FM Type
`"text2text"`	Single-Label classification applications	Text2Text
`"text2text_multi_label"`	Multi-Label classification applications	Text2Text
`"text2text_qa"`	Sequence-label classification applications	Text2Text
`"qa"`	Sequence-label classification applications	QA
`"pdf_text2text_extractor"`	PDF candidate-based extraction applications	Text2Text
`"doc_vqa"`	PDF candidate-based extraction applications	DocVQA

Copyright © 2025 Snorkel AI, Inc. All rights reserved

Warm start

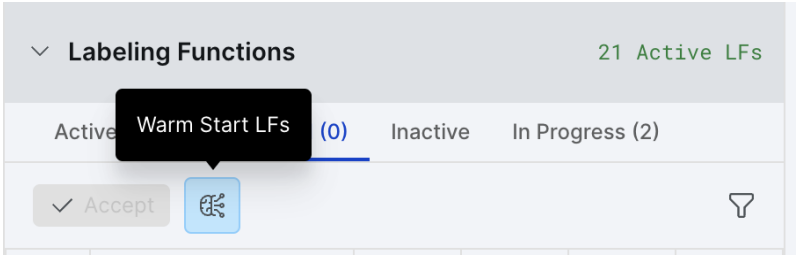
Warm start is the Snorkel Flow tool for getting your first training labels using state of the art foundation model (FM) based techniques. It's designed to give you a high-coverage labeling function (LF) where the foundation model is forced to give its best guess on every datapoint. This starting point can then be improved upon by the addition of other Snorkel Flow labeling functions.

> NOTE
>
> FM warm start supports the following application types: Single-label text classification, multi-label text classification, and single-entity sequence tagging.

> WARNING
>
> Warm start is not accessible within **Develop (Studio)** on a CPU-based deployment of Snorkel Flow. To use warm start without a GPU, you must use the SDK. See Using external models and Warm start SDK for more information.

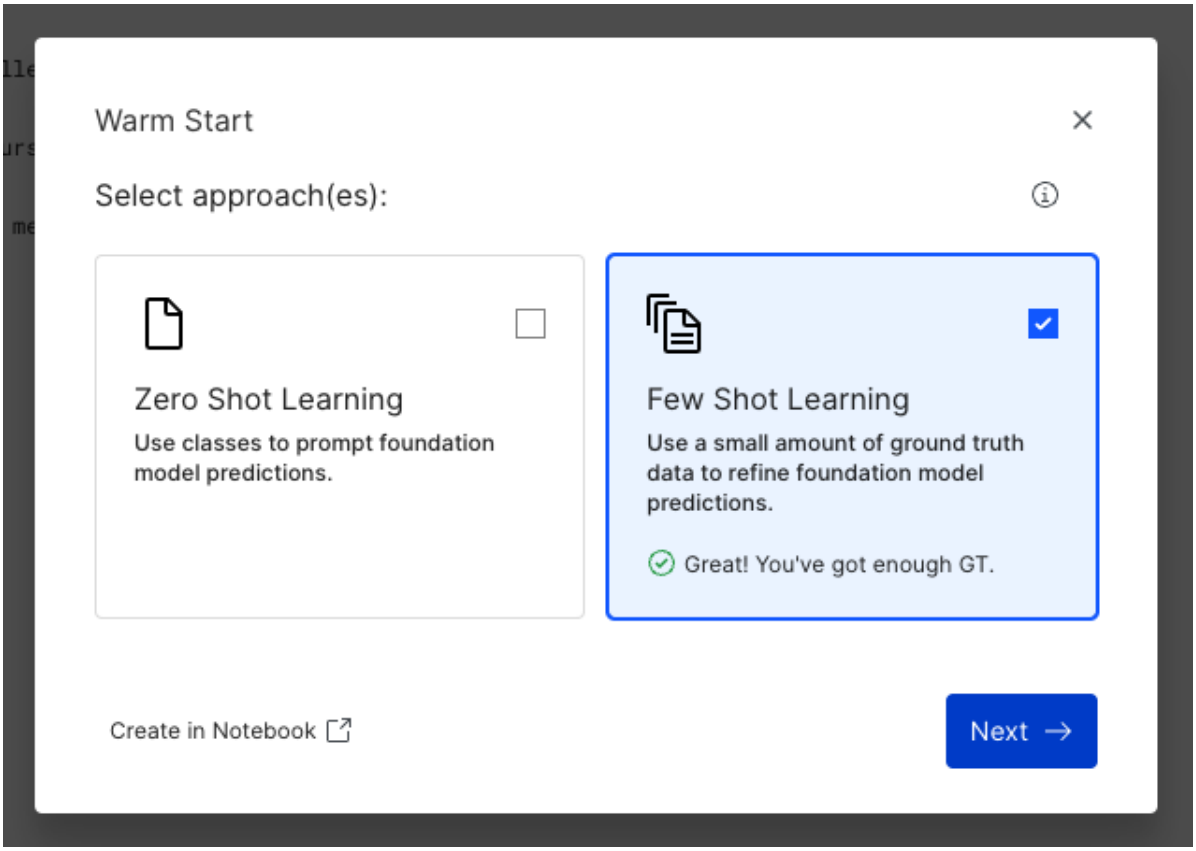
Running warm start

To access warm start in the UI, head to the **Labeling Functions** accordion inside your application. Then under **Suggested**, click the **Warm Start LFs** icon shown below.



This will launch the modal, where you select either zero-shot or few-shot learning. These techniques are described in the next section.

Copyright © 2025 Snorkel AI, Inc. All rights reserved



Follow the steps and at the final stage click **Create LFs**. This will kick off a long-running process to label your data using a foundation model. You can then monitor the progress in the **Labeling Functions** tab, under **In Progress**. If the LF is not showing, try clicking the refresh button.

Warm LF types

There are two types of LFs that can be created from the warm start wizard: zero-shot learning and few-shot learning.

Zero-shot learning describes the process of using only problem metadata, such as class names and descriptions, to make a prediction of the label for each datapoint.

Few-shot learning describes the process of additionally using a few (in the order of 10) labeled examples per class to make a better prediction of the label for each datapoint.

Zero-shot LFs

Zero-shot labeling functions don't require any labeled training data and instead utilize the label names to make an initial vote on each datapoint. These LFs will likely be less accurate than few-shot LFs, but may generalize better to unseen distributions of data. In order for them to work well they require sensible label names. For example, if the task is to categorize news articles, sensible label names could include: ["Sport", "Science & Technology", and "Business"].

Few-shot LFs

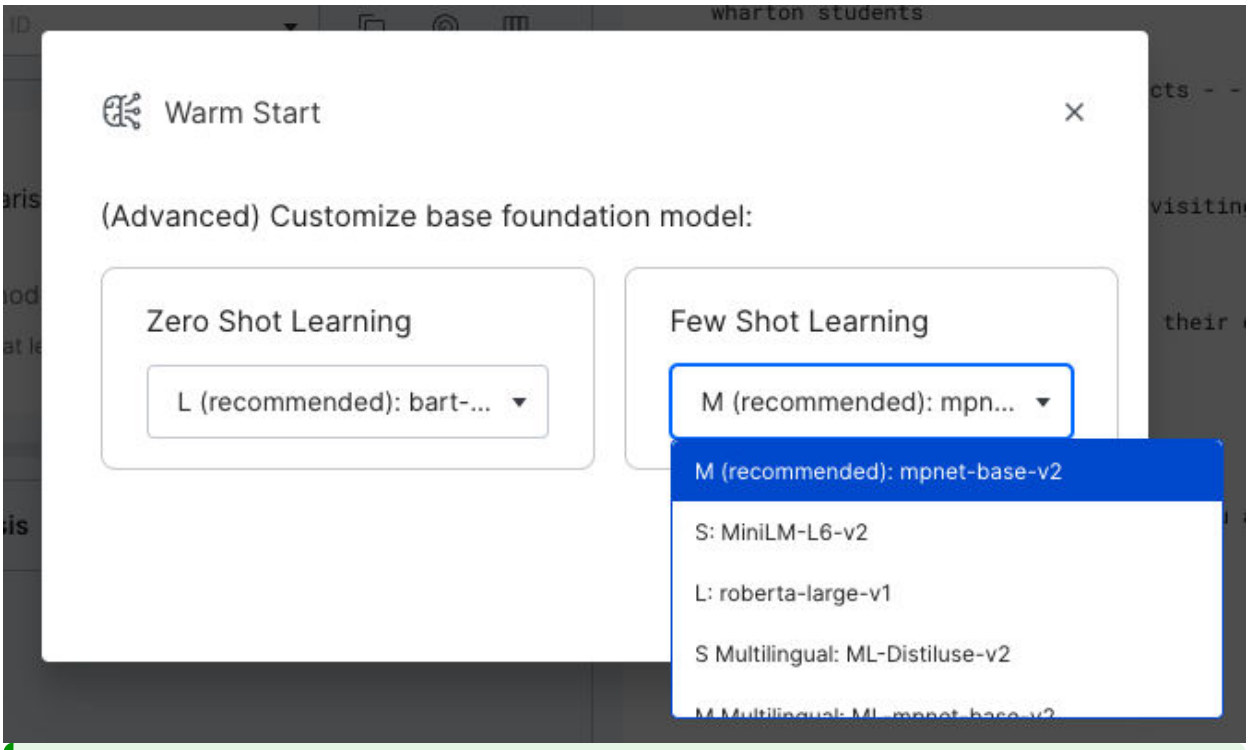
Few-shot labeling functions leverage existing ground truth labels to fit a model to your task. These tend to be more accurate than zero-shot LFs. Few-shot LFs are currently the recommended approach if you have enough ground truth labels. At least **five** labels are required per class, but no more than 50 labels per class will be used to ensure that the LFs generalize well.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

Model selection

Each method has a set of pre-trained models that you can select from. These models serve as the base for any information that will be distilled into the LFs created. Tradeoffs can be made between different choices:

- **Accuracy:** larger models tend to be more accurate.
- **Speed:** larger models will be slower to run.
- **Domain:** specialized models (e.g., Multilingual) may be more appropriate for your problem, and lead to better accuracy than the more general model.



> **TIP**
>
> For more advanced configuration, use the Warm start (SDK).

Model selection has a big effect on the wait time for, and accuracy of, the warm start LF. For document classification, below is a handy guide to help you pick.

Zero-shot LF models

Here are the models that are available for zero-shot LFs.

L (recommended): FLAN T5 Large: (google/flan-t5-large):

- A large English model that usually achieves good performance.
- Expect it to complete in 30mins - 2 hours. For especially large datasets with long documents, it can take longer.

M: FLAN T5 Base: (google/flan-t5-base):

- A medium size English model.
- It runs 2x faster than our recommended model.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

- It typically achieves a 10% relative drop in warm start F1 score compared to our recommended model.

Few-shot LF models

Here are the models that are available for few-shot LFs.

M (recommended): mpnet-base-v2 (sentence-transformers/paraphrase-mpnet-base-v2):

- A medium size English model with good all-round performance.
- Expect it to complete in 30mins - 2 hours. For especially large datasets with long documents, it can take longer.

S: MiniLM-L6-v2: (sentence-transformers/all-MiniLM-L6-v2):

- A small English model.
- It runs 3x faster than our recommended model.
- It typically achieves a 25% relative drop in warm start F1 score compared to our recommended model.

L: roberta-large-v1: (sentence-transformers/all-roberta-large-v1):

- A large English model.
- It runs 2.5x slower than our recommended model.
- It can achieve an larger warm start F1 score compared to our recommended model.

S Multilingual: ML-Distiluse-v2: (sentence-transformers/distiluse-base-multilingual-cased-v2):

- A small multilingual model.
- It supports 50+ languages: ar, bg, ca, cs, da, de, el, en, es, et, fa, fi, fr, fr-ca, gl, gu, he, hi, hr, hu, hy, id, it, ja, ka, ko, ku, lt, lv, mk, mn, mr, ms, my, nb, nl, pl, pt, pt-br, ro, ru, sk, sl, sq, sr, sv, th, tr, uk, ur, vi, zh-cn, zh-tw.
- It runs 15% slower than our recommended model.
- It typically achieves a 25% relative drop in warm start F1 score on English tasks compared to our recommended model.

M Multilingual: ML-mpnet-base-v2: (sentence-transformers/paraphrase-multilingual-mpnet-base-v2):

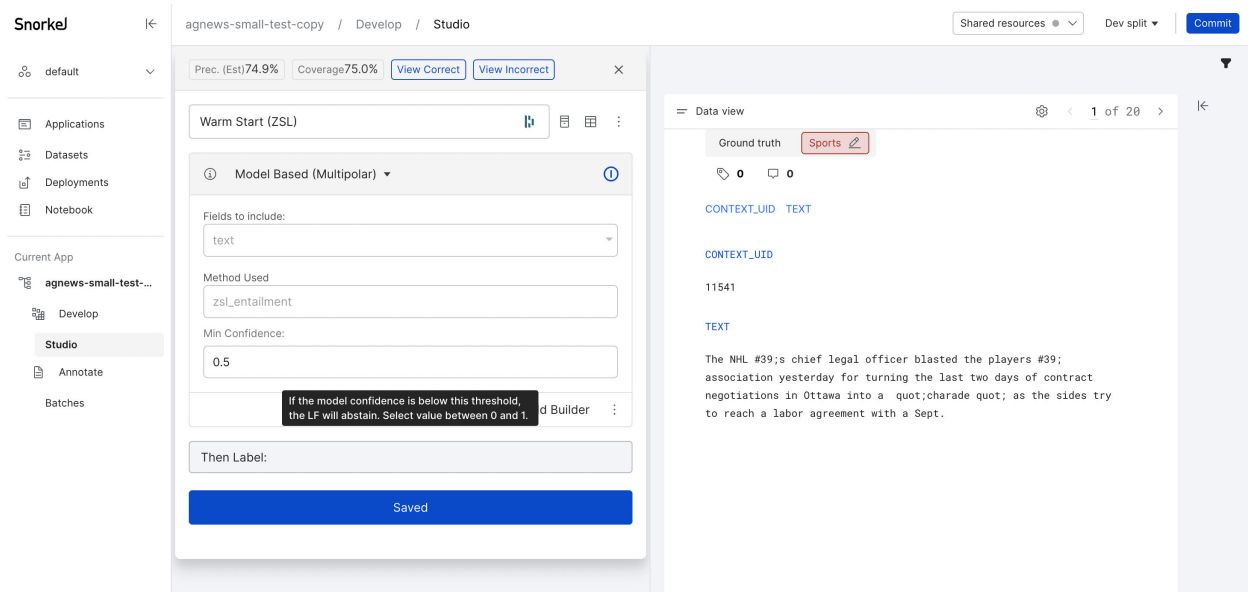
- A medium size multilingual model.
- It supports 50+ languages: ar, bg, ca, cs, da, de, el, en, es, et, fa, fi, fr, fr-ca, gl, gu, he, hi, hr, hu, hy, id, it, ja, ka, ko, ku, lt, lv, mk, mn, mr, ms, my, nb, nl, pl, pt, pt-br, ro, ru, sk, sl, sq, sr, sv, th, tr, uk, ur, vi, zh-cn, zh-tw.
- It runs 70% slower than our recommended model.
- It typically achieves a 10% relative drop in warm start F1 score on English tasks compared to our recommended model.

Editing warm start LFs

Once a warm start LF is active, you can edit the confidence threshold to further customize it. The confidence value reflects the foundation model's certainty in its predictions. Increasing the threshold trades off coverage for precision:

- Higher confidence thresholds mean that the model must be more certain before making a prediction. This increases precision, but decreases coverage.
- Lower confidence thresholds mean that the model can make more predictions, even when less certain. This increases coverage, but decreases precision.

Copyright © 2025 Snorkel AI, Inc. All rights reserved



For more advanced warm start capabilities and granular configuration, see Warm start (SDK).

Copyright © 2025 Snorkel AI, Inc. All rights reserved

Warm start (SDK)

Warm start is the Snorkel Flow tool for getting your first training labels using state of the art foundation model (FM) based techniques. You can create labeling functions (LFs) by applying zero-shot learning (ZSL) and few-shot learning (FSL) on your data. This provides baseline LF results to feed into a machine learning model.

Most customers kick-off FM warm start models using the UI. However, warm start can also be triggered through the notebook. The SDK provides additional functionality and more control over the ZSL/FSL process, enabling you to specify:

- Which data features to include in the model (default: all text columns)
- Which data splits to run inference over (default: all active datasources)
- Whether to output multiple unipolar LFs or a single multi-polar LF (default: multi-polar LF)
- The base foundation model to use (default: our recommended model)
- Whether to hide the warm start LFs and register the votes as a Snorkel Flow model (default: no)

> **NOTE**
>
> FM warm start supports the following application types: Single-label text classification, multi-label text classification, and single-entity sequence tagging.

Running warm start

To begin, import our SDK and retrieve the relevant model node for your application:

```
import snorkelflow.client_v3 as sf
ctx = sf.SnorkelFlowContext.from_kwargs() # Import context to our current application

APP_NAME = "APPLICATION_NAME" # Name of your application
NODE_UID = sf.get_model_node(APP_NAME) # Model node used in the application
```

To retrieve the available warm start methods for your node, along with their descriptions, use the `get_available_warm_start_methods()` function. An example of this is shown below:

Copyright © 2025 Snorkel AI, Inc. All rights reserved

```
>>> results = sf.get_available_warm_start_methods(NODE_UID)
>>> print(results)
Note: All possible warm start methods for this node type are supported.
Methods:
         zsl_prompt: A zero-shot learning method that uses the class names to
automatically generate a prompt template which is used to predict the label
     of each datapoint. For example, if the class names are ['cat', 'dog',
'bird'],
     we may ask the model `Is this a cat, dog, or bird?`, the model's output
is
     then used automatically to make a prediction. Recommended models include:
     ['google/flan-t5-large', 'google/flan-t5-base']

     ...

>>> print(results.methods["zsl_prompt"])
Description:
    A zero-shot learning method that uses the class names to
automatically generate a prompt template which is used to predict the label
of each datapoint. For example, if the class names are ['cat', 'dog', 'bird'],
we may ask the model `Is this a cat, dog, or bird?`, the model's output is
then used automatically to make a prediction.
Any generic sequence-to-sequence huggingface model can be used for ZSL Prompt
however, it's recommeded that the model included classification tasks in the
training process.
Recommended Models:
    google/flan-t5-large: FLAN (Fine-tuned LAnguage Net) T5 (Text-To-Text
Transfer Transformer) is a transformer-based architecture where all NLP
training tasks are reframed into a unified text-to-text format and learned
jointly without the need for separate task-specific heads, as in BERT-style
models. The large variant has 780M parameters.
        google/flan-t5-base: FLAN (Fine-tuned LAnguage Net) T5 (Text-To-Text
Transfer Transformer) is a transformer-based architecture where all NLP
training tasks are reframed into a unified text-to-text format and learned
jointly without the need for separate task-specific heads, as in BERT-style
models. The base variant has 250M parameters.
```

From this, you'll be provided with a list of methods that you can use for warm start. We recommend running warm start over just the dev and valid dataset splits initially to gauge how well it performs before running on all of your data. To do this, pass ["dev", "valid"] to the `run_warm_start` method along with your node uid, warm start method name, and the base foundational model. More information about the available parameters and an example can be seen below:

Copyright © 2025 Snorkel AI, Inc. All rights reserved

```
Signature:
sf.run_warm_start (
    node :  int ,
    warm_start_method :  str ,
    foundation_model :  str ,
    lf_name :  str  =   'Warm Start SDK'  ,
    splits :  Union [ List [ str ]  ,  NoneType ]   =   None  ,
    columns :  Union [ List [ str ]  ,  NoneType ]   =   None  ,
    one_lf_per_class :  bool  =   False  ,
    sync :  bool  =   True  ,
    allow_unsupported_foundation_model :  bool  =   False ,
    ** additional_model_kwargs :  Any ,
 )   ->  str
 Docstring:
Kick off an FM Warm Start job on a given node.

Parameters
----------
node
    Node uid to run Warm Start on.
warm_start_method
    The string identifier of the Warm Start method to use. If you are
    unsure, try running
    sf.get_available_warm_start_methods(NODE_UID) first to see
    which methods are available for this node.
foundation_model
    The Foundation Model to use within the Warm Start method. For
    example, openai/gpt-4 or bigscience/T0pp
lf_name
    Name of the created Labeling Function.
splits
    Splits to run Warm Start inference over. Defaults to all, ["train",
    "dev", "valid", "test"].
columns
    Text fields to use in Warm Start. Defaults to all text columns
    except in Information Extraction nodes where it defaults to
    [LEFT_CONTEXT, SPAN_PREVIEW, RIGHT_CONTEXT].
one_lf_per_class
    If True, a separate LF will be created for each class. This allows
    for a finer level of control over the Warm Start output.
sync
    If True, method will block until the Warm Start job is complete.
    Note the job progress can always be be monitored within the 'In
    Progress' LFs table on Studio or via sf.poll_job_status(job_uid).
allow_unsupported_foundation_model
    If True, run Warm Start for the specified foundation_model even if
    the model is not supported. This allows additional control to
    bypass foundation model arg validation.
```

Copyright © 2025 Snorkel AI, Inc. All rights reserved

```
additional_model_kwargs
    Additional kwargs to pass to the Foundation Model. For example, to
    increase the context window size specify the `max_input_tokens`
    kwarg.

Returns
-------
job_uid : str
    The uid of the Warm Start job which can be used to monitor progress
    with sf.poll_job_status(job_uid).

Example
-------
>>> sf.run_warm_start(NODE_UID, "zsl_prompt", "google/flan-t5-base")
Note the job progress can always be be monitored within the 'In Progress' LFs
table on Studio or via sf.poll_job_status('XXX')
```

You can then monitor the progress within **Develop (Studio)** in the **Labeling Functions** tab, under **In Progress**. If the LF is not showing, try clicking the refresh button.

Running inference on new data

You can run an existing warm start model on new data through the SDK. To do so, the data source must be active on your application and included within a split. You can then make use of the following code snippet to kick off warm start inference on the specified splits:

```
>>> LF_UID = sf.get_lf_uid(node_uid, "Warm Start (SDK)")
>>> results = sf.run_lf_inference(NODE_UID, LF_UID, ["train", "test"])
Note the job progress can be monitored with sf.poll_job_status('XXX')
```

> NOTE
>
> Run the empty SDK method followed by a question mark to see it's documentation. For example, `sf.run_lf_inference?`

Any data within the split that this warm start LF has already processed will be automatically retrieved from the cache. This saves compute time and allows for this method to be used for failure retries if say HuggingFace went down while the LF was processing.

Registering a warm start LF as a Snorkel Flow model

To help with error analysis, you may want to register the warm start LF as a model:

Copyright © 2025 Snorkel AI, Inc. All rights reserved

```
# set the only active LF to be the warm start LF
sf.archive_lfs(node_uid)
ctx.tdm_client.put(f"/nodes/{node_uid}/active-lfs/{warm_start_lf_uid}", json=
{"state": "active"})

# create a training set out of the LF
lf_pkg = sf.package_lfs(node_uid, "Warm Start LF Package")
rsp = sf.add_training_set(node_uid, lf_package_version=lf_pkg)

# register the warm start model
model_uid = sf.register_model(
  node_uid,
  description="Warm Start model predictions",
  training_set=rsp["training_set_uid"],
  name="Warm Start",
)

# add the LF predictions to the model
import pandas as pd
predictions = pd.concat([
  sf.get_training_set(node_uid, rsp["training_set_uid"], split,
include_data_columns=[])
  for split in ["train", "dev", "valid", "test"]
])
sf.add_predictions(
  node_uid,
  x_uids=list(predictions.index),
  predicted_labels=list(predictions["training_set_labels"]),
  model_uid=model_uid,
)

# hide the warm start LF (this can also be done via the UI)
ctx.tdm_client.post("/user-settings", json={
  "user_uid": user_uid,
  "application_uid": app_uid,
  "node_uid": node_uid,
  "settings": {"global_preferences": {"should_show_warm_start_lfs": False}}
})
```

Remote warm start / GPT baseline

If you don't have access to local GPUs, you can utilize a remote model with an automatically generated ZSL prompt to initiate warm start for your problem. Please note that this requires setting up a third-party integration within Snorkel Flow. For more information, see Utilizing external models.

Given that this approach may involve third-party expenses when employing models from providers such as OpenAI, we recommend estimating the cost of running warm start on a portion or the entirety of your dataset. You can use the `get_warm_start_cost` command to provide an approximate USD total for the operation:

Copyright © 2025 Snorkel AI, Inc. All rights reserved

```
>>> results = sf.get_warm_start_cost(
     node=NODE_UID,
     configs=[{"method": "zsl_prompt_remote", "provider": "openai", "model":
"openai/gpt-4"}],
     splits=["dev", "valid"]
    )
>>> results[0].additional_cost
63.20
```

Once you have linked your OpenAI account and are comfortable with incurring the specified third-party cost, you can initiate the remote warm start job using the same `run_warm_start` method that was explained at the beginning of this page. Be sure to specify the `zsl_prompt_remote` warm start method when doing so.

```
sf.run_warm_start(NODE_UID, "zsl_prompt_remote", "openai/gpt-4", splits=
["dev"])
```

Progress can be tracked via the In Progress LFs table.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

Utilize embeddings

This page walks through how to create and use embeddings in Snorkel Flow. You can utilize embeddings across your end-to-end workflow: understanding your data, labeling programmatically, and model training.

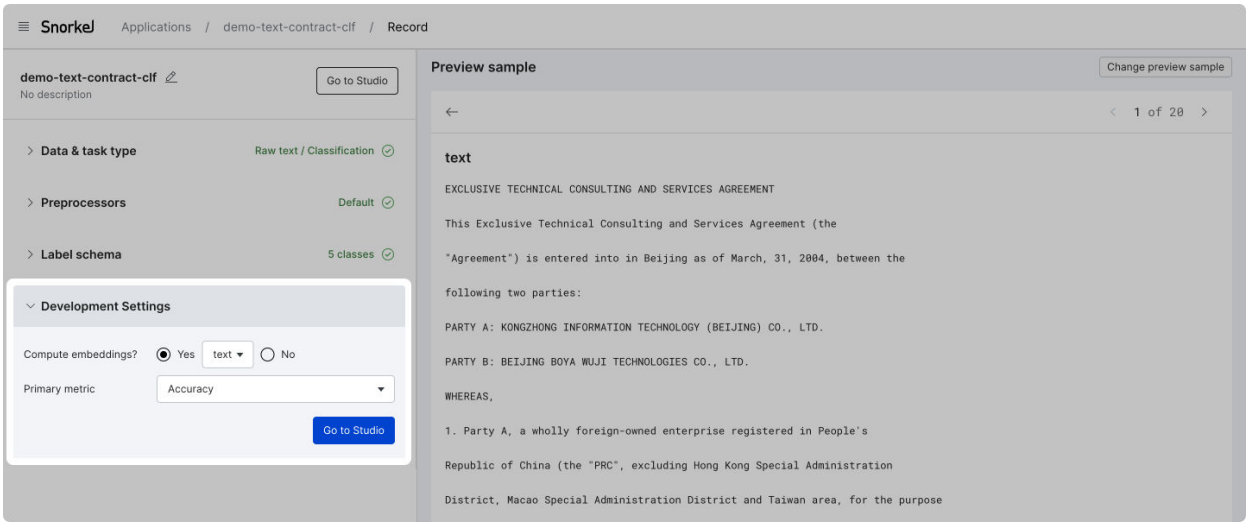
There are two ways to compute embeddings:

- During application creation using the guided workflow
- During model development in Studio

Once you have computed your embeddings, you can then utilize them in Studio!

Compute embeddings during application creation

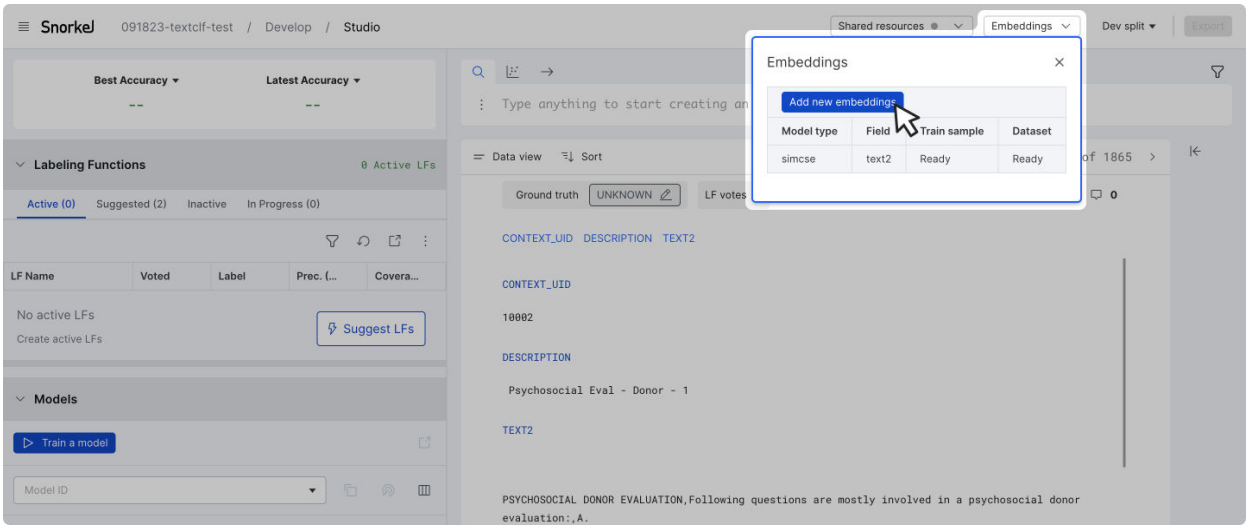
When you create an application using the guided workflow, the compute embeddings option will appear at the Development Settings step. This is set to **Yes** by default, with the field selection being the primary text/PDF field. The embeddings are calculated using SimCSE.



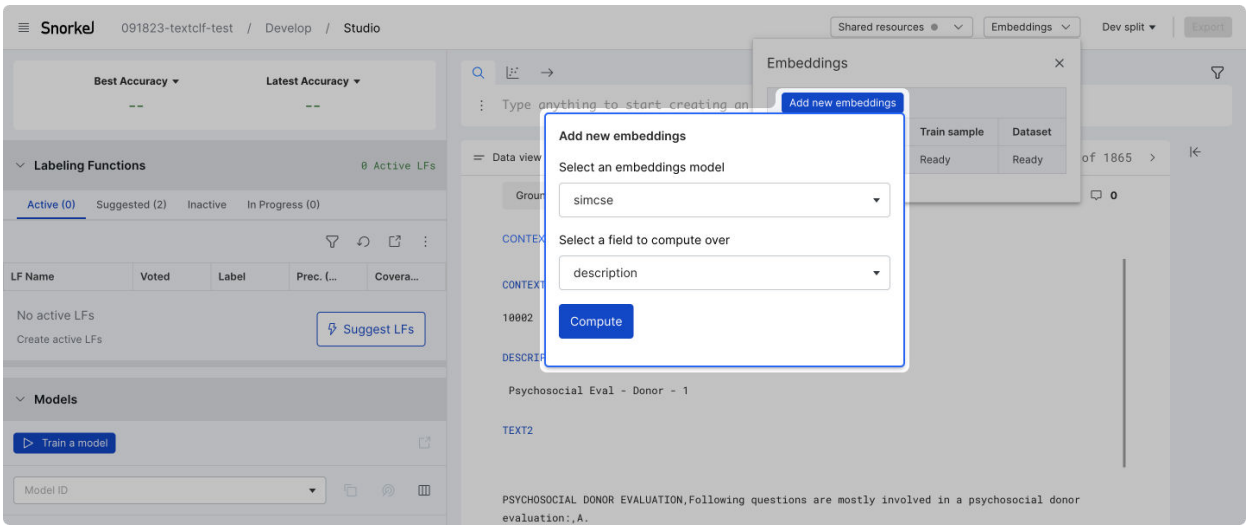
Compute embeddings in embedding home

You may also opt to compute additional embeddings during model development in Studio. To do so, click the **Embeddings** dropdown on the top-right corner of your screen, then click **Add new embeddings**. For sequence tagging applications, you can calculate RAG embeddings. For more information, see Prompting with document chunking (RAG).

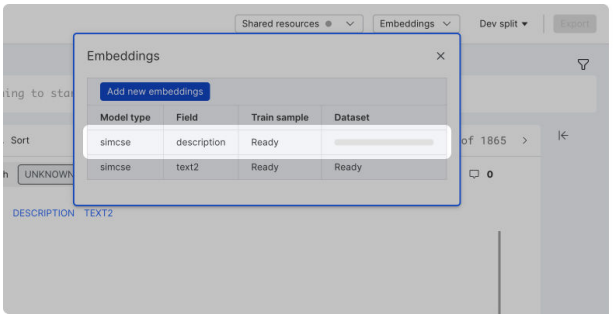
Copyright © 2025 Snorkel AI, Inc. All rights reserved

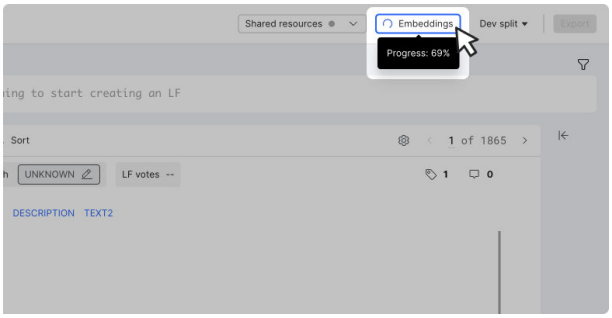


In the modal, select the desired parameters, then click **Compute**.



You can track calculation progress in the table or by hovering over the Embeddings dropdown. Once the calculation is complete, you can use the new embeddings throughout Studio.



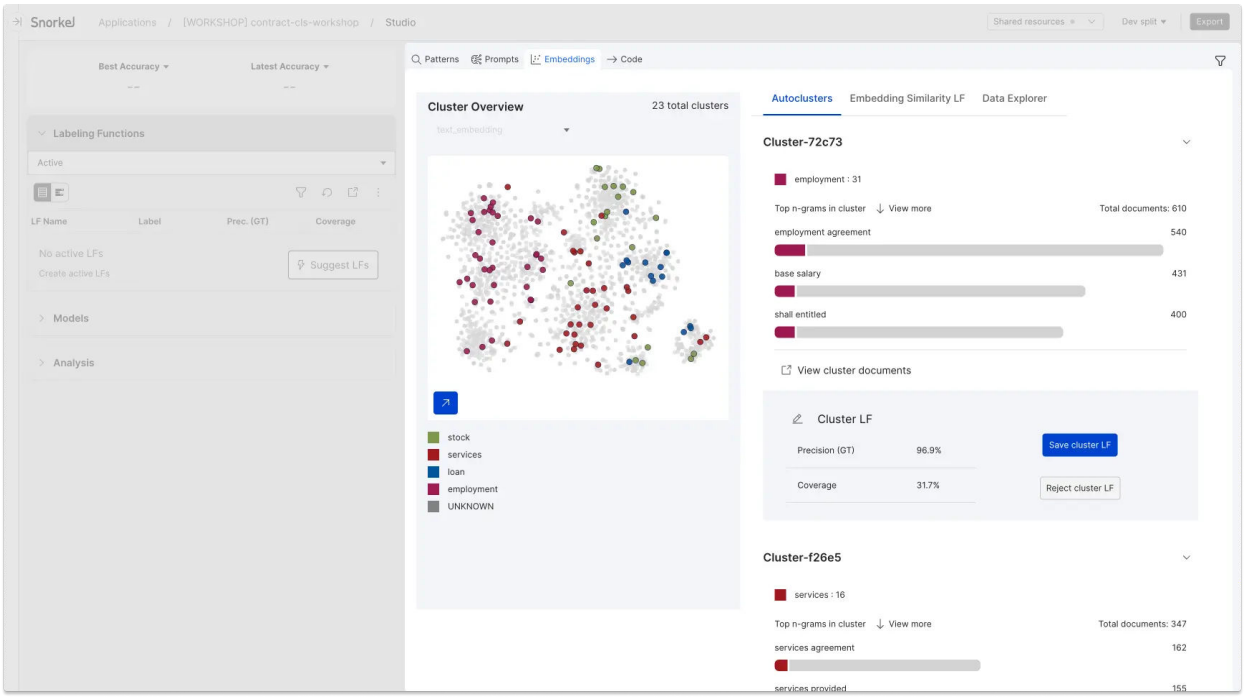


Embedding-powered features in Studio

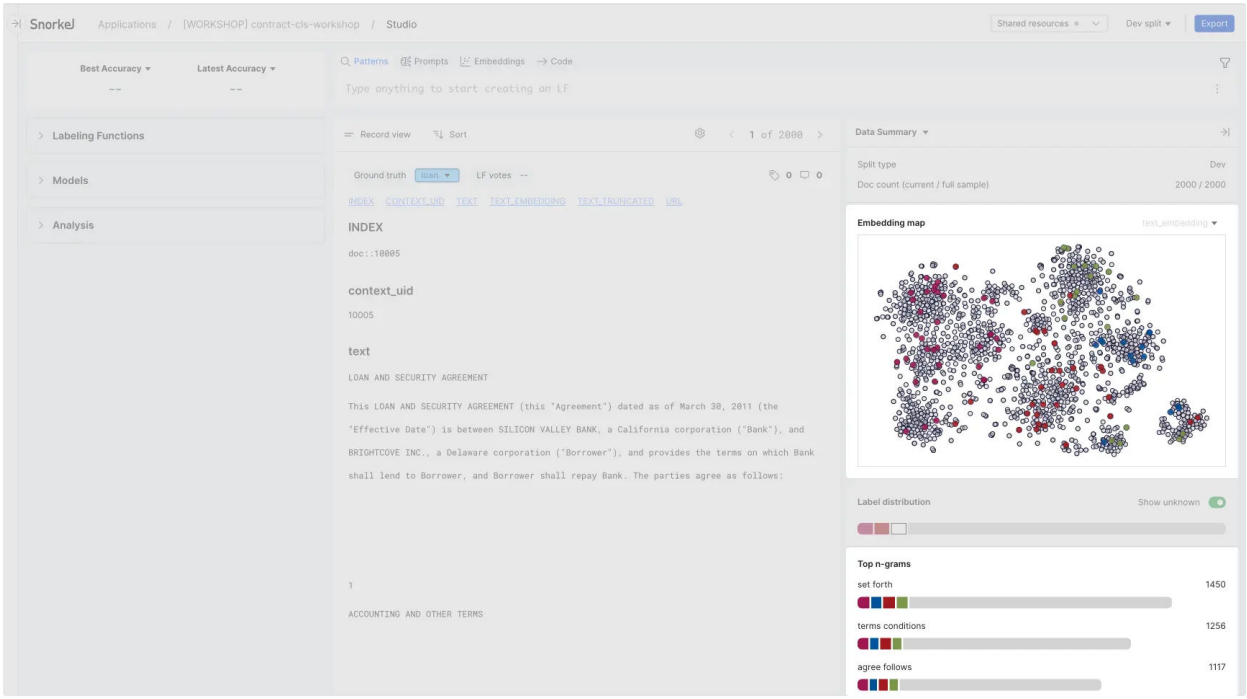
Here is a list of features in Studio that are powered by embeddings:

- You can use the embedding-based cluster workflow to identify groups of similar datapoints, which can then be used to create cluster LFs.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

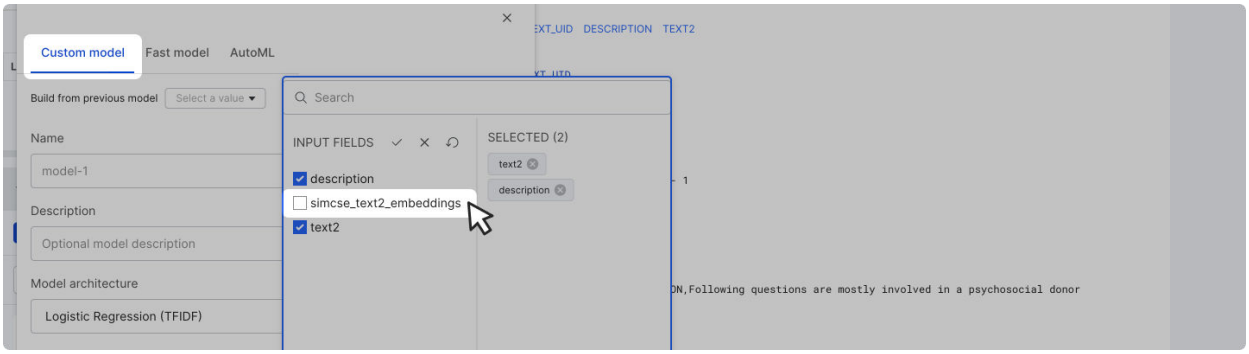


- You can view an embedding map as well as the top n-grams in the Data summary pane.



- You can use the embedding field as an input when model training.

Copyright © 2025 Snorkel AI, Inc. All rights reserved



Supported use cases

The table below shows the availability of embedding compute and embedding-powered features per use case in the 2024.R2 LTS (v0.93) release:

	Raw text classification	Raw text candidate extraction	Raw text sequence tagging	PDF classification	PDF extraction	All multi-label cases
Compute embeddings (simcse, spacy, clip)	✅	✅	❌	✅	❌	❌
Compute RAG embeddings	❌	❌	✅	❌	❌	❌
Embedding-based cluster workflow (Cluster view)	✅	✅	❌	✅	❌	❌
Embedding map (in Data Summary Pane)	✅	✅	❌	❌	❌	❌
Top n-gram (in Data Summary Pane)	✅	✅	❌	❌	❌	❌
Embedding field for model training	✅	✅	✅	❌	❌	❌

Copyright © 2025 Snorkel AI, Inc. All rights reserved

Embedding based cluster LFs

The embedding based cluster workflow allows you to supercharge your programmatic labeling by leveraging embeddings to build labeling functions (LFs) in your Snorkel Flow application.

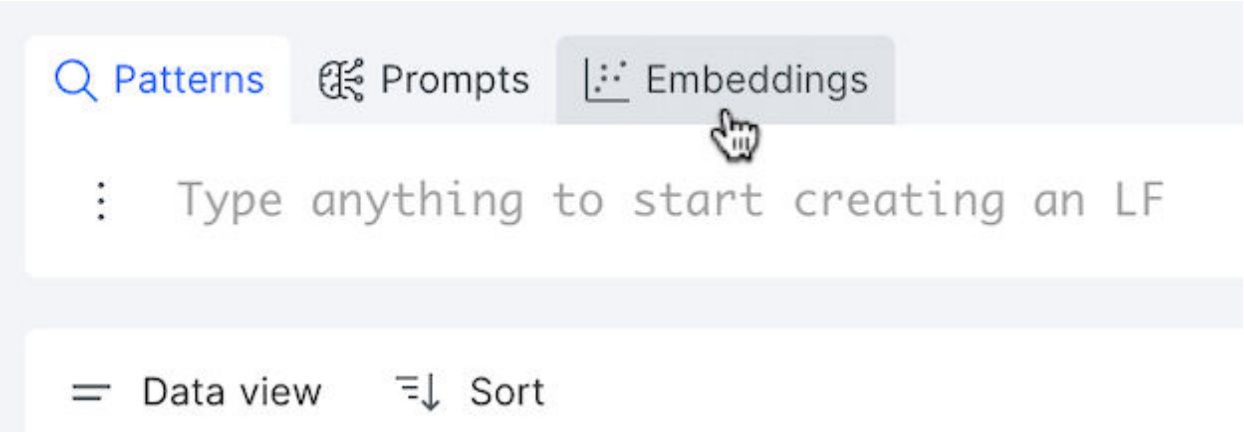
> NOTE
>
> To use the embedding based cluster workflow, you'll need to add an EmbeddingFeaturizer or RegisterCustomEmbedding upstream of your model node.

> NOTE
>
> Ground truth labels are required to generate clusters. Generally, this workflow works best with a mix of data points that have ground truth labels and data points that are unlabelled.

> NOTE
>
> Snorkel Flow does not support embedding-based cluster LFs for text sequence tagging and multi-label applications.

Using embedding based cluster workflow in Studio

In Studio, the embedding based cluster workflow helps you identify themes of similar data points and then directly action on them by creating cluster LFs. Additionally, we have tools to find documents similar to any input text, as well as drag-to-discover functionality to easily explore your embeddings. To enter the embedding based cluster workflow, click the **Embeddings** icon and text within the LF composer toolbar as shown below.



Autoclusters

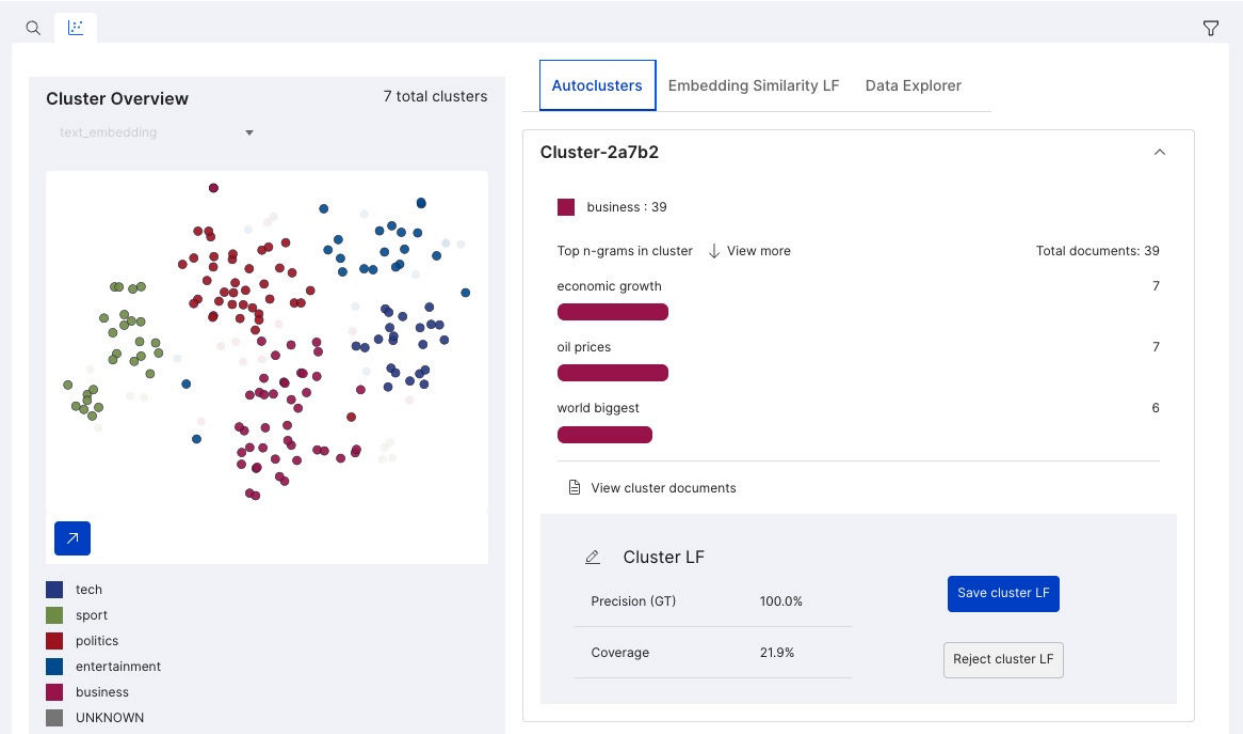
Autoclusters provide auto-suggested cluster LFs. The two main panels of this view are:

1. **Cluster Overview**
   - Provides a 2 dimensional visual representation of the embeddings.
   - Allows you to choose which embedding field to work with.
   - Allows you to select any point to bring up the related cluster detail in the LF panel.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

2. Cluster Details
   - Lists the auto-suggest cluster LFs sorted by coverage.
   - When a cluster is selected, it highlights the related data points in the **Cluster Overview**.
   - Allows you to drill down to specific example documents in the cluster.
   - Shows the top n-grams and ground truth proportions in the cluster.
   - Shows the precision and coverage metrics of the preview LF.
   - **Save cluster LF** saves the LF to your active LF list.
   - **Reject cluster LF** minimizes the details from further listings with the possibility of reevaluation.

> **NOTE**
>
> Clusters are automatically recomputed whenever the active set of data points changes. This includes resampling the development set, applying a filter, or previewing a text-search-based LF.

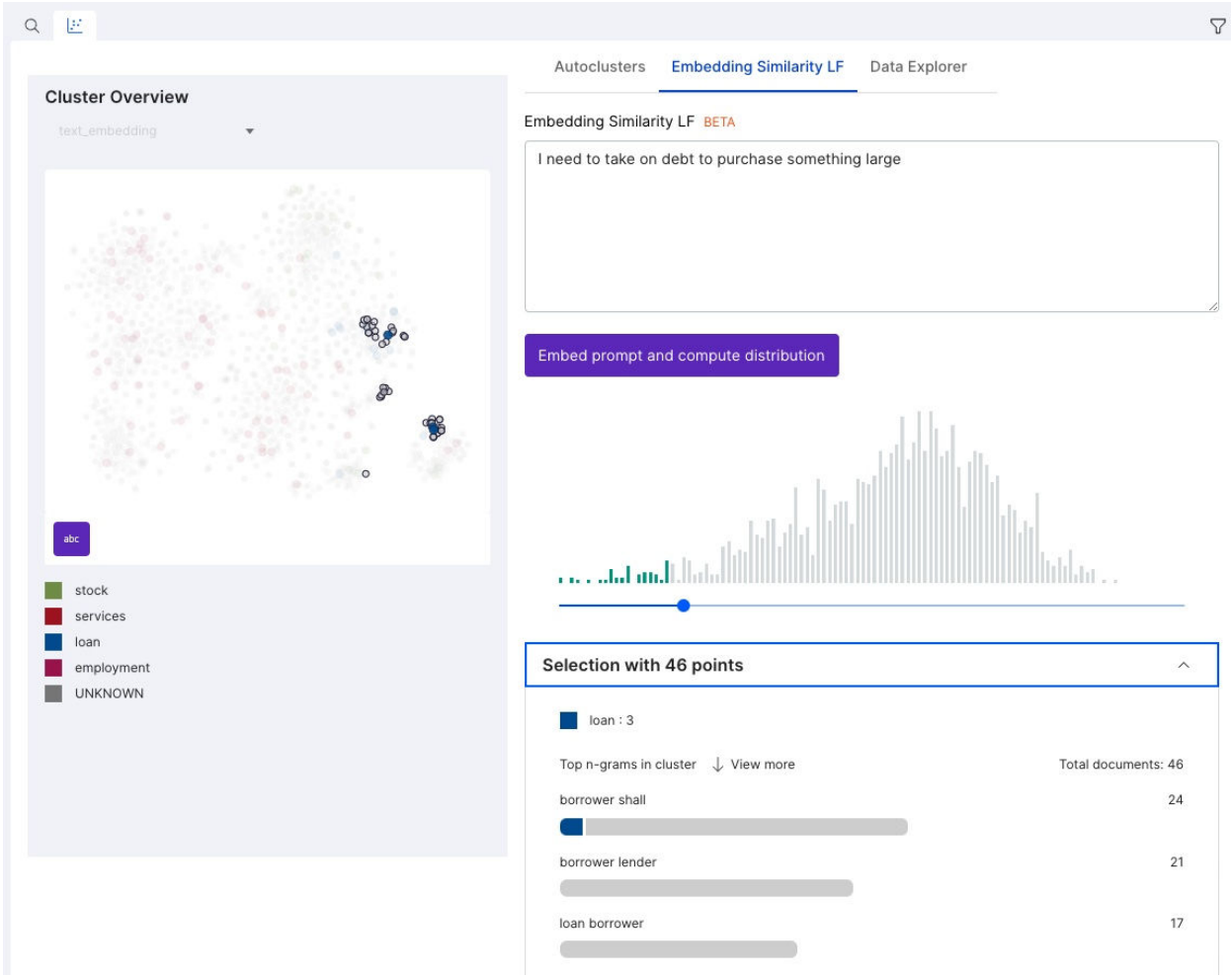


(Beta) Embedding similarity LF

In addition to the automatically generated clusters, you can also create your own clusters by using the **Embedding Similarity LF** tab. This tab lets you enter any text, for example, a sentence from a document that you want to find similar examples for. Snorkel will then compute the embedding of the entered text, calculate the similarity between each data point and the entered text embedding, and create an interactive histogram of the results. You can select a threshold beneath the histogram to see the pertinent data points, and then create a cluster LF from the selected data points.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

> **NOTE**
>
> This functionality does not work with RegisterCustomEmbedding as the entered text embedding must be calculated on demand.



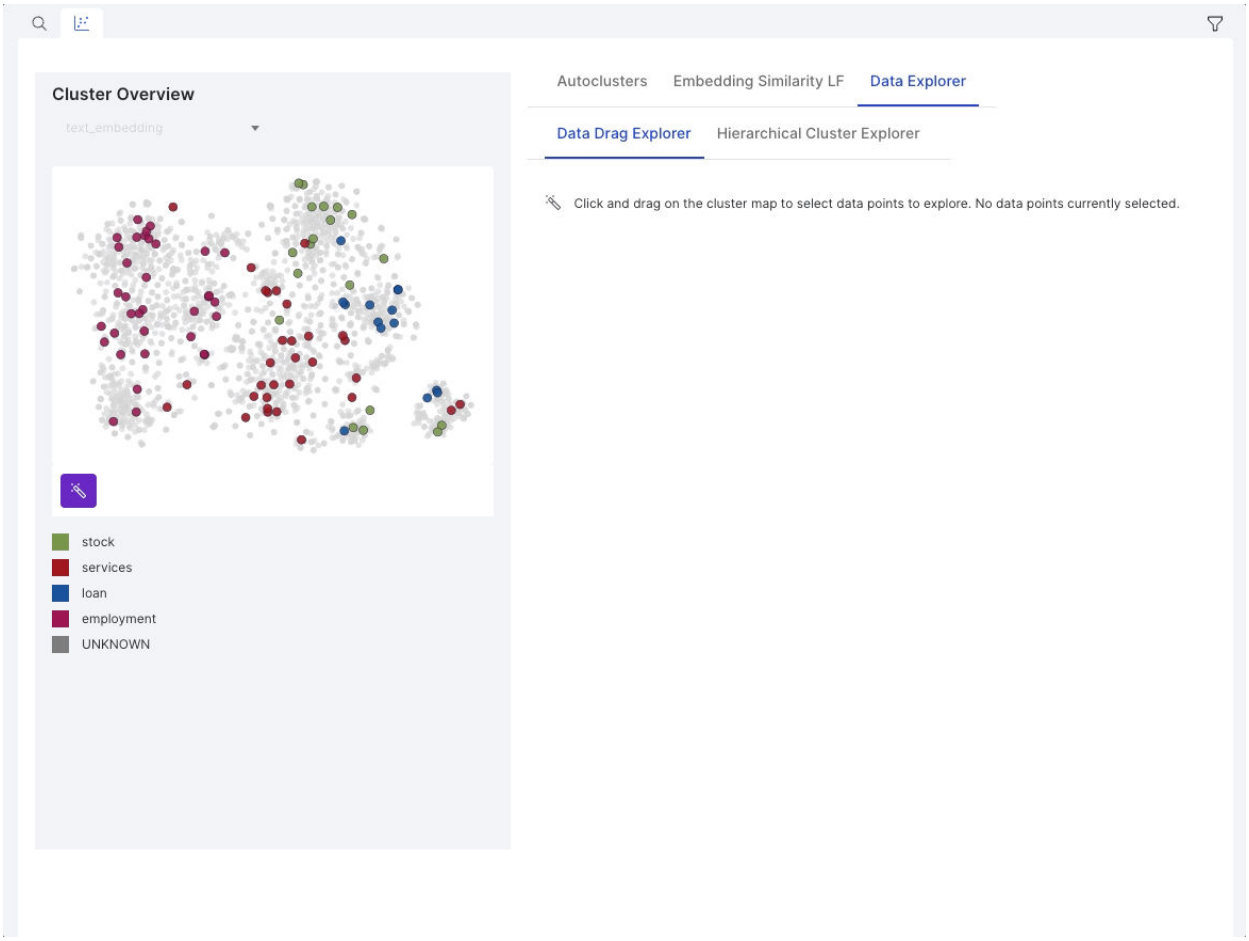
Data explorer

The **Data Explorer** extends your ability to leverage clusters by allowing you to visually choose your own cluster or calibrate the auto cluster settings.

Data drag explorer

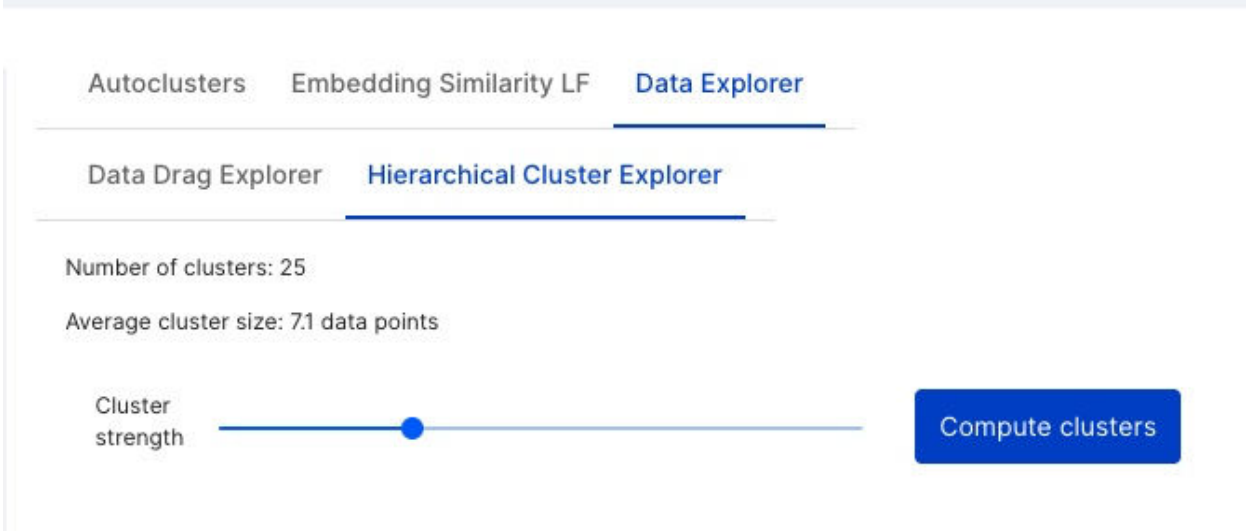
The **Data Drag Explorer** allows you to click and drag the points that you want to cluster on the cluster map. This results in a selection of points which you can create an LF from or tag for further use.

Copyright © 2025 Snorkel AI, Inc. All rights reserved



Hierarchical cluster explorer

The **Hierarchical Cluster Explorer** allows you to calibrate the **Cluster strength** to your needs, which adjusts the average cluster size in data points. After each computation of clusters, you can choose clusters to create LFs from just as with the auto suggested cluster LFs.



Copyright © 2025 Snorkel AI, Inc. All rights reserved

Introduction to labeling functions (LFs)

Labeling functions (LFs) in Snorkel Flow allow users to programmatically label data. Labeling data with LFs is faster than asking subject matter experts to label it by hand.

LFs allow users to specify the conditions in which Snorkel Flow will apply a particular label to a data point. This is known as a *vote* for that data point to receive that label. If a data point doesn't meet the conditions for a particular LF, the LF abstains by voting `UNKNOWN`.

The Snorkel Flow labeling model then combines all LF-assigned votes and other signals to decide how to label each data point. The labeling model estimates LF accuracies and aggregates LFs to calculate label votes without relying on ground truth labels.

LFs can be renamed, edited, or deleted.

LF composer toolbar

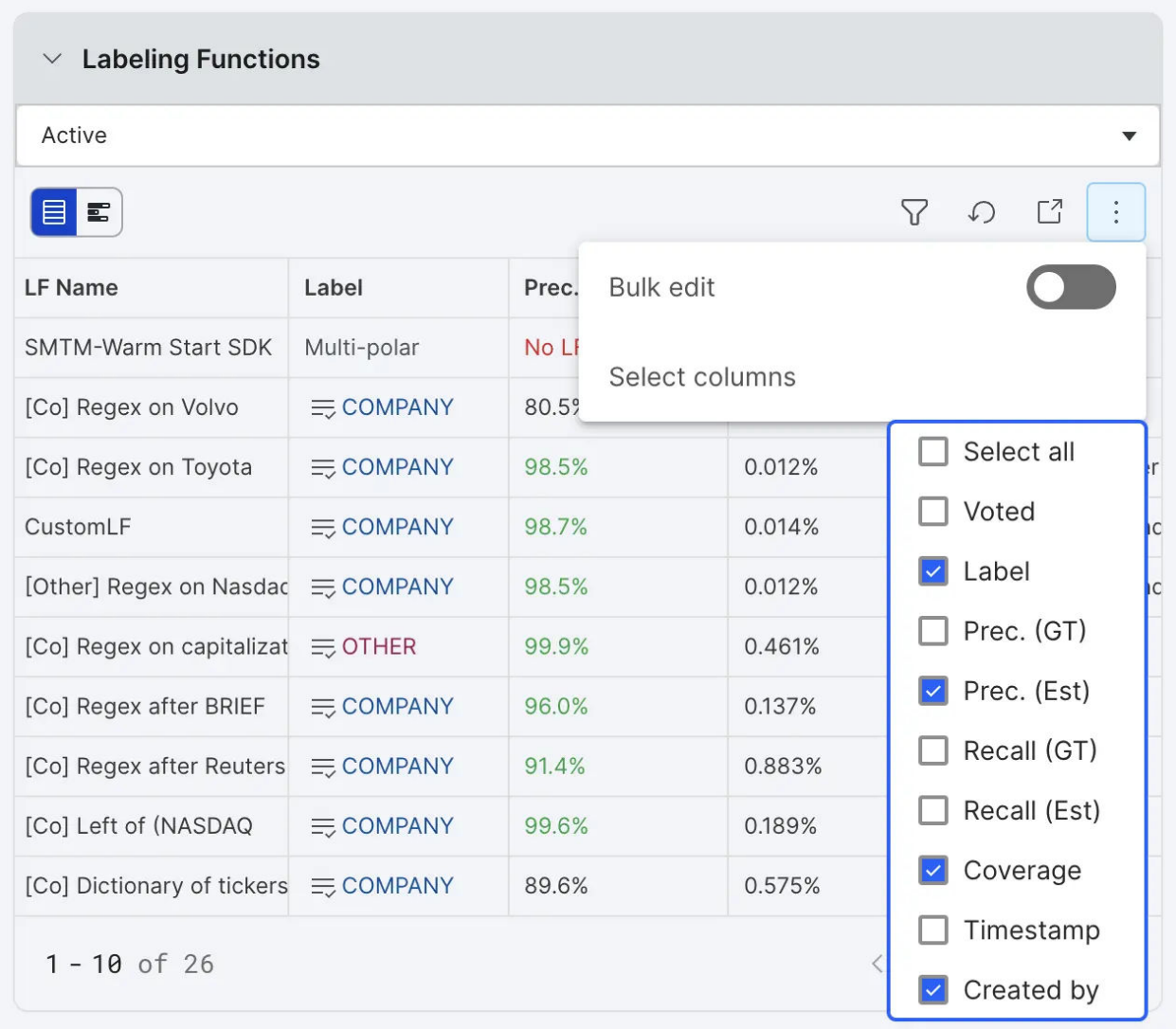
LFs can be created using several tools, including:

1. Search-based
2. Embeddings-based
3. Foundation model-based

LF summary

Once LFs have been created, details about the LFs are available in the LFs table in the left sidebar. These details include the LF name, LF vote, LF metrics, LF author, and the time the LF was created. The table columns can be customized by clicking the three dots in the top right and selecting the desired columns. More strategies for LF management can be found in the article on LF management.

Copyright © 2025 Snorkel AI, Inc. All rights reserved



- **LF Name**: The name for this LF
- **Label**: The label assigned by the labeling function
- **Voted** (specific to the datapoint selected in Data Viewer tab):
  - Gray ✓ voted, but no GT exists
  - Green ✓ voted correctly based on assigned GT
  - Red x voted incorrectly based on assigned GT
- **Precision (GT)**: The (number of correct LF votes) / (total number of LF votes), based on the existing GT labels. This is only an estimate based on a potentially very small dev set. The learned precision Snorkel Flow will use to generate probabilistic labels for data points won't be calculated until a label package is created and can be viewed on the LF Packages page.
- **Precision (Estimated)**: The same as Precision (GT) but using estimated GT labels (based on LF votes) wherever manual GT labels do not exist. (Total LF votes for class in agreement with estimated GT labels) / (Total LF votes for class).
- **Recall (GT)**: The (number of correct LF votes) / (total number of examples in class), based on the existing GT labels.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

- **Recall (Estimated)**: The same as Recall, but using estimated GT labels (based on LF votes) wherever manual GT labels do not exist. (Total LF votes for class in agreement with majority) / (Total majority votes for class).ƒ
- **Coverage**: The percentage of data points in the current split that this labeling function votes on.
- **Count (GT)**: The total number of LF votes generated by this LF on the current split.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

LF management tab

The Labeling Functions (LF) Management Tab in Snorkel Flow provides an interface to organize and manage labeling functions (LFs), crucial for your Snorkel Flow applications. It allows you to filter, sort, and evaluate LFs, helping you streamline the labeling process and optimize model performance.

By managing LFs effectively, you ensure consistent, high-quality labels, which are vital for improving your model's accuracy.

Now, let's explore the key features and functionalities of the LF Management Tab.

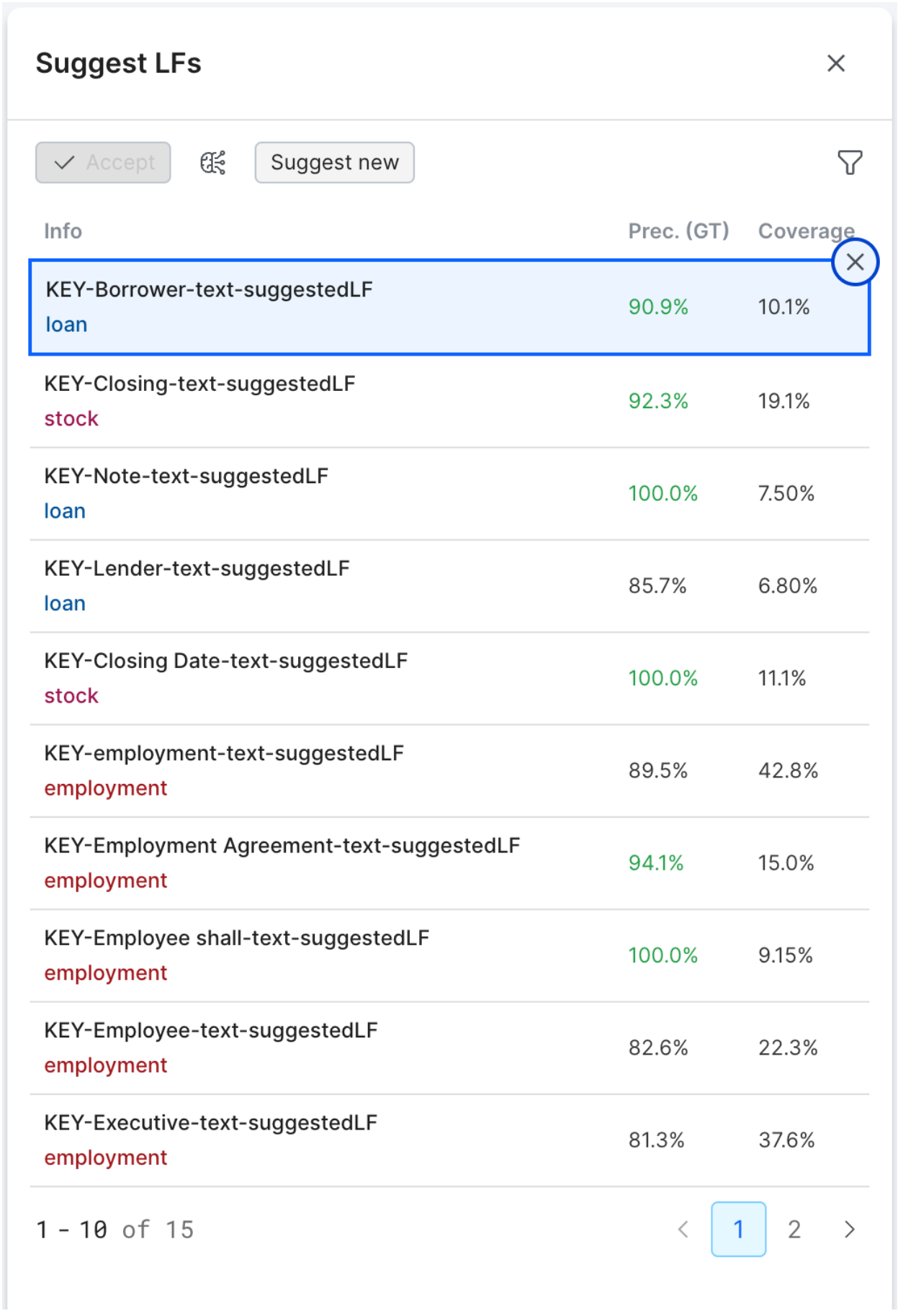
As you develop your model in **Develop (Studio)**, the **Labeling Functions** accordion provides a range of tools to view and manage your Labeling Functions.

There are 4 types of Labeling Functions (LFs) in Snorkel Flow:

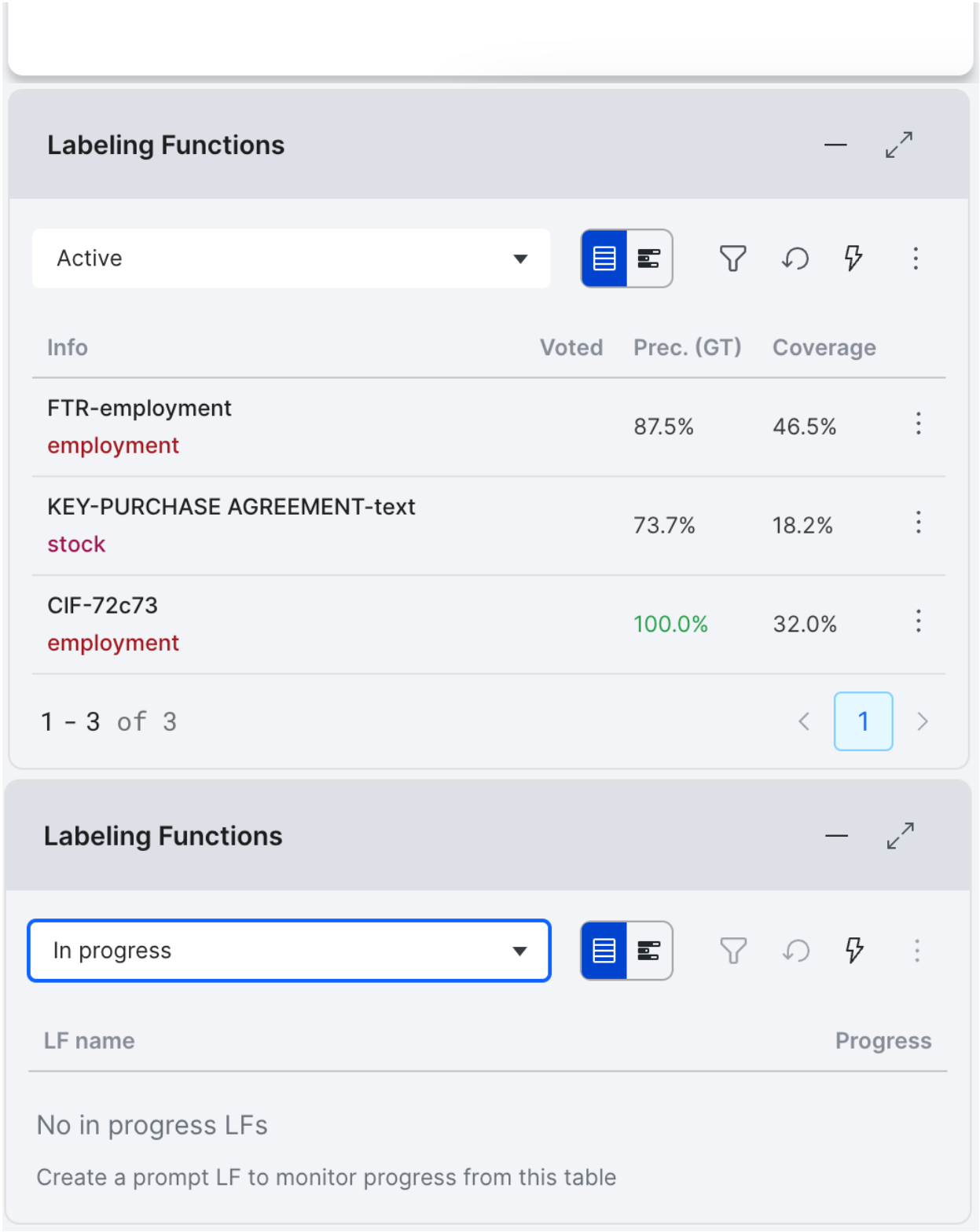
- **Active**: The set of LFs actively used in labeling. When creating a new training set this is the group of LFs that will be packaged for model training.
- **Suggested**: LFs suggested by the Snorkel Flow based on your data. You can modify the parameters of these suggestions as needed.
- **Inactive**: LFs that you have archived from the active list.
- **In Progress**: LFs that are being generated and have not yet been activated.

All LFs are displayed in a table format, with different tables for each type.

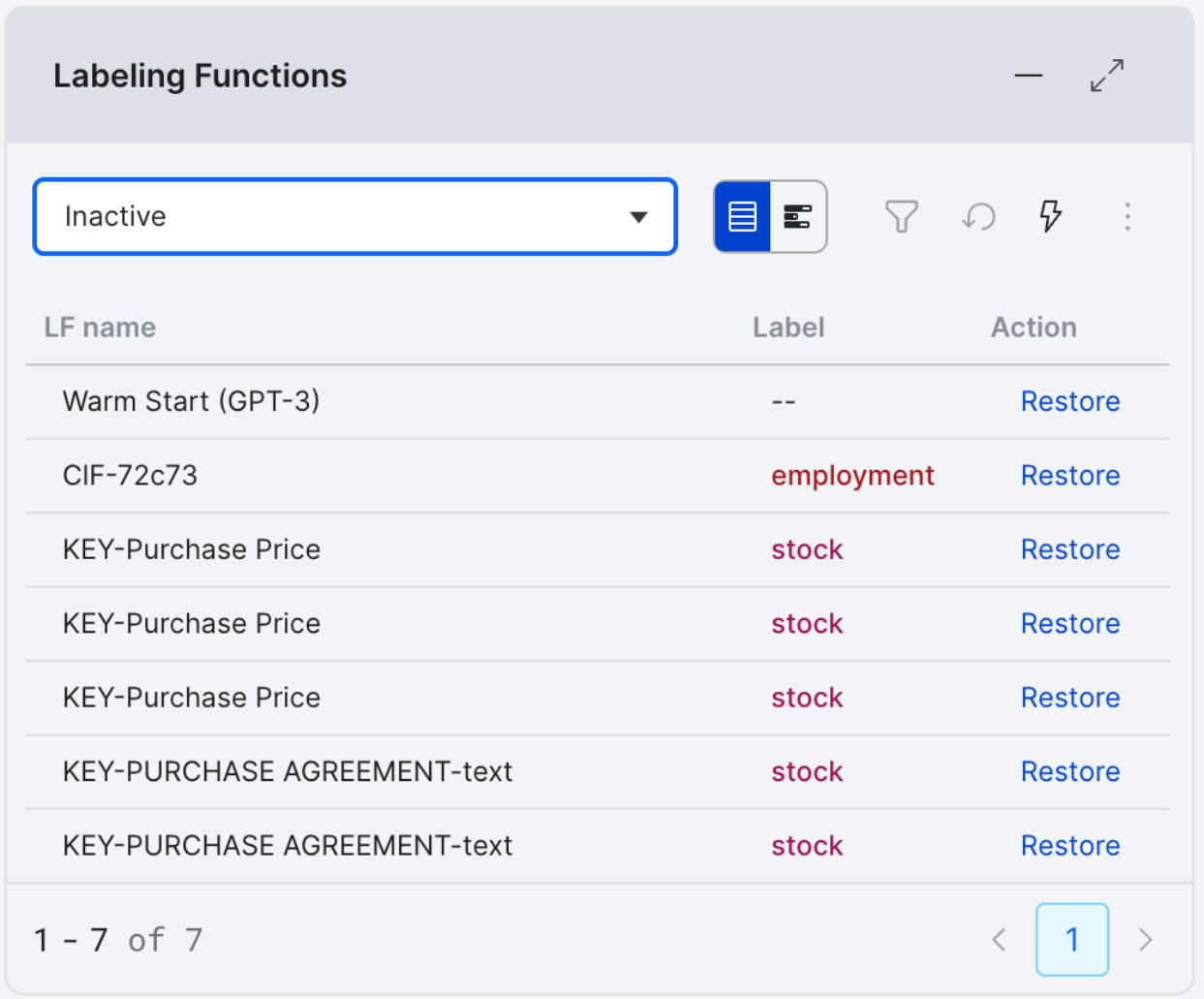
Copyright © 2025 Snorkel AI, Inc. All rights reserved



Copyright © 2025 Snorkel AI, Inc. All rights reserved



Copyright © 2025 Snorkel AI, Inc. All rights reserved



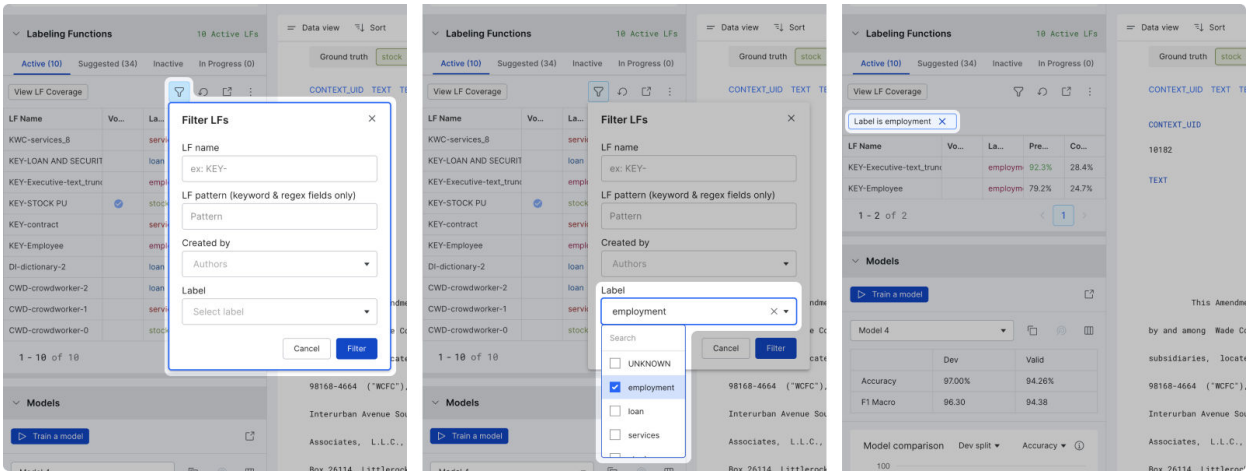
The following section explains most commonly used functions in the Labeling Function tables:

- Filter LFs
- Sort LFs
- View active LF details
- Rename LFs

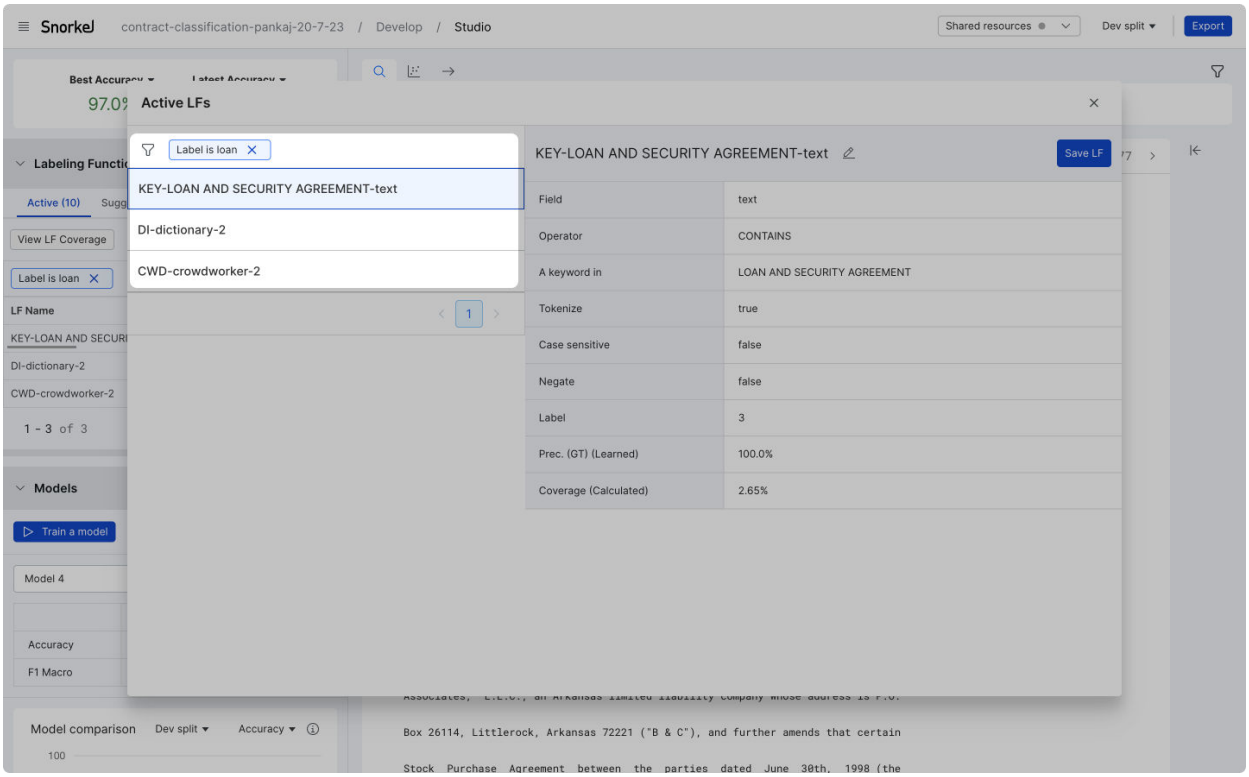
Filter LFs

To filter LFs in the table, click on the **filter** icon above the table. You can filter by LF name, pattern, author, and label. Once applied, that filter appears above the table, and the table itself shows only the LFs that match the filter.

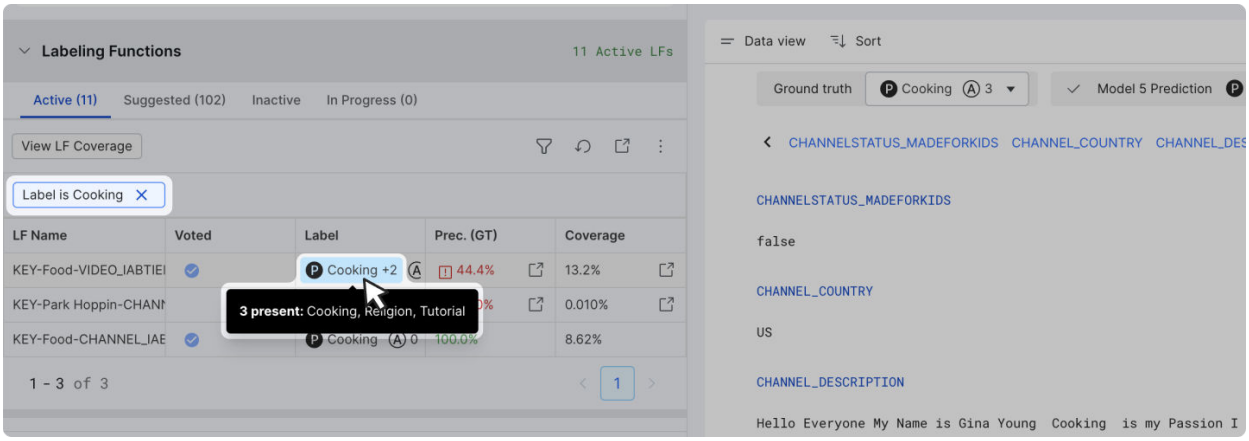
Copyright © 2025 Snorkel AI, Inc. All rights reserved



Filter LF is available in the **Active** and **Suggested** tables. It is also available in the LF details modal.



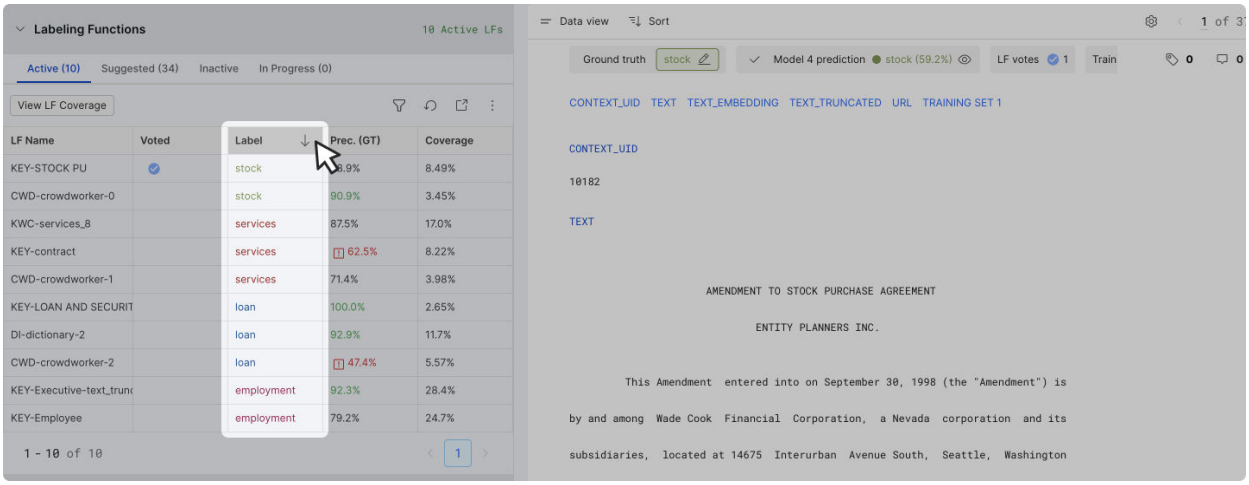
When filtering by label for multi-label apps, we will show that label first followed by a count of other labels.



Copyright © 2025 Snorkel AI, Inc. All rights reserved

Sort LFs

Users can sort LFs, including multi-label LFs, by clicking on the "Label" header in the LF table.

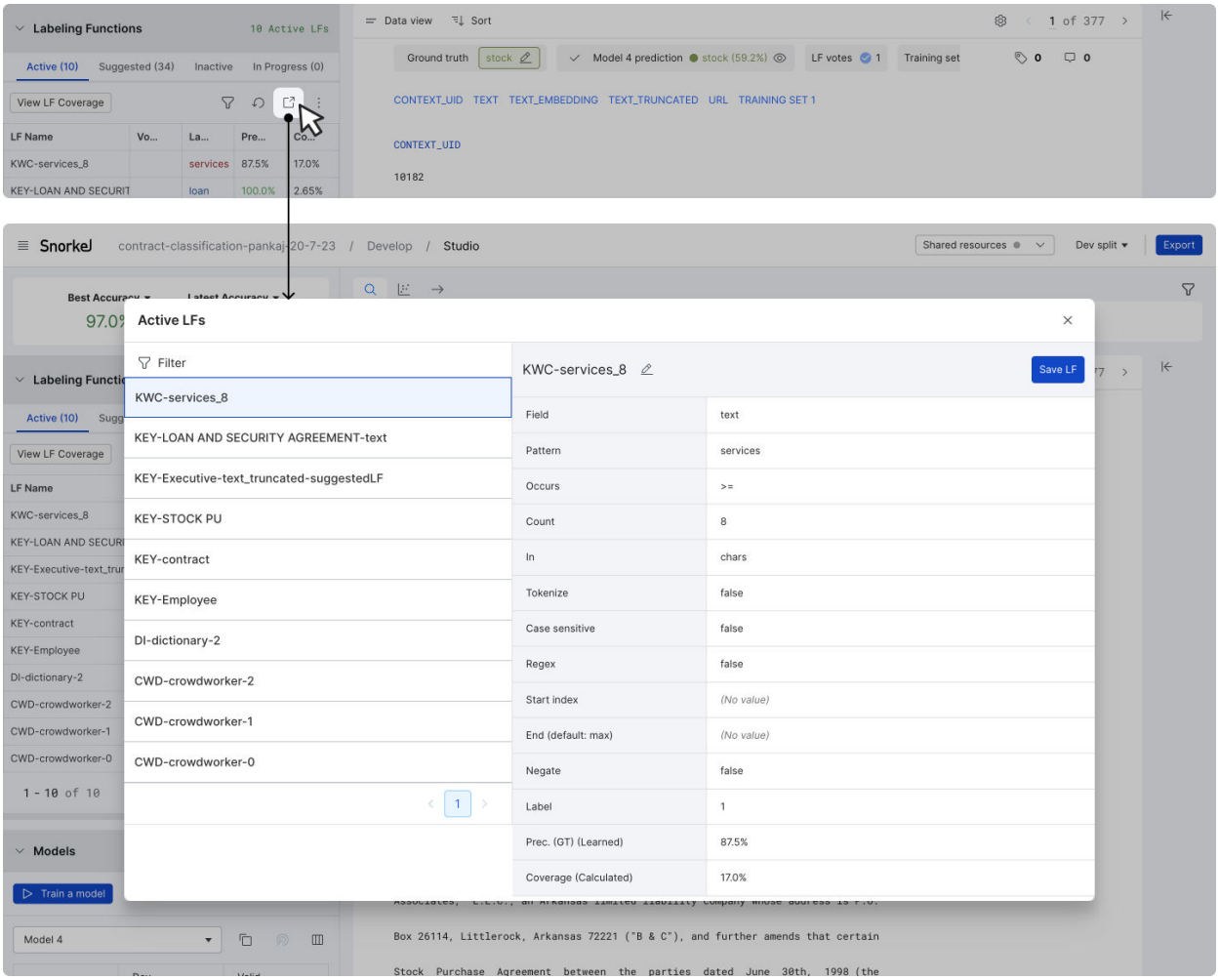


Multi-label LFs are sorted by the voted labels alphabetically. If an LF votes on more than one label, we will first look at the present labels alphabetically, and then at absent labels alphabetically to determine the order of the multi-label LFs.

View active LF details

Click on the launch icon in the table to open a model where details of each LF can be reviewed in a table format.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

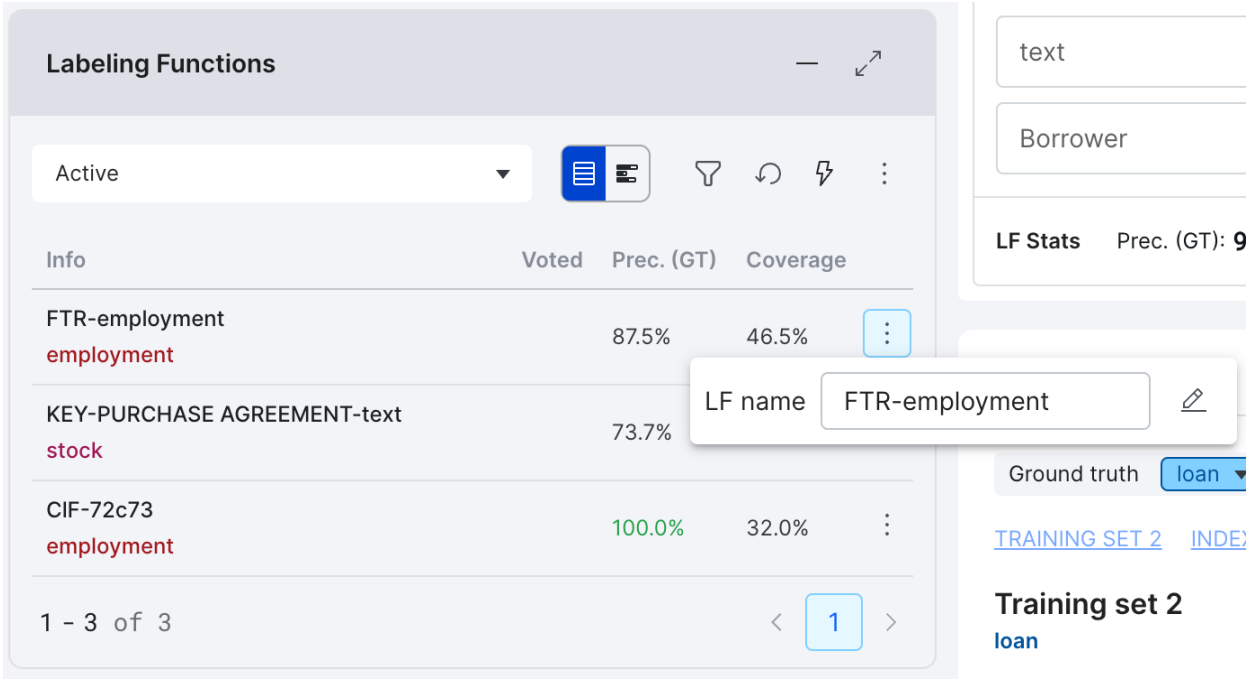


Rename LFs

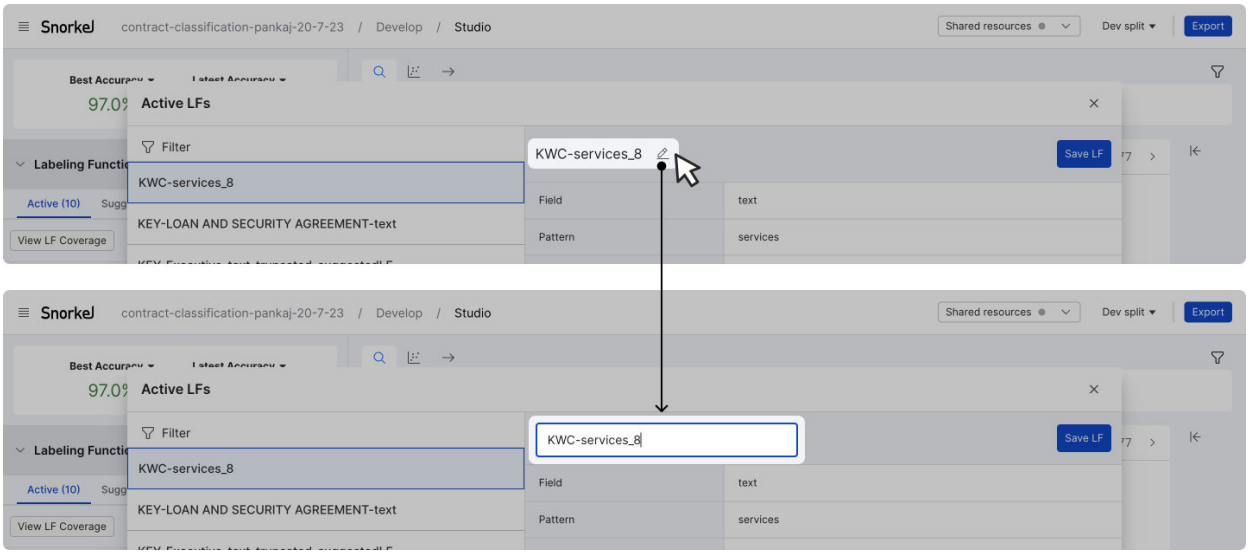
There are multiple locations where you can rename an LF: in the LF table, in the active LF modal, and in the LF composer.

In the LF table, click on an LF (LF edit mode), click on the overflow icon and select **Rename this LF** in the menu. Keyboard **Enter** to submit edits.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

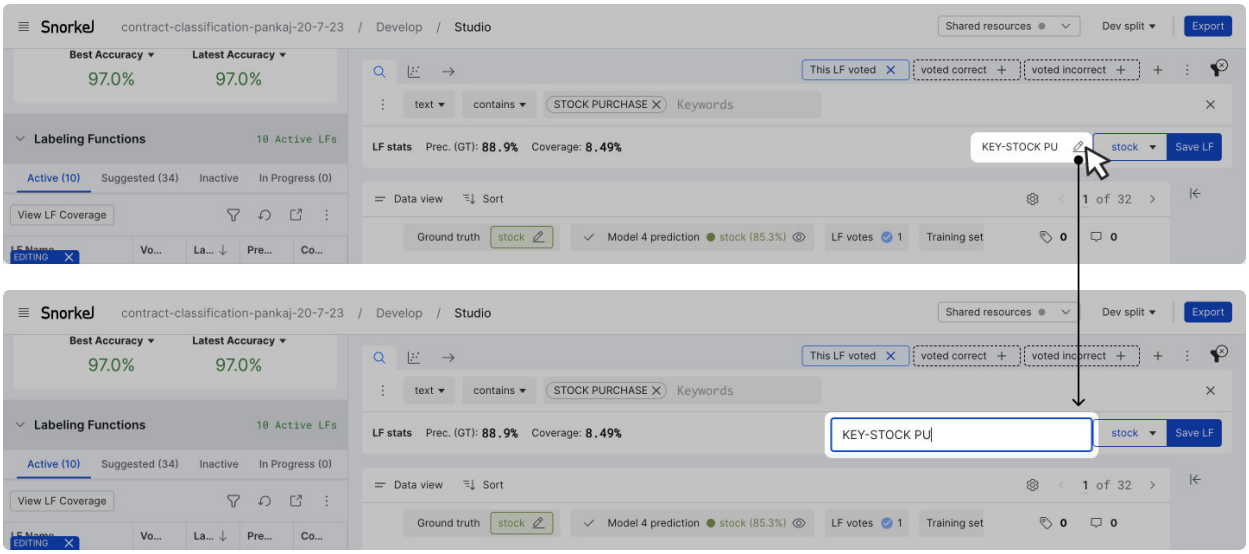


In the active LF modal, click on the **pencil (edit)** icon to enter name edit mode. Keyboard **Enter** to submit edits.



In LF edit mode, go to the LF composer and click on the **pencil (edit)** icon to enter name edit mode. Keyboard **Enter** to submit edits.

Copyright © 2025 Snorkel AI, Inc. All rights reserved



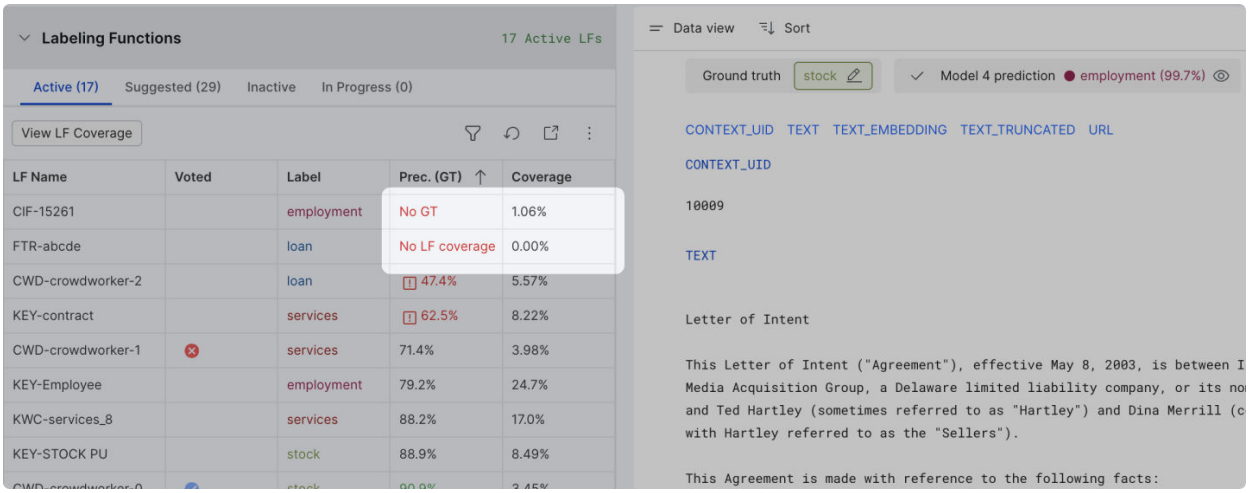
LF metrics

Single label applications

Both the LF table and the LF composer displays Prec. (GT) and Recall (GT) for each LF by default. Numbers above 90.0% is colored green (considered high); numbers below 70.0% is colored red ewith a warning icon (considered low).

Prec. (GT) and Recall (GT) values may not be numerics in the following conditions:

- **No LF coverage**: when a LF does not vote any data points in the current split, Prec. (GT) and Recall (GT) cannot be calculated and reads **No LF coverage**.
- **No GT**: when none of the data point a LF voted on has any Ground Truth, Prec. (GT) and Recall (GT) cannot be calculated and reads **No GT**.



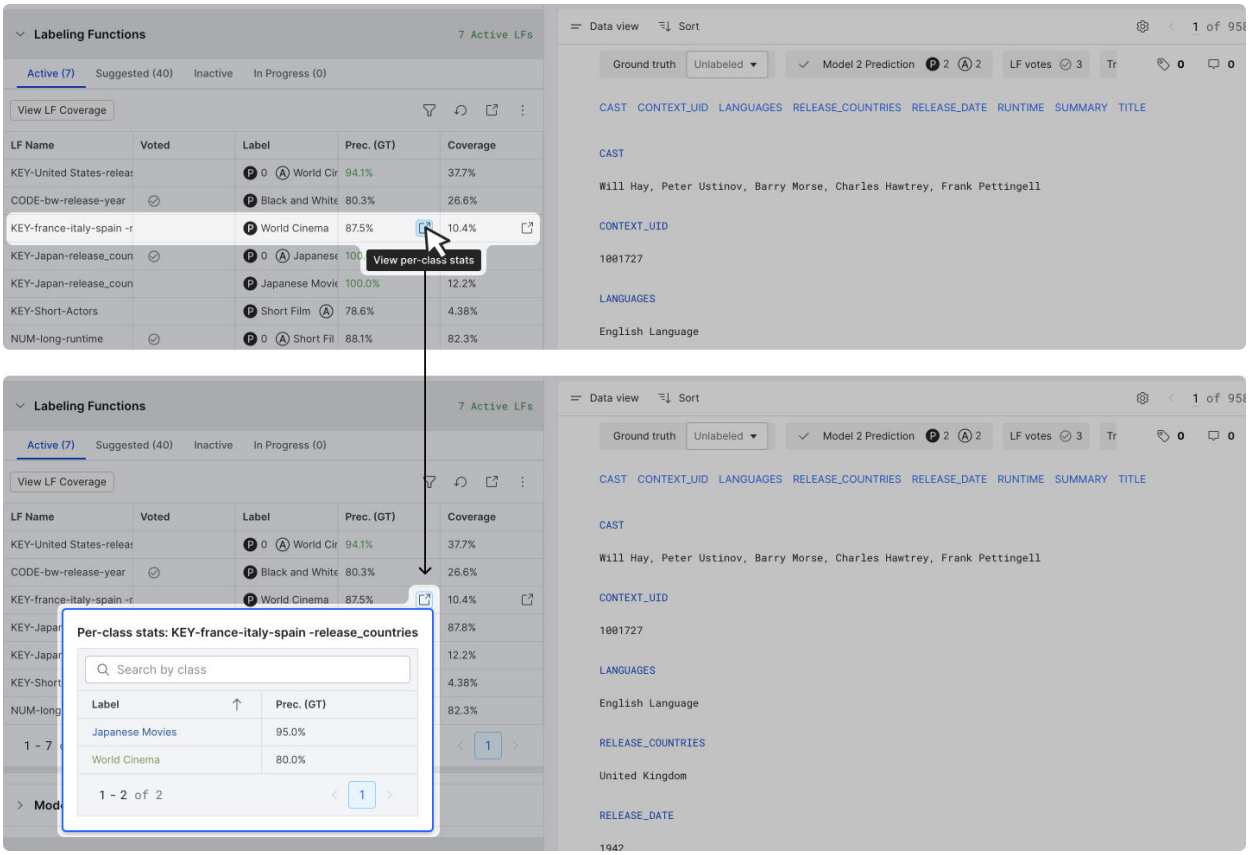
Multi-label applications: standard & per-class metrics

In multi-label applications, two types of metrics are available:

- **Standard metrics**: in a multi-label setting, LF metrics are displayed in the macro-averaged form (i.e. average of per-class metrics over all classes). We only calculate these metrics over points where neither the Ground Truth nor the prediction are `ABSTAIN`s.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

- **Per-class metrics:** when a LF votes on multiple classes, click on the **launch** icon to see the metrics breakdown per class.

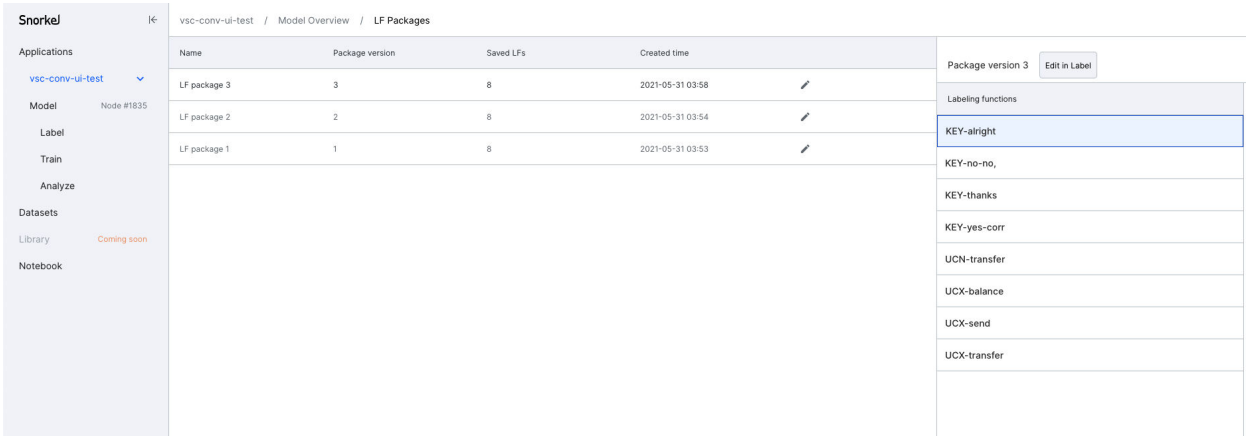


Copyright © 2025 Snorkel AI, Inc. All rights reserved

LF package

Manage LF packages

In the LF Packages pane, you can look at the Labeling Function (LF) packages that you have created from **Develop (Studio)**.



The right pane shows the learned accuracies of the different LFs in the latest LF package that has been used to label a data split.

In case you want to revert to an old LF Package, you can click **Restore LF package [n]** to overwrite the current set of LFs in **Develop (Studio)**.

> ⚠ WARNING
>
> If you click **Restore LF package**, your current set of LFs will be lost. We recommend that you create an LF Package with the current set of LFs before loading a new version.

Edit package

You can click on the pencil symbol to edit the name of an LF package.

Apply labeling functions

Click on **Label** dataset to have your data labeled using any existing LF package you have created.

- **Package Name**: Name of the LF package.
- **Fraction Cores to Use** and **Partition Size (MB)** are variables for Dask data parallelism. This represents the fraction of cores to use to train the model. For example, if your machine runs on 8 cores and the fraction is 0.5, you can train your model on `8 * 0.5 = 4` cores in parallel.
- **Data Start Date** and **Data End Date**: If these two values are entered, label only the data sources whose datestamps are within the range. We recommend using the date format `YYYY-MM-DD`.
- **Tuning splits**: The split you want to tune the hyperparameters for your model. Currently, Snorkel Flow isn't supporting hyperparameters tuning, so ignore this option.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

Code LFs

In Snorkel Flow, labeling functions (LFs) help automate the process of labeling data. Snorkel Flow provides modular building blocks, like AND and OR logic blocks, to create many types of LFs. However, in cases where your labeling logic isn't fully captured by the available LF builders, you can write custom Python functions.

These custom LFs, referred to as code LFs, allow you to directly define labeling logic and integrate it into your project.

Here is an example code LF. Use `@labeling_function()` to define your LF, and `sf.add_code_lf()` to add the LF to the model node for your application.

```
import snorkelflow.client as sf
from snorkel.labeling.lf import labeling_function

@labeling_function()
def my_code_lf(x):
    if "drug" in x.body:
        return "SPAM"
    return "UNKNOWN"

# Save to a model node
sf.add_code_lf(node, my_code_lf)
```

Copyright © 2025 Snorkel AI, Inc. All rights reserved

Custom LFs

Custom LF builders via python SDK

This section describes how to register a custom LF builder using Jupyter Notebook, and then use that LF builder in **Develop (Studio)**Studio.

Defining the classes

To develop a custom LF builder, we need to define 2 classes; a custom LF template and a custom LF template schema. To define a custom LF template schema, we write a class via the SDK that inherits from the `TemplateSchema` class, in which we define the input fields, their data types, optional default values, and optional validators. This is similar to a pydantic BaseModel class.

For example, if we want to define a custom LF builder that can label based on a numeric field value comparison (say, "if field num_images in the dataset is > 4, then label as spam"), we will need 3 inputs to the LF: the data field (e.g., num_images), the comparison operator (e.g., >), and the value to compare (e.g., 4).

Along with these, a doc string in this custom LF template schema class can optionally specify UI rendering in **Develop (Studio)**. For example, the order in which the inputs should render can be specified by the docstring. Input fields should be enclosed in square brackets. If input names has a word field in it, a dropdown list with all available data fields is rendered in the UI. If only integer fields should be rendered, specify (int) beside the name, similarly (string) for string fields. See the doc string Field [field](int) is [operator] [value] used below.

```
from templates.utils import TemplateSchema
from pydantic import validator
class CustomNumericTemplateSchema(TemplateSchema):
    '''
    Field [field](int) is [operator] [value]
    '''
    field: str
    operator: str = "="
    value: float

    @validator("operator")
    def check_operator(cls, op: str) -> str:
        if op not in [">", "<", "="]:
            raise ValueError(f"Invalid operator {op}. Should be one of the [
>, <, =]")
        return op
```

Validator defined here will check for valid operator input. The doc string in the above example shows up as follows:

Copyright © 2025 Snorkel AI, Inc. All rights reserved

To define a custom LF template, we write a class via the SDK that inherits from the `Template` class. We need to define several necessary fields and methods as part of this class: `template_type`, `abbreviation`, `description`, `menu_type`, `docs_link`, `template_schema`, `__init__`, `check`.

`template_type` is a unique identifier for LF builder and should always start with custom

`abbreviation` is used to distinguish between LF builders in **Develop (Studio)** easily.

`description` to specify a description for the LF builder. Shown in the UI help tool tip.

`menu_type` is a dictionary with three required fields: `name`, `value`, and `category`. `Value` should be a unique `template_type`. `Category` is used to group LF builders in **Develop (Studio)** and filter by application type.

`docs_link` is an optional user guide link for the LF builder. This is linked in the help tool tip

`template_schema` is the template schema class name that should be used with this LF builder.

`__init__` is a required method that gets the TemplateConfig input which is a dict of all the inputs defined in the TemplateSchema. Fetch this input data so it can be used in the check method.

`check` is a required method that should return List[Tuple[int, int]] for sequence tagging labeling (span votes with span_start and span_end. Example : [(1, 5), (6, 10)])

or a boolean for all other kinds of labeling (true if the LF should vote for the label, false otherwise). See the example below:

Copyright © 2025 Snorkel AI, Inc. All rights reserved

```
from templates.template import Template, TemplateConfig
import pandas as pd
class CustomNumericTemplate(Template):
    """Custom LF Template based on numerical comparisons"""

    template_type = "custom_numeric"
    abbreviation = "NUM"
    description = "If [field] has a numeric value that is [>, =, etc.]
[value], then label."
    menu_type = {
        "name": "Custom Numeric Builder",
        "value": template_type,
        "category": ["custom"],
    }
    docs_link = (
        "/user_guide/reference/label/lf_builders.html#"
        + "numeric-builder-numerical-lfs"
    )
    template_schema = "CustomNumericTemplateSchema"

    def __init__(self, template_config: TemplateConfig) -> None:
        """LF Template based on numeric comparisons.

        Heuristic:
        "If x[field] [OPERATOR] [value], return True"
        """
        self._field = template_config["field"]
        self.op_str = template_config["operator"]
        self._value = template_config["value"]

    def check(self, x: pd.Series) -> bool:
        field_value = x[self._field]
        if self.op_str == "=":
            return field_value == self._value
        elif self.op_str == ">":
            return field_value > self._value
        elif self.op_str == "<":
            return field_value < self._value
        return False
```

Registering the class

After defining the LF Template and LF Template schema class, we are now ready to register them for use in Snorkel Flow.

We do so by calling the `snorkelflow.client.add_lf_template_class` SDK method and pass in our class definitions `CustomNumericTemplate` and `CustomNumericTemplateSchema`.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

```
import snorkelflow.client as sf
sf.add_lf_template_class(
   CustomNumericTemplate, CustomNumericTemplateSchema,
)
```

Copyright © 2025 Snorkel AI, Inc. All rights reserved

Multi-polar LFs

This page walks through how to create multi-polar LFs and multi-polar crowdworker LFs, and explains when you should take advantage of this flexibility to create highly complex LFs.

Multi-polar LFs can be used to incorporate advanced logic into LFs, allowing an individual LF to vote differently based on different conditions. These LFs can be developed in the in-platform notebook.

What is a multi-polar LF?

The key difference between multi-polar LFs and a typical uni-polar LF is the ability to vote differently based on different conditions:

- **Uni-polar** labeling functions can only output a single label from a given label space.
- **Multi-polar** labeling functions can output any label from a given label space.

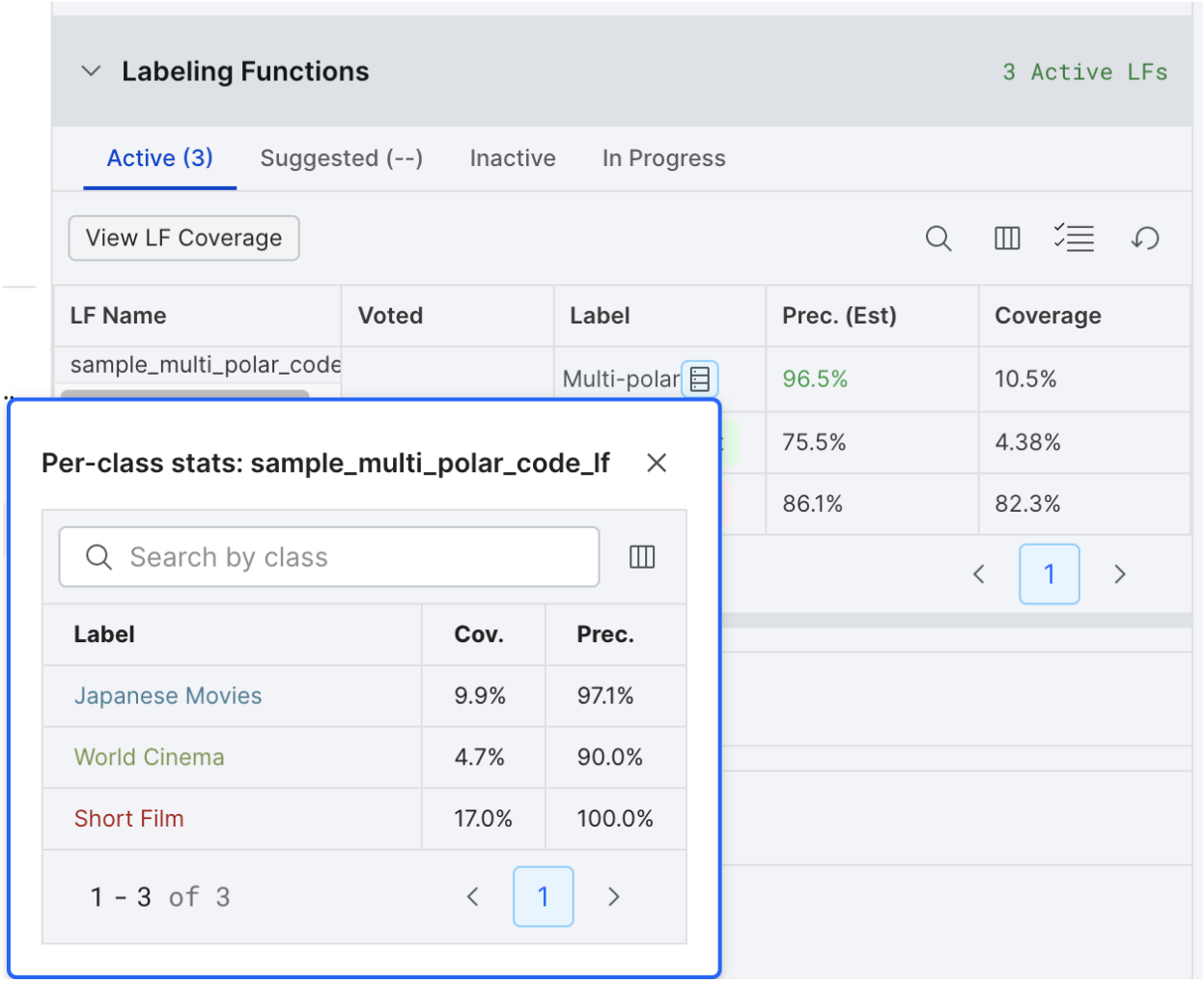
Here are some examples of multi-polar LFs:

- Code that evaluates the similarity between a data point and each class in a consistent way, and then votes for the "most similar" class.
- Crowdworker labels from the same source/vendor.

One of the objectives of multi-polar LFs is to reduce the number of LFs that are needed for high-cardinality use cases (e.g., a classification application with a large number of possible classes).

Once you create a multi-polar LF, you can view per-class statistics in the LF summary pane. Click the icon on the right side of the Label column to view the coverage and precision metrics by class name.

Copyright © 2025 Snorkel AI, Inc. All rights reserved



Why use multi-polar LFs?

Multi-polar LFs are generally more performant and provide more accurate measurement than creating a larger number of separate uni-polar LFs.

Performance

Some labeling functions are naturally multi-polar, for example, crowdworker LFs, model-based LFs, and embedding-based LFs. The cardinality for these functions tends to be high, which causes usability and performance issues if you're restricted to using only uni-polar labeling functions.

For single label applications, being limited to purely uni-polar labeling functions means creating as many LFs as the cardinality of the application. This means that for an application with a cardinality of over 1000, 1000 LFs would be necessary. Multi-polar LFs reduce the amount of labels that are needed and therefore increase performance.

Accurate Measurement

Using multi-polar LFs provides one consolidated measure of precision and coverage.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

How to create a multi-polar LF

You can create multi-polar LFs using the in-platform notebook. Use `@labeling_function()` to define your LF, and `sf.add_code_lf()` to add the LF to the model node for your application.

Here is an example of a multi-polar LF for a multi-label classification application:

```
@labeling_function()
def sample_multi_polar_code_lf(x):
    if x.release_countries is not None and "Hong Kong" in
x.release_countries:
        return {
            'World Cinema': "PRESENT",
            'Japanese Movies': "ABSTAIN",
            'Short Film': "ABSTAIN",
            'Black and White': "ABSTAIN"
        }
    elif x.languages is not None and "Japanese" in x.languages:
             return {
            'World Cinema': "ABSTAIN",
            'Japanese Movies': "PRESENT",
            'Short Film': "ABSTAIN",
            'Black and White': "ABSTAIN"
        }
    elif x.runtime < 50:
            return {
            'World Cinema': "ABSTAIN",
            'Japanese Movies': "ABSTAIN",
            'Short Film': "PRESENT",
            'Black and White': "ABSTAIN"
        }
    return {
            'World Cinema': "ABSTAIN",
            'Japanese Movies': "ABSTAIN",
            'Short Film': "ABSTAIN",
            'Black and White': "ABSTAIN"
        }

lf = sf.add_code_lf(node, sample_multi_polar_code_lf, is_multipolar=True)
```

Multi-polar crowdworker LFs

The upload format for multi-polar crowdworkers supports input for multiple labels at a time. Compared to the standard crowdworker builder, the multi-polar crowdworker allows for both single-label and multi-label inputs.

Here is an example CSV file of a multi-polar crowdworker for a multi-label classification application:

Copyright © 2025 Snorkel AI, Inc. All rights reserved

```
uid,label
doc::10009844,"{""Black and White"": ""PRESENT"", ""Japanese Movies"":
""ABSTAIN"", ""Short Film"": ""ABSTAIN"", ""World Cinema"": ""ABSTAIN""}"
doc::10016997,"{""Black and White"": ""ABSTAIN"", ""Japanese Movies"":
""ABSENT"", ""Short Film"": ""PRESENT"", ""World Cinema"": ""ABSTAIN""}"
doc::10004664,"{""Black and White"": ""ABSTAIN"", ""Japanese Movies"":
""ABSTAIN"", ""Short Film"": ""ABSTAIN"", ""World Cinema"": ""ABSTAIN""}"
```

Note

The CSV file does not have to cover all `x_uids`. If an `x_uid` is not covered, a crowdworker LF votes abstain. For example, `doc::1000708` is not included in the CSV file above. Therefore the LF votes abstain for all ground truth classes.

For multi-label applications, the label of any row should be a serialized dictionary with all ground truth classes.

- `doc::10009844,"{""Black and White"": ""PRESENT""}"` is not a valid form. You cannot just say that one of the classes is present.
- `doc::10009844,"{""Black and White"": ""PRESENT"", ""Japanese Movies"": ""ABSTAIN"", ""Short Film"": ""ABSTAIN"", ""World Cinema"": ""ABSTAIN""}"` is a valid form. All ground truth classes are specified in the label.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

Suggested LFs

Introduction to autosuggest in Snorkel Flow

This article introduces Suggested labeling functions (LFs) and provides a reference for all Suggested LF types. It explains what types of Suggested LFs are useful and available, based on your uploaded data. Suggested LFs bootstrap your data development by suggesting LFs automatically based on your uploaded data. They can help you quickly get started with adding LFs after you upload a new dataset. This guide walks through automated strategies and how best to incorporate them into your iterative, data-centric development process.

What is autosuggest?

Autosuggest within Snorkel Flow can be divided into two broad categories: few-shot and zero-shot learning strategies.

- **Few-shot Learning**: Snorkel Flow expects at least some labeled documents per class to recommend high-quality LFs to the user. These labeled documents or data points must be available within your train sample (i.e., dev set).
- **Zero-shot Learning**: No labeled data is required. These strategies rely on the class names that are incorporated into a Foundation Model (FM) or a simple pattern-based lookup in select fields. For more information about FM support in Snorkel Flow, see Foundation model suite.

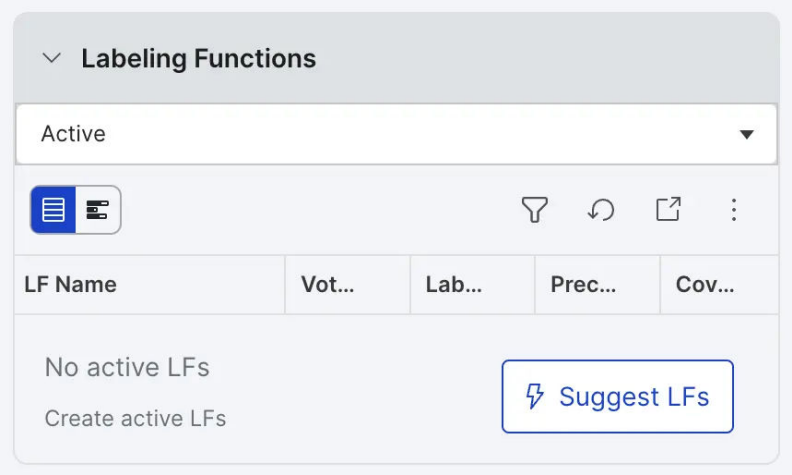
Specific strategies for both categories are outlined in the Strategies section below.

Where can I find autosuggest?

If you don't yet have any LFs in your application, then you can find the **Suggest LFs** button in the **Active** tab of the LFs section.

If you already have some active LFs in your application, then you must navigate to the **Suggested** tab of the LFs section to find the **Suggest new** button.

Copyright © 2025 Snorkel AI, Inc. All rights reserved



When to use autosuggest?

Each application template has a default, “push-based” suggestion strategy that runs when you load your application in **Develop (Studio)**. This assumes that your application has at least some labeled data in your dev split (a sample of the train split). Once the default LF suggestion strategy has returned some Suggested LFs, they will remain available for review. If you’d like to refresh the suggestions, you may generate them manually by selecting the suggestion strategy and options from the **Suggested** pane.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

Copyright © 2025 Snorkel AI, Inc. All rights reserved

*Figure 3: LF suggestions are populated below the **Suggested** pane.*

Once generated, you can review and accept or reject the LFs. You also have the option to automatically accept all suggested LFs from a given strategy.

How does Autosuggest work?

Most of the existing Autosuggest LF strategies are supervised approaches, meaning you need at least a few ground truth (GT) examples for each class labeled before you can use the approach. The more GT you have in your sample, the better the suggested LFs will be in terms of generalization. If you don't have enough GT, there's a chance that your LFs could overfit to your small labeled set. In addition, some strategies may require an operator or two in the DAG before you can employ the suggestion strategy. These operators featurize the data (e.g., add embeddings for a specific field) and cache them for later use.

See the Strategies section below to get more detail on each method.

- Text classification
- PDF classification (OCR and native PDF)
- Multi-label classification
- Text extraction
- Sequence tagging
- PDF extraction (hOCR and native PDF)
- Utterance / conversation classification

Autosuggest strategies by application template

Text classification

LF Suggestion Strategies	Input	Description
Simple Keyword	Text fields	Given an ngram range, Keyword LFs are generated from selected text fields that are tokenized (case sensitive, English stopwords excluded) and run through a CountVectorizer. Keywords are then sorted by the count of the word in class X minus the count of the word in all other classes.
Simple Keyword/Numeric	Float or int fields	In addition to the Simple Keyword strategy, uses int or float-based fields, and calculates a linear threshold (e.g., x > 10) that achieves a minimum desired precision. **Note**: This is the default strategy for this application template.
TFIDF	Text fields	Generate a keyword LF that is based on the sorted logistic regression coefficients of absolute value. Model features are generated using TF-IDF.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

LF Suggestion Strategies	Input	Description
Keyword Count	Text fields	Similar to TFIDF strategy, but instead, a Keyword Count LF is created based on the frequency of the keyword.
Model-base generator (experimental)	Text or numeric fields	The standard options include a Logistic Regression model trained over text (TF-IDF) or numeric features.
Embedding Based	Embedding field	An SVM model is trained over embeddings computed over a text field. Requires adding the EmbeddingFeaturizer operator to the DAG.
Embedding Nearest Neighbor	Embedding field	For each datapoint we compute the nearest class based on the embedding of the class. We use cosine similarity to compute the nearest neighbor. Output of this strategy is a multipolar LF.
Hierarchical TF-IDF	Text fields	We compute top ngrams for each class using TFIDF in a hierarchical fashion and create a multipolar keyword based LF.
Search for Class Name (experimental)	Text fields	For each label does a case insensitive lookup of the name in the provided fields and returns the most commonly occurring label as the LF.

PDF classification (OCR and native PDF)

LF Suggestion Strategies	Input	Description
Simple Keyword	Text fields	Given an ngram range, Keyword LFs are generated from selected text fields that are tokenized (case sensitive, English stopwords excluded) and run through a CountVectorizer. Keywords are then sorted by the count of the word in class X minus the count of the word in all other classes.
Simple Keyword/Numeric	Float or int fields	In addition to the Simple Keyword strategy, uses int or float-based fields, and calculates a linear threshold (e.g., x > 10) that achieves a minimum desired precision. **Note**: This is the default strategy for this application template.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

LF Suggestion Strategies	Input	Description
TFIDF	Text fields	Generates a keyword LF that is based on the sorted logistic regression coefficients of absolute value. Model features are generated using TF-IDF.
Model-base generator (experimental)	Text or numeric fields	The standard options include a Logistic Regression model trained over text (TF-IDF) or numeric features.
Embedding Based:	Embedding field	An SVM model is trained over embeddings computed over a text field. Requires adding the EmbeddingFeaturizer operator to the DAG.
Embedding Nearest Neighbor	Embedding field	For each datapoint we compute the nearest class based on the embedding of the class. We use cosine similarity to compute the nearest neighbor. Output of this strategy is a multipolar LF.
Hierarchical TF-IDF	Text fields	We compute top ngrams for each class using TFIDF in a hierarchical fashion and create a multipolar keyword based LF.
Search for Class Name (experimental)	Text fields	For each label, does a case-insensitive lookup of the name in the provided fields and returns the most commonly occurring label as the LF.

Multi-label classification

LF Suggestion Strategies	Input	Description
Simple Keyword	Text fields	Given an ngram range, Keyword LFs are generated from selected text fields that are tokenized (case sensitive, English stopwords excluded) and run through a CountVectorizer. Keywords are then sorted by the count of the word in class X minus the count of the word in all other classes. **Note**: This is the default strategy for this application template.
Simple Keyword/Numeric	Float or int fields	In addition to the Simple Keyword strategy, uses int or float-based fields, and calculates a linear threshold (e.g., x > 10) that achieves a minimum desired precision.
TFIDF	Text fields	Generates a keyword LF that is based on the sorted logistic regression coefficients of absolute value. Model features are

Copyright © 2025 Snorkel AI, Inc. All rights reserved

LF Suggestion Strategies	Input	Description
		generated using TF-IDF.
Model-base generator (experimental)	Text or numeric fields	The standard options include a Logistic Regression model trained over text (TF-IDF) or numeric features.
Search for Class Name (experimental)	Text fields	For each label does a case-insensitive lookup of the name in the provided fields and returns the most commonly occurring label as the LF.

Text extraction

LF Suggestion Strategies	Input	Description
Simple Keyword	Text fields	Given an ngram range, Keyword LFs are generated from selected text fields that are tokenized (case sensitive, English stopwords excluded) and run through a CountVectorizer. Keywords are then sorted by the count of the word in class X minus the count of the word in all other classes.
Simple Keyword/Numeric	Float or int fields	In addition to the Simple Keyword strategy, uses int or float-based fields, and calculates a linear threshold (e.g., x > 10) that achieves a minimum desired precision.
TFIDF	Text fields	Generates a keyword LF that is based on the sorted logistic regression coefficients of absolute value. Model features are generated using TF-IDF.
Span-Based	span_text field	Generates keywords from span_text and its corresponding context windows. **Note**: This is the default strategy for this application template.
Model-base generator (experimental)	Text or numeric fields	The standard options include a Logistic Regression model trained over text (TF-IDF) or numeric features.
Search for Class Name (experimental)	Test fields	For each label does a case insensitive lookup of the name in the provided fields and returns the most commonly occurring label as the LF.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

Sequence tagging

LF Suggestion Strategies	Input	Description
Embedding-based	Embedding field	An SVM model is trained over embeddings computed over candidate spans. Requires adding two operators to the DAG: a featurizer that creates a new column with candidate spans (e.g., NounChunkSpanFeaturizer, VerbPhraseFeaturizer) and the EmbeddingCandidateFeaturizer that converts the candidate features into embeddings. **Note**: This is the default strategy for this application template.
Spacy NER / Property-based	Spacy generated "doc" field OR "NER" field from Spacy or BERT	Requires the SpacyPreprocessor to extract these entities. Must select the "doc" field OR "NER" fields from Spacy or BERT when using this strategy.

PDF extraction (hOCR and native PDF)

LF Suggestion Strategies	Input	Description
Simple Keyword	Text fields	Given an ngram range, Keyword LFs are generated from selected text fields that are tokenized (case sensitive, English stopwords excluded) and run through a CountVectorizer. Keywords are then sorted by the count of the word in class X minus the count of the word in all other classes. **Note**: This is the default strategy for this application template.
Simple Keyword/Numeric	Float or int fields	In addition to the Simple Keyword strategy, uses int or float-based fields, and calculates a linear threshold (e.g., x > 10) that achieves a minimum desired precision.
Model-base generator (experimental)	Text or numeric fields	The standard options include a Logistic Regression model trained over text (TF-IDF) or numeric features.
Search for Class Name (experimental)	Test fields	For each label does a case-insensitive lookup of the name in the provided fields and returns the most commonly occurring label as the LF.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

Utterance / conversation classification

LF Suggestion Strategies	Input	Description
Simple Keyword	Text fields	Given an ngram range, Keyword LFs are generated from selected text fields that are tokenized (case sensitive, English stopwords excluded) and run through a CountVectorizer. Keywords are then sorted by the count of the word in class X minus the count of the word in all other classes.
Simple Keyword/Numeric	Float or int fields	In addition to the Simple Keyword strategy, uses int or float-based fields, and calculates a linear threshold (e.g., x > 10) that achieves a minimum desired precision.
TFIDF	Text fields	Generates a keyword LF that is based on the sorted logistic regression coefficients of absolute value. Model features are generated using TF-IDF.
Model-base generator (experimental)	Text or numeric fields	The standard options include a Logistic Regression model trained over text (TF-IDF) or numeric features.
Embedding-based	Embedding field	An SVM model is trained over embeddings computed over a text field. Requires adding the EmbeddingFeaturizer operator to the DAG.
Embedding Nearest Neighbor	Embedding field	For each datapoint we compute the nearest class based on the embedding of the class. We use cosine similarity to compute the nearest neighbor. Output of this strategy is a multipolar LF.
Hierarchical TF-IDF	Text fields	We compute top ngrams for each class using TFIDF in a hierarchical fashion and create a multipolar keyword based LF.
Conversation-Based Input	Utterance field	Returns keywords based on the same strategy as the Keyword strategy, but the input field consists of joined utterances (no speaker names) **Note**: This is the default strategy for this application template.
Search for Class Name (experimental)	Test fields	For each label does a case-insensitive lookup of the name in the provided fields and returns the most commonly occurring label as the LF.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

Best practices

1. Label more data. Since most of the suggestion strategies require some GT labels, increasing the amount of labeled data and re-running the suggestion strategies can generate higher-quality LFs, even when running the same strategy.
2. Think carefully about which fields you feed to the suggestion strategies. For example, if you've preprocessed your data and truncated a text field, passing both the truncated and untruncated fields to the suggestion algorithm will likely return similar LFs.
3. Try to incorporate multiple different LF autosuggest strategies to increase the diversity of signals. Simple, interpretable techniques like Keyword or Numeric LFs combined with more sophisticated, model, or embedding-based strategies can improve overall performance.

FAQ

1. I created a new application and see zero suggestions in the **Suggested** pane. Why?

   There are three possibilities:
   1. Your application does not have enough labeled data per class.
   2. The LF suggestion strategy didn't identify any LFs that meet the minimum performance thresholds.
   3. The DAG does not have the required operators for suggested strategies to perform. If you manually re-run the LF suggestion strategy and no new LFs are identified, you will see a message stating this. To fix this, you can try a different LF suggestion strategy and/or label more GT and re-run the suggestion strategy.

2. What if suggested LFs are similar to active or other suggested LFs?

   On the one hand, it can still be helpful to manually ensure that the suggested LFs that you accept are unique and not duplicative with existing LFs. You can also do this proactively by being mindful of duplicative signals in the fields you feed to the automated suggestion strategies. On the other hand, it's not absolutely necessary to deduplicate similar LFs since the label model has the ability to differentiate between LF signals by examining their agreements and disagreements.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

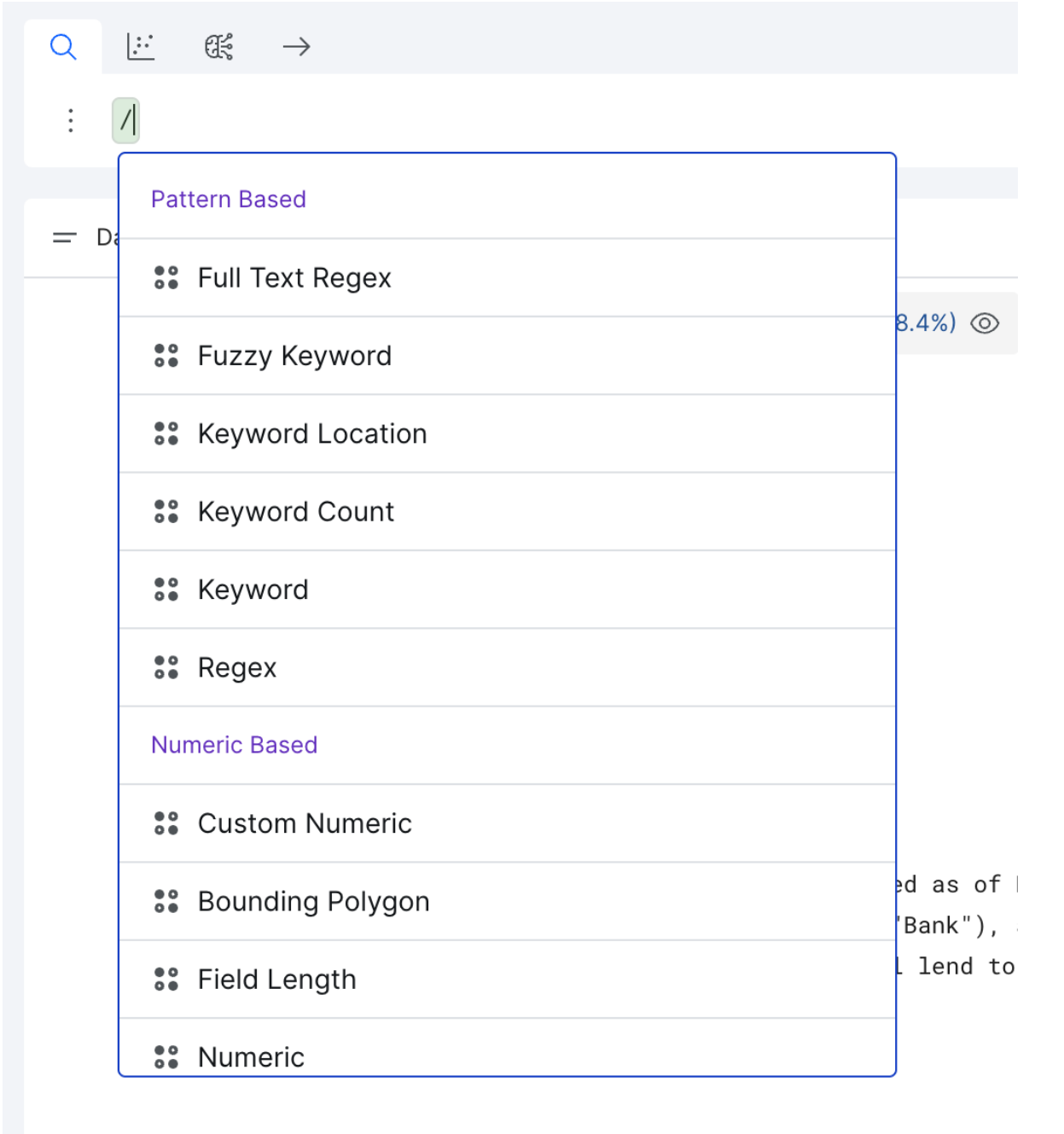
Introduction to search based LFs

Writing a labeling function (LF)

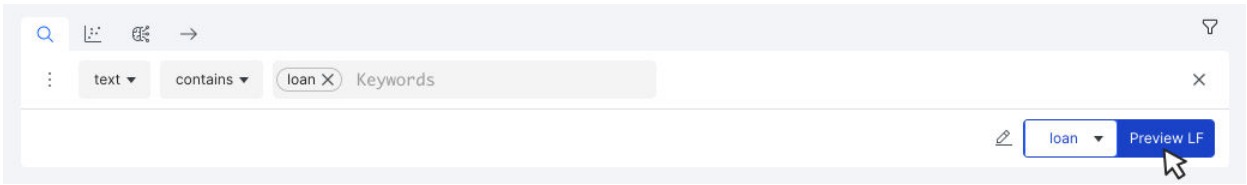
Navigate to the **Pattern based search workflow** LF composer tab and type anything into the search bar to start writing a labeling function. Each application comes with a list of recommended builders that you can choose from.

To view all builders available in the application you're working on, type **/** into the search bar and select a builder type.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

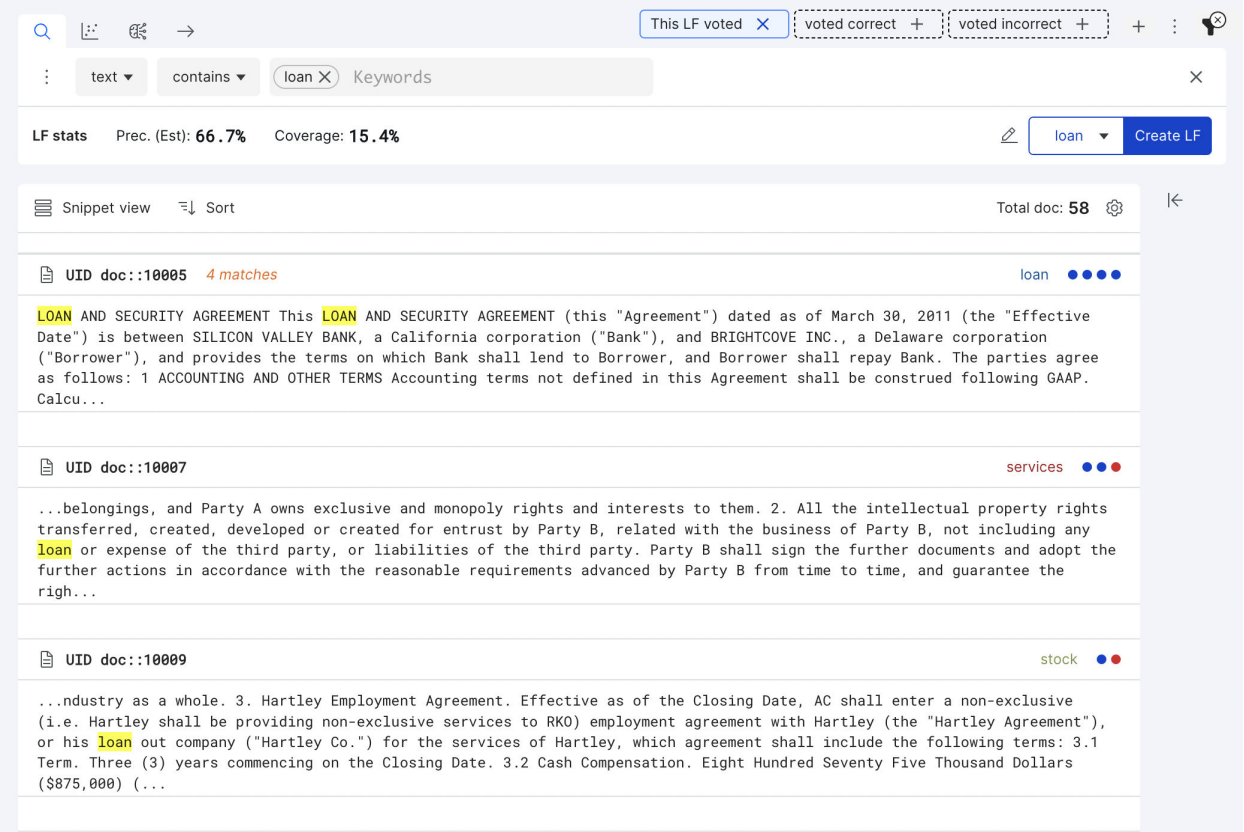


Click on **Preview LF** to preview your LF, located in the bottom right of the search bar.

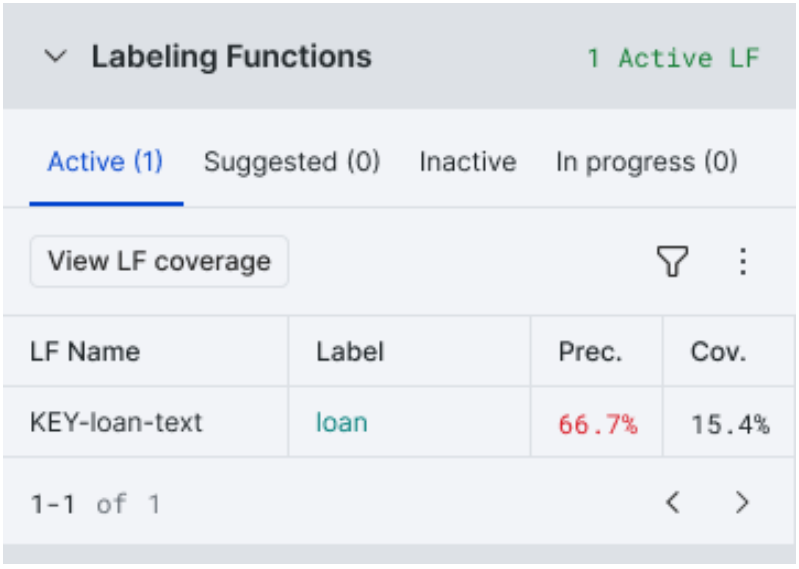


Preview LF will return results indicating where your draft LF will vote. The **Preview LF** button have now become the **Create LF** button.

Copyright © 2025 Snorkel AI, Inc. All rights reserved



Once you're done previewing and refining your draft LF, click **Create LF**. The search bar will clear, and your labeling function will appear in the **Active LFs** list under the LF summary section.



Understanding labeling function (LF) metrics

Preview LF metrics via LF composer

When previewing an LF you will see the following information displayed in LF composer:

- **Prec. (Est)**: The estimated precision of the previewed LF based on correct votes relative to any GT and other LF signals. See LF summary for details.

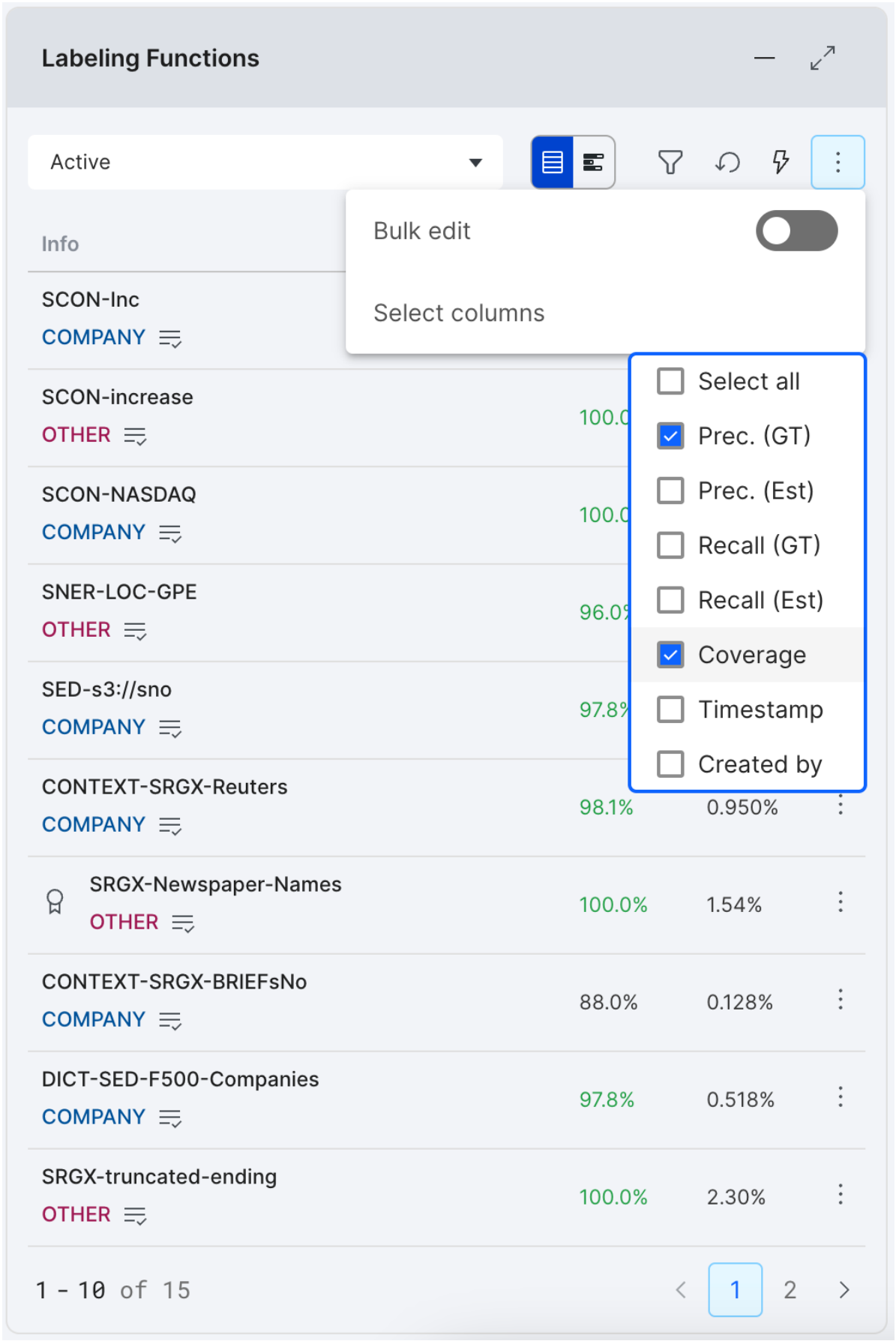
Copyright © 2025 Snorkel AI, Inc. All rights reserved

- **Coverage**: The percentage of data points this LF votes on.
- **voted correct** filter chip: Apply a filter to show examples that were correctly voted on by this LF.
- **voted incorrect** filter chip: Apply a filter to show examples that were incorrectly voted on by this LF.

LF summary metrics

The **Labeling Functions** summary pane contains the following metrics (on the current split) that can be accessed via the **Columns** icon in the top right corner:

Copyright © 2025 Snorkel AI, Inc. All rights reserved



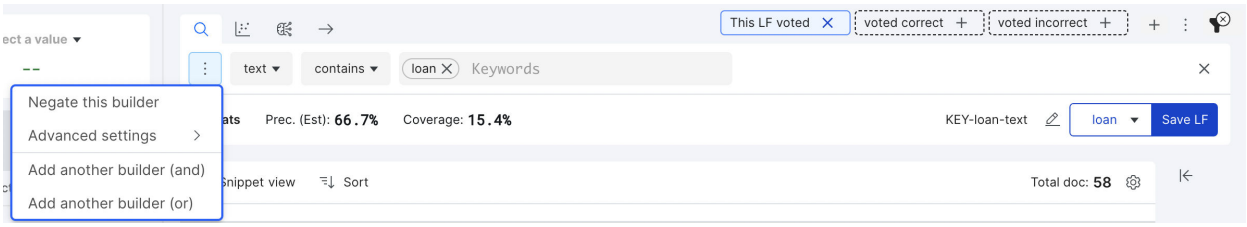
Copyright © 2025 Snorkel AI, Inc. All rights reserved

- **Labels**: The label assigned by the labeling function
- **Voted** (specific to the datapoint selected in Data Viewer tab):
  - Gray ✓ voted, but not GT exists
  - Green ✓ voted correctly based on assigned GT
  - Red X voted incorrectly based on assigned GT
- **Precision (GT)**: The (number of correct LF votes) / (total number of LF votes), based on the existing GT labels. This is only an estimate based on a potentially very small dev set. The learned precision Snorkel Flow will use to generate probabilistic labels for data points won't be calculated until a label package is created and can be viewed on the LF Packages page.
- **Precision (Estimated)**: The same as Precision (GT) but using estimated GT labels (based on LF votes) wherever manual GT labels do not exist. (Total LF votes for class in agreement with estimated GT labels) / (Total LF votes for class).
- **Recall (GT)**: The (number of correct LF votes) / (total number of examples in class), based on the existing GT labels.
- **Recall (Estimated)**: The same as Recall, but using estimated GT labels (based on LF votes) wherever manual GT labels do not exist. (Total LF votes for class in agreement with majority) / (Total majority votes for class).
- **Coverage**: The percentage of data points in the current split that this labeling function votes on.
- **Count (GT)**: The total number of LF votes generated by this LF on the current split.

Additional labeling function (LF) functionalities

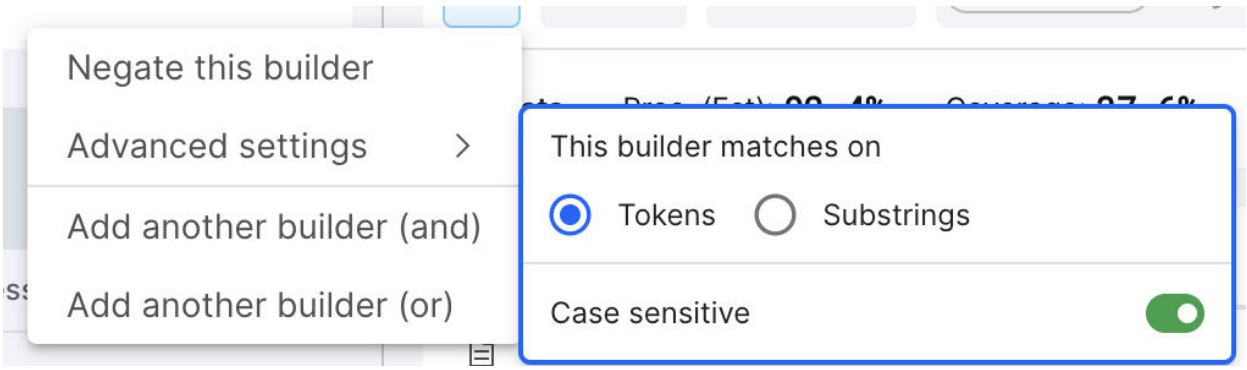
LF builder settings

Each LF will have additional configuration options available in the three-dot menu on the left side of the search bar. Clicking a active LF from the LF summary pane will also expose it in the search bar to access these options. These settings are "sticky"— they persist until for a specific user and application until they're changed.



- **Negate this builder**: Only votes on data points that don't match the specified condition.
- **Advanced settings**: Each builder has its own additional settings, and the most common advanced settings among them are:
  - **Case sensitive**: if enabled, the texts should be treated case-sensitive. By default, all texts are case-insensitive.
  - **Regex**: if enabled, the string you enter will be treated as regex.
- **Add another builder (and/or)**: enables you to add a new row to the search bar to combine different types of labeling function builders within one labeling function.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

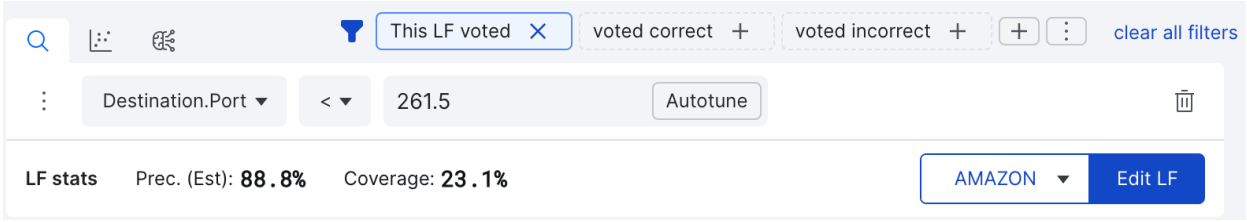


Keyword and Dictionary Builders have additional settings for how to match user-inputted words or phrases:

- **Tokens**: The builder will look to match the entire token (e.g., the entire word/phrase) and not vote on a label if the word or phrase appears as part of another complete word. Tokens are split on spaces.
- **Substring**: The builder will look to see if the word or phrase appears in the field, whether it's a substring of another word or not.

Advanced LF options

Some LFs support advanced interactions like "autotune" for numeric LFs or "refresh" for dictionary LFs. To access these advanced configuration options, click the available buttons from any LF builder template that supports these options (see Numerical LF builders as an example).



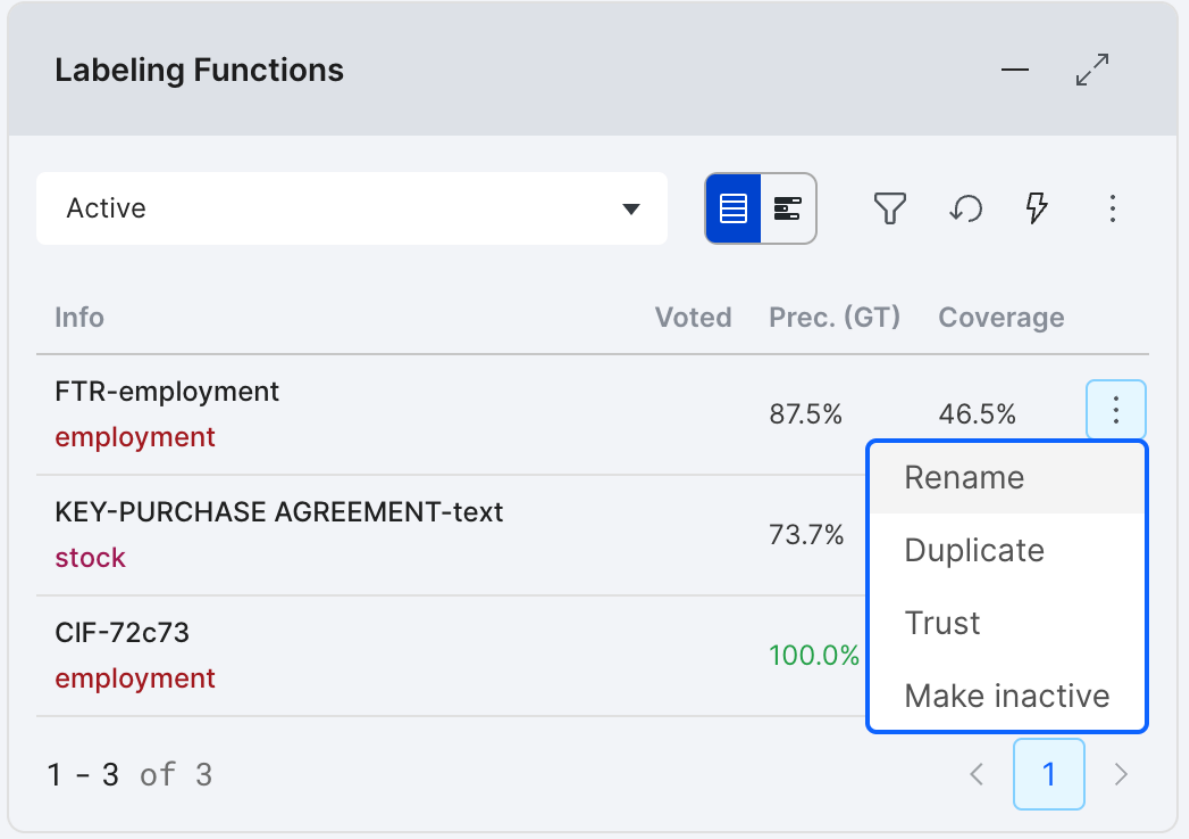
Phrase match highlighting

For pattern-based LF builders, the matching phrases will be highlighted in the current data point in the **Data view** tab for doc-based applications and under the **Span** tab for span-based applications. Only the Keyword, Regex, and Span Context builders currently support this.

LF settings

For each labeling function, there are a few settings available in the three-dot menu to the left of each saved LF row in the Active LF list.

Copyright © 2025 Snorkel AI, Inc. All rights reserved



- **Rename this LF**: Opens an edit field for you to rename a LF. Note that each LF name has to be unique.
- **Trust this LF**: Tells the labeling model to default vote on datapoints with the label specified by this LF.
- **Archive this LF**: Sends the LF to **Inactive LFs** tab.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

External resource LF builders

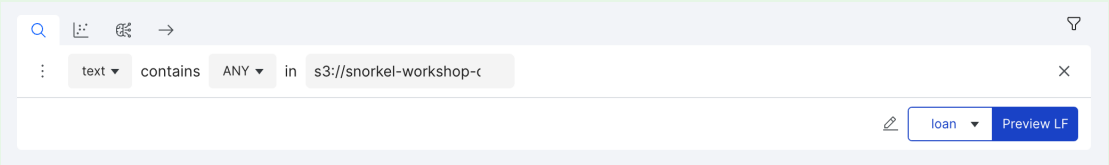
This article describes the set of external resource LF builders that are available for classification and information extraction applications.

Dictionary builder

Label data points that contain all, any, or none of the words from a dictionary, provided through a TXT or CSV file. If you have a CSV that contains words for multiple words classes, you can use the **Dictionary Generator**.

> **NOTE**
> We label data points where keywords in an external dictionary (e.g. a text file with a newline-separated list of loan terms) appear in the text.

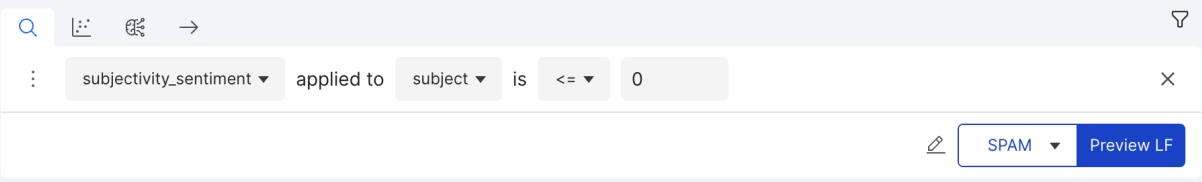
> **TIP**
> The location of the TXT or CSV file needs to be an S3 file.

> **TIP**
> 
>

External model builder

Label data points using the predictions from a machine-learning model, such as sentiment analysis models from TextBlob and NLTK.

> **NOTE**
> For the email-spam application, we find that many spam emails contain subjects that convey negative feelings. We label data points where the subject has a negative sentiment score as predicted by the external text analysis mode.



Crowdworker builder

Label data points using crowdworker annotations, as a new-line separated list of UIDs, each with `doc::` prefix.

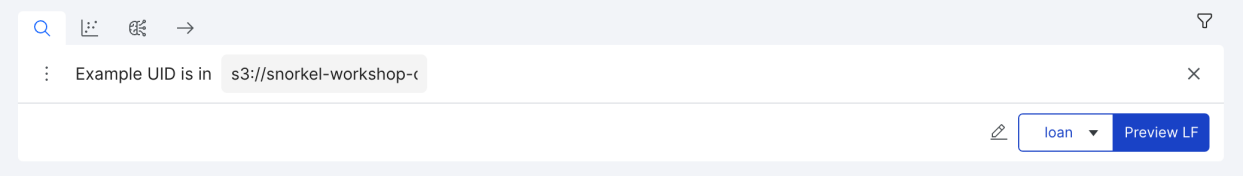
Copyright © 2025 Snorkel AI, Inc. All rights reserved

For example, say we want to annotate rows with UID `1`, `2` and `3` as label `X`. We will first create a file, in the format shown below, and upload it to S3.

```
doc::1
doc::2
doc::3
```

The crowd-worker builder will then take in a path to the file and label all the examples in the file as `X`.

> **NOTE**
> The current form supports file input for a single label at a time.



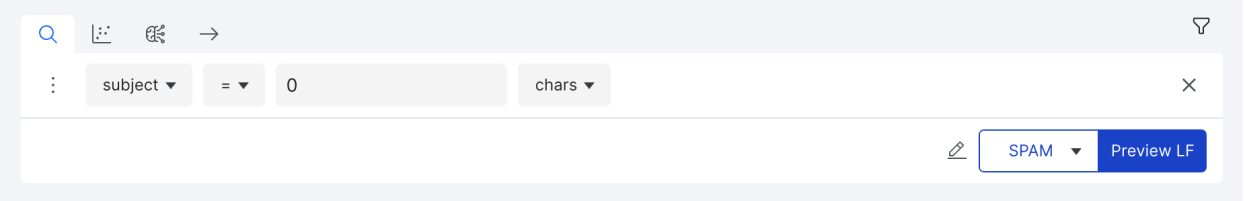
Copyright © 2025 Snorkel AI, Inc. All rights reserved

Numerical LF builders

This article describes the set of numeric labeling function (LF) builders that are available for classification and information extraction applications.

Field length builder

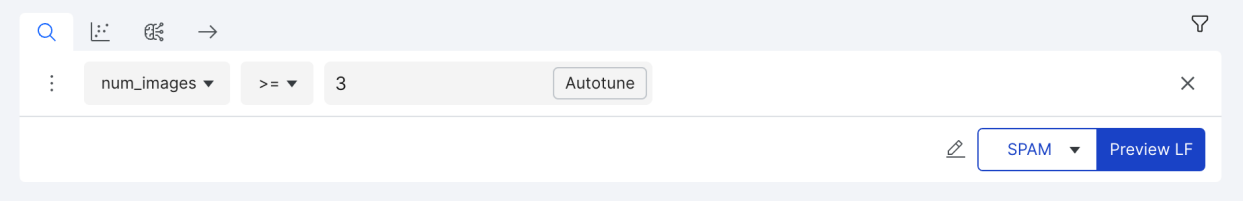
Label data points based on the length of a specific field according to units (e.g., subject, body etc.)



Numeric builder

Label data points that match a numerical condition based on a numerical field. Using a non-numerical field will likely result in abstains for all data points. Selecting a field and label will enable the autotuner functionality, which will calculate the optimal numerical condition for the field and label based on labeled examples in the dev set. It will match or exceed the specified minimum precision (if possible) while maximizing recall.

> **NOTE**
>
> For the email-spam application, if the email contains at least 3 images, it's likely SPAM.



Bounding polygon builder

Label data points that are inside the polygon created by tracing the specified coordinates in the provided order, or on one of the vertices of the polygon. At least 3 coordinates must be provided.

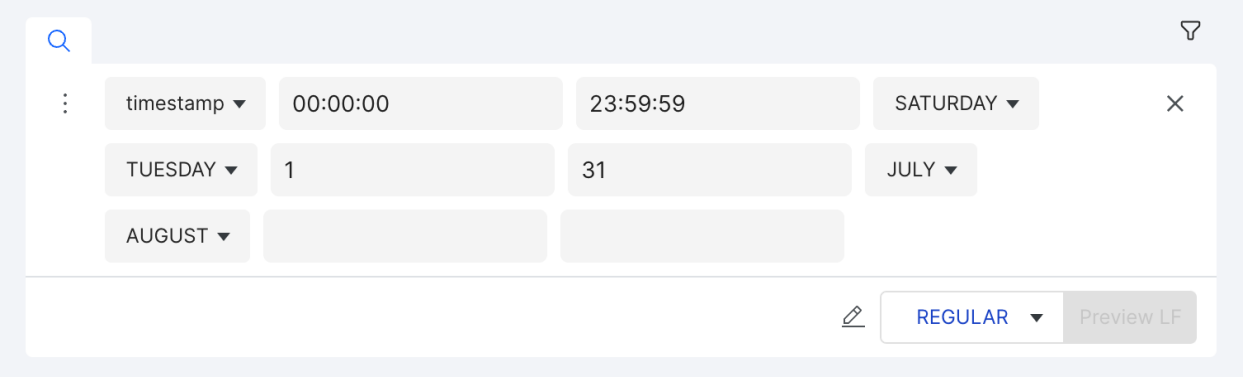
Copyright © 2025 Snorkel AI, Inc. All rights reserved

Other search LF builders

This page describes the set of other search based LF builders that are available for classification and information extraction applications.

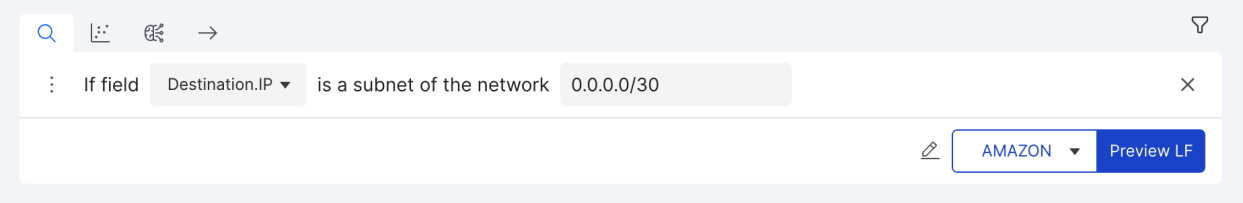
Timestamp builder (time series LF)

Label data points that fall within the specified periodic time intervals (start and end are both inclusive). The ranges (excluding years) are cyclic, meaning that, for example, specifying a start day of Saturday and an end day of Tuesday will select all Saturdays, Sundays, Mondays, and Tuesdays that satisfy the other constraints.



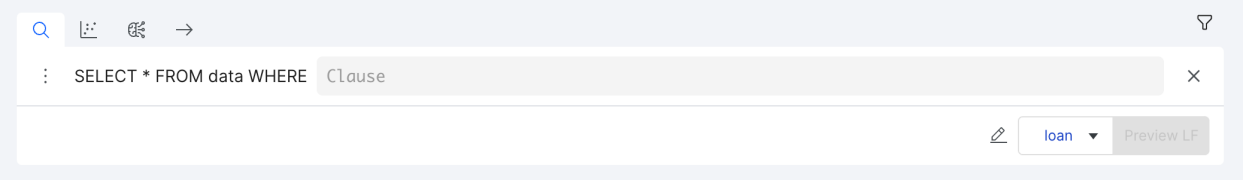
IP address builder (network LF)

Label data points based on whether an IP address in the specified field is a subnet of the specified IP network, represented by an IP address with a prefix (e.g., 10.0.0.0/30). Both IPv4 and IPv6 are supported.



SQL query builder (table based LF)

Label data points using a SQL query to return relevant rows of the dataset to mark them as having a label. This can be helpful or engineers and data anaylsts whose workflows have provided them with commonly used queries to find the segment of the data they care about.



Copyright © 2025 Snorkel AI, Inc. All rights reserved

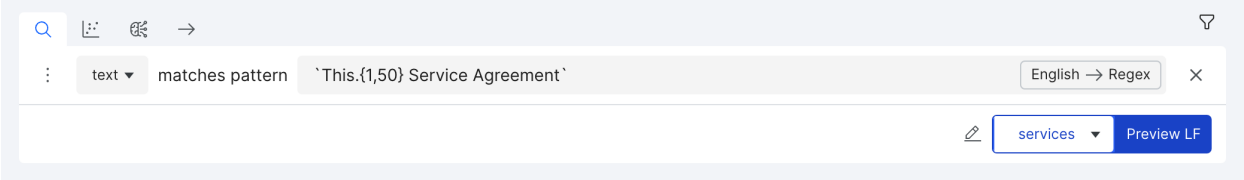
Pattern based LF builders

This page describes the basic set of pattern based LF builders that are available for classification applications.

Regex builder

Label data points that match a user-defined regex over a specific data field. This builder highlights phrases that match the regex pattern in the data point in the data viewer to help iterate. https://regex101.com is also a good resource for regexes. The builder also supports fuzzy matching in regex patterns (see https://pypi.org/project/regex/).

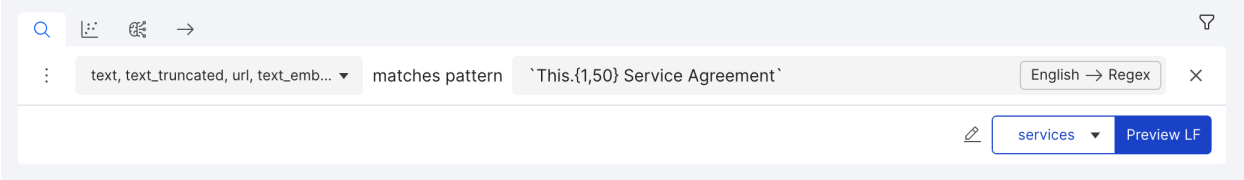
> NOTE
>
> To find service documents in the contract classification dataset, we might label data points that match the regex pattern `` `This.{1,50} Service Agreement` ``



Full text regex builder

Label data points that match a user-defined regex over a specific set of data fields.

Whereas the Regex Builder searches on one field, the Full Text Regex Builder can search through multiple or all fields at once. For example, you could search both the Title and Text body of a news article with a single LF. By default, all fields are selected (hence the “full text” match).



Keyword builder

Label data points that satisfy a condition with respect to at least one in a list of keywords you provide. There are three conditions that we support in-app:

- CONTAINS: if the specified field contains at least a keyword.
- CONTAINS LINE MATCHING: if one of the lines in the specified field matches a keyword.
- EQUALS: if the specified field equals a keyword.

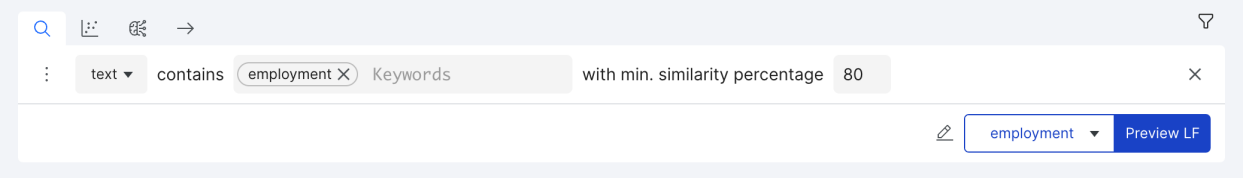
> NOTE
>
> Note that the Keyword builder allows a maximum of 3 keywords/phrases per builder.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

> NOTE
>
> For the contract classification application, we can look for the word `employment` to label documents as `EMPLOYMENT` type.

Fuzzy keyword builder

Label data points that contain a string that's similar to at least one of a list of keywords you provide, within the specified similarity ratio using the SequenceMatcher library. This is particularly useful for documents obtained through OCR.

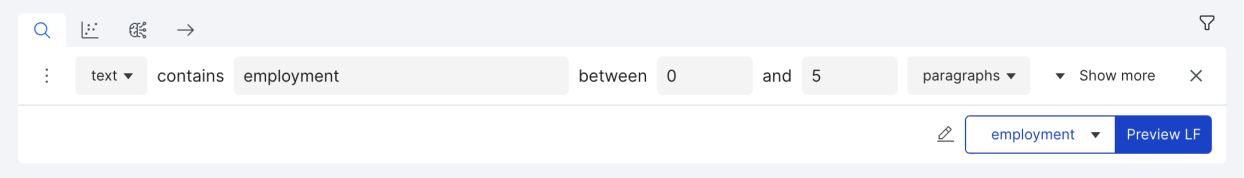


Keyword location builder

Label data points based on whether a pattern appears in a specific section (lines, paragraphs etc.) of a field. Words are split using the space character (s), lines using the newline character (n), sentences based on selected punctuation ([.?!]s), and paragraphs based on two new line characters (nn+).

Advanced option: You can specify how often the pattern should appear in that location!

> NOTE
>
> For the contract classification application, we look whether the word `employment` occurs in the first 5 paragraphs.



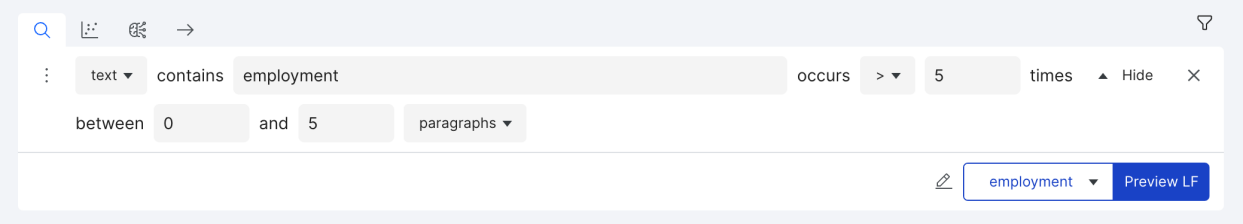
Keyword count builder

Label data points based on the frequency of pattern in a field.

Advanced option: You can specify the specific section of the field you want to count the pattern frequency!

> NOTE
>
> For the contract classification application, we look whether the word `employment` occurs more than 5 times in the first 5 paragraphs.

Copyright © 2025 Snorkel AI, Inc. All rights reserved



Copyright © 2025 Snorkel AI, Inc. All rights reserved

Rich document LF builders

If your application utilizes rich document structure (see the Extraction from PDFs: Extracting balance sheet amounts tutorial), the following LF builders are available. These builders support heuristics based on structural characteristics of the document with respect to the span.

Rich document expression builder

Label data points by evaluating the given expression. The expression uses Python syntax, and can reference any dataframe columns, as well as special variables `SPAN` (representing the span) and `PATTERN1`, `PATTERN2`, and so on (referencing the first match for each given regex). The special variables have properties `top`, `bottom`, `left`, `right`, `center`, `middle`, `page_id`, `line_id`, `par_id`, `row_id`, `char_start`, `char_end`, `text`. If any of the regex patterns has no match, the LabelingFunction will abstain.

In the example below, the LF has regex patterns `["Fair", "[Vv]alue"]` and expression `PATTERN1.top <= PATTERN2.top and PATTERN1.top > PATTERN2.top - 100 and SPAN.left > PATTERN1.left - 100 and SPAN.right < PATTERN1.right + 100 and SPAN.top > PATTERN1.top`. It will find documents that have the words `Fair` and `Value` (or lowercase value) with Fair not too far below Value, and label spans that are roughly below Fair.

This builder can also be used for PDF classification applications, but without the special `SPAN` variable.

For example, consider the LF with patterns `["Assets", "Liabilities"]` and expression `PATTERN1.left == PATTERN2.left and len(hv_lines.dfs_vert[page_idx]) >= 2`. This matches any page with `Assets` and `Liabilities` in the same column (vertically aligned), AND at least 2 vertical lines (suggesting the presence of a table). This LF could be used to identify whether a document is a balance sheet in a document classification task.

Additional Examples

- Label any span whose `bottom` is within 10 pixels of the pattern `Term`
  - Patterns: `Term`
  - Expression: `abs(PATTERN1.bottom - SPAN.bottom) < 10`
- Label any span whose `bottom` is 10 pixels above the pattern `Term` and has `Imp` in it's text.
  - Patterns: `Term`
  - Expression: `PATTERN1.bottom < SPAN.bottom + 10 and 'Imp' in SPAN.text`
- Label any span whose lowercase text is `credit` and whose document has `Term` in the first 1000 pixels.
  - Patterns: `Term`, `Credit`
  - Expression: `PATTERN1.bottom < 1000 and PATTERN2.text.lower() == SPAN.text.lower()`
- Label any doc which has `Term` in the first 1000 pixels of the first page
  - Patterns: `Term`
  - Expression: `PATTERN1.bottom < 1000 and page_index == 0`

Copyright © 2025 Snorkel AI, Inc. All rights reserved

Rich document bounding box

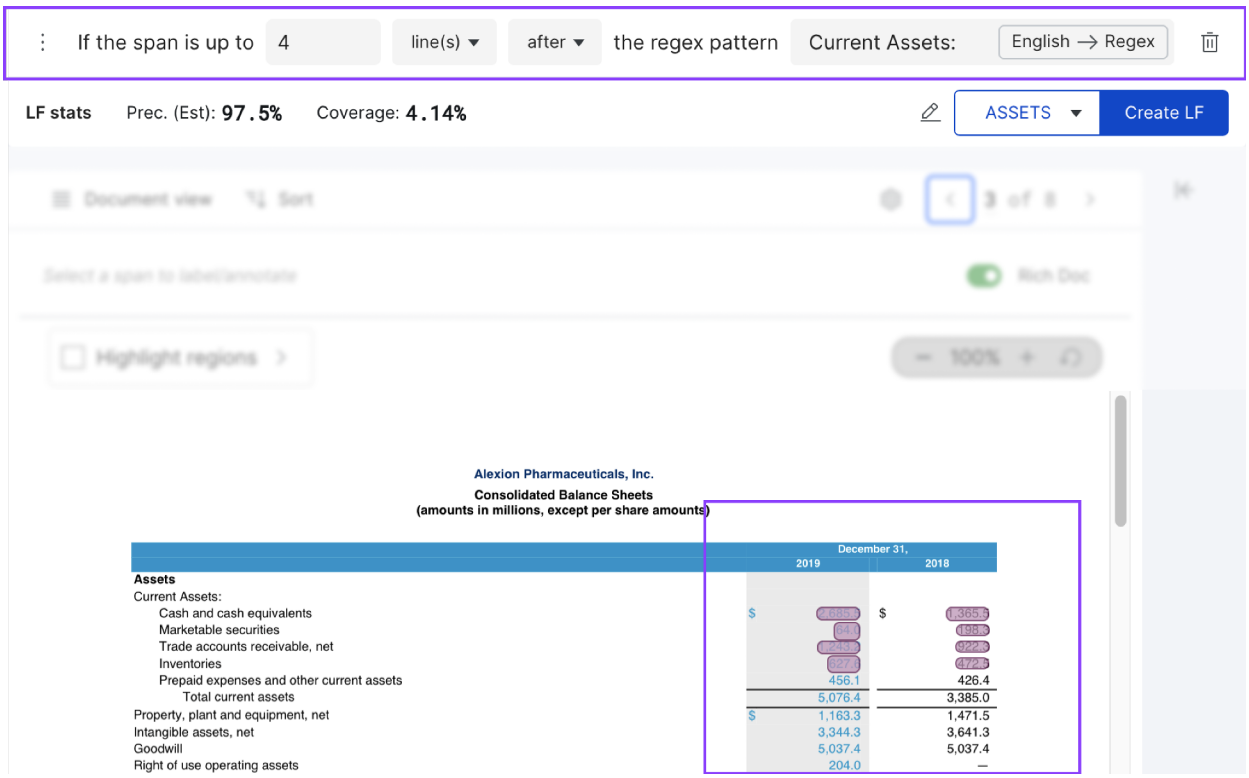
Label documents based on the existence of a given regular expression at a specific location in the page. The bounding box coordinates can be found by hovering the cursor over the words in the Data View. In the example below the LF checks if regex pattern “Loan.15Agreement” lies within a bounding box defined by coordinates “0, 0, 2400, 1000”. We compare all occurences of the regex pattern with the coordinates. This builder can only be used for Rich Doc classification applications.

Span regex proximity

Label data points based on the existence of a given regular expression in their vicinity. This checks if the span is within some number (called the window size) of units (lines/paragraphs/areas) in a specified direction (before, after, or either direction) of the specified regex. If the window size is 0, only text in the span’s own line/paragraph/area will be considered.

In the example below, this LF labels the 8 spans that are up to `4 LINES` after the regex pattern `Current Assets:` as `ASSETS`.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

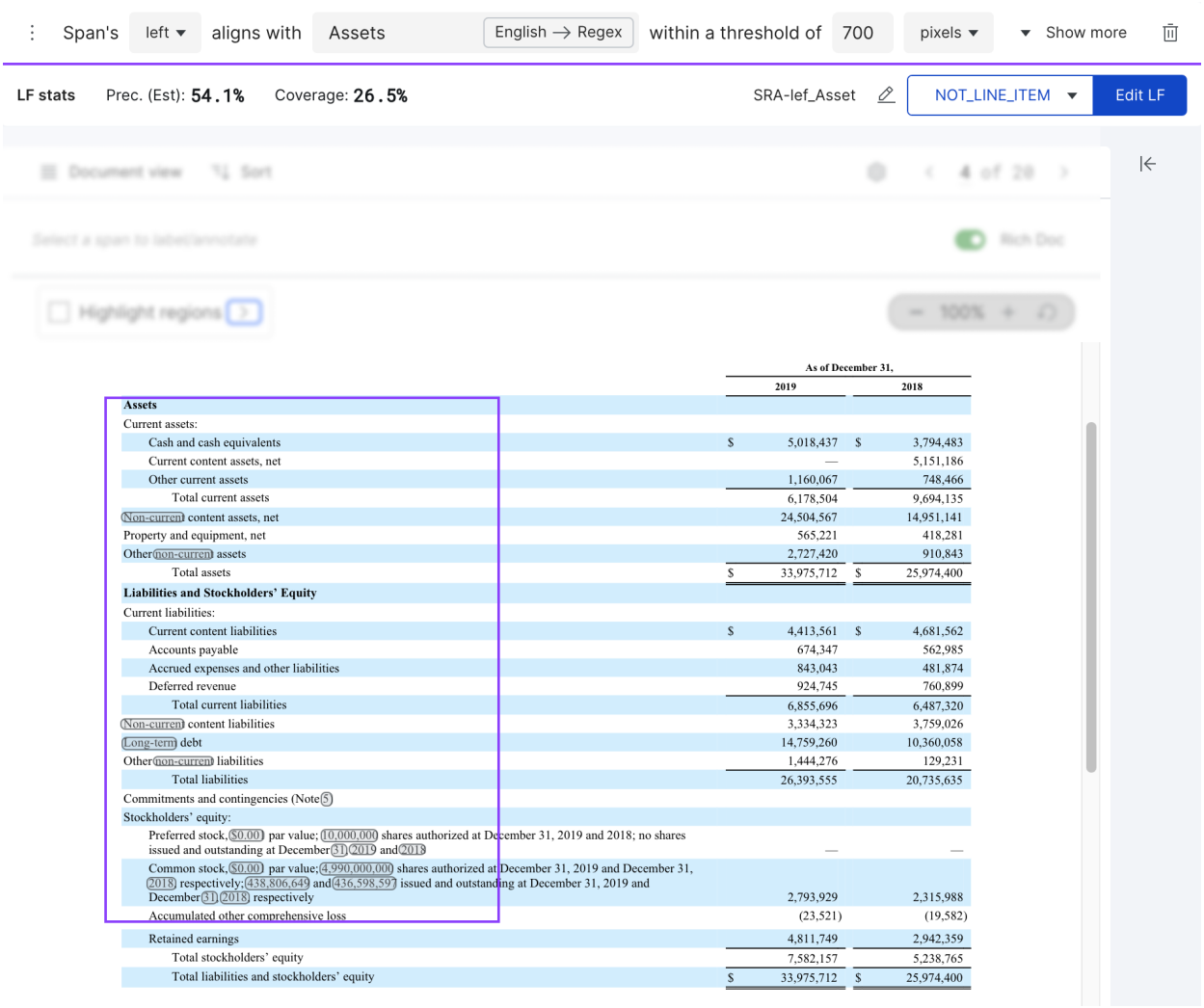


Span regex alignment

Label data points based on whether or not a given regular expression is aligned with the span in some way. Select a location (LEFT / CENTER / RIGHT / TOP / MIDDLE / BOTTOM) to compare between the bounding box of the span and the boundings boxes of any matches for your regular expression. Also select a threshold for how close the two coordinates need to be to be considered aligned. This comparison can be made in PIXELS or PAGE_PERCENT, a percentage (0-100) of the page's total width/height, depending on which dimension is relevant for the given location. You may also optionally limit matches to the specified scope of the Span, and/or in only the given direction (e.g., if your threshold is 100 pixels, location is TOP, and threshold direction is UP_ONLY, then only matches whose top boundary are within 100 pixels *above* (not below) the Span on the page will be matched. If no threshold direction is specified, then matches within the threshold are allowed in both directions (LEFT *and* RIGHT, or UP *and* DOWN).

As is the example below, this LF labels the spans whose `LEFT` aligns with the regex pattern `Assets` (case-sensitive), within a margin of `700 PIXELS`, as `INVALID`. These spans are highlighted in grey in the figure.

Copyright © 2025 Snorkel AI, Inc. All rights reserved



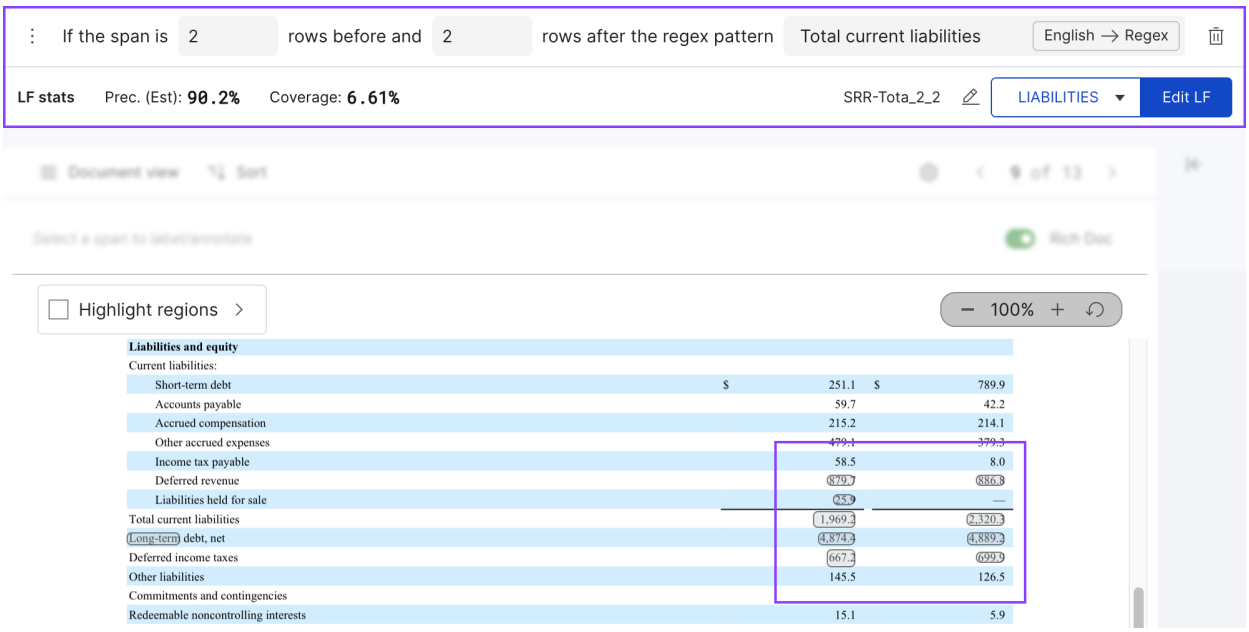
	As of December 31,	
	2019	2018
Assets		
Current assets:		
Cash and cash equivalents	$ 5,018,437	$ 3,794,483
Current content assets, net	—	5,151,186
Other current assets	1,160,067	748,466
Total current assets	6,178,504	9,694,135
Non-current content assets, net	24,504,567	14,951,141
Property and equipment, net	565,221	418,281
Other non-current assets	2,727,420	910,843
Total assets	$ 33,975,712	$ 25,974,400
Liabilities and Stockholders' Equity		
Current liabilities:		
Current content liabilities	$ 4,413,561	$ 4,681,562
Accounts payable	674,347	562,985
Accrued expenses and other liabilities	843,043	481,874
Deferred revenue	924,745	760,899
Total current liabilities	6,855,696	6,487,320
Non-current content liabilities	3,334,323	3,759,026
Long-term debt	14,759,260	10,360,058
Other non-current liabilities	1,444,276	129,231
Total liabilities	26,393,555	20,735,635
Commitments and contingencies (Note 5)		
Stockholders' equity:		
Preferred stock, $0.00 par value; 10,000,000 shares authorized at December 31, 2019 and 2018; no shares issued and outstanding at December 31, 2019 and 2018	—	—
Common stock, $0.00 par value; 4,990,000,000 shares authorized at December 31, 2019 and December 31, 2018 respectively; 438,806,649 and 436,598,597 issued and outstanding at December 31, 2019 and December 31, 2018 respectively	2,793,929	2,315,988
Accumulated other comprehensive loss	(23,521)	(19,582)
Retained earnings	4,811,749	2,942,359
Total stockholders' equity	7,582,157	5,238,765
Total liabilities and stockholders' equity	$ 33,975,712	$ 25,974,400

Span regex row (Rich-document based LFs)

Label data points based on whether or not the span is within a certain number of rows of a given regular expression. Set rows before and rows after to specify the window of rows to search, relative to the span's current row. If both rows before and rows after are set to 0 only spans in the same row, that match the regex pattern, will be labeled.

In the example below, this LF labels the 10 spans that are at most 2 rows before or 2 rows after the regex pattern `Total current liabilites` as `LIABILITIES`.

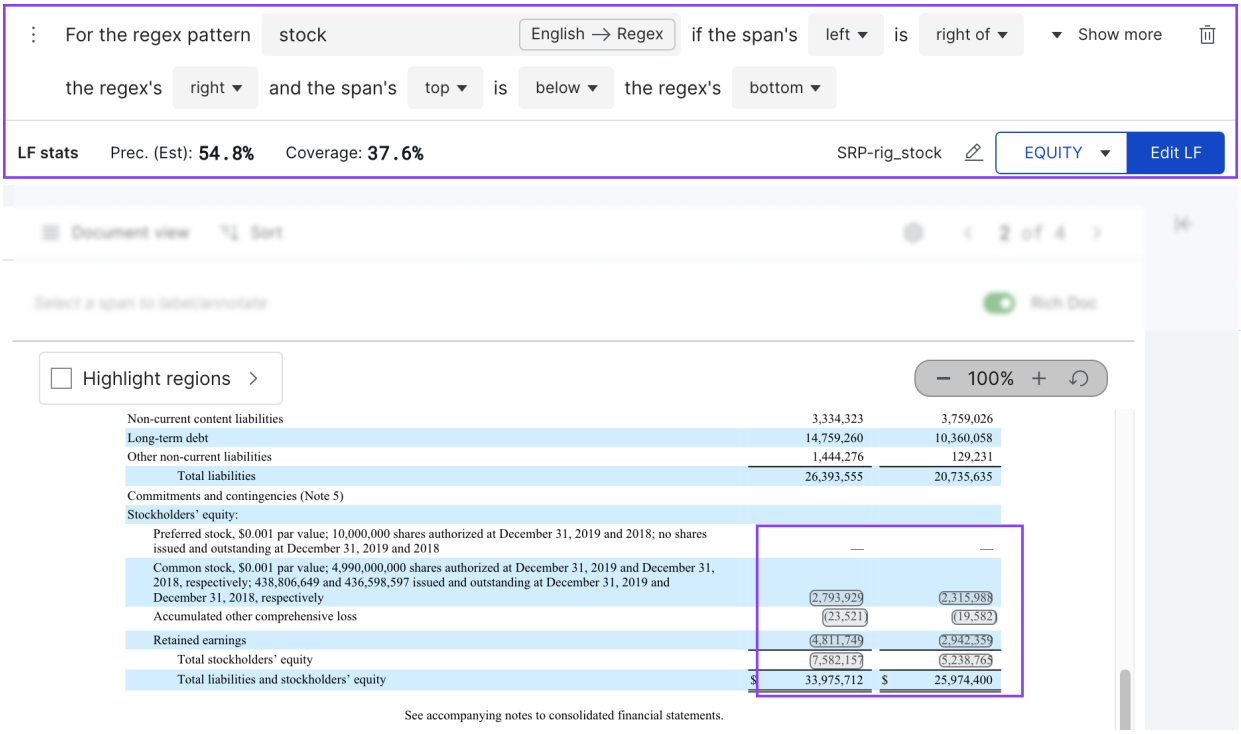
Copyright © 2025 Snorkel AI, Inc. All rights reserved



Span regex position

Label data points based on whether the span is at a specific direction with respect to the given regular expression. Select two location attributes (LEFT / CENTER / RIGHT / TOP / MIDDLE / BOTTOM) of the span to compare with two location attributes of any matches to the regular expression. You may optionally select the scope of the comparison (PAGE / AREA / PARA / LINE). You can enable only the first, only the second, or both conditions under the Advanced options.

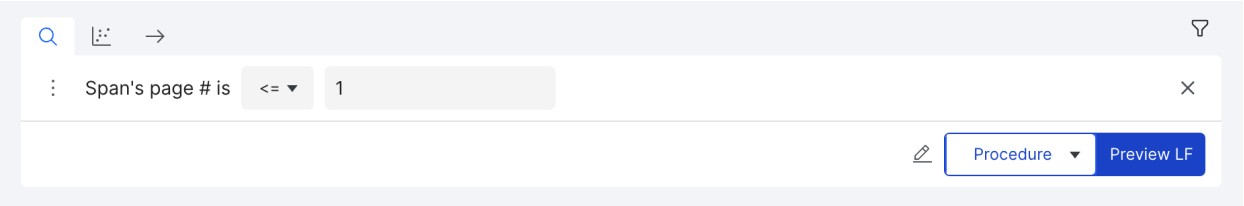
In the example below, this LF labels 14 spans that lie below and to the right of the regex pattern `stock` as `EQUITY`.



Copyright © 2025 Snorkel AI, Inc. All rights reserved

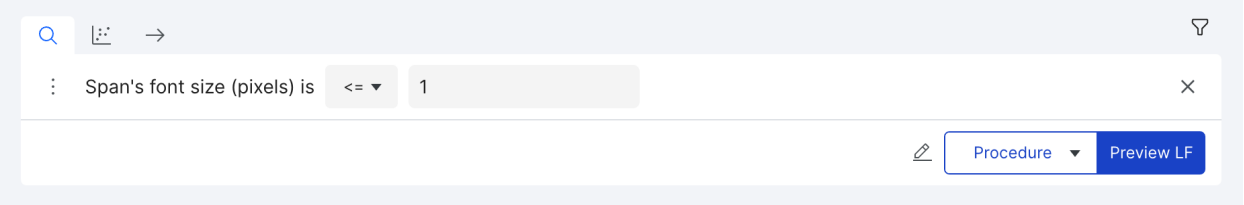
Span page

Label data points based on the page number of the given span, using 1-indexing (the first page of the document is considered page 1, not page 0).



Span font size

Label data points based on the font size of the given span (in pixels, not points), based on the value reported in the hOCR.



Copyright © 2025 Snorkel AI, Inc. All rights reserved

(Beta) Word LF builders

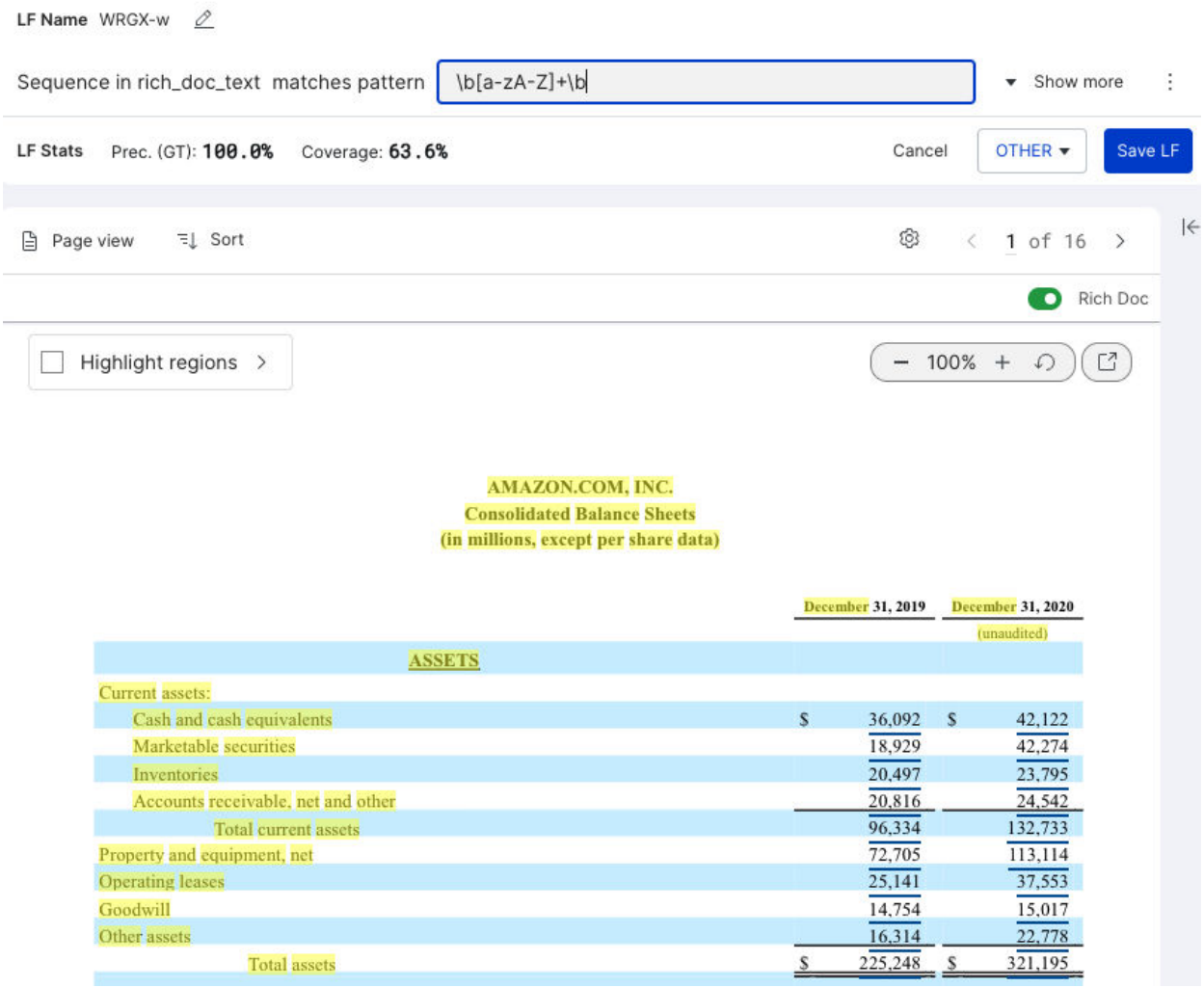
> **NOTE**
> This is a beta feature available to customers using a Snorkel-hosted instance of Snorkel Flow. Beta features may have known gaps or bugs, but are functional workflows and eligible for Snorkel Support. To access beta features, contact Snorkel Support to enable the feature flag for your Snorkel-hosted instance.

This article describes the set of word-based labeling function (LF) builders that are available for PDF word classification applications.

Word-based regex builder

Label words present in the page based on regex patterns applied over the page text. This builder highlights words that match the regex pattern in the data viewer to help with iteration.

This example uses the word-based regex builder to find all alphabetical words in the page using the regex pattern `\b[a-zA-Z]+\b`.



AMAZON.COM, INC.
Consolidated Balance Sheets
(in millions, except per share data)

	December 31, 2019	December 31, 2020 (unaudited)
ASSETS		
Current assets:		
Cash and cash equivalents	$ 36,092	$ 42,122
Marketable securities	18,929	42,274
Inventories	20,497	23,795
Accounts receivable, net and other	20,816	24,542
Total current assets	96,334	132,733
Property and equipment, net	72,705	113,114
Operating leases	25,141	37,553
Goodwill	14,754	15,017
Other assets	16,314	22,778
Total assets	$ 225,248	$ 321,195

Copyright © 2025 Snorkel AI, Inc. All rights reserved

Word-based expression builder

Label words present in the page based on a regex pattern and a Python expression that is evaluated on the pattern and words. The expression uses Python syntax and the following special variables:

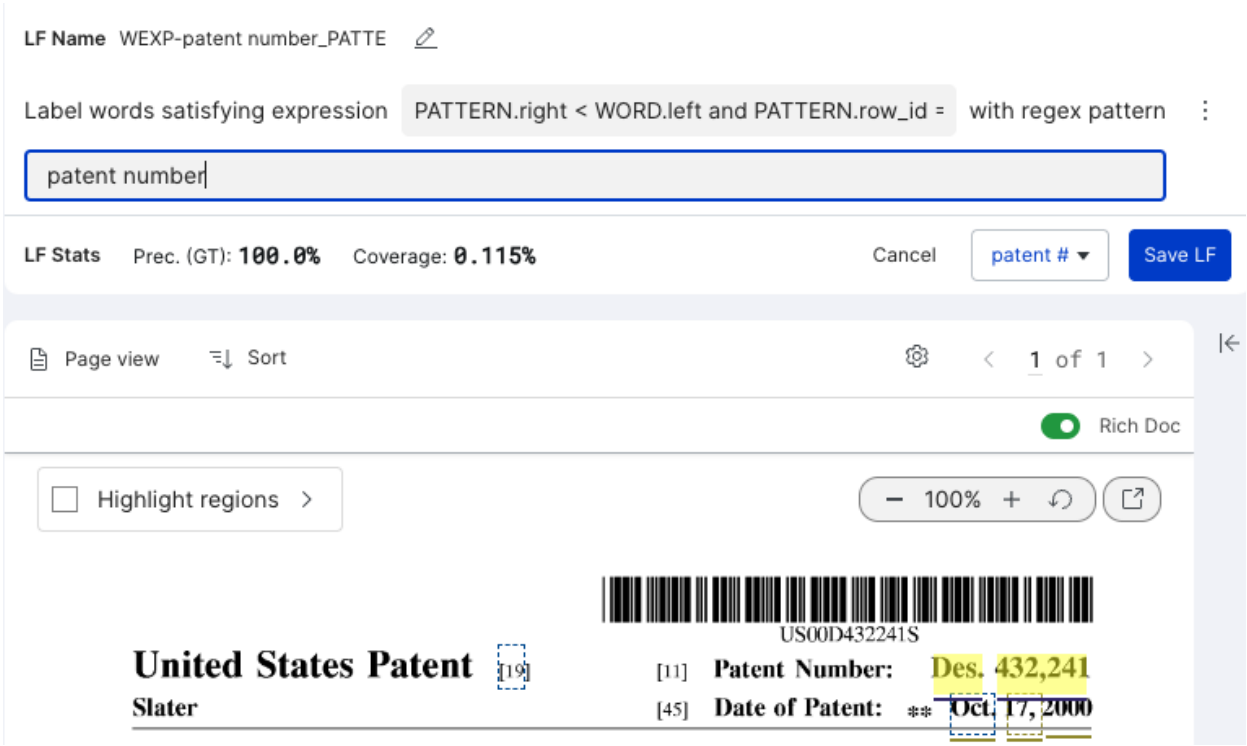
- `WORD`: the word to evaluate
- `PATTERN`: the regex pattern

The special variables have the following properties:

- `top`, `bottom`, `left`, `right`: the bounding box of the word
- `center`: the center of the word (midpoint between left and right)
- `middle`: the middle of the word (midpoint between top and bottom)
- `char_start`, `char_end`: the character start and end indices of the word
- `line_id`: the line index of the word
- `par_id`: the paragraph index of the word
- `area_id`: the area index of the word
- `page_id`: the page index of the word
- `row_id`: the row index of the word, which is useful for multi-column documents

This builder highlights words that match the expression in the data viewer to help with iteration.

This example uses the word expression builder to find the patent number in the page. The regex pattern is `patent number` and the expression is `PATTERN.right < WORD.left and PATTERN.row_id == WORD.row_id`. This builder looks for words that are to the right of "patent number" and in the same row.

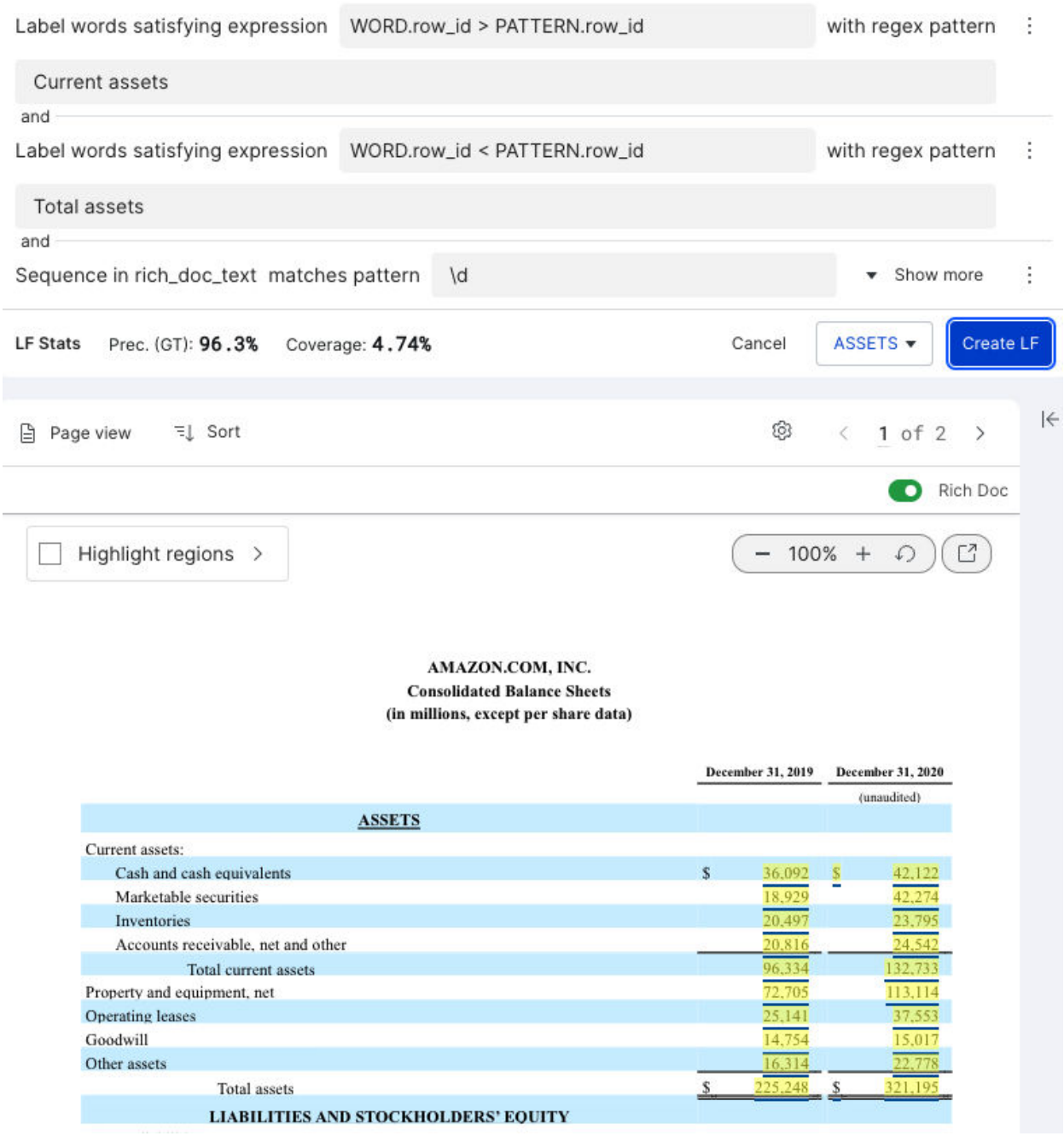


This builder is also effective when combined with the word regex builder above. This example finds all asset amounts in the balance sheet by combing multiple LFs:

- **Word regex LF** to find all numeric words in the page, using the regex pattern `\d`.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

- **Word expression LF** to find the words that are below the term "Current assets" using the pattern `Current assets` and the expression `WORD.row_id > PATTERN.row_id`.
- **Word expression LF** to find the words that are above the term "Total assets" using the pattern `Total assets` and the expression `WORD.row_id < PATTERN.row_id`.



AMAZON.COM, INC.
Consolidated Balance Sheets
(in millions, except per share data)

	December 31, 2019	December 31, 2020 (unaudited)
ASSETS		
Current assets:		
Cash and cash equivalents	$ 36,092	$ 42,122
Marketable securities	18,929	42,274
Inventories	20,497	23,795
Accounts receivable, net and other	20,816	24,542
Total current assets	96,334	132,733
Property and equipment, net	72,705	113,114
Operating leases	25,141	37,553
Goodwill	14,754	15,017
Other assets	16,314	22,778
Total assets	$ 225,248	$ 321,195
LIABILITIES AND STOCKHOLDERS' EQUITY		

Copyright © 2025 Snorkel AI, Inc. All rights reserved

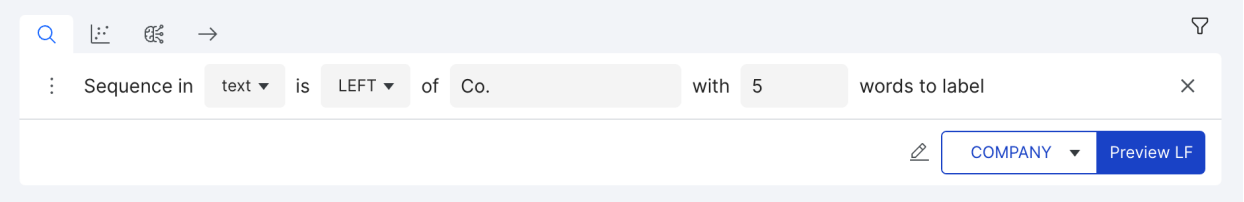
Sequence LF builders

This article describes the set of builders that are available for sequence tagging applications (see the Sequence tagging: Extracting companies in financial news articles tutorial).

Sequence context builder

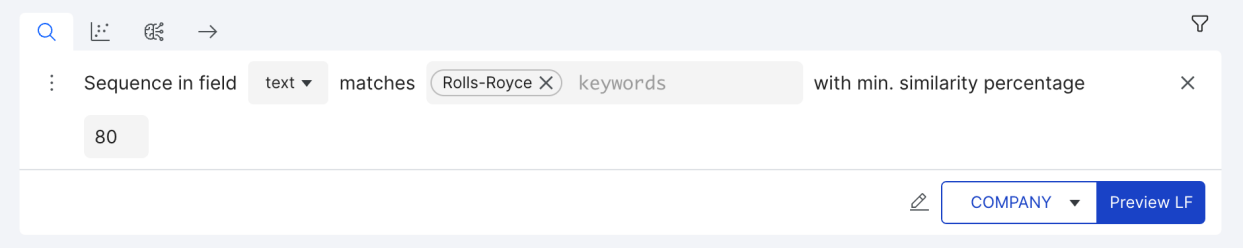
Label tokens based on the text surrounding a regular expression pattern. This labels the specified number of tokens if they are `[LEFT]`, `[RIGHT]`, or `[LEFT OR RIGHT]` of the provided pattern.

> **TIP**
> You can combine the "Show LF Votes" button and the "View [In]Correct" filter to see which tokens are being labeled by this LF.



Sequence fuzzy keyword builder

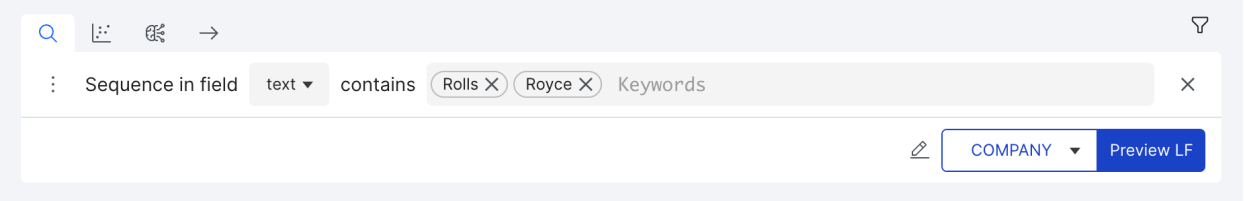
Label tokens that are similar to one or more of the list of keywords (max of three) you provide, within the specified similarity ratio using the SequenceMatcher library. This is particularly useful for documents obtained through OCR.



Sequence keyword builder

Label a sequence of one or more tokens that match any words or phrases you provide.

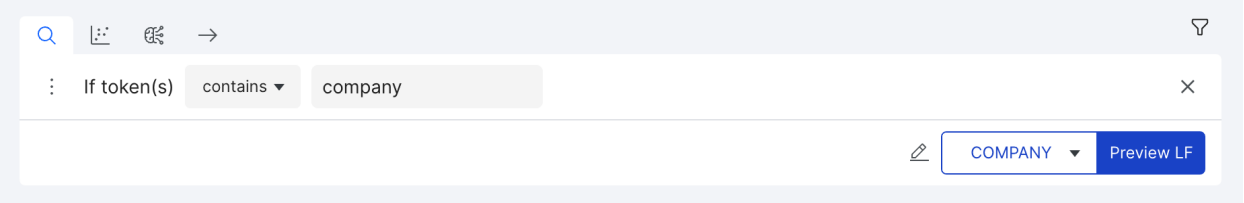
> **NOTE**
> The Sequence Keyword builder allows a maximum of three (3) keywords/phrases per builder. To add more, you can * combine additional Sequence Keyword Builders using `OR`, * or use the Sequence Entity Dict Builder.



Copyright © 2025 Snorkel AI, Inc. All rights reserved

Sequence substring expansion builder

If a token `[matches]` the whole value, `[contains]` the value in it, `[starts with]`, or `[ends with]` the value, then specify the class to vote for. The value can be toggled as regex or not.



Sequence spaCy prop builder

This LF builder incorporates the spaCy properties of tokens for labeling them.

Prerequisite

In your DAG, add a `SpacyPreprocessor` node before the `Model` node to create spaCy properties for your data. The `SpacyPreprocessor` "Field" and "Target field" values are `text` and `doc`, respectively.

Label tokens that satisfy these conditions:

- The tokens in this spaCy field (`doc`).
- Is tagged with any `SPACY_TAGS`, such as `VERB` or `ADJ` for part-of-speech tags. For a complete list of tags, see the spaCy glossary

The dropdown has commonly used spaCy tags. If you have added any other spaCy tags in your application, add these as free text.

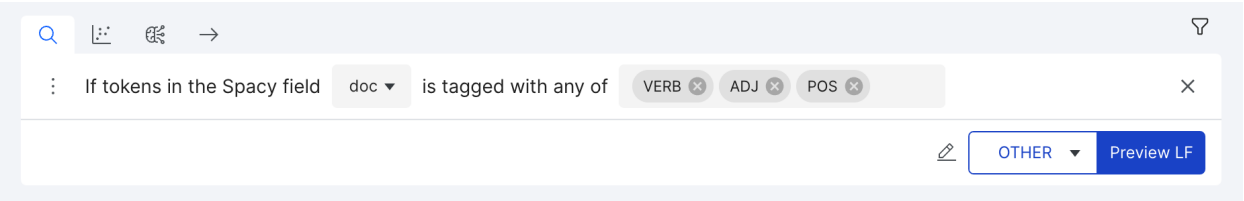
The data viewer in the spaCy field shows any tags you added. For example, the spaCy field `doc` in the data viewer shows custom tags `NNP`, `IN`, `NNPS`, and `_SP:`


```
{"id":47,"start":254,"end":255,"tag":",","pos":"PUNCT","morph":"PunctType=Comm","lemma":",","dep":"punct","head":45},
{"id":48,"start":256,"end":263,"tag":"NNP","pos":"PROPN","morph":"Number=Sing","lemma":"Society","dep":"conj","head":45},
{"id":49,"start":264,"end":266,"tag":"IN","pos":"ADP","morph":"","lemma":"of","dep":"prep","head":48},
{"id":50,"start":267,"end":276,"tag":"NNP","pos":"PROPN","morph":"Number=Sing","lemma":"Petroleum","dep":"compound","head":51},
{"id":51,"start":277,"end":286,"tag":"NNPS","pos":"PROPN","morph":"Number=Plur","lemma":"Engineers","dep":"pobj","head":49},
{"id":52,"start":286,"end":287,"tag":"_SP","pos":"SPACE","morph":"","lemma":"\n","dep":"dep","head":51},
```


Advanced option: You can specify the spaCy properties that fit your use case. Snorkel Flow supports `POS` (part-of-speech), `DEP` (dependencies), and `TAG`.

Example

If the tokens are tagged as `VERB` or `ADJ` under `POS` part-of-speech tags, they are not likely company names. We can label these tokens as `OTHER`.



Sequence NER builder

This LF builder incorporates the named entity recognition resources for labeling spans.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

Label spans that satisfy these conditions:

- The spans in this field (`NER field`)
- is tagged with any of (`NER properties`)

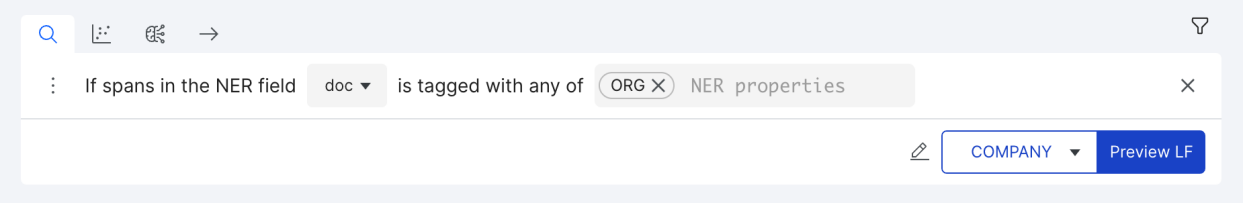
Prerequisite: In your DAG, add a `SpacyPreprocessor` node before the `Model` node to create Spacy properties for your data. The `SpacyPreprocessor` takes the your text field as "Field" and outputs "Target field" as `doc`, in this case, select `NER field` as `doc`. Alternatively, add a custom featurizer in the DAG to add NER field with the following format:

> NOTE
>
> NER field for text `Apple is looking at buying U.K. startup for $1 billion` is a JSON dictionary with "ents" as key, and value is a list of entities; Each entity is a dictionary with 'start', 'end' and 'label' as keys.

Note

```
{
 "ents": [
   {'start': 0, 'end': 5, 'label': 'ORG'}, # Apple
   {'start': 27, 'end': 31, 'label': 'GPE'}, # U.K.
   {'start': 44, 'end': 54, 'label': 'MONEY'}, # $1 billion
 ]
}
```

If the spans are tagged as `ORG`, they are likely company names. We can label these spans as `COMPANY`. In this example, "Apple" would be labeled as `COMPANY`.



Sequence entity dict builder

Label all tokens containing patterns from a dictionary provided through a JSON file.

Only the values of the JSON are used by the labeling function; the keys can be used for organizational purposes, but are otherwise ignored. See the example below for reference.

Label tokens that satisfy these conditions:

- The tokens in this field (`text`)
- Contains the patterns from this file (`file_path`)

Note

Example

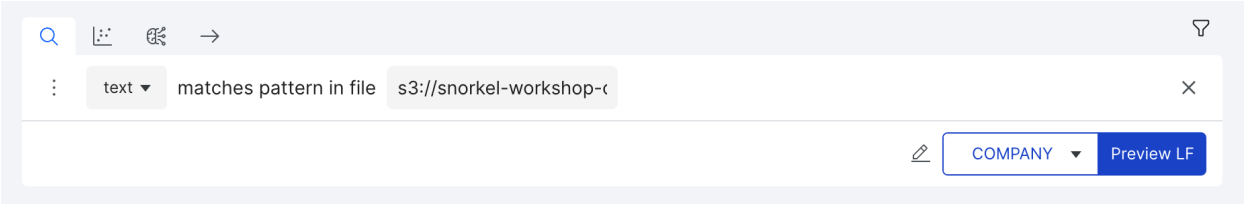
Our JSON file maps Fortune 500 stock tickers to company name aliases. The keys (stock tickers) are ignored for the purposes of this labeling function. If any of the values in the JSON match a span in the

Copyright © 2025 Snorkel AI, Inc. All rights reserved

document, we label the matched tokens as `COMPANY`. Here is an example of the JSON file: s3://snorkel-workshop-data/financial-news/f500_ticker_key_fixed.json

```
{
 "WMT": [
 "Walmart",
 "www.walmart.com",
 "Wal Mart Stores Inc"
 ],
 "XOM": [
 "Exxon Mobil",
 "www.exxonmobil.com",
 "Exxon Mobil Corp"
 ],
 ...
}
```

> **TIP**
> The location of the JSON file needs to be an S3.



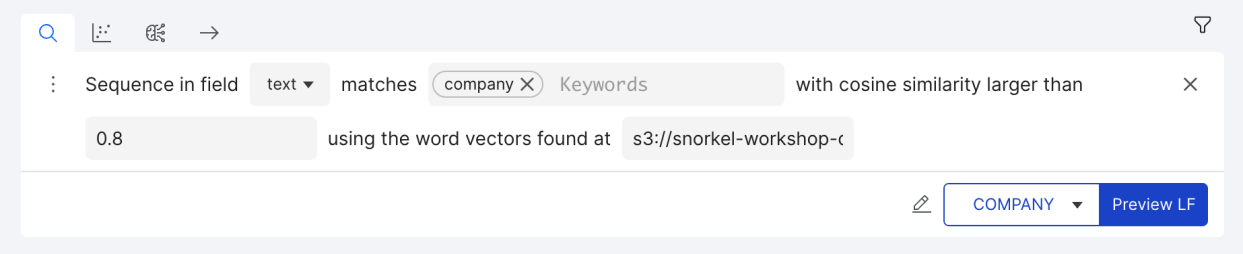
Sequence word vector builder

Label all tokens with a cosine similarity score greater than or equal to the provided threshold. This score is calculated between the associated word vectors loaded from the provided file path.

- **Keywords** - The keywords that will be compared to the tokens within the text field
- **Cosine Similarity** - Indicates the threshold at which a token will be considered similar enough to one of the keywords so as to be labeled. A score of 1.0 represents an identical word. Typical scores for matching are in the 0.4 to 0.8 range.
- **Word Vector Path** - Indicates the S3 or MinIO path to a file containing pre-trained word vectors. This file should be formatted such that each line contains a single token followed by the vector values delimited by spaces.

Note
Sample word vector file format for 4-dimensional vectors: scale 0.96014 -0.18144 0.22938 0.28215 solution 1.1908 0.23095 -0.169 0.043158 banking 0.17391 -0.69398 0.11051 0.7731 ...

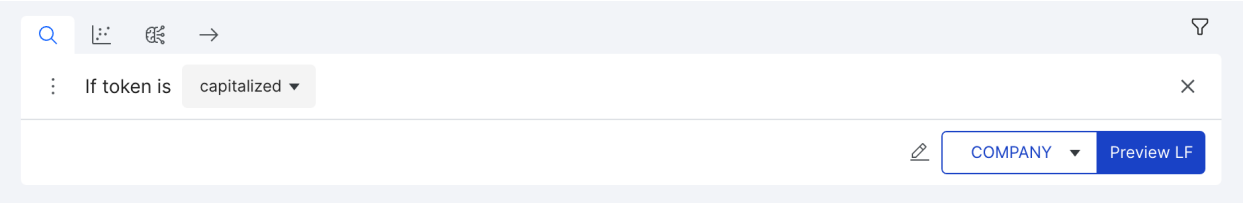
Copyright © 2025 Snorkel AI, Inc. All rights reserved



Sequence letter case builder

Vote the class of a token based on whether it's lowercase, uppercase, or fully capitalized. By default, this LF uses the regex `\b` word boundary to tokenize. You can set the Word Boundary Tokenizer to off under the advanced menu, which will use a spaCy tokenizer instead.

The nearby tokens are merged by default; you can turn off Merge Nearby Tokens under the advanced menu, which will vote on individual tokens.



Copyright © 2025 Snorkel AI, Inc. All rights reserved

Span based LF builders

This article describes the basic set of span based LF builders that are available for text extraction applications. They include all builders for classification applications but with extra support for span or contexts around the spans. There are also Span Builders – **Span Content Builder**, **Span Context Builder**, **Span Location Builder** – made especially for extraction applications. In this section, we cover these three builders and the span-aware **Regex Builder**.

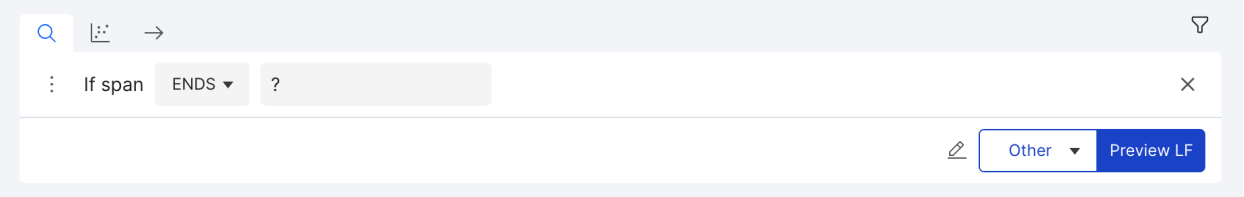
Span content builder

Label data points based on the content of the span to see if it:

- Matches exactly (`MATCHES`)
- Contains (`CONTAINS`)
- Starts with (`STARTS`)
- Ends with (`ENDS`)

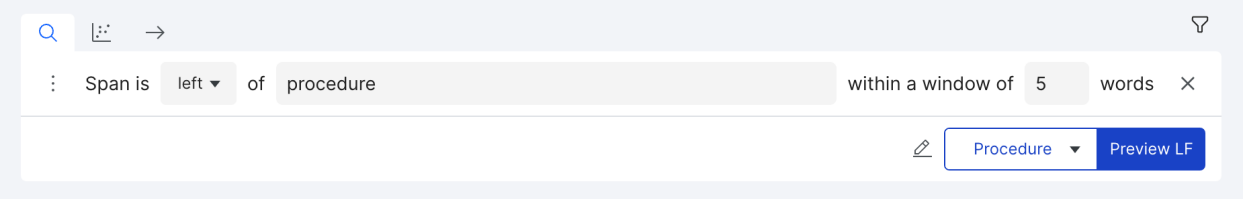
With the given string.

> **NOTE**
> An extracted candidate is more likely a question if it ends with a question mark.



Span context builder

Label data points based on the text surrounding the extracted spans. This checks if the span is `[LEFT] / [RIGHT] / [LEFT OR RIGHT]` of any in the list of the given strings, within the specified number of words.



Span location builder

Label data points based on whether the text of the extracted span appears in the specific location in the document. Words are split using the space character (s),lines using the newline character (n), sentences based on selected punctuation([.?!]s), and paragraphs based on two new line characters (nn+).

Advanced option: You can specify the frequency of the span within that location!

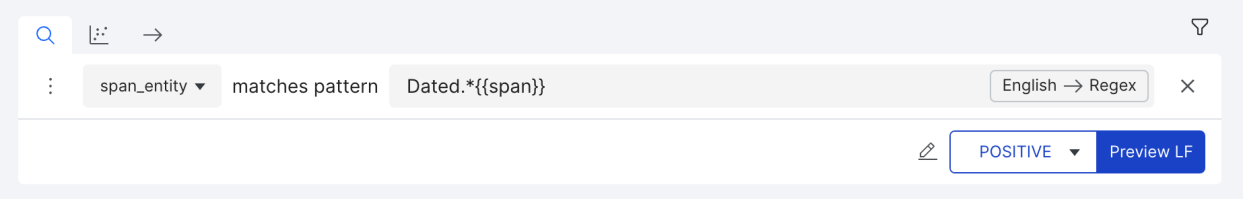
Copyright © 2025 Snorkel AI, Inc. All rights reserved

> **NOTE**
>
> If the text in the span appears in the first 5 paragraphs, it is probably the date associated with a contract (the execution date for loan agreements).

Regex builder

This is similar to the Regex Builder for classification applications, except that you can use the special marker `{{span}}` to reference the span's text.

> **NOTE**
>
> To match all the data points where the text span follows the word Dated, use the regex: `Dated.*{{span}}`.



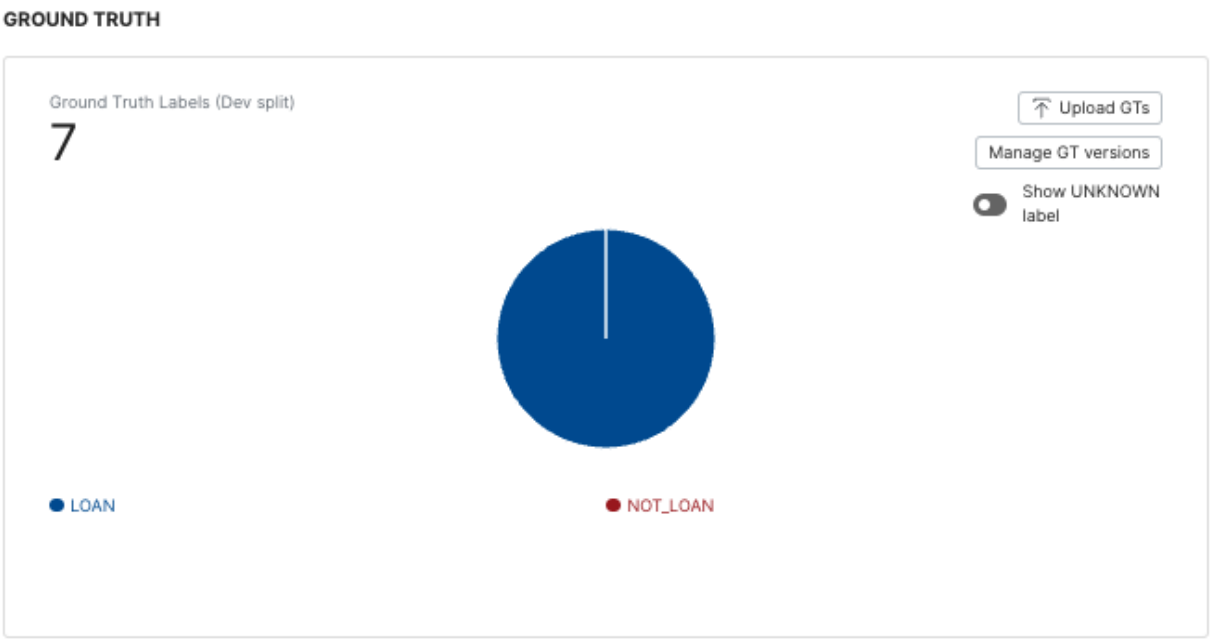
Copyright © 2025 Snorkel AI, Inc. All rights reserved

Versioning ground truth

This lesson demonstrates how to save multiple different versions of Ground Truth for your data. This applies to use cases involving various iterations on the Ground Truth, where users can save and load previous versions of Ground Truth for future model results. We provide walk-throughs for how to version (i.e., save, load, delete) Ground Truth in both the UI and the SDK.

Versioning ground truth in the UI

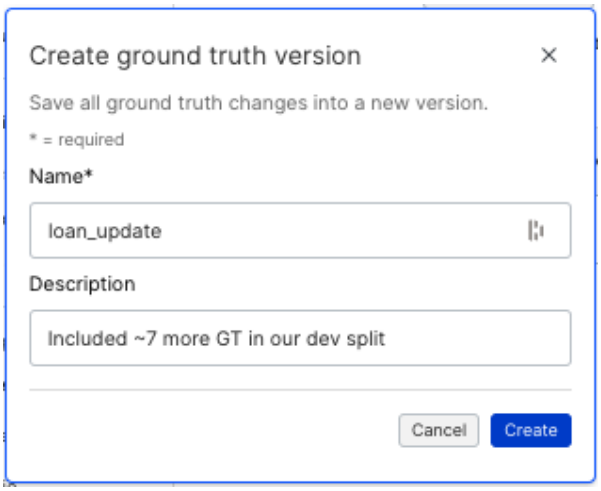
We can manage our Ground Truth by navigating to the **Develop page** of an application and finding the **Manage GT versions** button in the **Ground Truth** section to access the **Ground Truth** versions page. The **Ground Truth** section displays the label distribution for each label in our schema for our specified data split. The example below shows the total number of LOAN and NOT_LOAN data points in our dev split.



Clicking on the **Manage GT versions** brings you to the full list of Ground truth versions saved for your application. You can load a previously saved ground truth version by simply clicking **Load**, or delete an existing ground truth version by simply clicking the trash icon. The small green checkmark signifies the currently active ground truth version used for model results.

To save the current ground truth as a new version, use **Create GT Version** button to provide a unique name and description for the version.

Copyright © 2025 Snorkel AI, Inc. All rights reserved



Apart from manually created ones, ground truth versions are also automatically created when batch annotations are saved or when a model is trained on ground truth that is not already saved as a version. Auto-saved versions are prefixed with **auto**.

Versioning ground truth in the SDK

The SDK provides methods to save, load, list, and delete ground truth versions.

- `gts.create_ground_truth_version` SDK method can save the current ground truth as a new version.

```
import snorkelflow.client as sf

node = sf.get_model_node(APP_NAME) # Grab model node from desired application

sf.gts.create_ground_truth_version(
      node=node, # Model node in application
      name='new_gt', # Name of new ground truth version
      description='Recent GT additions to data' # Description for new ground
truth version
)
```

- `gts.load_ground_truth_version` SDK method can be used to load a specific version of ground truth.

```
import snorkelflow.client as sf

node = sf.get_model_node(APP_NAME) # Grab model node from desired application

sf.gts.load_ground_truth_version(
      node=node, # Model node in application
      gt_version_uid=1576 # UID for ground truth version, found in UI
)
```

- `gts.delete_ground_truth_version` SDK method can be used to delete a specific version of ground truth.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

```
import snorkelflow.client as sf

node = sf.get_model_node(APP_NAME) # Grab model node from desired application

sf.gts.delete_ground_truth_version(
      node=node, # Model node in application
      gt_version_uid=1576 # UID for ground truth version, found in UI
)
```

- `gts.list_ground_truth_versions` SDK method can be used to list all the versions of ground truth that have been saved for a node.

```
import snorkelflow.client as sf

node = sf.get_model_node(APP_NAME) # Grab model node from desired application

sf.gts.list_ground_truth_versions(
      node=node # Model node in application
)
```

Copyright © 2025 Snorkel AI, Inc. All rights reserved

Using data slices

A slice is a filtered subset of data rows that share a specific characteristic, like a topic, language, or error type. By using filters, users can create slices to focus on high-impact data areas for analysis, debugging, and targeted model training.

There are two types of slices:

- **Manual slices:** Individual datapoints are directly assigned to a slice.
- **Programmatic slices:** Applies a slicing function across the dataset to automatically group relevant datapoints. Slicing functions can be composed from pre-built templates.

> NOTE
>
> Slices have some limitations:
> - You cannot create batches from a slice.
> - Slices cannot be directly used for model training.
> - Slice definitions cannot be exported between projects.

When to use slices

There are two primary use cases for using slices within Snorkel Flow—annotator and data scientist.

Annotator

As an annotator, you can create slices to focus on specific data segments that need careful labeling or review, which helps prioritize data that aligns with specific project goals.

- For more information about how to annotate, see Walkthrough for annotators.
- For more information about how to review annotations, see Walkthrough for reviewers.

Data scientist

As a data scientist, you can use slices to analyze model performance on specific subsets and improve model accuracy in targeted areas. In the **Evaluate** page, data scientists can track model performance across slices, helping identify where the model is excelling or needs improvement. In **Studio**, data scientists can filter data by slice to focus on high impact subsets for training and error analysis. Continue reading to learn more about how to create and use slices through the UI and SDK.

Where to find slices

- Annotation Suite
- Studio Page
- Evaluate Page
- Batches

Edit slice membership

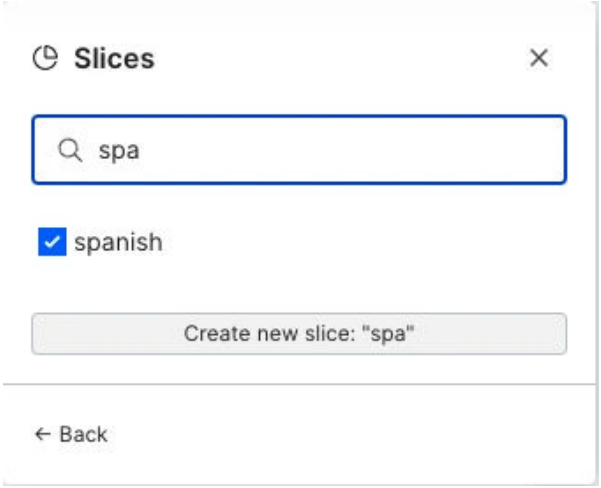
You can search for a slice, add a new slice, or check and uncheck current slices for the specific datapoint.

1. Select the **Slices** icon.

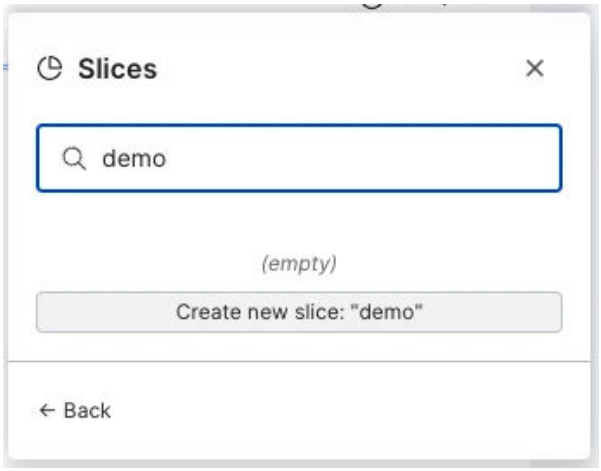
Copyright © 2025 Snorkel AI, Inc. All rights reserved

2. Select the **Edit slice membership** button.

3. Edit the slice membership:
   - If you want to search for a slice, type the slice name in the **Search or add slice** bar:



   - If you want to add a new slice, type the slice name in the **Search or add slice** bar and select **Create new slice**:



   - If you want to check or uncheck current slices for the specific datapoint, select the checkbox next to the slice name.

Manage slices

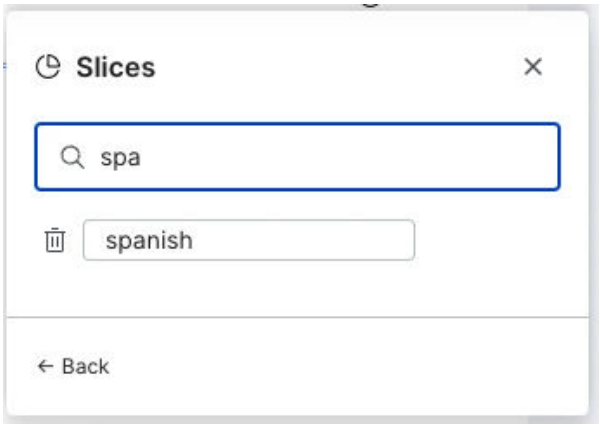
You can search for a slice, delete a slice, or edit the name of a slice.

1. Select the **Slices** icon.

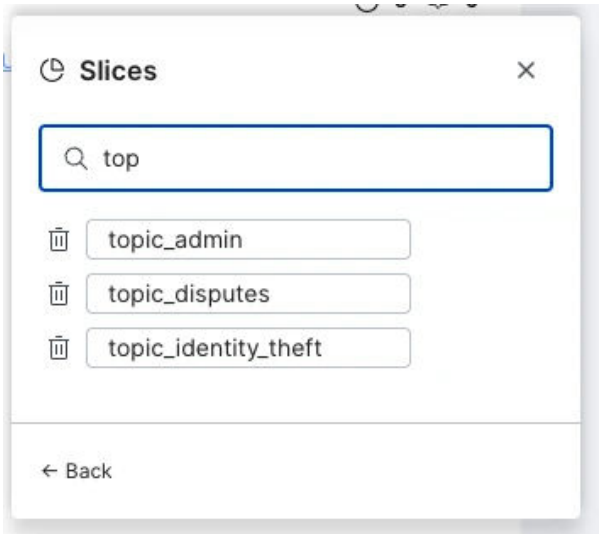
2. Select the **Manage slices** button.

3. Manage the slice:
   - To search for a slice, type the slice name in the **Search** bar:

Copyright © 2025 Snorkel AI, Inc. All rights reserved



- To delete a slice, select the **Delete** icon to the left of to the slice name:



- To edit the name of a slice, type the updated name in the specific slice bar.

Manage slices using the SDK

In the SDK, use the `slice` class to create and manage subsets of data slices within a dataset. Slices can be manually or programmatically defined and accessed through the `create()` and `get()` methods.

1. Initialize a slice. Use `Slice.create()` to define a new slice by specifying the dataset, name, and optional description. Add a configuration (`SliceConfig`) to define a slicing function, allowing for automatic assignment of datapoints to the slice based on criteria. This approach is programmatic slices.

2. Add or remove datapoints.
   - **Manually:** Use `add_x_uids(x_uids)` to add specific datapoints by UID to a slice.
   - **Programmatically:** Configure the slice using `SliceConfig` to automatically assign datapoints based on the criteria.

3. Retrieve and list slices:
   - Use `Slice.get(dataset, slice)` to retrieve a specific slice.
   - Use `Slice.list(dataset)` to list all slices associated with a dataset.

4. Updating and managing slices:

Copyright © 2025 Snorkel AI, Inc. All rights reserved

- Modify properties like name or description using `update([name, description, config])`.
- Remove datapoints by UID from a slice using `remove_x_uids(x_uids)`.

Programmatic slices

To create a programmatic slice through the Python SDK, write a slicing function by using `Slice.create` and the `SliceConfig` object that defines the criteria for including datapoints in a slice.

> NOTE
>
> We only support Regex and Keyword templates for programmatic slices.

This code snippet is an example of how to create a programmatic slice using the regex template:

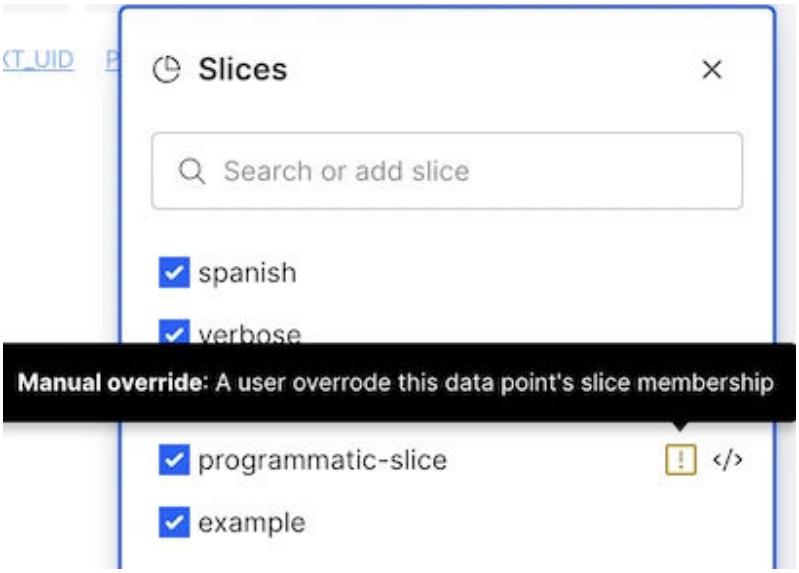
```
from snorkelflow.sdk.fine_tuning_app import FineTuningApp
from snorkelflow.sdk.slices import Slice, SliceConfig
from templates import RegexTemplateSchema
from snorkelflow.utils.graph import DEFAULT_GRAPH

app_name = 'evaluation_workflow_example'
ft_app = FineTuningApp.get(app_name)
topic_disputes_slice = Slice.create(name="topic_disputes",
   dataset=ft_app.dataset_uid,
   config=SliceConfig(
      templates=[
            RegexTemplateSchema(
               field="question",

regex_pattern=r"\b(appeal|appealed|dispute|disputed|disputes)\b",
               case_sensitive=False,
            )
      ],
      graph=DEFAULT_GRAPH,
   )
)
```

- **Slice Overrides:** Allows you to adjust slice memberships manually when an automatic slicing function doesn't fully capture the desired datapoints. When viewing a datapoint, you can use the checkbox to manually override its inclusion in a slice, either adding it to a slice it missed or removing it from one it shouldn't be in.

Copyright © 2025 Snorkel AI, Inc. All rights reserved



- **Inclusion List:** Ensures specific datapoints are part of a slice, regardless of the slicing function's criteria.
- **Exclusion List:** Excludes specific datapoints from a slice, even if they meet the automatic slicing function's criteria.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

(Beta) Prompt development Overview

> NOTE
> This is a beta feature available to customers using a Snorkel-hosted instance of Snorkel Flow. Beta features may have known gaps or bugs, but are functional workflows and eligible for Snorkel Support. To access beta features, contact Snorkel Support to enable the feature flag for your Snorkel-hosted instance.

Prompt development, sometimes called prompt engineering, is the process of designing and refining inputs to guide AI models, such as large language models (LLMs), to produce high-quality, task-specific outputs. In Snorkel Flow, prompt development workflows empower users to create, experiment with, and optimize prompts iteratively. This ensures efficient, precise, and reproducible outcomes for downstream tasks.

A prompt development workflow begins with uploading a text dataset, selecting an LLM, and crafting system and/or user prompts to tailor the model's response. With prompt versioning, users can iterate on their designs, enabling continuous improvement of AI-driven results.

Prompt development enables users to harness the full potential of AI models by guiding them to produce accurate, efficient, and contextually relevant outputs. By optimizing prompts, it reduces the need for extensive fine-tuning, accelerates workflows, and ensures models align with specific business goals or user needs.

Components of Prompt Development

Input Dataset

A prompt development workflow uses an input dataset. This dataset serves as context or examples for the LLM to work with. An input dataset for prompt development should contain relevant data points or scenarios that the LLM can use to align its output with your goals. One example of an input dataset is question and context pairs that can be used for chatbot development by providing an LLM with a prompt and the context needed to answer domain-specific questions.

Best practices

When creating input datasets, it is important to follow best practices:

- Use clear column names that map to LLM processes and predictions. For example, `User Input`, `Context`, and `Expected Response`.
- Make sure you **enable multi-schema annotations** when uploading the input dataset.
- Make sure you select the train split when uploading the input dataset.

> NOTE
> Before creating a prompt development workflow, upload the dataset you plan to use. Learn more about uploading datasets.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

LLMs

The prompt development workflow allows you to experiment with different LLMs to identify the model and prompt combination that delivers the highest quality results.

Prompts

There are multiple types of prompts in prompt development:

- **System prompt:** Sets the overarching tone, rules, or constraints for the LLM. Examples include:
  - Instructing the model to adopt a persona. `You are a financial assistant.`
  - Ensuring responses adhere to specific styles or formats. `Format this using the Google Developer Style Guide.`
- **User prompt:** Focuses on the primary task or request. These prompts add task-specific instructions, often referencing input dataset columns dynamically. Examples include:
  - For **simple tasks** not requiring global rules or a consistent persona. `Translate the following text into French: {Text}.`
  - When the **task or output is fully defined** in the user prompt. `Extract all ingredients from the following recipe: {Recipe}.`

Using multiple prompts together

While user prompts can suffice for simple tasks, combining system and user prompts is recommended for:

- **Consistency:** The system prompt ensures predictable behavior. For example:
  - System Prompt: `You are a data assistant summarizing technical documents.`
  - User Prompt: `Summarize the following text: {Text}.`
- **Complex workflows:** System prompts provide global instructions while user prompts handle granular tasks. For example:
  - System Prompt: `You are an expert in natural language processing. Always respond in JSON format.`
  - User Prompt: `Extract named entities from the following text: {Text}.`

Features and benefits

Key features

- **Workflow Management:** Create, view, and manage prompt workflows via the Prompts page.
- **Dataset Selection:** Choose datasets with clearly displayed columns.
- **Prompt Experimentation:** Test different LLMs, system prompts, and user prompts.
- **Prompt Versioning:** Save, view, and compare prompt iterations.
- **Prompt Version Favoriting:** Star your favorite prompt versions.
- **Prompt Version Renaming:** Rename prompt versions to help track different iterations.
- **SME Feedback:** You can create batches for SME annotation, and view ground truth annotations provided by SMEs on input data and LLM responses for each data point, directly from the **Develop prompt** page.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

- **Multiple Views:** View your input data, LLM responses, and annotations in a table or one data point at a time.

Workflow capabilities

1. **Run Prompts:** Execute prompts and view LLM responses.
2. **Iterative Refinement:** Update prompts iteratively for continuous improvement.
3. **Comparison View:** Analyze multiple LLM responses side-by-side.
4. **Incorporate SME Feedback:** Request and view subject matter expert annotations to improve prompts.
5. **Output Integration:** View output datasets on the **Datasets** page.

Known limitations

These are the known limitations for prompt development:

- Only **text datasets** are supported.
- Datasets can contain a maximum of **400 rows**.
- We assume 100% of your data belongs to **train split**.
- Data is viewable for only one data source at a time.

What's next?

You can learn to create a prompt development workflow.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

(Beta) Create prompt development workflow

> NOTE
> This is a beta feature available to customers using a Snorkel-hosted instance of Snorkel Flow. Beta features may have known gaps or bugs, but are functional workflows and eligible for Snorkel Support. To access beta features, contact Snorkel Support to enable the feature flag for your Snorkel-hosted instance.

Prompt development, sometimes called prompt engineering, is the process of designing and refining inputs to guide AI models, such as large language models (LLMs), to produce high-quality, task-specific outputs. A prompt development workflow begins with uploading a text dataset, selecting an LLM, and crafting system and/or user prompts to tailor the model's response. With prompt versioning, users can iterate on their designs, enabling continuous improvement of AI-driven results.

Prerequisite

Enable required models via the **Foundation Model Suite** to set up and manage external models. Learn more about using external models.

Create a prompt development workflow

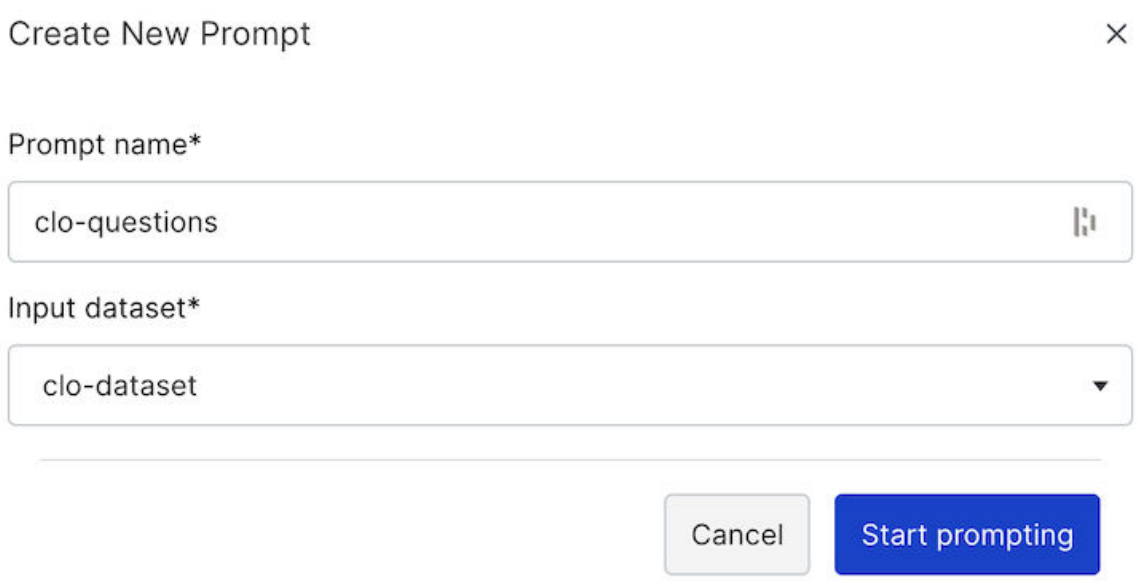
Upload input dataset

1. Navigate to the **Datasets** page.
2. Select **Upload new dataset**.
3. Select the train split when uploading.

Create a prompt workflow

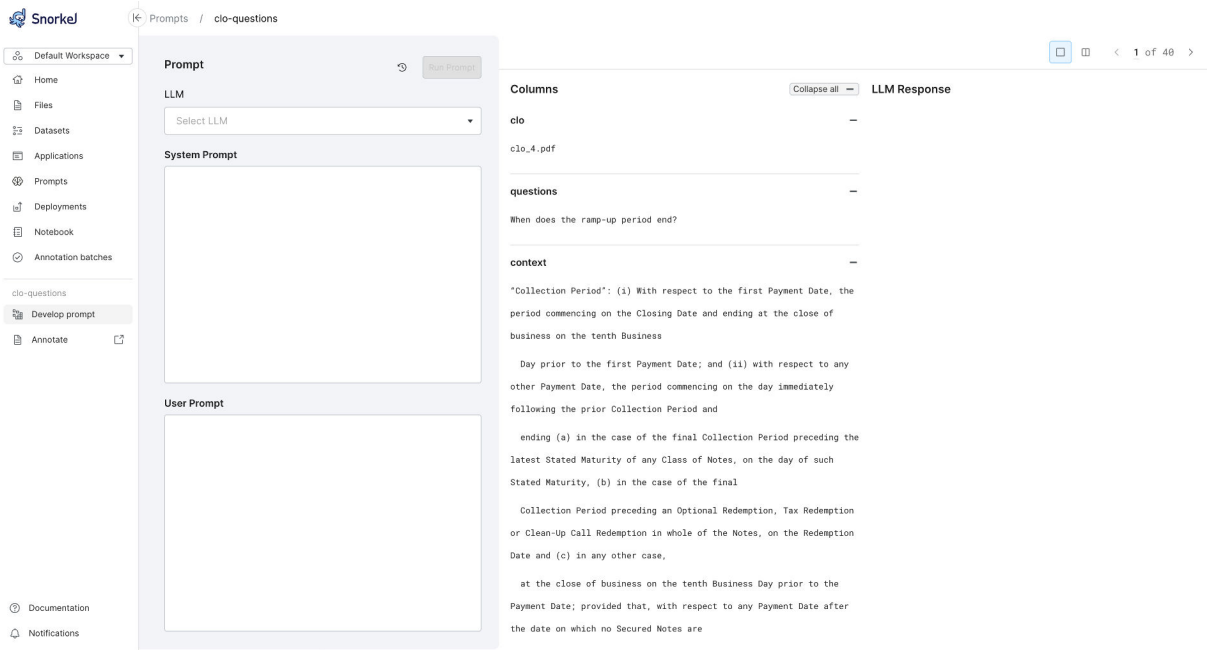
1. Go to the **Prompts** page.
2. Select **Create Prompt**.

Copyright © 2025 Snorkel AI, Inc. All rights reserved



3. Name your workflow.
4. Associate it with an input dataset.

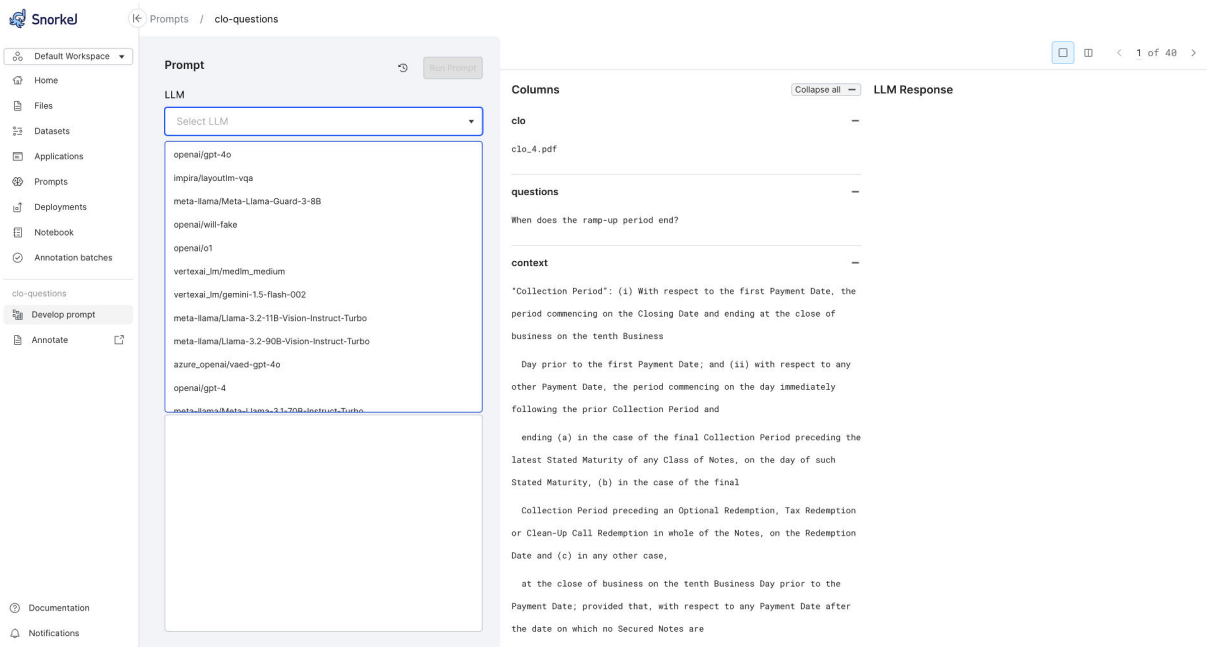
You will see the initial prompt workflow page:



Select model

1. Choose an LLM from the dropdown menu.

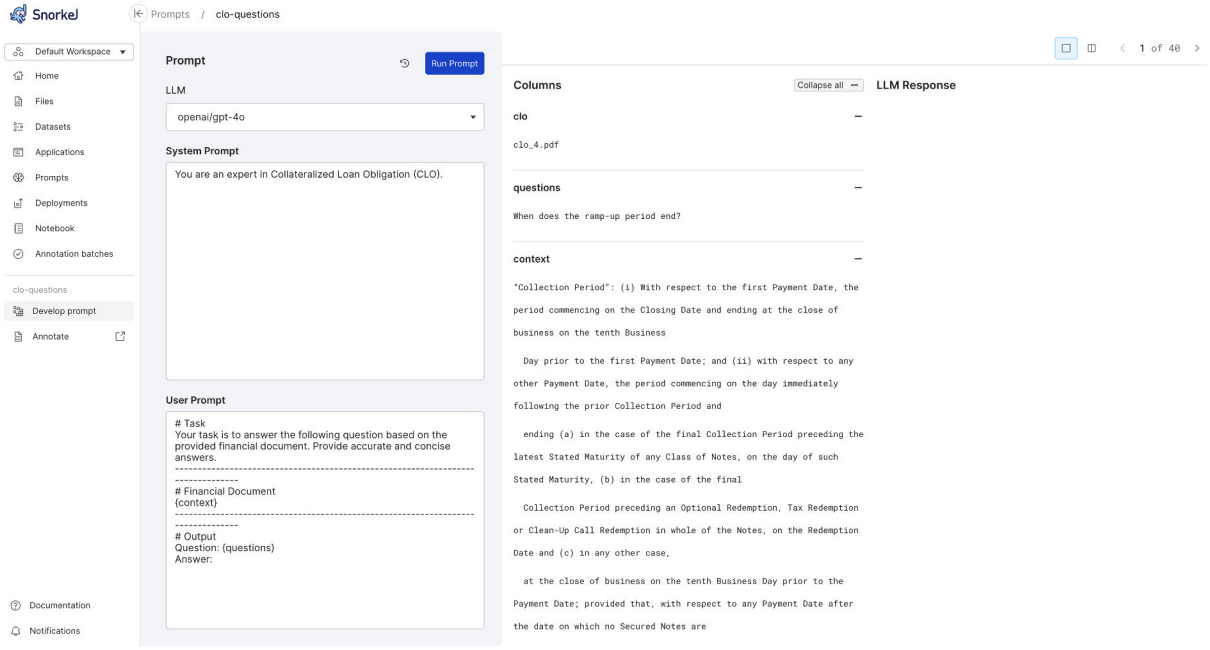
Copyright © 2025 Snorkel AI, Inc. All rights reserved



2. Switch models as needed to optimize results.
3. If required, enable additional models via the Foundation Model Suite.

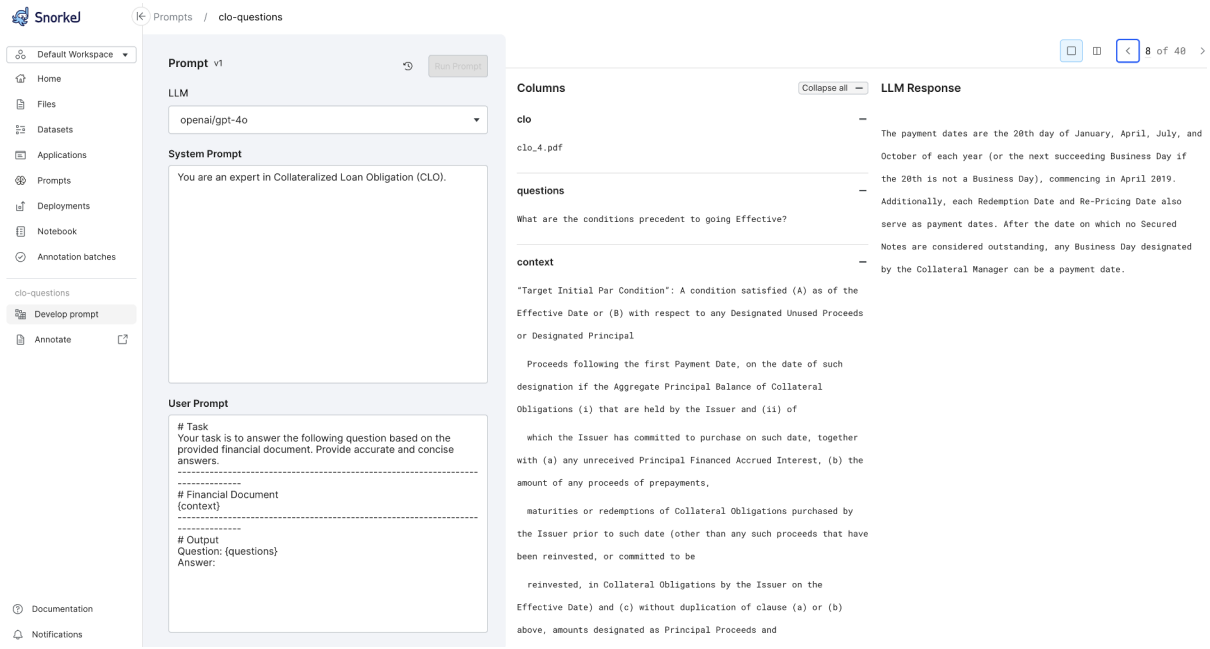
Enter and run prompts

1. Configure prompts, using system prompts, user prompts, or both as needed. For more about prompts, see Prompt development overview.



2. Execute the workflow and review responses for each input in single data point or table view.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

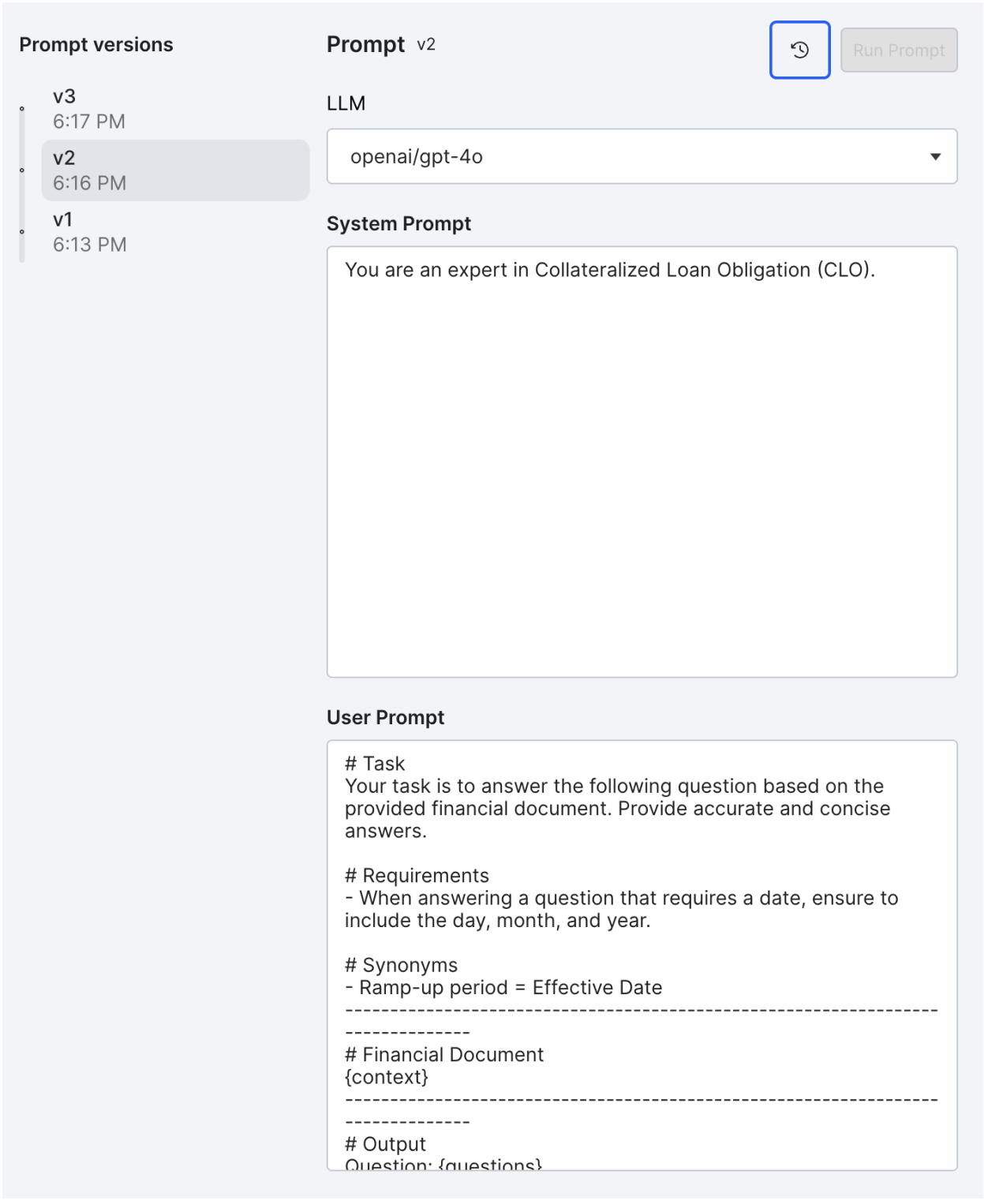


3. To improve responses, iterate your prompts or LLM settings and re-run workflows.

Manage prompt versions

Select the **time back icon** ( ) to access version history and compare prompts and responses over time.

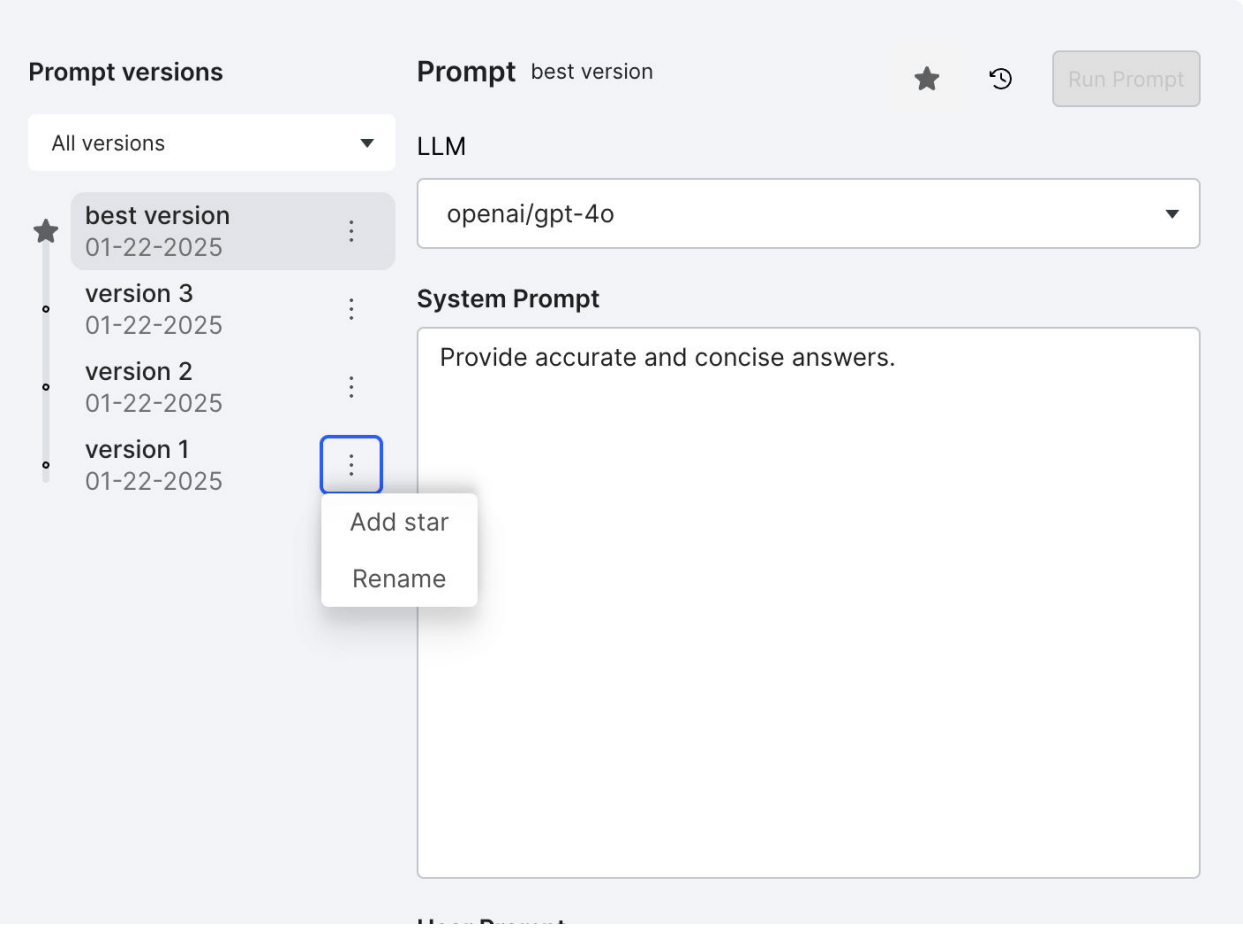
Copyright © 2025 Snorkel AI, Inc. All rights reserved



Favorite and rename prompt versions

Use the prompt versioning feature to add stars and custom names for prompts to help you compare prompts and responses over time.

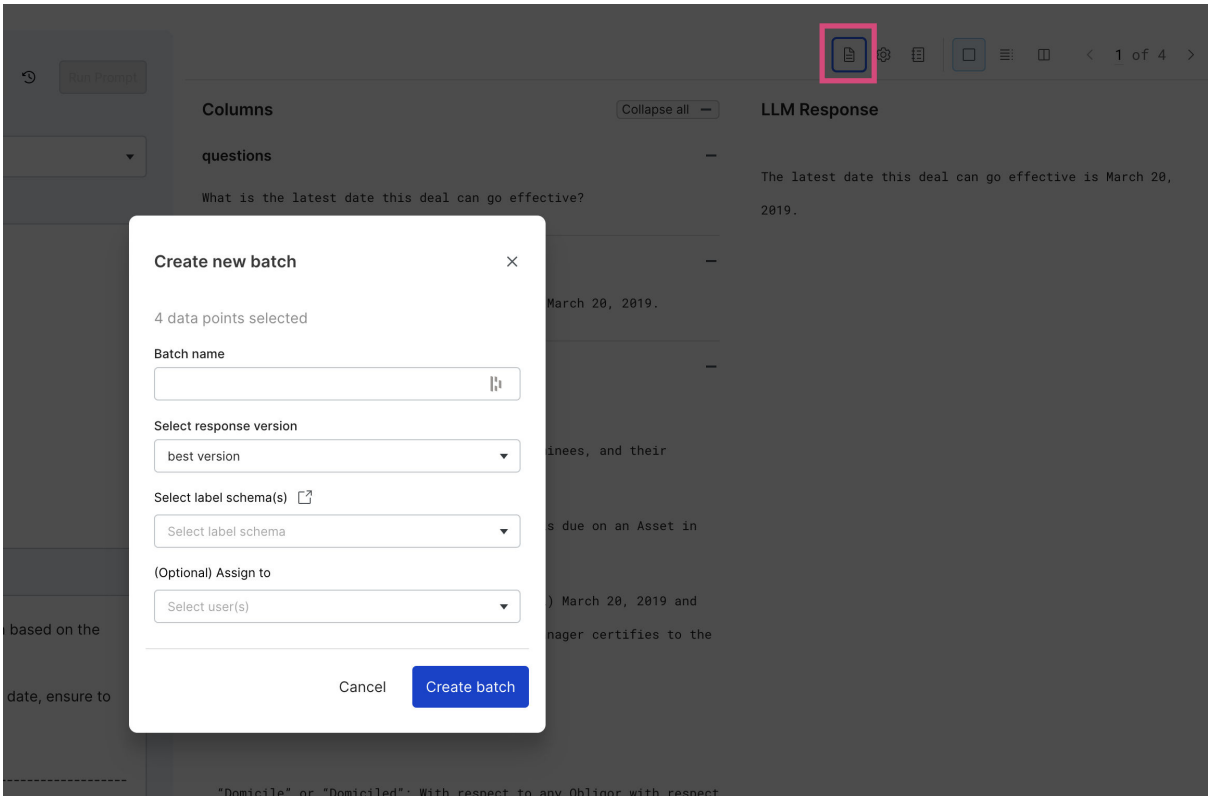
Copyright © 2025 Snorkel AI, Inc. All rights reserved



Incorporate SME feedback to improve prompts

1. Select the **Create new batch** icon to send a batch of prompts and responses to SMEs for annotation.

Copyright © 2025 Snorkel AI, Inc. All rights reserved



2. View ground truth provided by SMEs directly from the **Develop prompt** page to improve on prompts. The SME feedback is displayed next to the **LLM Response**. You can view ground truth provided on the input dataset and the LLM responses.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

LLM Response v49

Hello! Thank you for considering ABC Bank for your personal loan needs. We're here to help you find the right solution to fit your goals.

ABC Bank provides a variety of personal loans tailored to meet diverse needs. The Home Improvement Loan is available for renovations and upgrades, offering loan amounts up to $50,000. The Debt Consolidation Loan helps simplify multiple payments into a single, manageable loan with options up to $30,000. For covering tuition and other education-related expenses, the Education Loan provides funding up to $40,000. Additionally, the Personal Needs Loan offers flexible funding for events, large purchases, and other personal expenses, with amounts available up to $25,000.

Our loans feature competitive rates, terms ranging from 12 to 60 months, and a quick and convenient application process. You can apply online, at any branch, or over the phone, with many applications processed within one business day.

If you have any questions or would like further assistance, feel free to reach out. We're here to help!

Annotations

GT on LLM response

Personal loan	Yes
Mortgage loan	Yes
Car loan	No

Rationale

This response provided helpful information on personal and mortgage loans, but missed car loan.

GT on input data

Personal loan	Yes
Mortgage loan	Yes
Car loan	No

Golden response

This is user's attempt at answering the query in free-form text format, which can be used as a comparison against what the LLM outputs. This may be a part of the input dataset, or may be added later in the annotation suite.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

Copyright © 2025 Snorkel AI, Inc. All rights reserved

Configure and train models

This page walks through how to configure, train, and tune machine learning models on your labeled dataset in **Develop (Studio)**.

After you create and develop labeling functions for your dataset, you can then begin training models!

Once you kick off you model, you can then:

- View your model training progress.
- View and analyze your model results.

Model training in **Develop (Studio)**

To train a model, select **Train a model** in the Models pane. If you have already trained a model, and want to make minor adjustments to it without recreating the model from scratch, select the model from the dropdown, and then click the icon. This will automatically populate the model options based on the model selected.

We provide three modes for model training within the Snorkel Flow application:

- **Fast model**: Runs a single logistic regression model with no customizations for quick model results and iterations.
- **Custom model**: Allows you to customize your model architecture, hyperparameters, and hyperparameter searches.
- **AutoML**: With one click, allows you to run and exhaustive search over hyperparameters and model architectures, returning the best performing model

See Supported modeling libraries for more information about the different modeling libraries that we support.

Fast model training

The **Fast model** option runs a basic logistic regression model on a sample of your training data. The model runs significantly fast, making this an excellent option to quickly iterate and assess models while actively developing new labeling functions (LFs). You can select the following training options:

- **Training Set size**: The number of data points on which to run the model.
- **Auto-train Fast Models**: Turn this on to automatically train a fast model on a sample of your data when your set of active LFs changes.

Custom model training

The **Custom model** option gives you full control over the model architecture, hyperparameters, and hyperparameter searches. We offer default model configs for several commonly used models, which provides a good starting point for model training.

The following sections detail the options that are available to customize your model.

General options

- **Build from previous model**: Select a previously executed model to automatically populate the model options based on its settings. This allows you to make minor adjustments to the existing model without the need to recreate it from scratch.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

- **Name**: A name for your model. The model's name is unique per model, per application. Model names cannot be longer than 30 characters.
- **Description**: An optional description of your model.
- **Model architecture**: The model type (e.g., logistic regression).
- **Training sets**: The training set of Snorkel labels that you'd like to train on. Choose from an existing set, or create a new training set.
- **Input fields**: The columns of the data that you want to train your model on. This option is unavailable for span-aware models.

Note
The order of the fields matter for BERT models because they truncate the sequence length to the specified `max_seq_len`. Therefore, you should put the columns with more important information or those of shorter length first.

Features libraries

- **Add LF labels for model training and inference**: This option is only available for Scikit-Learn models. If selected, it will make the LF labels (i.e., labels that the LF outputs in the selected training set) available to the model as features. The model is likely to overfit to the LF labels, so this option is only recommended for LFs with high coverage and accuracy. Models that are trained with this option are currently not available for application export.

Train options

> NOTE
> Model configs that are computationally expensive have been marked with a `*`, to indicate they may have long training times without a GPU.

- **Oversample data to match class distribution on valid split**: if checked, Snorkel Flow will oversample the generated labels to match the class distribution of the valid split. This option is useful for applications with imbalanced classes.
- **Override LF labels with ground truth labels where available**: If checked, Snorkel Flow will override the generated labels with ground truth labels for model training, whenever they are available.
- **Tune decision threshold on valid split**: This option is only available for binary applications. If checked, it will use the valid split to pick a decision threshold that maximizes the primary metric (either accuracy or F1 score) on the valid split. This is especially useful for applications where there is significant class imbalance, as often occurs in text extraction applications.
- **Train noise-aware model using probabilistic labels**: This option is only available for Scikit-Learn models and cannot be combined with the **Oversample data to match class distribution on valid split** option. If selected, it will train the selected model using probabilistic training labels that the label model outputs, instead of integer predictions. This option can be helpful for applications with high cardinality (a large number of classes).
- **Filter out low confidence labels**: This option filters out Snorkel-generated labels that have low-confidence (i.e., close to random).
- **Include dev split when training**: This option uses data points in the dev set to train the model, in addition to data points in the train set. Note that using this option can result in the trained model being overfit to the dev set, and therefore the suggestions in the Analysis pane will not be as effective.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

- **Predict on dev only**: This option runs predictions on the dev split only. This speeds up the model inference step. If you select True, then to treat the dev split as a holdout dataset, also set **Include dev split when training** to **False**.
- **Max. Runs**: This option specifies the maximum number of models to train with configs that are sampled from the specified hyperparameter search space.

Model options

As you make selections for your model architecture, train options, etc., the model config will automatically update to reflect those changes. You can also manually customize the JSON to define your desired model or to define a hyperparameter search. Clicking the **Reset** button will reset the config back to the default values for the options that you selected.

Hyperparameter search

You can define a grid search over hyperparameter options in the model config. To specify a set of hyperparameters to search over, set the value of the hyperparameter in the config to be a dictionary with a single key `"SEARCH"`, whose value is the list of values to search over.

> NOTE
>
> You can only do hyperparameter search if you provide a valid split.

For example, to search over models with regularization parameter `C` of `1, 10, and 100`, set `"C": {"SEARCH": [1, 10, 100]}`. Snorkel Flow will then train three models:

- one with `C = 1`
- one with `C = 10`
- one with `C = 100`

Snorkel Flow will then report the hyperparameter options and metrics for the model with the best performance on the valid split.

You can also search over combinations of hyperparameters. For example, if you set `"penalty": {"SEARCH": ["l1", "l2"]}` and `"C": {"SEARCH": [1, 10, 100]}` in the model config, Snorkel Flow will train six models, using each combination of `"penalty"` in `["l1", "l2"]` and regularization parameter `"C"` in `[1, 10, 100]`.

> NOTE
>
> By default, Snorkel Flow searches a maximum of 8 trials at a time, so if you do specify more than 8 settings, it will randomly sample 8 models to run.

An example configuration of a hyperparameter search is shown below:

Copyright © 2025 Snorkel AI, Inc. All rights reserved

```
{
  "classifier": {
    "classifier_cls": "LogisticRegression",
    "classifier_kwargs": {
      "C": { "SEARCH": [1, 10, 100] },
      "penalty": { "SEARCH": ["l1", "l2"] },
      "solver": "liblinear"
    }
  },
  "text_vectorizer": {
    "vectorizer_cls": "CountVectorizer",
    "vectorizer_kwargs": {
      "ngram_range": [1, 2],
      "n_features": 250000,
      "lowercase": false
    }
  }
}
```

Supported Model Options

For the model train options, the `classifier_kwargs` key can contain any supported third-party keyword arguments. Transformer models are trained using their `Trainer` class. Read the transformer `TrainArguments` for a list of supported arguments. Additionally, view model-specific arguments for any `sklearn` models.

Unique Snorkel Flow Transformer Arguments

- load_best_model_at_the_end: This is a `TrainArgument` supported argument; however, when a user specifies this arg, Snorkel Flow will also pass in values for `metric_for_best_model`, `evaluation_strategy`, and `save_strategy`. This is done for convenience since those parameters are required to load the best model.
- early_stopping: This is not a `TrainArgument` supported argument. This leverages a `CallBack` for early stopping. By default, this is `True`.
- early_stopping_patience: This is not a `TrainArgument` supported argument. This is a parameter passed to the `CallBack` for early stopping. By default, this is `2`.

AutoML model training

The AutoML training option allows you to run an exhaustive search over hyperparameters and model architectures with one click, returning the best performing model.

> NOTE
>
> To run AutoML, you must have ground truth labels in your valid split.

To start an AutoML run:

Copyright © 2025 Snorkel AI, Inc. All rights reserved

1. From the Models pane, click **Train new model**, then select the **AutoML** tab.
2. The default selection for search strategy is **Grid Search**. You can also choose **Bayesian Optimization** from the dropdown, which will start a smart range search on the corresponding search space. See the Ray tune API docs for more information about Bayesian optimization tuning.
3. The default selection for model architecture is **Logistic Regression**, but you can search over multiple model architectures using the **Model Architectures** drop down. Additional model architectures that are available are **XGBoost**, **DistilBERT**, and **Fast Logistic Regression**.
4. The default selection for input fields is preset, but you can choose which fields are being used in the **Input Fields** drop down. Your fields preferences will be saved.

AutoML defaults to the latest training set. The progress of model runs can be tracked from the **Models** side bar. AutoML jobs are large and can take 10-12 hours to complete. However, while AutoML runs, you can:

- See the best model that has been trained so far. You can register that model, and start working with it as the AutoML job continues.
- View all the configs that have been trained so far under the **best model trained so far** panel. This allows you to view the search space for the AutoML run, and understand how different hyperparameters impact the model score.

Once an AutoML run is complete, you can see the best performing model across all the models trained in the job. If you hover next to the model name, you can see what hyperparameters returned the best performing model.

Span aware models

For extraction and entity classification applications, we provide span aware models that use features from the extracted spans and their contexts. To adjust the default options, set the values in the model config (under Model options) under `extraction_options`:

- `left_context_len`: The number of characters to the left of the span to include in the context.
- `right_context_len`: The number of characters to the right of the span to include in the context.
- `mask_span`: Replace the span text itself with a special `[Span]` token. This helps prevent over-fitting to the exact value of the spans in the dataset.

Distributed training with Ray Train

Beginning in version 0.91, Snorkel Flow supports distributed training with Ray Train for certain models. Enabling this feature moves model training from a single-threaded process in model-trainer, to a distributed job on the ray-gpu cluster. This is designed to initially support the HuggingFace BERT-family of models for single label classification.

> NOTE
>
> Given its experimental nature, this feature may lead to issues or unexpected behaviors. If problems arise, disable Ray Train in your user settings and retry the training job. Additionally, for support with large training jobs, administrators can contact the success team. Any feedback on the feature is welcome.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

When to consider enabling Ray Train

This feature is experimental, and turned **off** by default. Due to its development stage, it is recommended that you keep this feature turned off to maintain stability in your model training processes. That being said, here are some use cases where you may want to turn **on** Ray Train:

- **You have a single label classification application**. Currently this is the only application type that is supported.
- **You are training a BERT or DistilBERT model**. Logistic regression, XGBoost, and all other model frameworks will not be affected by Ray Train.
- **You are facing out-of-memory errors with large datasets**. This may show up as a snackbar error or crash in your model training job.
- **Snorkel Flow has access to two or more GPUs**. Ray Train will only lead to performance gains if your instance has access to multiple GPUs.

How to enable Ray Train

Follow these steps to enable Ray Train on your Snorkel Flow instance:

1. Click your user name in the bottom right corner of your screen.
2. Click **User Settings**, then click **Feature Flag Management**.
3. Toggle the **RAY_TRAIN** flag to the on position.

After activation, model training can be configured as usual. Training jobs for BERT and DistilBERT will automatically run on the distributed system. They will potentially run slower, but have a reduced memory footprint.

Troubleshoot model training

You may encounter the following error when training a model:

```
Cannot train model: the resulting training set has labels for only one class
[X] (XXXX sample(s)).
Need labels for more than one class to train a model.
```

This error occurs when the label model is only confident in producing labels for a single class.

This can happen for a few reasons:

- One class is much more dominant than the others, so the label model is never confident in the other class labels because they are rare.
- The labeling functions for the other classes have low coverage and compete significantly with a high-coverage labeling function for the dominant class.
- The label model estimates that labeling functions for the other classes are very imprecise.

To fix this error, you can:

- (Preferred) Write more labeling functions for the other classes. This may include adding more training data in the areas with low coverage. This is the preferred solution because it will help you improve the quality of your model.
- (Alternative) Set **Filter out low confidence labels** to **False**. However, keep in mind that this will introduce low-confidence labels to your end model.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

Copyright © 2025 Snorkel AI, Inc. All rights reserved

Model training progress stages

This guide explains the training progress stages you will encounter when you initiate model training in a Snorkel Flow Application. Monitoring the training progress bar is useful as it helps you estimate the time remaining in training and understand the duration of each stage. This information allows you to efficiently manage your time, possibly engaging in other tasks or refining labeling functions while the model trains.

After you have configured your model and initiated training, the model progress bar displays various stages based on whether you are training with a new dataset or using AutoML. Once training is complete, you can view and analyze the model results.

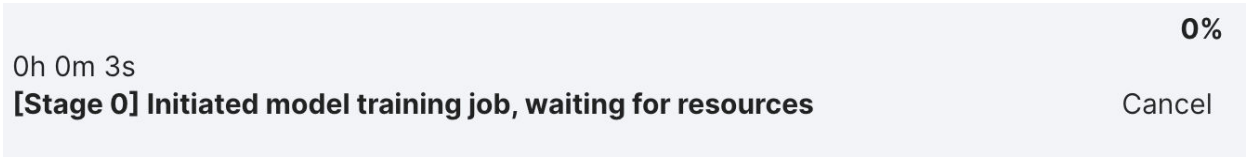
Standard Training Progress Stages

These are the typical stages after you start training a single model in a Snorkel Flow Application.

> NOTE
>
> If you start training a model from an existing training set, only Stages 3-6 will be applied.

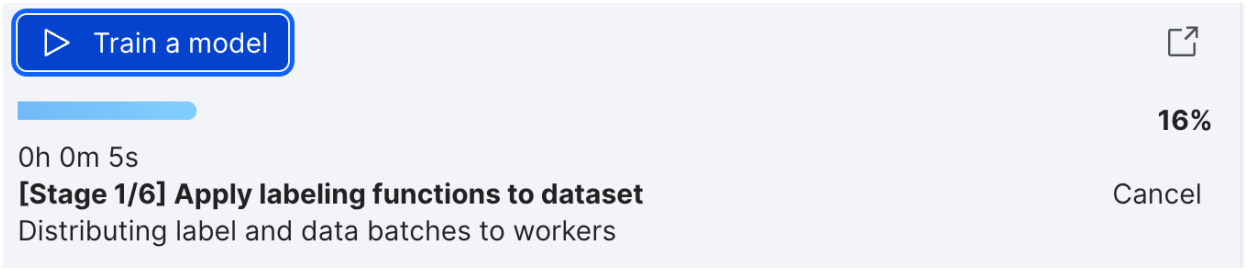
Initiated model training job, waiting for resources

This stage indicates that your modeling job is in the queue and awaiting available computing resources.



Apply labeling functions to dataset

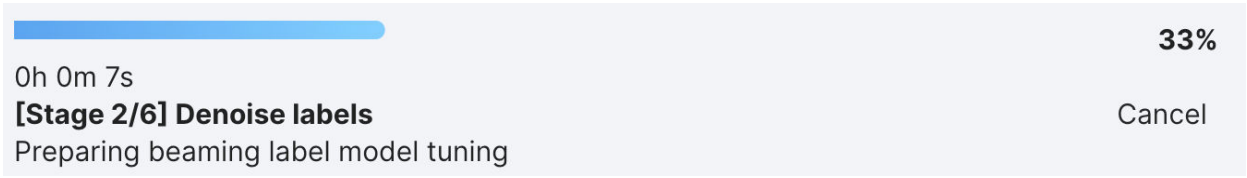
Snorkel Flow processes all your labeling functions across your training, validation, and/or development datasets. The duration of this stage depends on the number of labeling functions and the size of your datasets.



De-noise labels

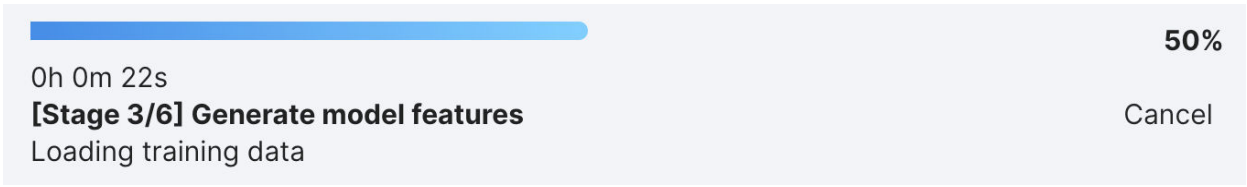
The weak supervision algorithms aggregate all labeling functions and denoise them, generating a final programmatically labeled training dataset. If you have enabled the option to **Train noise-aware model using probabilistic labels**, this stage might take longer than usual.

Copyright © 2025 Snorkel AI, Inc. All rights reserved



Generate model features

During this phase, Snorkel Flow tokenizes and vectorizes your data to prepare it for the training phase.



Train model

The chosen model architecture is applied to the vectorized data. Updates such as **Step X of Y** indicate the completion of mini-batches and training updates. Complex models, like BERT, require more time compared to simpler models, like logistic regression or XGBoost.

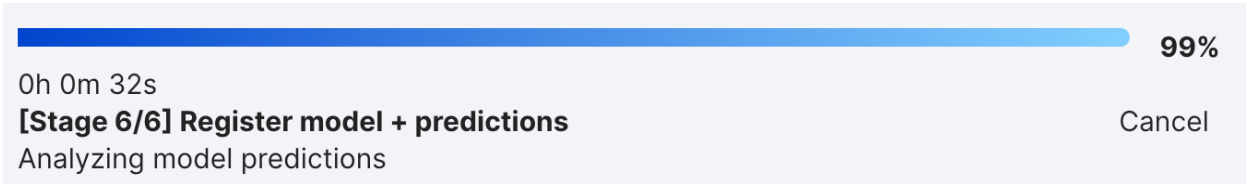
Perform model inference

Once training is complete, Snorkel Flow runs the model on your dataset to generate predictions for evaluation. To accelerate this process, you can set **Predict on dev only** to **True**. This setting ensures that predictions are generated for only the development set, treating it as a holdout dataset for more quick error analysis.

Set **Include dev split when training** to **False** to exclude the development data from the training set, ensuring that there isn't training data leakage.

Register model + predictions

The final stage involves saving your model and predictions and calculating evaluation metrics. Although the progress bar might show 99%, completing this stage can take longer than expected, especially with large datasets or complex prediction tasks.

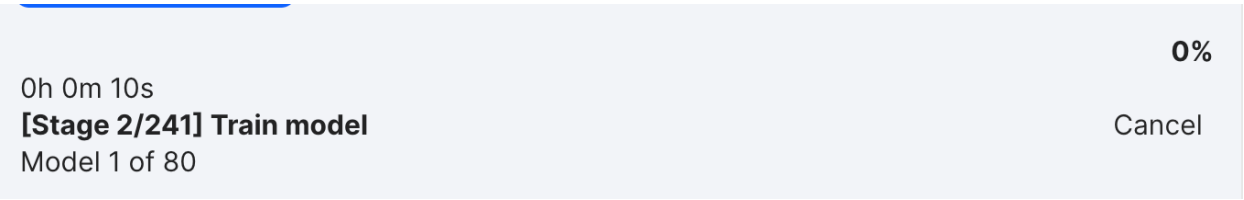


AutoML training process stages

When using AutoML for hyperparameter tuning, the progress stages displayed become much larger, reflecting the extensive number of stages due to the iteration of models. Each model iteration involves three main stages—featurization, training, and inference—repeated across all models, plus the final step

Copyright © 2025 Snorkel AI, Inc. All rights reserved

of model registration. Label function application and denoising are conducted once at the beginning of the process. Here is an example of what you might observe during this process:



Progress Percentage Calculation

The progress percentage during AutoML training is calculated using this formula:

```
progress % = (current stage / (total number of stages + within stage
progress)) * 100
```

For example, if you are currently in the model training stage, which is the fourth stage of six total stages, the initial progress would be calculated as `4/6 = 66%`. As the training progresses to step 50 out of 100 steps within this stage, the calculation adjusts to reflect the intermediate progress, moving the progress calculation to `4.5/6 = 75%` as it approaches the next stage.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

Model training in the SDK

This guide details the process for training models outside the Snorkel Flow UI using the Python Software Development Kit (SDK). This method is particularly useful for users seeking to implement specialized models not included in the default model offerings, or to apply advanced modeling techniques not supported by the UI. Training with the SDK provides the flexibility to experiment with various model types using programmatically labeled data from Snorkel Flow.

The Snorkel Flow SDK allows for seamless integration with any modeling pipeline, enabling the training of custom models with datasets curated in Snorkel Flow. After training, models and their predictions can be registered back to the platform, allowing users to conduct detailed error analyses and make iterative refinements to their models and datasets, enhancing overall model performance.

For more information, see the following resources:

- Tutorials
- Model zoo
- Supported modeling libraries
- SDK reference

Copyright © 2025 Snorkel AI, Inc. All rights reserved

Best practice: Continuous model validation

This article provides a comprehensive guide to continuous model validation, emphasizing its critical role in maintaining the accuracy and effectiveness of your deployed Snorkel Flow models. Continuous model validation refers to the regular assessment of a model to ensure it continues to perform as expected despite changes in the data it processes or in its operational environment. This article details methodologies for regularly assessing model performance, which equips you to identify and address deviations caused by changes in the underlying data, also known as data drift. Continuous model validation ensures the long-term reliability of machine learning models used in production. Snorkel recommends you be conduct model validation periodically or when you suspect performance degradation. Adhering to these practices sustains operational success because the validation helps mitigate the risk of performance decline in response to changing data landscapes and shifting business needs.

Importance of model validation

Continuous model validation is critical in the machine learning lifecycle. This best practice ensures that models maintain high performance despite changes in incoming data. Regular checks help identify deviations from expected performance metrics like Accuracy, F1 Score, and Precision, which can degrade due to shifts in the underlying data. For example, a model predicting seasonal temperatures might require adjustments as conditions change throughout the year. Consistent evaluation is indispensable for long-term reliability and provides insights into setting the right validation frequency based on your model's impact and the data it processes.

Establishing a model validation cadence

It is crucial to choose an appropriate validation frequency. The cadence should reflect the importance of the model's output and the dynamic nature of the data it handles. These are the key considerations for developing your model validation schedule:

1. **Severity of Impact:** Models critical to business operations or with high stakes in output accuracy should undergo more frequent validations.
2. **Nature of Data:** Models using data that frequently updates or significantly changes should be validated more often to quickly adapt to new conditions.
3. **Volume of Data:** High-volume data applications are more susceptible to variations in model performance, necessitating shorter intervals between validations.

Snorkel AI recommends collaborating with your Snorkel Machine Learning Success Manager to tailor a validation strategy that best supports your needs.

Steps for Effective Model Validation

To ensure comprehensive validation, follow these steps as they fit your specific application requirements:

1. **Assess the latest baseline metrics.** Start by reviewing the current model's latest baseline metrics. These metrics are helpful to compare the performance of the model on new production data.
2. **Identify representative data samples.** Gather a diverse dataset from production that covers various times, locations, and all relevant labels. Aim for a random yet representative sample.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

3. **Label the validation dataset.** If not already labeled, manually label the new dataset in Snorkel Flow for direct validation use.
4. **Evaluate model performance.** Analyze how the current model performs with the new data. If the results fall below acceptable thresholds, consider revising the labeling functions and model configurations.
5. **Refine and retrain your model.** Use these insights to update your labeling functions and retrain the model using the enriched data set within Snorkel Flow.

Note: While Snorkel Flow facilitates steps 1, 3, 4, and 5, you will need to conduct step 2 externally.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

Supported modeling libraries

This page describes the three major modeling libraries that Snorkel Flow supports: Scikit-Learn, XGBoost, and HuggingFace's Transformers. If you think that your application requires a custom library please reach out to a Snorkel representative.

Scikit-Learn

Snorkel Flow supports the logistic regression and k-nearest neighbors classifiers from the Scikit-Learn library, as set by the `classifier_cls` option. To configure the classifier, pass keyword arguments as a dictionary to the `classifier_kwargs` option, which is forwarded directly to the classifier initializer. See the Scikit-Learn logistic regression documentation and the Scikit-Learn k-nearest neighbors documentation for information about all supported keyword arguments.

To perform classification on text fields, we also support several Scikit-Learn vectorizers to transform the text into features. These are set by the `vectorizer_cls` option, which currently supports CountVectorizer, HashingVectorizer, and TfidfVectorizer. To configure the vectorizers, pass keyword arguments as a dictionary to the `vectorizer_kwargs` option, which is forwarded directly to the classifier initializer.

An example model configuration for Scikit-Learn is shown below:

```
{
  "classifier": {
    "classifier_cls": "LogisticRegression",
    "classifier_kwargs": {
      "C": 10,
      "penalty": "l2",
      "solver": "liblinear",
      "random_state": 123
    }
  },
  "text_vectorizer": {
    "vectorizer_cls": "HashingVectorizer",
    "vectorizer_kwargs": {
      "ngram_range": [
        1,
        2
      ],
      "n_features": 250000,
      "lowercase": false
    }
  }
}
```

XGBoost

Snorkel Flow supports XGBoost for gradient boosted trees through the XGBClassifier in the Scikit-Learn Python API. You can set this using the `classifier_cls` option. Similar to logistic regression from the Scikit-Learn library, you can configure the XGBoost classifier by passing keyword arguments as a

Copyright © 2025 Snorkel AI, Inc. All rights reserved

dictionary to the `classifier_kwargs` option. The dictionary is forwarded directly to the classifier initializer. See the XGBClassifier documentation for information about all supported keyword arguments.

To perform classification on text fields, the same vectorizers from the Scikit-Learn library are supported, as defined in the Scikit-Learn section.

An example model configuration for XGBoost is shown below:

```
{
  "classifier": {
    "classifier_cls": "XGBoostClassifier",
    "classifier_kwargs": {
      "n_estimators": 100,
      "max_depth": null
    }
  },
  "text_vectorizer": {
    "vectorizer_cls": "CountVectorizer",
    "vectorizer_kwargs": {
      "ngram_range": [
        1,
        2
      ],
      "max_features": 250000,
      "lowercase": false
    }
  }
}
```

HuggingFace Transformers

Snorkel Flow supports pre-trained BERT classification models through the HuggingFace's Transformers library. You can specify a pre-trained model with the `pretrained_model_name` option. The HuggingFace documentation lists the full set of options for pre-trained models.

> **TIP**
> We recommend using DistilBERT models because of their smaller model size and lower compute cost.

To configure the HuggingFace Tokenizer, we support the `tokenizer_kwargs` option, which is passed to the initializer and supports the options that are available in the pretrained HuggingFace Tokenizers documentation. To configure the Adam Optimizer, we support the `adamw_kwargs` option, which is passed to the initializer and supports the options that are available in the PyTorch AdamW documentation.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

> **TIP**
>
> We recommend keeping the `weight_decay = 0.0` within the AdamW optimizer because we do not want to apply weight decay to the bias and layer norm parameters. The external `weight_decay` option correctly excludes these parameters.

The remaining options are used to configure model training and inference, which we support via PyTorch. In particular, the `freeze_bert_embeddings` option enables you to train only the final classification layer. This option can yield good performance at lower computational cost.

An example model configuration for HuggingFace Transformers is shown below:

```
{
 "pretrained_model_name": "distilbert-base-uncased",
 "tokenizer_kwargs": {
 "do_lower_case": true
 },
 "adamw_kwargs": {
 "lr": 0.00005
 },
 "weight_decay": 0,
 "freeze_bert_embeddings": true,
 "max_sequence_length": 32,
 "num_train_epochs": 1,
 "train_batch_size": 64,
 "max_grad_norm": 1,
 "gradient_accumulation_steps": 1,
 "lr_warmup_steps": 0,
 "predict_batch_size": 64
}
```

Copyright © 2025 Snorkel AI, Inc. All rights reserved

View and analyze model results

This page walks through the model results and metrics that are available in the Models pane. The Analysis pane provides additional charts, metrics, and guidance on how to refine your models. See Analysis: Rinse and repeat for more information.

Once you've trained a model, if you have ground truth, the **Best Accuracy** and **Latest Accuracy** are displayed in the top left corner of your application. If you don't have ground truth, then the values display `--`. This allows for an easy comparison between the current model's performance and the best-performing model among all iterations. You can use the drop downs to change the metrics that are shown.

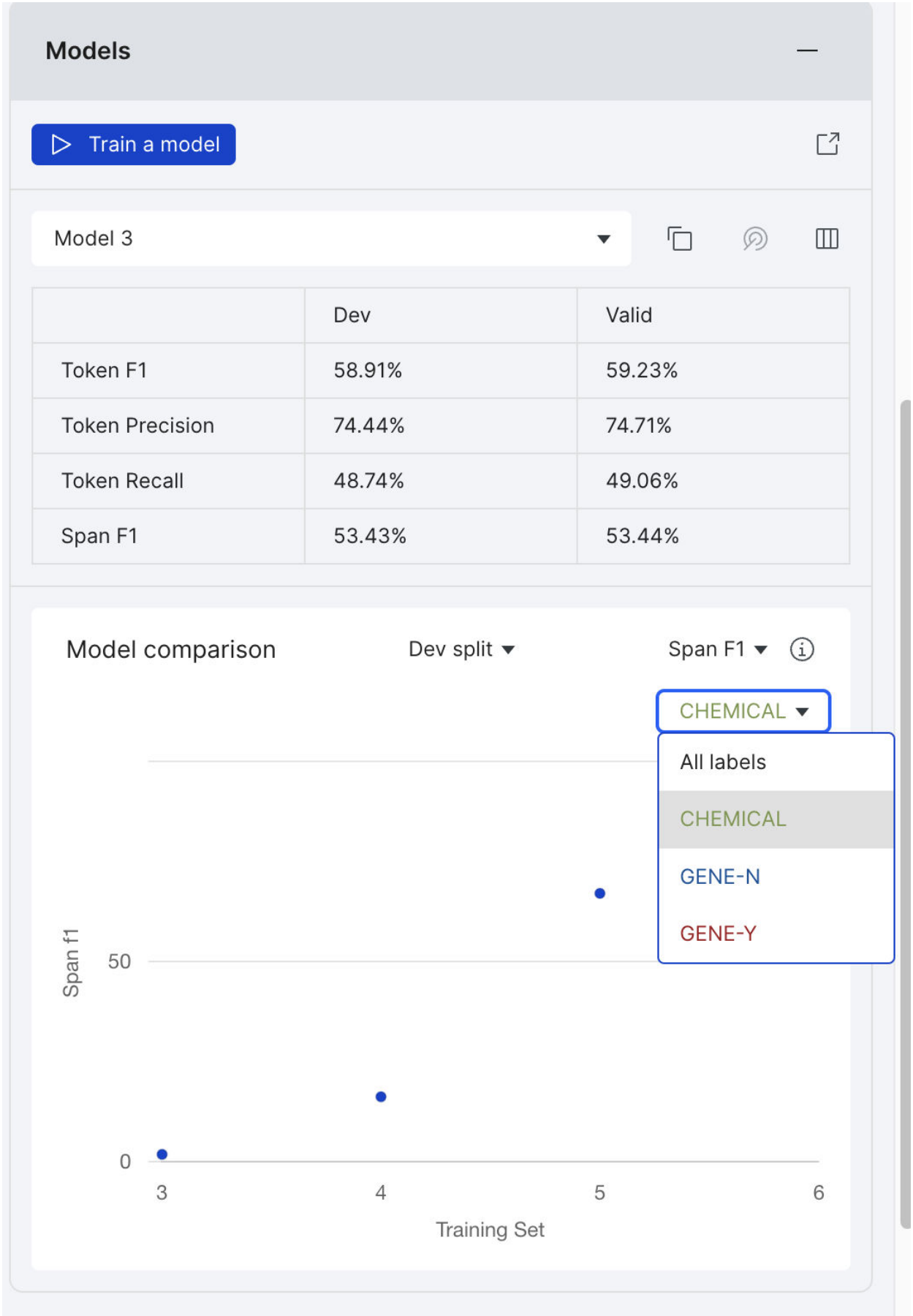
Models pane

By default, the **Accuracy** and **F1 Macro** for your dev and valid splits for the most recent model are shown in the table. You can change these metrics in the Advanced options of the Models modal.

By default, the **Model comparison** chart shows the F1 macro of the dev split for all models. This provides a good visual comparison of how well your models are performing and whether new iterations are improving model performance. You can use the drop downs to change which split and which metric is shown in the graph.

By default, the class-level dropdown is set to "All labels," which displays metrics at an aggregate level across all classes. To view class-level performance, select individual classes from the dropdown menu.

Copyright © 2025 Snorkel AI, Inc. All rights reserved



Copyright © 2025 Snorkel AI, Inc. All rights reserved

The Models pane offers additional options:

- For more information about your model, click the icon in the Models pane to bring up the Models modal.
- If you have already trained a model, and want to make minor adjustments to it without recreating the model from scratch, select the model from the dropdown, and then click the icon. This will bring up the Train a model modal and automatically populate the model options based on the model selected.
- To restore the labeling functions (LFs) that were active when when the model with the best performance was trained, click the icon. This will overwrite the current set of active LFs.
- To view model metrics for the test set, click the icon. This is hidden by default to avoid looking at test set scores while hyperparameter tuning and training different models. We recommend having another held out test set outside Snorkel Flow for evaluating final performance as well.

Models modal

For more information about your model, click the icon in the Models pane to bring up the Models modal. Here, you can view and customize metrics for all models that have been trained.

The following options are available when you open the Models modal. To view more options, click the icon or click **Advanced options**.

- **Show Test**: View the model metrics for the test set. This is hidden by default to avoid looking at test set scores while hyperparameter tuning and training different models. Select **Hide Test** to remove the test set metrics from the table.
- **Commit**: Once you are finished iterating and developing models, click **Commit** to save the model into the application pipeline. Committing a model is required for it to be used in deployment. In addition, you need to commit a model to be able use as input to downstream nodes (e.g., in a hierarchical child block).
- Hover over the icon view the config of the search space for which the best performing model was returned. Click the icon to copy the config to the clipboard.

More options

Click the icon to view more options:

- **View search space config**: View the config of the search space for which the best performing model was returned.
  - **Copy config**: In the **Search space config** modal, select this to copy the config to the clipboard.
- **Edit model name/description**: Edit the model name and description.
- **Export as CSV**: Download the predictions from the model for all splits as a CSV file.
- **Delete this model**: Delete the selected model. It will ask you to confirm before proceeding with deletion.

Advanced options

Click **Advanced options** in the top right corner of the modal to expose the advanced options bar.

- **Sort by**: Choose any metric on the valid set in the dropdown to sort the models. If there are ties, then the order is determined first by F1, and then by accuracy scores.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

- **Slices Filter**: Use the dropdown to view model metrics on subsets of the data based on slices that you have created. Note that **No Slices Assigned** refers to all data points that do not have an error analysis slice applied to them.
- **Postprocessors**: When turned on, this applies all post processors for the model node before calculating model metrics. This is turned on by default.
- **Model Metrics**: Selecting metrics from the **Metric types** option changes the metrics that are displayed in the tables in the modal as well as in the Models pane. Selecting a metric from the **Primary metric** option changes the metric that is optimized for during hyperparameter search and threshold setting.
  - You can also register custom metrics in the Python SDK using `snorkelflow.client.metrics.register_custom_metric`.
- **Enable model metrics cache**: Check the box to enable caching for your model metrics.

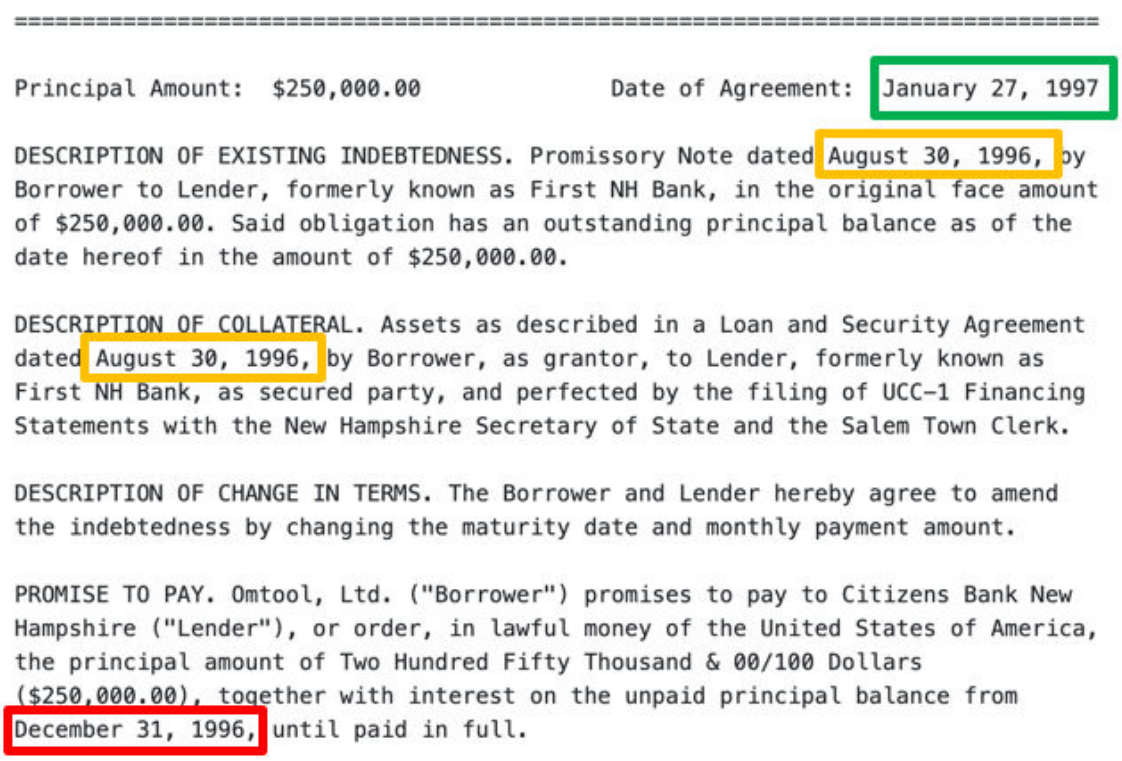
[Extraction] document-level operators and metrics

For information extraction applications, we often have different levels at which we'd like to evaluate final model performance:

- **Span level**: The number of candidate spans that were classified correctly. For example, in the Information extraction: Extracting execution dates from contracts tutorial, this would be how many dates we correctly marked as either POSITIVE (an execution date) or NEGATIVE (not an execution date).
- **Document level**: This measures Snorkel's performance at extracting one (or several) values from the overall document. For example, the execution date of the document.

In many settings, such as the Information extraction: Extracting execution dates from contracts tutorial, we care less about classifying all dates correctly (the span level score), and instead want to find the single execution date of the document with high accuracy (the document level score).

Copyright © 2025 Snorkel AI, Inc. All rights reserved



==========

Principal Amount: $250,000.00 Date of Agreement: January 27, 1997

DESCRIPTION OF EXISTING INDEBTEDNESS. Promissory Note dated August 30, 1996, by Borrower to Lender, formerly known as First NH Bank, in the original face amount of $250,000.00. Said obligation has an outstanding principal balance as of the date hereof in the amount of $250,000.00.

DESCRIPTION OF COLLATERAL. Assets as described in a Loan and Security Agreement dated August 30, 1996, by Borrower, as grantor, to Lender, formerly known as First NH Bank, as secured party, and perfected by the filing of UCC-1 Financing Statements with the New Hampshire Secretary of State and the Salem Town Clerk.

DESCRIPTION OF CHANGE IN TERMS. The Borrower and Lender hereby agree to amend the indebtedness by changing the maturity date and monthly payment amount.

PROMISE TO PAY. Omtool, Ltd. ("Borrower") promises to pay to Citizens Bank New Hampshire ("Lender"), or order, in lawful money of the United States of America, the principal amount of Two Hundred Fifty Thousand & 00/100 Dollars ($250,000.00), together with interest on the unpaid principal balance from December 31, 1996, until paid in full.

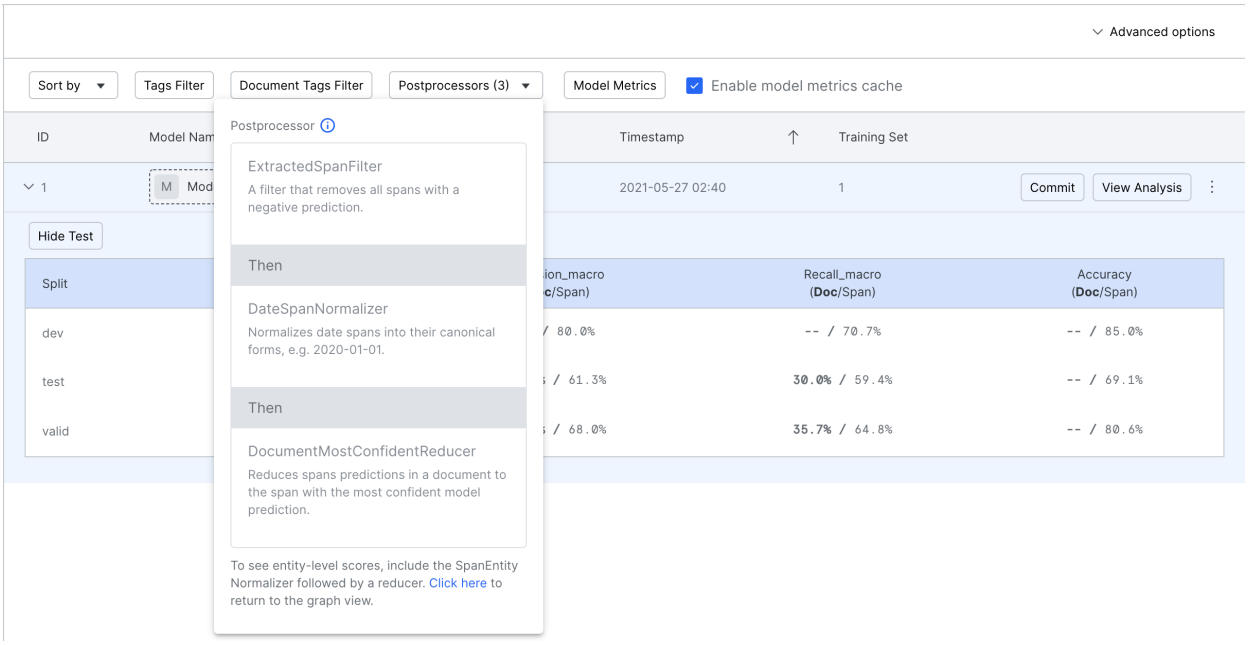
While labeling function and model development leverage span-level candidates in Snorkel Flow, many downstream applications often rely more heavily on document-level extractions. As a result, Snorkel Flow provides a number of operators to convert raw, span-level classifications into document-level extractions. See Post processors in extraction blocks for information on how to set this up for your application.

Document level metrics

By using some combination of normalizers and reducers, you can convert span-level predictions and ground truth into document-level extractions. These are then used to compute document-level accuracy and F1.

These options are available under **Advanced options**. The Models table will display document-level scores (`Doc`) besides span-level scores (`Span`).

Copyright © 2025 Snorkel AI, Inc. All rights reserved



[Entity classification] entity-level operators and metrics

Similar to the extraction setting, Snorkel Flow supports operators and metrics at a higher-level than spans:

- **Span-level**: The number of candidate spans that were classified correctly.
- **Entity-level**: This measures Snorkel's performance of extracting one (or several) values for each document and entity. For example, the mention of a company entity for a particular news article.

Similar to document-level post processors and metrics, Snorkel Flow supports entity-level varieties that produce (document, entity) pairs. More information about entity-level operators can be found in Post processors in extraction blocks.

Entity level metrics

After applying an entity-level reducer, you can convert span-level predictions and ground truth into entity-level extractions. These extractions are then used to compute document-level accuracy and F1 macro.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

GenAI Evaluation workflow overview

Evaluation for GenAI output is inherently challenging because GenAI model responses are varied and context-dependent.

For example, let's say that you are evaluating a chatbot prototype for external release. The chatbot is supposed to answer questions about different health insurance plans that your company offers. You likely have a set of sample answers from internal testing that you need to evaluate, and the answers are likely quite varied. This example use case is explored in detail in the Evaluate GenAI output tutorial.

An evaluation process that can drive meaningful business decisions and improvements about a GenAI model must be:

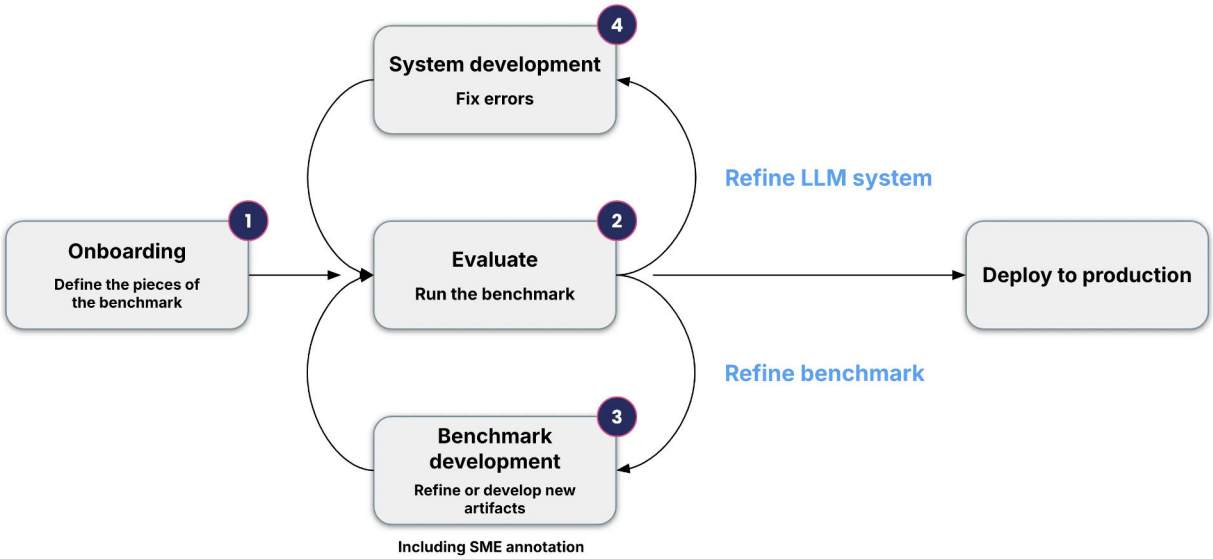
- **Specialized:** Because the potential output of a GenAI model is so varied and context-dependent, evaluation must be customized to suit the specific use case.
- **Fine-grained:** Evaluations should be detailed and focus on individual criteria that matter.
- **Actionable:** The results of the evaluation must provide insights that lead to improvements in the model.

Without a structured evaluation framework, it's difficult to assess model performance accurately or improve it effectively. Snorkel applies these principles in our evaluation workflow to provide a comprehensive and flexible system to continuously monitor, test, and improve AI models, making them reliable tools for enterprise applications.

Snorkel's evaluation framework provides a quick start with smart defaults so you can get an initial evaluation immediately. As your evaluation needs evolve, Snorkel's evaluation suite enables you to craft a more powerful and customized benchmark within the same workflow.

Workflow

Snorkel's evaluation framework follows four phases:



Read a summary of the evaluation workflow below, or dive into the details of each step by following the links:

Copyright © 2025 Snorkel AI, Inc. All rights reserved

1. Onboarding artifacts: Set up the benchmark by defining artifacts, which can be customized: criteria, reference prompts, data slices, and evaluators. For access to our smart defaults on the evaluation criteria, evaluators, and slices, contact your Snorkel representative.
2. Creating the initial evaluation benchmark: Run the evaluation, leveraging programmatic labels and fine-grained analysis to assess the model's performance across various dimensions.
3. Refining the benchmark: Refine the existing artifacts and gather ground truth labels from subject matter experts to validate and improve existing artifacts.
4. Improving your GenAI system: Refine the LLM system through prompt development, RAG tuning, or LLM fine tuning.

This structured process allows AI systems to be reliably tested and refined before being deployed in enterprise environments. By following this process, enterprises can ensure that their GenAI models are aligned with critical business needs and standards.

Tutorial

To follow along with an example of how to use Snorkel's evaluation framework, see the Evaluate GenAI output tutorial.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

Onboarding and defining the GenAI evaluation artifacts

Evaluation for GenAI output begins with onboarding, where you define the key elements of the benchmark. This includes identifying the evaluation criteria, defining data slices, gathering reference prompts, and writing evaluators.

To follow along with an example of how to use Snorkel's evaluation framework, see the Evaluate GenAI output tutorial.

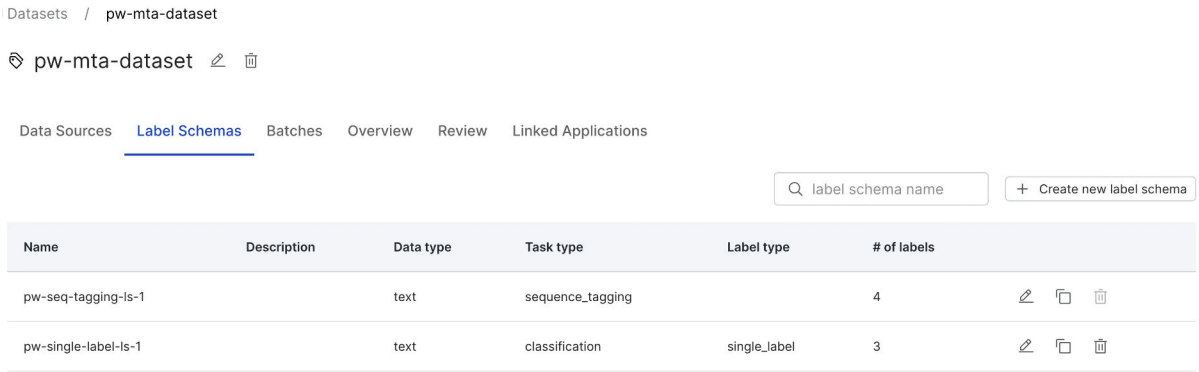
Criteria

These are the elements that the model's output will be evaluated against:

- **Correctness:** Is the output factually accurate?
- **Compliance:** Does the output meet regulatory and policy requirements?
- **Robustness:** Is the model resistant to adversarial attacks?
- **Fairness:** Does the model avoid harmful biases?

To add a new criteria:

1. Select the **Datasets** page.
2. Select your dataset.
3. Select the **Label Schemas** tab.
4. Select the **Create new label schema** button on the top right.
5. Fill out the forms in the dialog according to your criteria.



Data slices

Data slices represent specific segments of the dataset that are particularly important for business objectives. Data slices could be based on:

- Topic (e.g., administrative queries vs. dispute queries).
- Language (e.g., responses in Spanish).
- Custom slices relevant to enterprise-specific use cases.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

Reference prompts

A set of prompts that the model will be tested against. The prompts should be representative of all the types of questions you want your model to perform well on. These can be collected, curated, authored, and/or generated. Prompts can include questions that were:

- Mined from historical data (user query streams from real questions asked to a chatbot, historical query logs from questions asked to human agents, etc).
- Provided by Subject Matter Experts (SMEs).
- Synthetically generated.

Each time you iterate on your fine-tuned model, the responses to the prompts will change, but the prompts themselves will remain consistent across iterations.

Evaluators

These are functions for assessing if an LLM's response satisfies your defined criteria. Each evaluator takes in a datapoint and returns true or false. They can include:

- A prompted LLM serving as a judge, commonly referred to as LLM-as-a-judge.
- Automated comparison to an SME-annotated "gold" dataset (e.g. embedding or LLM similarity score between new LLM and gold SME responses).
- An off-the-shelf classifier for checking PII, toxicity, or specific compliance criteria.
- A heuristic rule.
- A custom predictive model built with programmatic supervision.

(Optional) Ground truth

To get started quickly, you can complete onboarding without gathering ground truth labels. This allows you to more quickly generate your first benchmark evaluation before involving subject matter experts (SMEs) in annotation. When you run the benchmark first, you have a better idea of how and where to use your SME time impactfully. In the refinement step, we explain the best way to gather ground truth labels to aid in your evaluation workflow. However, if you already have ground truth labels, you can also include them in this onboarding stage, which will result in more signal in your first evaluation benchmark.

Next steps

Once you have defined your criteria, data slices, reference prompts, and evaluators, you can run your first evaluation. See running your first evaluation for more information.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

Run an initial evaluation benchmark

Once you've completed artifact onboarding, it's time to set the benchmark for this GenAI model.

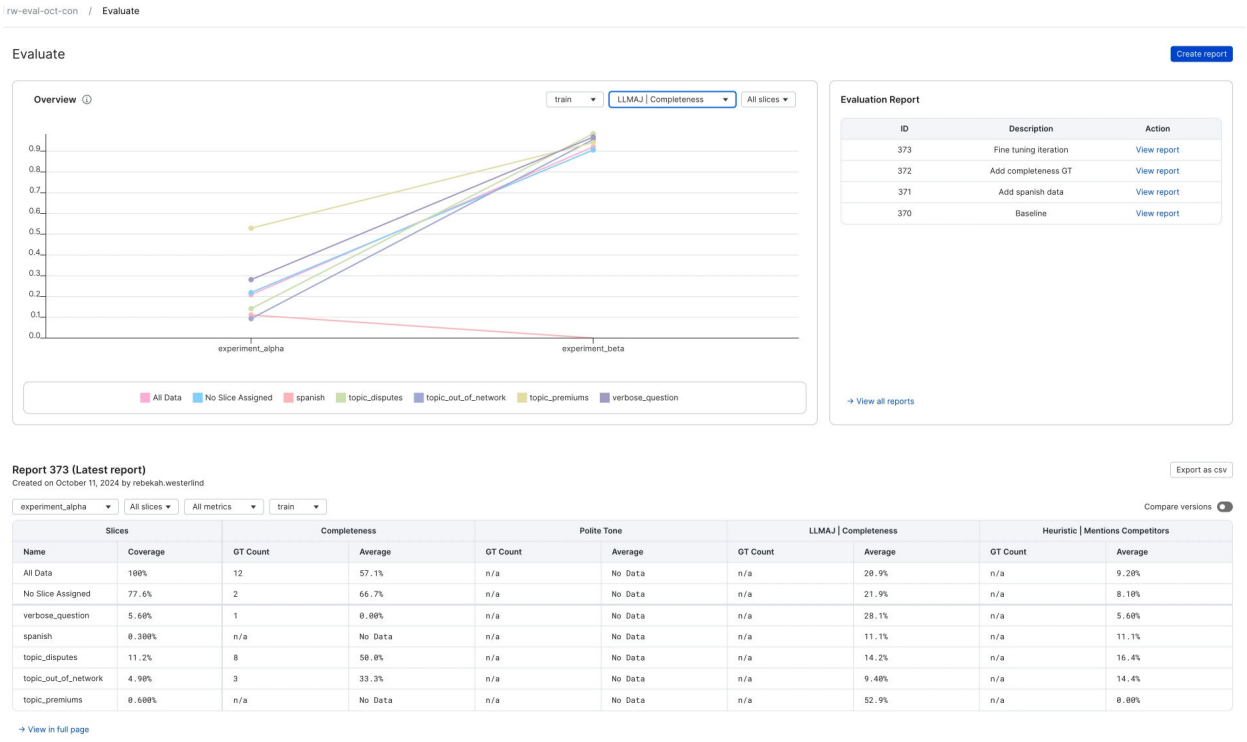
Run the first evaluation to assess the model's performance. The data, which are the reference prompts and responses from the current version of the model, are fed into the evaluation and you can see performance across your slices and criteria.

To follow along with an example of how to use Snorkel's evaluation framework, see the Evaluate GenAI output tutorial.

Evaluate overview

The **Evaluate** page has three areas:

- **Iteration Overview**, or performance plot
- **Recent Iterations** table
- **Latest report** data



Iteration overview

The iteration overview is a plot that shows how your performance has changed over recent iterations. Different data splits, evaluators, and slices can be selected so you can focus on what you care about. This is a helpful image to share with those interested in how your project is going. When you run the first evaluation, you will see points rather than lines in the plot. Once you have run multiple iterations, you will see lines connecting the points so you can visually track trends in performance.

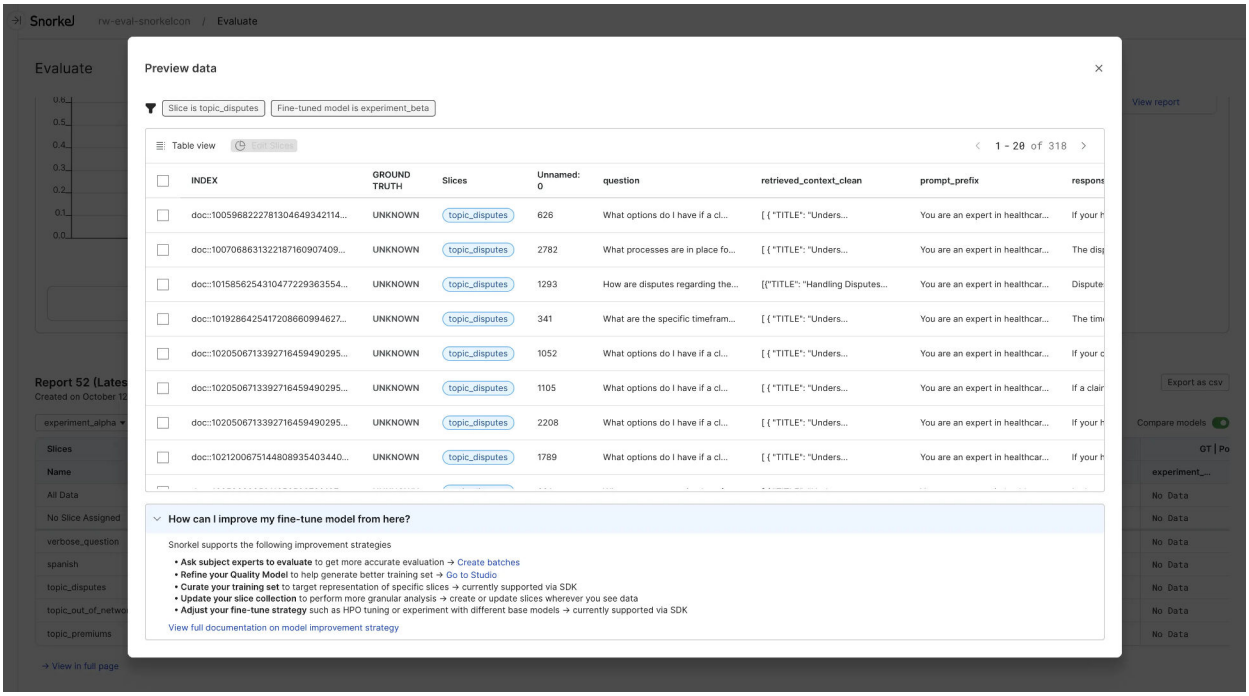
Copyright © 2025 Snorkel AI, Inc. All rights reserved

Recent iterations table

The recent iterations table shows the ten most recent evaluation reports. It can be a useful way to see what steps you've taken recently to improve the performance of your model.

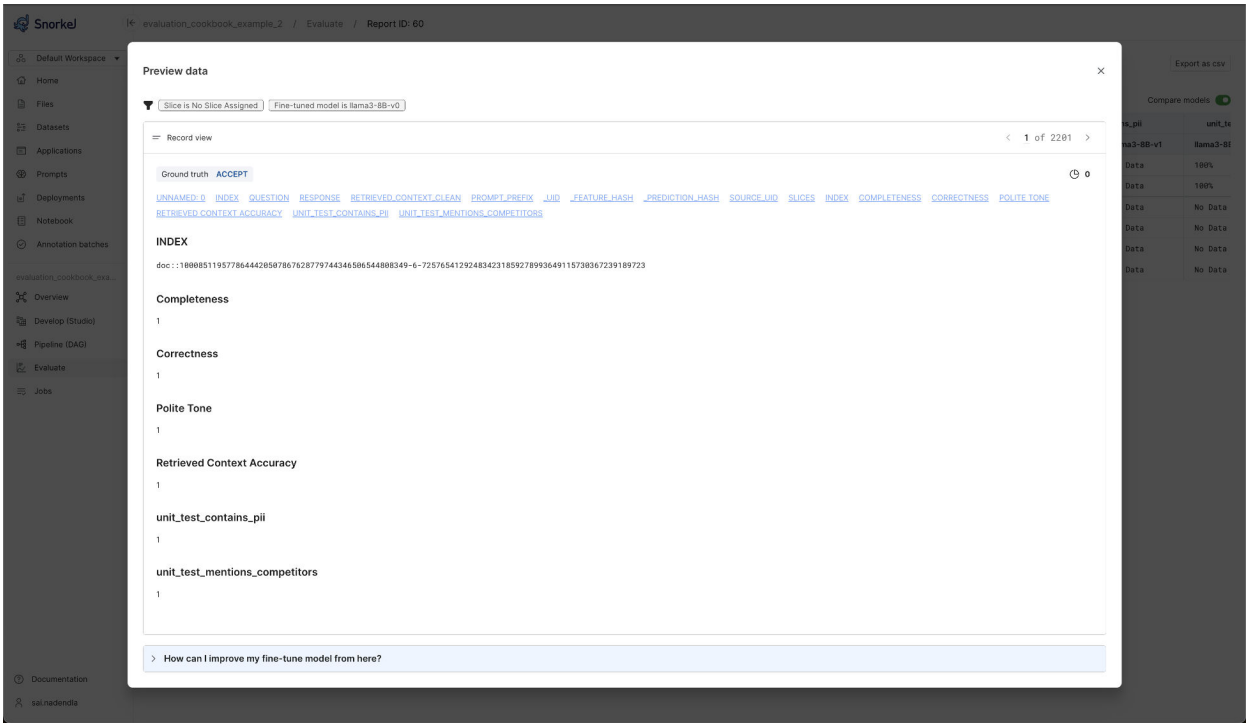
Latest report data

The latest report data is shown at the bottom of the page. It shows how each slice is performing for each evaluator. Select a cell in the data table to open a more detailed view of the data in that iteration and slice. Each cell in the data table showcases datapoint-level metrics for the criteria specified. Selecting a cell provides the detailed metric scores for the individual datapoint.



Datapoint view:

Copyright © 2025 Snorkel AI, Inc. All rights reserved



Next steps

Next, After running the initial evaluation, you may need to iteratively refine the benchmark. This step is iterative, with the end goal of having a benchmark that fully aligns with business objectives, so your measurements of the GenAI model's performance against it are meaningful.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

Refine the evaluation benchmark

After running the initial evaluation, you may need to refine it. This step is iterative, with the end goal of having a benchmark that fully aligns with business objectives, so your measurements of the GenAI model's performance against it are meaningful.

Artifact Refinement involves refining or developing new evaluation artifacts:

- Ground truth labels: Add ground truth labels to ensure evaluators are accurate.
- Criteria: Adjust the criteria based on performance or new enterprise requirements.
- Data slices: Add new slices based on identified gaps or underperforming areas.
- Prompts; Refine prompts to ensure that the evaluation covers all necessary aspects of the use case.

For each artifact, here are some refinement steps you might consider. To follow along with an example of how to use Snorkel's evaluation framework, see the Evaluate GenAI output tutorial.

Ground truth labels

After running the first round of evaluators, Snorkel recommends collecting a small number of ground truth labels for the relevant, defined criteria to ensure the evaluators are accurate. To do this:

1. Create a batch with that specific criterion.
2. Assign your subject matter experts to review that batch so you can collect ground truth labels from your subject matter experts in that batch. Commit these labels as ground truth. For more information, read about annotation.
3. Add ground truth evaluators and re-run the evaluation.
4. Compare the ground truth evaluator with the programmatic evaluator for that criterion. If the numbers are similar, you can trust the evaluator going forward and don't need to collect ground truth labels for this criterion in each iteration. If the numbers are not similar, you should compare the places where the ground truth and evaluator disagree and improve the evaluator to better understand these situations.

> NOTE
>
> If your evaluators are already validated, you can skip subject matter expert (SME) annotation. For example, an enterprise-specific pre-trained PII/PHI model may not require SME annotation for the use case.

Ideally, each evaluator reaches trusted status in the early phases of an experiment, and can be used to expedite the iterative development process. Snorkel recommends re-engaging domain experts for high leverage, ambiguous error buckets throughout development and in the final rounds of development as a pipeline is on its way to production.

How to improve evaluators

Certain criteria may be too difficult for a single evaluator. For example, an organization's definition of "Correctness" may be so broad that developers find that an Evaluator does not accurately scale SME preferences. In cases like this, Snorkel recommends one of the following:

- Break down the criteria into more fine-grained definitions that can be measured by a single evaluator.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

- Rely on high-quality annotations for that criteria during development.
- Collect gold standard responses and create a custom evaluator to measure similarity to the collected gold standard response.

Criteria

Sometimes new criteria surface, or it becomes clear that the definition of a criterion should be adjusted. To add a new criterion, follow the steps in the onboarding guide. To edit a criterion, select the pencil icon next to an existing label schema and change the name or description.

Reference prompts

You can add more prompts by uploading another datasource.

More best practices for refining the benchmark

- **If most of your data isn't captured by data slices**: Consider refining or writing new slicing functions.
- **If a high-priority data slice is under-represented in your dataset**: Consider using Snorkel's synthetic data generation modules (SDK) to augment your existing dataset. Also consider retrieving a more diverse instruction set from an existing query stream or knowledge base.
- **If an evaluator is innaccurate**: Use the data explorer to identify key failure modes with the evaluator, and create a batch of these innaccurate predictions for an annotator to review. Once ground truth has been collected, you can scale out these measurements via a fine-tuned quality model or include these as few-shot examples in a prompt-engineered LLM-as-judge.
- **To scale a criterion currently measured via ground truth**: From the data explorer dialog inside the evaluation dashboard, select **Go to Studio**. Use the criterion's ground truth and Snorkel's Studio interface to write labeling functions, train a specialized model for that criterion, and register it as a custom evaluator. These fine-tuned quality models can also be used elsewhere in the platform for LLM Fine-tuning and RAG tuning workflows.

Next steps

Now that your benchmark is indicative of your business objectives, use it to measure and refine your LLM system.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

Refine the GenAI system based on evaluation insights

The end goal for GenAI output evaluation is to use the insights to refine your LLM system until it is production-worthy and meets your criteria.

Now that you have a trustworthy benchmark, you can use a variety to techniques to improve your GenAI system. These include:

- **LLM fine tuning:** Fine tuning allows you to change the LLM's parameters to adapt its performance to your criteria. Snorkel integrates with Amazon SageMaker, one fine-tuning option. Our LLM fine tuning and alignment tutorial has an example of how to use SageMaker to fine tune your LLM.
- **RAG tuning:** On request, Snorkel can provide an example notebook with instructions for using Snorkel to tune a RAG system.
- **Prompt development:** Snorkel Flow's prompt development features will be released in early 2025.

To follow along with an example of how to use Snorkel's evaluation framework, see the Evaluate GenAI output tutorial.

Once the model has been sufficiently improved, it can undergo another round of evaluation. Continue to track your evaluation progress until the system meets your performance thresholds.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

Creating good labeling functions

This article will cover both high-level guidance as well as specific tips for writing good labeling functions (LFs).

High-level guidance

Good LFs:

- Correct existing model or training dataset errors.
- Expand the coverage of programmatic labels for a label class.
- Have high precision.

> NOTE
>
> Precision does not need to be at 100%. Snorkel Flow's UI highlights the precision with red if it is too low.

- Can be crafted quickly. You should plan to write many LFs and perform many training iterations.

To write good LFs:

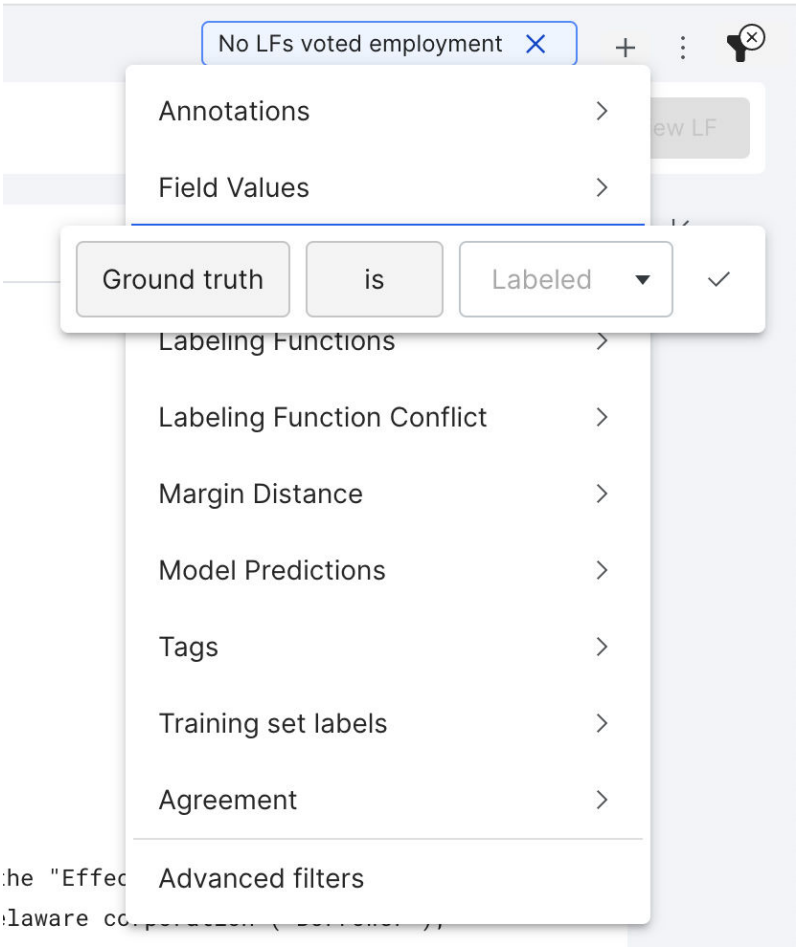
- Train a model as soon as possible. Then, use error analysis tools and filters to write "corrective LFs" to inject signal where the current model and/or training dataset needs it the most -- the existing errors.
- Start with the obvious and move quickly. For your first 10 LFs, try not to focus too much on perfecting each LF. You can always edit and perfect your LFs later.
- Do thorough data discovery. Use our extensive set of filters, clickable error buckets in our analysis tools, embeddings clusters, etc. to discover patterns in the data that you can encode as LFs.
- Create as many labeling functions as you can for each class. If a single rule could describe a class with 100% coverage and precision, you probably wouldn't need a model to begin with.
- Explore the wide variety of labeling function builders.

Labeling data when starting a new task

- Discover the data to get ideas for labeling functions. Start by paging through a few examples, filtering by ground truth class, and switching between data views.
- You can use filters to hone in on specific datapoints.

> NOTE
>
> Filtering by ground truth label will limit you to only data points with ground truth labels – you may also want to look through some of the unlabeled dataset to get ideas!

Copyright © 2025 Snorkel AI, Inc. All rights reserved



- Your goal should be to write one high coverage and empirical precision labeling function per class to train your first model and get started with error-guided data development.

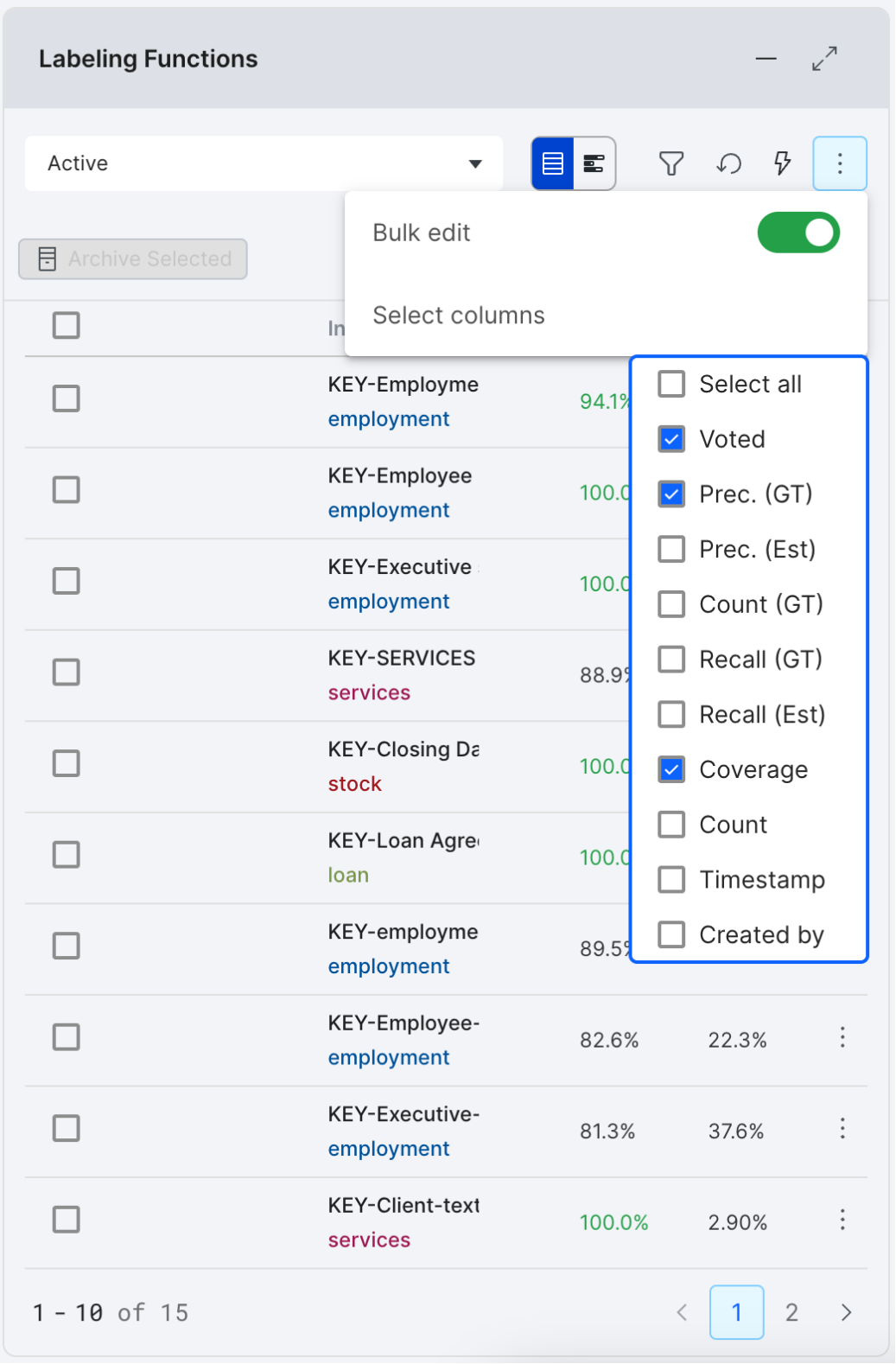
  If you have access to Foundation Models, an easy way to get started is to use a prompt LF to generate labels using the Foudnation Model's predictions.

- The **empirical precision** in Label Studio is just a rough estimate given the data in current split. Increasing the accuracy of your labeling functions will help the performance of your model, and as a loose guideline**, we recommend refining any LFs that show up as “red” in the interface**. To refine your LF:

  - Use the `View Incorrect` and `View Correct` filter buttons to see data points where the LF was incorrect. Try making the LF pattern more specific and re-saving the LF to overwrite the old one.
  - Add conditions to your LF using the **Add condition** and **Negate** options in the advanced options dropdown menu.
  - If you suspect your LF might be more accurate than the current estimates show, try filtering to where your LF voted but the ground truth label is UNKNOWN and supplying new ground truth labels for these examples. This might also reveal ways to refine your LF.

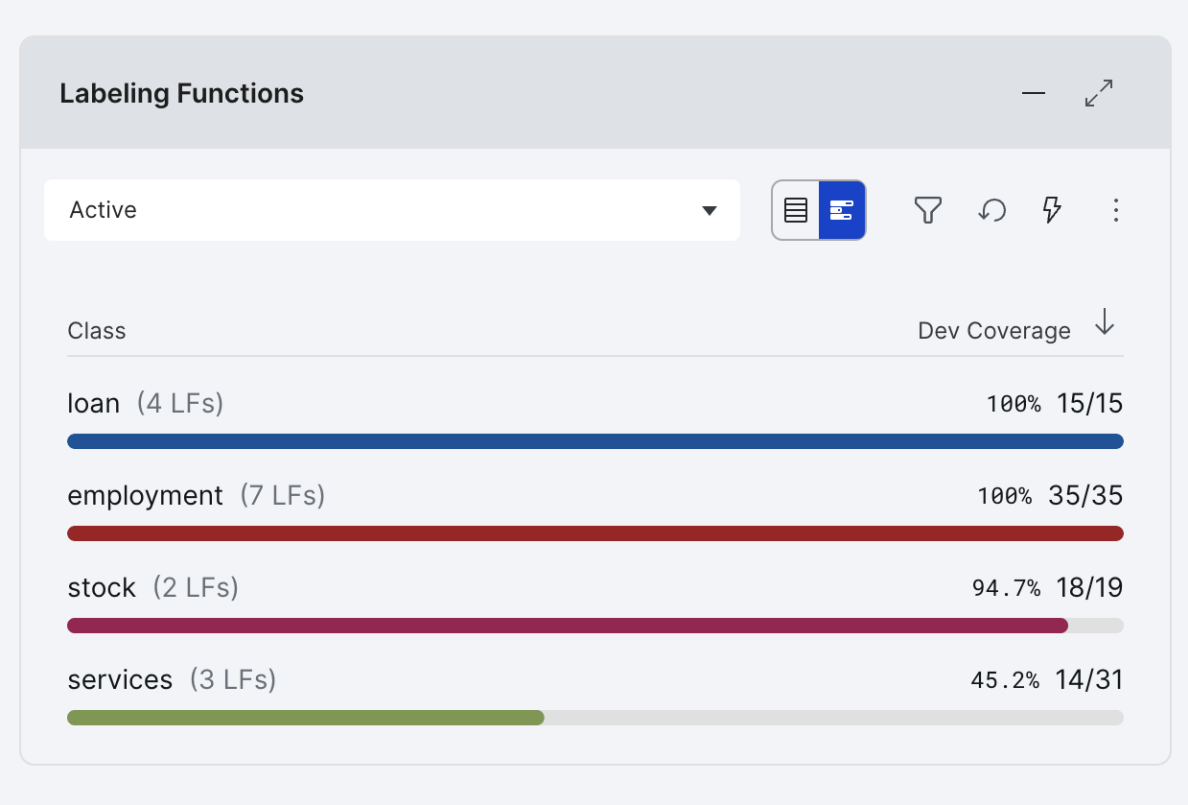
    > You can optionally view metrics on (GT) instead of estimated metrics

Copyright © 2025 Snorkel AI, Inc. All rights reserved



- It is not necessary to achieve 100% coverage -- models can generalize.

Copyright © 2025 Snorkel AI, Inc. All rights reserved



- As you move through, you can also assign Ground Truth (GT) labels easily with keyboard shortcuts.
  - GT labels will also be used for error analysis, identifying what types of examples your model currently makes mistakes on.
- Start error-guided analysis quickly.

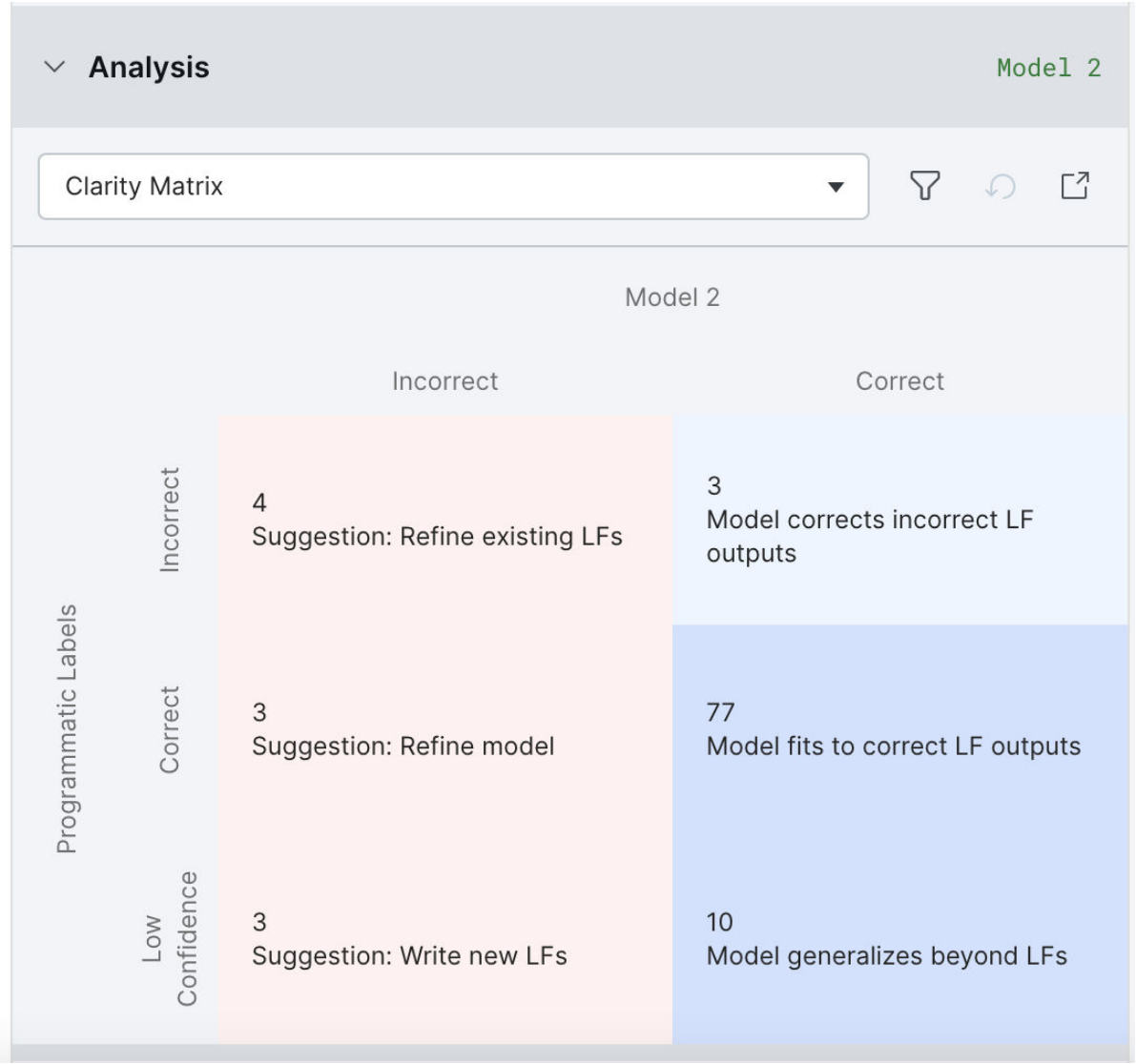
  The fastest way to do this is by training a model in SnorkelFlow on a Foundation Model's predictions using our prompt builder. There's no need to re-label what out of the box model already gets correct, and this will expose error buckets that can be easier to analyze and correct than the entire dev set or the entire labeled class.

Using the analysis pane to create LFs

After training a model, the **Analysis** pane will guide the refinement of existing LFs and the creation of new ones.

The **Clarity Matrix** shows how your model's predictions fared relative to the labels from your labeling functions (de-noised and combined by Snorkel Flow) at a high level:

Copyright © 2025 Snorkel AI, Inc. All rights reserved



- The upper-left cell with **Suggestion: Refine existing LFs** shows data points where the model is incorrect because the labeling functions were incorrect.

  Select this cell to view and refine the incorrect labeling functions.

- The middle-left cell with **Suggestion: Refine model** shows data points where the labeling functions were correct but the model was incorrect.

  If after iterating on your LFs, this cell has the most errors as indicated by a higher number and a more saturated color, then you should select this cell to return to the models page and try a more powerful and/or better-tuned model.

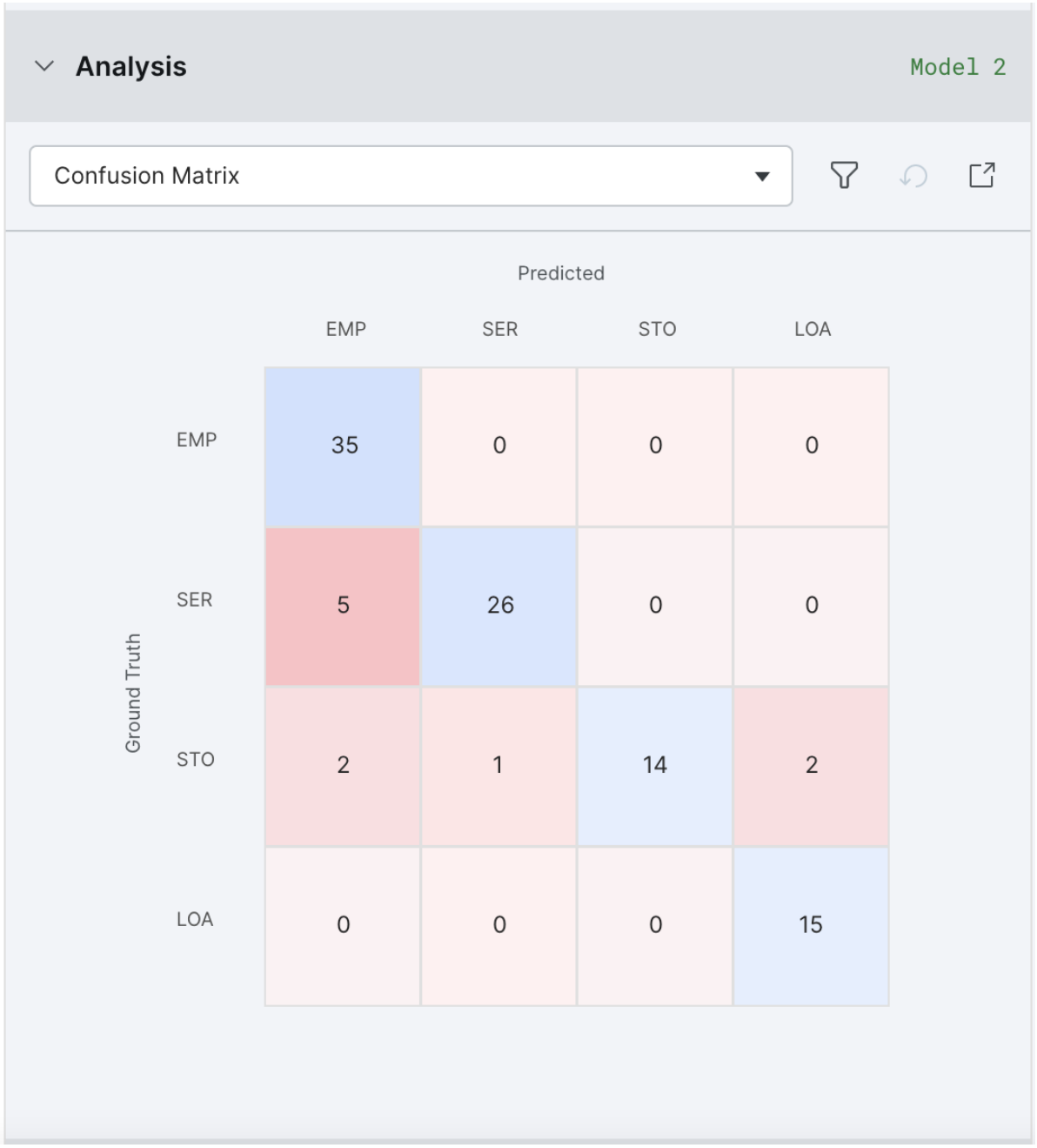
- The lower-left cell with **Suggestion: Write new LFs** shows data points where all of the labeling functions abstained, and the model did not successfully generalize.

  Select this cell to write new labeling functions that cover these data points.

The **Confusion Matrix** shows where the model confused one class for another (within the off-diagonal buckets). You can click on these to filter data points and write or edit labeling functions to correct this

Copyright © 2025 Snorkel AI, Inc. All rights reserved

error mode. For example, the below shows that 2 of the 15 loans are instead classified as employment. Clicking on that box would be a good place to start refining your labeling functions.



The **Error Correlations** view traces the model errors back to labeling functions most correlated with these errors. If an LF is correlated with a large number of errors, you may want to refine or delete it.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

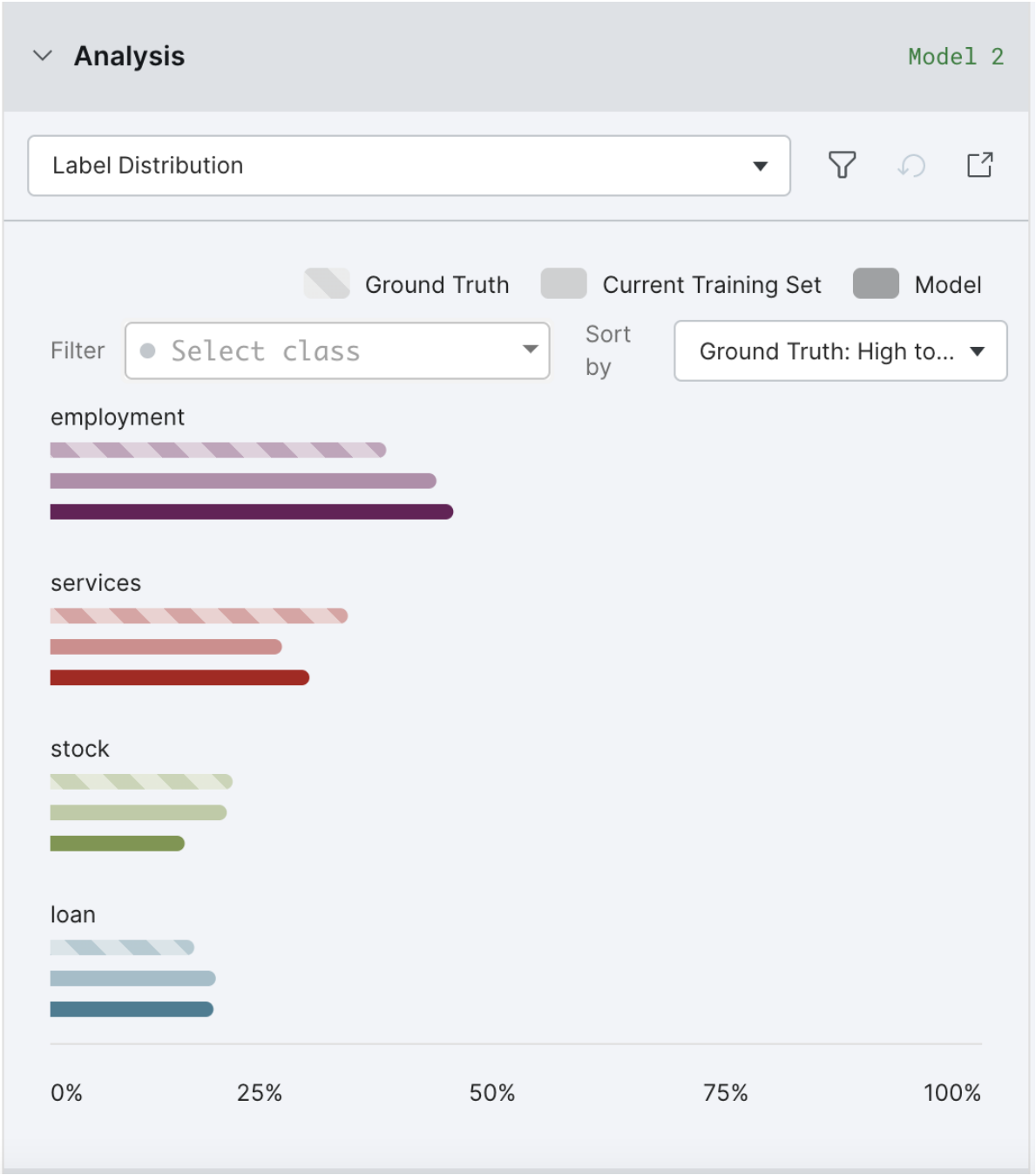
Error Correlation

LFs errors (valid)

Number of errors	Labeling function	Class	View errors
122	FTR-stock	employment	→
73	FTR-security	loan	→
46	sample_multi_polar_code_lf	--	
34	FTR-employment	employment	→
34	FTR-employment-1	employment	→
33	KEY-Executive-suggestedLF-test-hello-world-test	employment	→
16	KEY-Note-suggestedLF	loan	→
15	KEY-Closing Date-suggestedLF	stock	→
5	FTR-purchaseofshares-1	employment	→
	KEY-Customer-		

The **Label Distributions** view shows the relationship between the empirically-measured class distribution, and that of your labeling functions and model. If there are significant discrepancies, you may want to write more / fewer labeling functions for that class, and/or select **Oversample data to match class distribution on** valid split under the **Tune hyperparameter** options on the **Models** page.

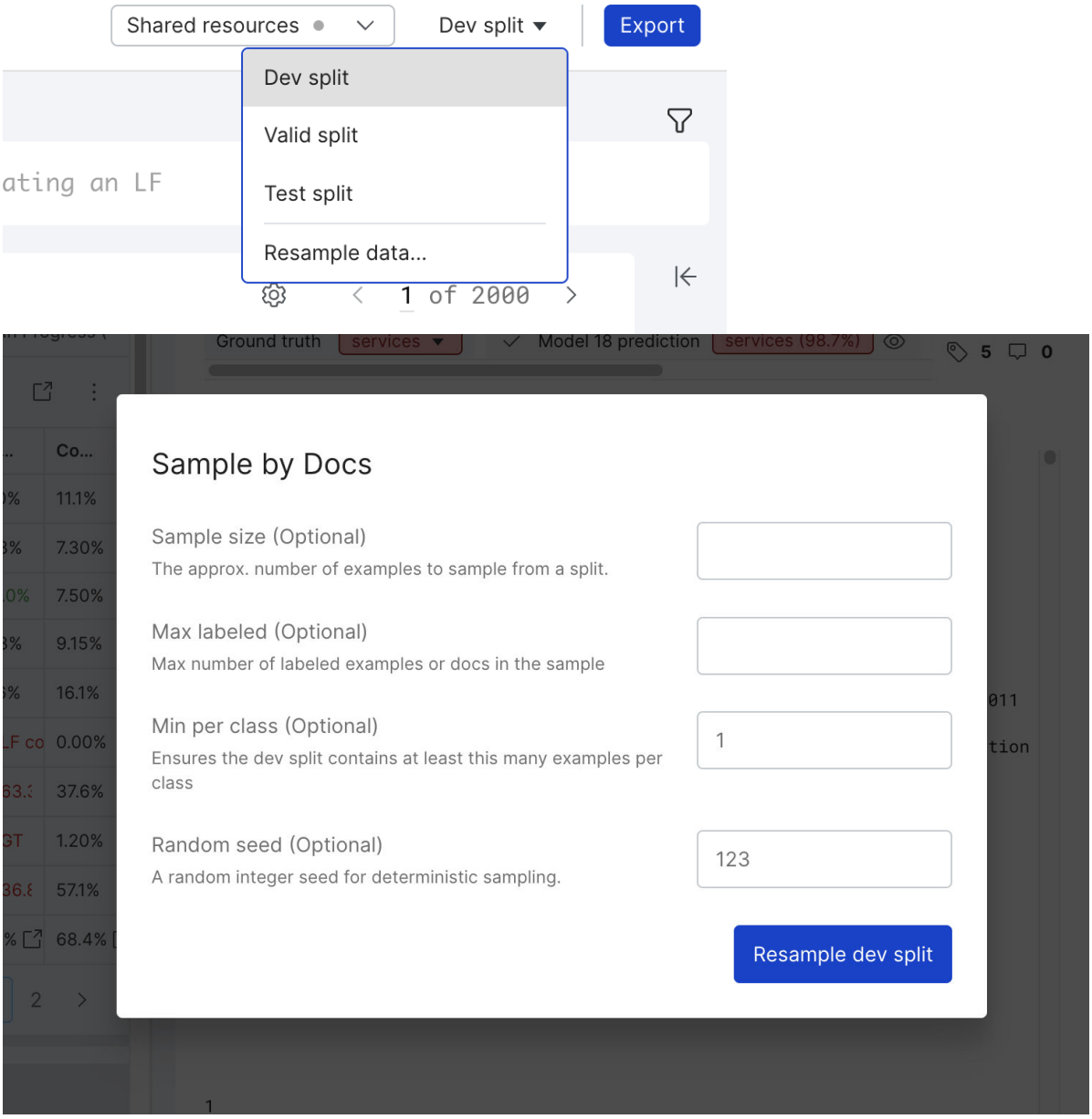
Copyright © 2025 Snorkel AI, Inc. All rights reserved



The longer you iterate, the more likely your LFs are to be tuned to the particular data points that you've labeled in your dev set (i.e., overfitting). For this reason, it's a good practice to periodically refresh your dev set by resampling your dev set from the train split using the resample option on the split selector. Note that since these data points will have already been seen by the model, we suggest training a new model to maximize the usefulness of the analysis tools.

To resample your data, click **Resample data...** from the split selector dropdown.

Copyright © 2025 Snorkel AI, Inc. All rights reserved



For more information about resampling data, see Application data control pane.

Tips and tools

Quantity over quality

Labeling functions do not need to be perfect- though having better labeling functions will in general improve the performance of your model, as will having more, and more diverse, labeling functions.

Labeling functions will in general be noisy and incomplete in their coverage. Snorkel Flow will denoise these labeling functions as well as handle conflicts and overlaps among them, and then train a final machine learning model to generalize to data points your labeling functions do not cover. Read about our "Label Model" for more information on this topic.

You can click on the "View LF Coverage" button to see the coverage by each class.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

Remove poorly performing labeling functions

While labeling functions that perform better than average can improve your end results, labeling functions that perform below average can add noise to your model and harm its performance. One way to handle this is to use the estimated accuracies in Label as a rough guide for very badly performing labeling functions, and try to remove or improve them, e.g. by adding boolean conditions to narrow them and make them more precise.

Improving existing Labeling Functions

After you've defined some labeling functions, you can improve their performance even further by using the below strategies:

1. Inject domain expertise directly
   - **Goal:** Increase training set coverage with less overfitting risk (not looking at specific examples for LF ideation)
   - **Strategy:** Reference external resources (e.g. 3+ digit SIC code categories from https://siccode.com/) to write keyword/regex labeling functions
2. Leverage "Trusted Labeling Functions" for label correction
   - **Goal**: "Correct" noisy labeling functions (functions with low precision, and perhaps high coverage)
   - **Strategy**: Write targeted labeling functions to override noisy labels, and select "Trust this labeling function" in the UI
3. Pass-thru servable labeling functions as modeling features
   - **Goal**: Augment modeling signal with high quality labeling functions
   - **Strategy**: Selectively pass LF outputs that are:
     - Servable: Available at inference time
     - High Precision: To avoid model overfitting
   - **How**: When defining model training configuration settings, in "Train New Model" pane, under "Feature Libraries", click "True", before clicking "Train Model" button.
4. Detect overfitting using estimated accuracies
   - **Goal**: Mitigate overfitting in settings without a validation split
   - **Strategy**: Remove or prune labeling functions with a large gap between Snorkel-estimated train and dev accuracy

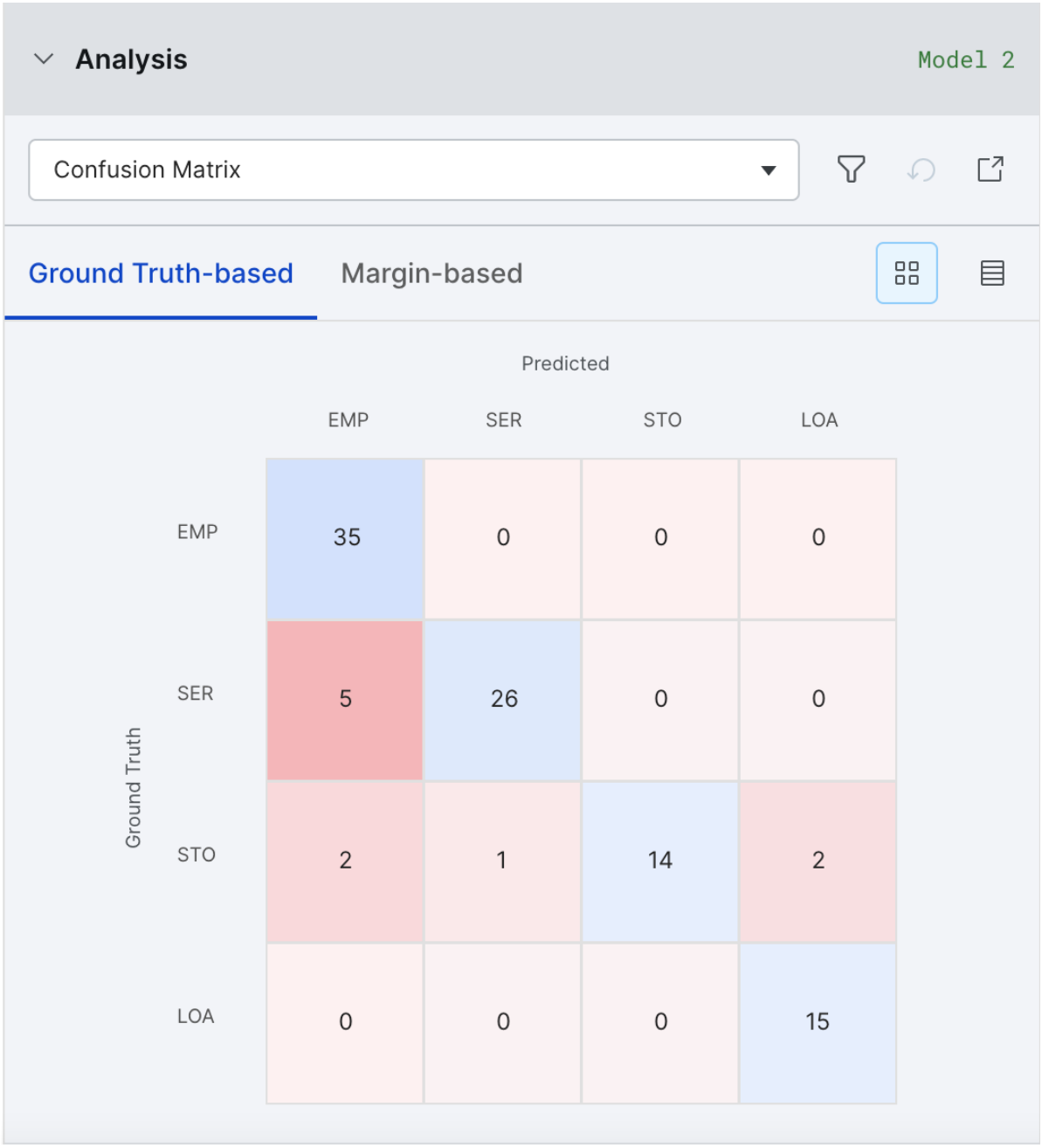
Check on overall coverage by changing the summary statistics

On the upper left corner of the screen, you can see the performance of the current set of labeling functions on the selected split as well as dataset statistics.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

Analysis: Rinse and repeat

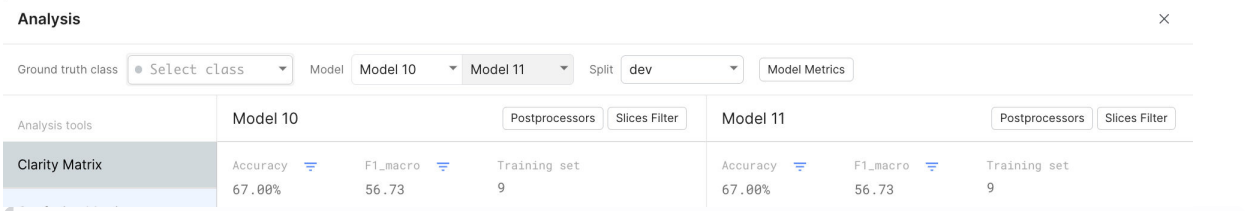
For a selected model in the **Train** pane, click **Analysis** to open the **Analysis** pane, where you can find analysis that helps you understand the performance of both your labeling functions and your models. It also offers suggestions on how to improve your end results.



In Snorkel Flow, this feedback can be directly acted upon by adding/modifying LFs or training a new model to target specific errors. Improving our model becomes an iterative, systematic process.

Clicking on **Open Analysis** will open the Analysis Modal. The Analysis Modal also allows you to compare multiple models and view the different analysis tools over different splits of the data.

Copyright © 2025 Snorkel AI, Inc. All rights reserved



> **NOTE**
>
> If a split does not have predictions or ground truth labels, some of the analysis tools will not be available.

The predictions from a given model are recorded at the time the model is trained. To get predictions for any new data points added to a dataset, you will need to train a new model. On the other hand, the metrics on the Analyze page are computed live. As a result, the metrics may change in the following ways:

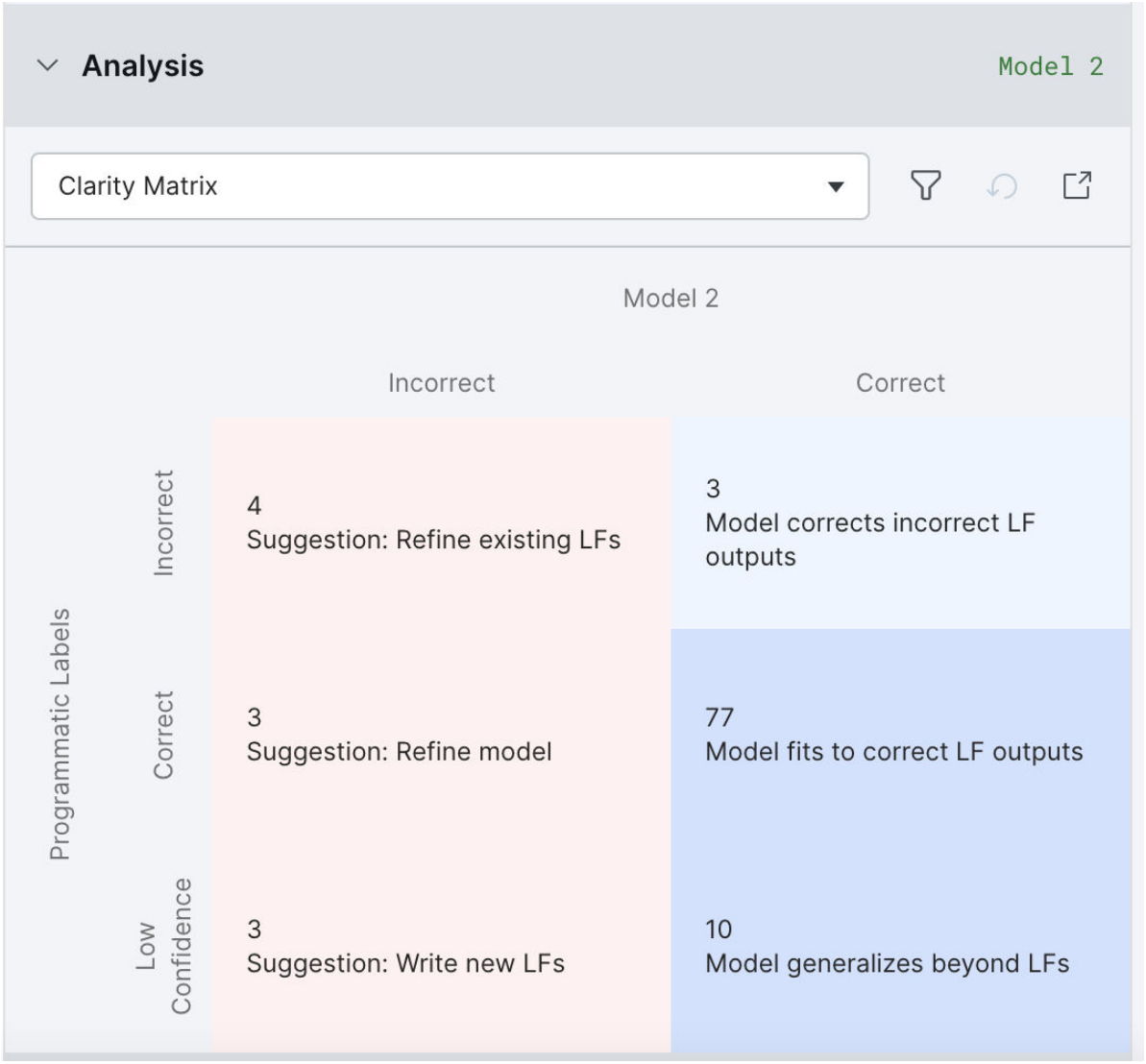
Scenario	Metrics Updated for Existing Model
New datapoint(s) added	No
New GT addded	Yes
Existing GT changed	Yes

Clarity matrix

The Clarity Matrix helps identify mistakes made by your model with respect to your label package. This allows you to identify a reasonable next step that would improve model performance.

Clicking on an element of the Clarity Matrix activates the relevant filters in **Develop (Studio)**.

Copyright © 2025 Snorkel AI, Inc. All rights reserved



By default, “All labels” is selected and the Clarity Matrix displays information for all class labels. If you are interested in a specific class label, please choose it from the dropdown at the top of the **Analysis Modal**.

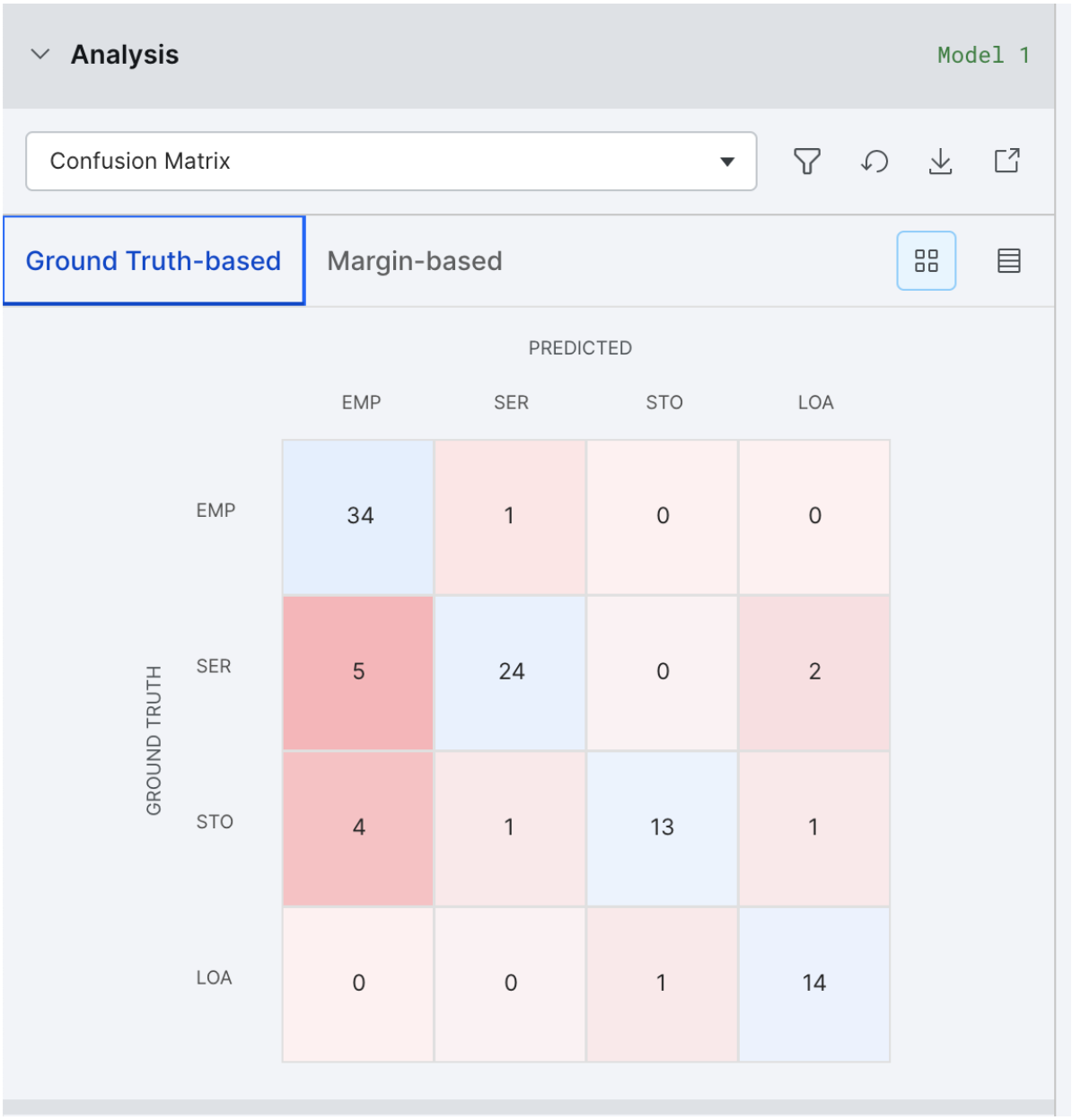
Confusion matrix

The Confusion Matrix helps to identify major buckets of errors being made by your model. There are two views available: Matrix View and Table View.

> **NOTE**
> Matrix View is not available for high cardinality tasks.

Clicking on an error bucket will activate filters in **Develop (Studio)** to see all the samples in that error bucket.

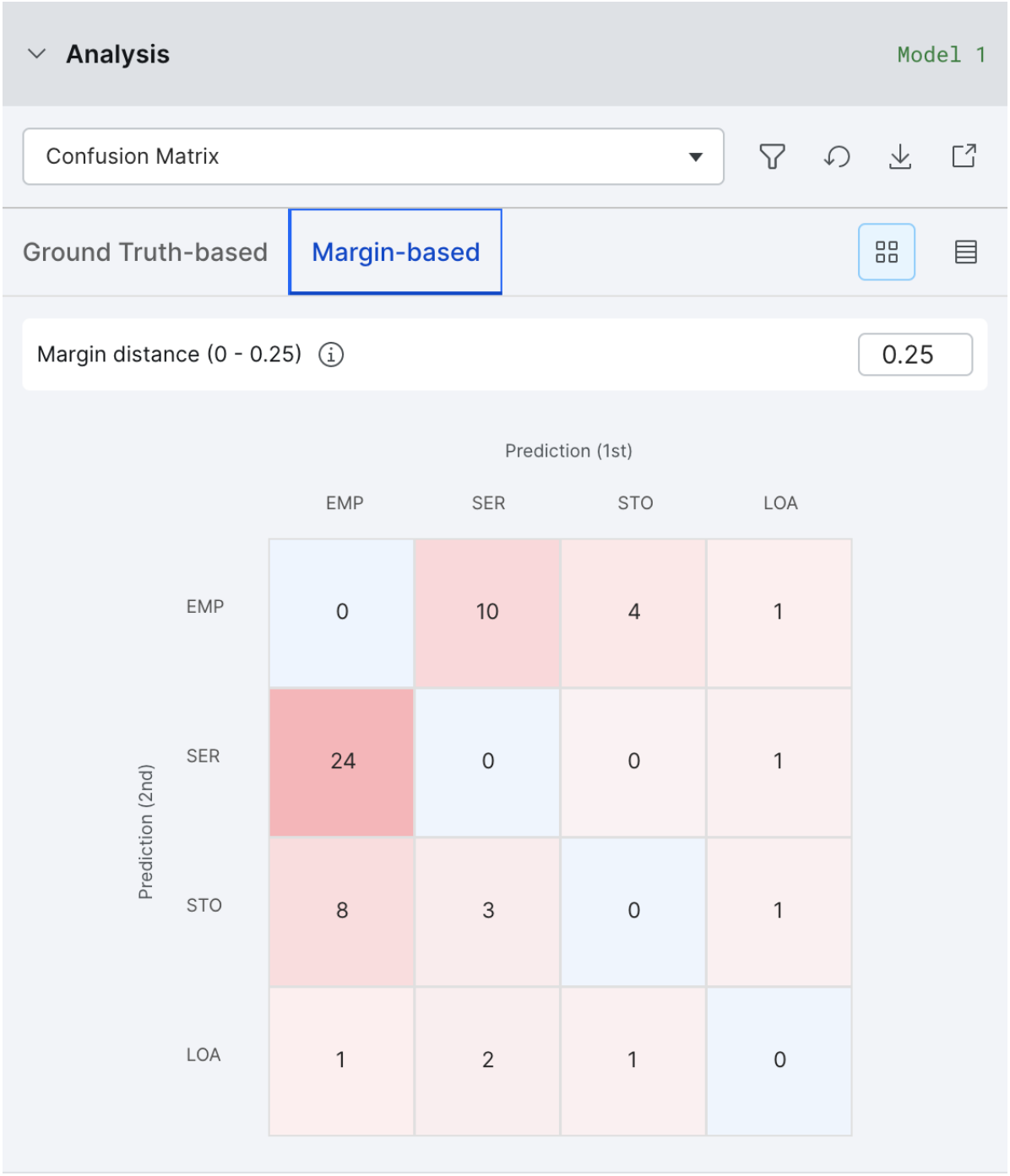
Copyright © 2025 Snorkel AI, Inc. All rights reserved



Margin confusion matrix

When you lack ground truth, the margin confusion matrix is a useful tool to identify *likely* errors in a similar way to the confusion matrix. Instead of using ground truth labels, the margin confusion matrix uses the predictions of the model and the measure of *margin* which is the difference between the. two most confident predictions. For example [0.5, 0.4, 0.1] has a margin of 0.5 - 0.4 = 0.1. The smaller the margin, the more confused the model is between the top two predictions. We can use this to find examples where the model is likely to be incorrect, or at least examples where the model would benefit from extra LF coverage.

Copyright © 2025 Snorkel AI, Inc. All rights reserved



LF error contributions

The LF Error contributions lists all labeling functions whose incorrect predictions match the incorrect predictions from the end model.

Clicking on one of the LFs takes you to the Label page with a filter that shows examples where that LF is incorrect.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

Error Correlation

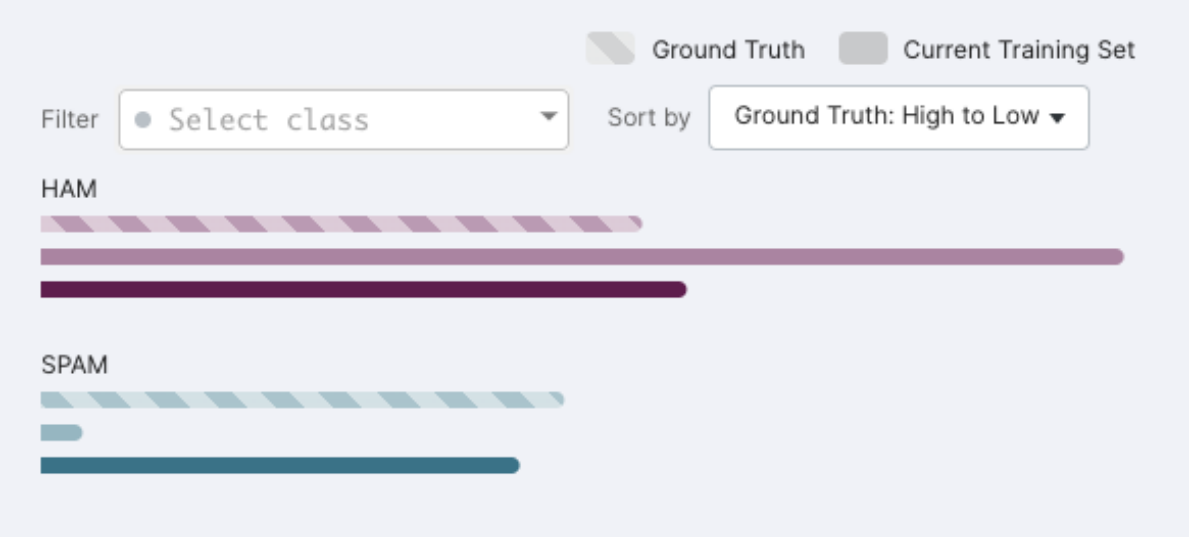
LFs errors (valid)

Number of errors	Labeling function	Class	View errors
122	FTR-stock	employment	→
73	FTR-security	loan	→
46	sample_multi_polar_code_lf	--	
34	FTR-employment	employment	→
34	FTR-employment-1	employment	→
33	KEY-Executive-suggestedLF-test-hello-world-test	employment	→
16	KEY-Note-suggestedLF	loan	→
15	KEY-Closing Date-suggestedLF	stock	→
5	FTR-purchaseofshares-1	employment	→
	KEY-Customer-		

Label distributions

Label Distributions shows both the raw counts (on hover) and percentages of each class based on the labels from ground truth, predicted by the training set that the model uses (label model output), and predicted by the current model.

Copyright © 2025 Snorkel AI, Inc. All rights reserved



Class level metrics

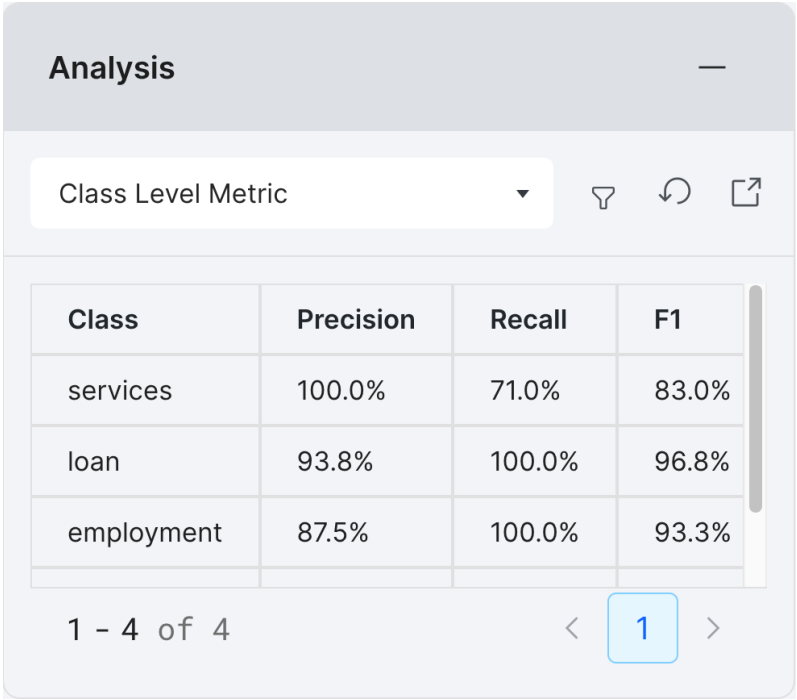
Class level metrics displays model performance on per-class basis to help decide where to best focus efforts.

These class level metrics are available in Snorkel Flow:

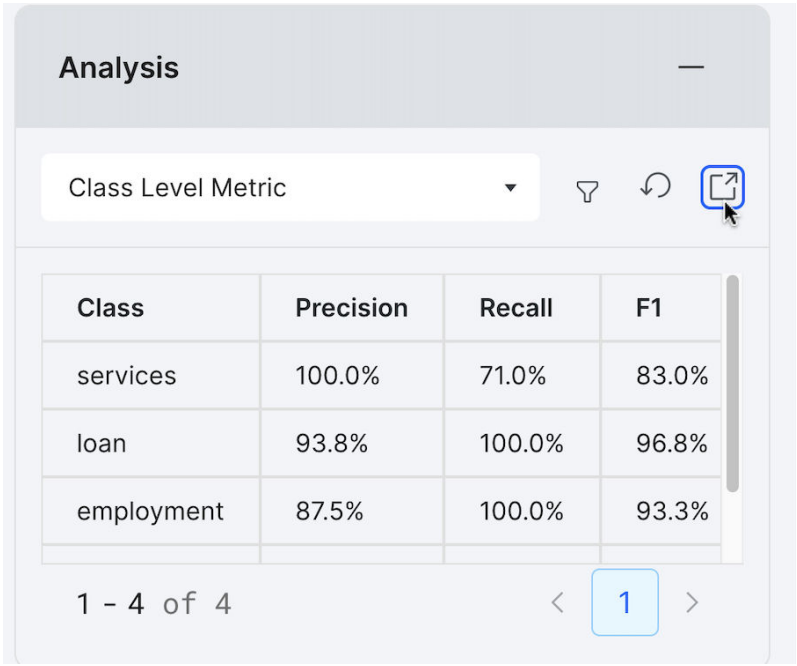
- Span precision
- Span recall
- Span F1
- Token precision
- Token recall
- Token F1

From the Analysis panel, you can view all available class level metrics within the table. Scroll the table horizontally to view them all.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

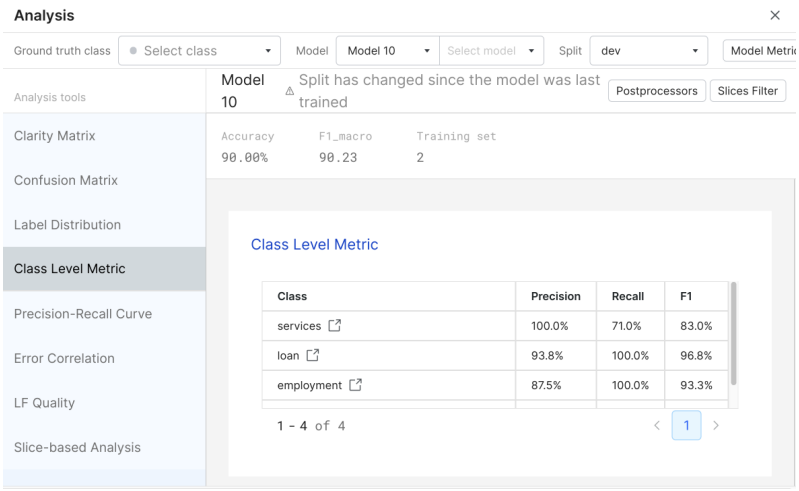


For a larger viewport and to compare metrics across models, select the launch icon to open the **Analysis** modal.

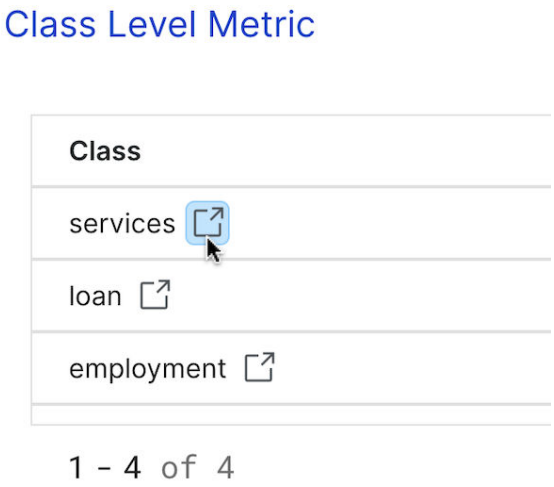


With the **Analysis** modal open, a table displays all of the class level metrics.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

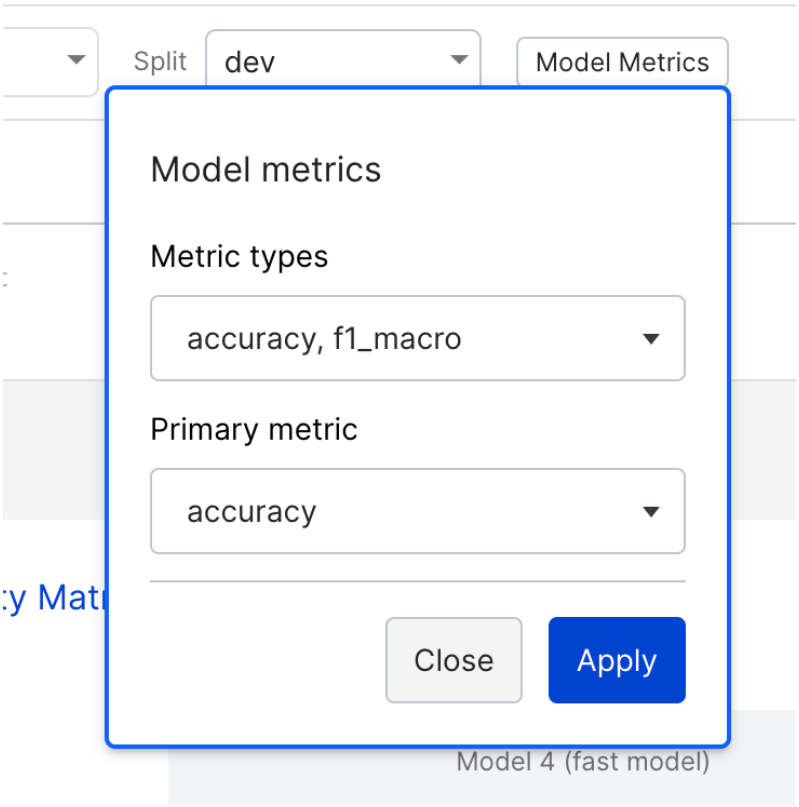


Select the icon next to a class name to view the **Label** page with the relevant filters activated.



To update the metric types you are interested in, select **Open Analysis** to open the **Analysis** modal, where you can select **Model Metrics** to access **Metric Types**.

Copyright © 2025 Snorkel AI, Inc. All rights reserved



In addition to the built-in metric types, you can register custom metrics via notebook. See `snorkelflow.client.metrics.register_custom_metric` in the SDK reference for more details and examples.

Generalization

This provides two analyses:

- The improvement in data coverage by the end model compared to the label model created with your labeling functions. For example, if the labeling functions you created cover 80% of your training data, the improvement would be 20% because the end model can cover 100% of your training data.
- The generalization gap between your `train` and `valid` splits. If the difference in the accuracies for this is greater than 5%, then a warning about overfitting to the dev set is raised. Note that the `train` split score ignores the datapoints in the `dev` set, so in some cases this warning may not appear even if the `dev` and `valid` scores have a difference more than 5%.

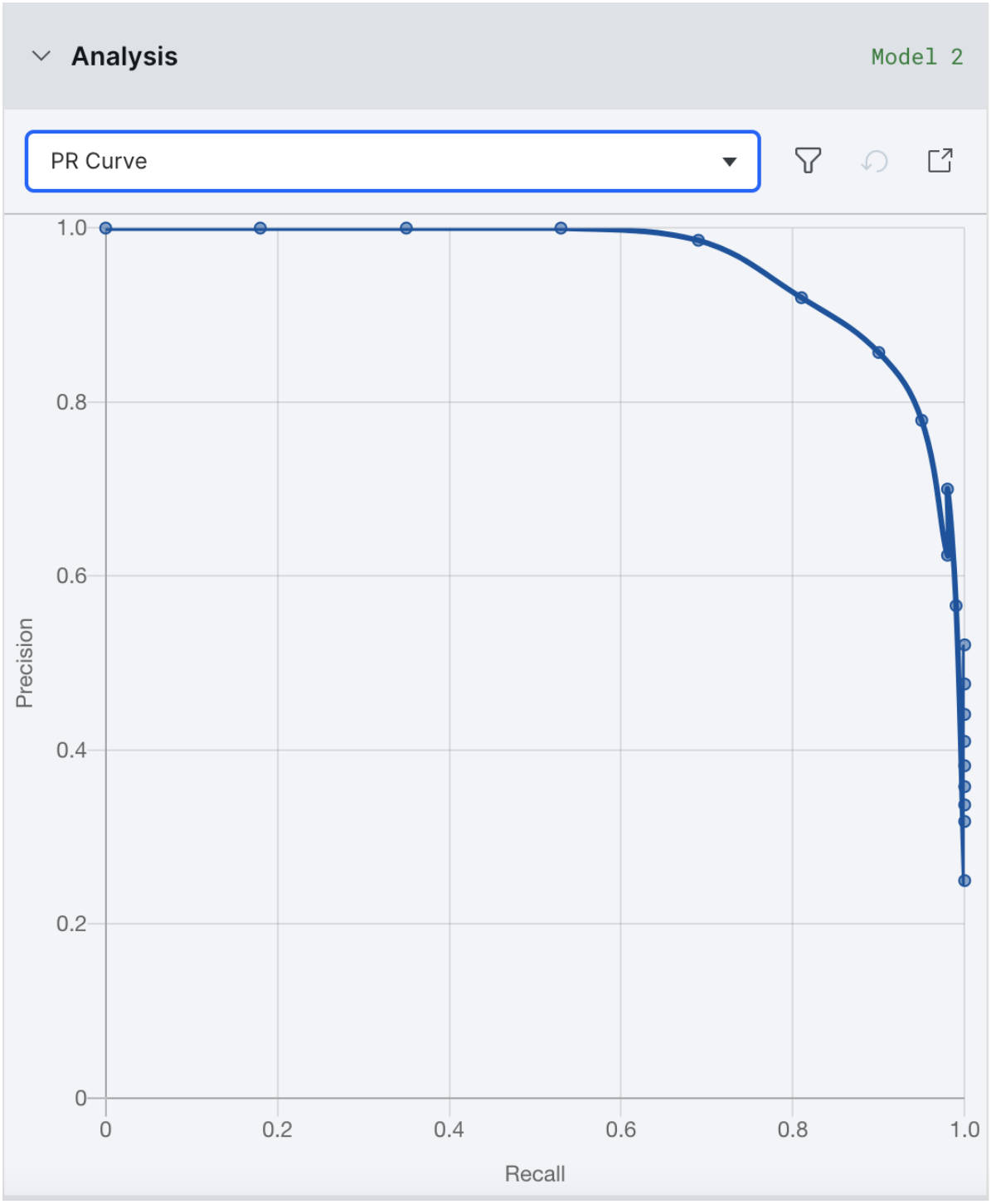
> NOTE
>
> Some datasets can have large differences between the dev and valid set performances due to differences in underlying data distributions. If that's the case, you might want to modify your data so that your dev and split come from the same distribution.

Precision-recall curve

The precision-recall curve plots the precision and the recall for different decision thresholds. This matters more for binary classification problems, and is especially useful for when your data suffers from class

Copyright © 2025 Snorkel AI, Inc. All rights reserved

imbalance, e.g. there are more negative samples than positive samples.



LF quality view

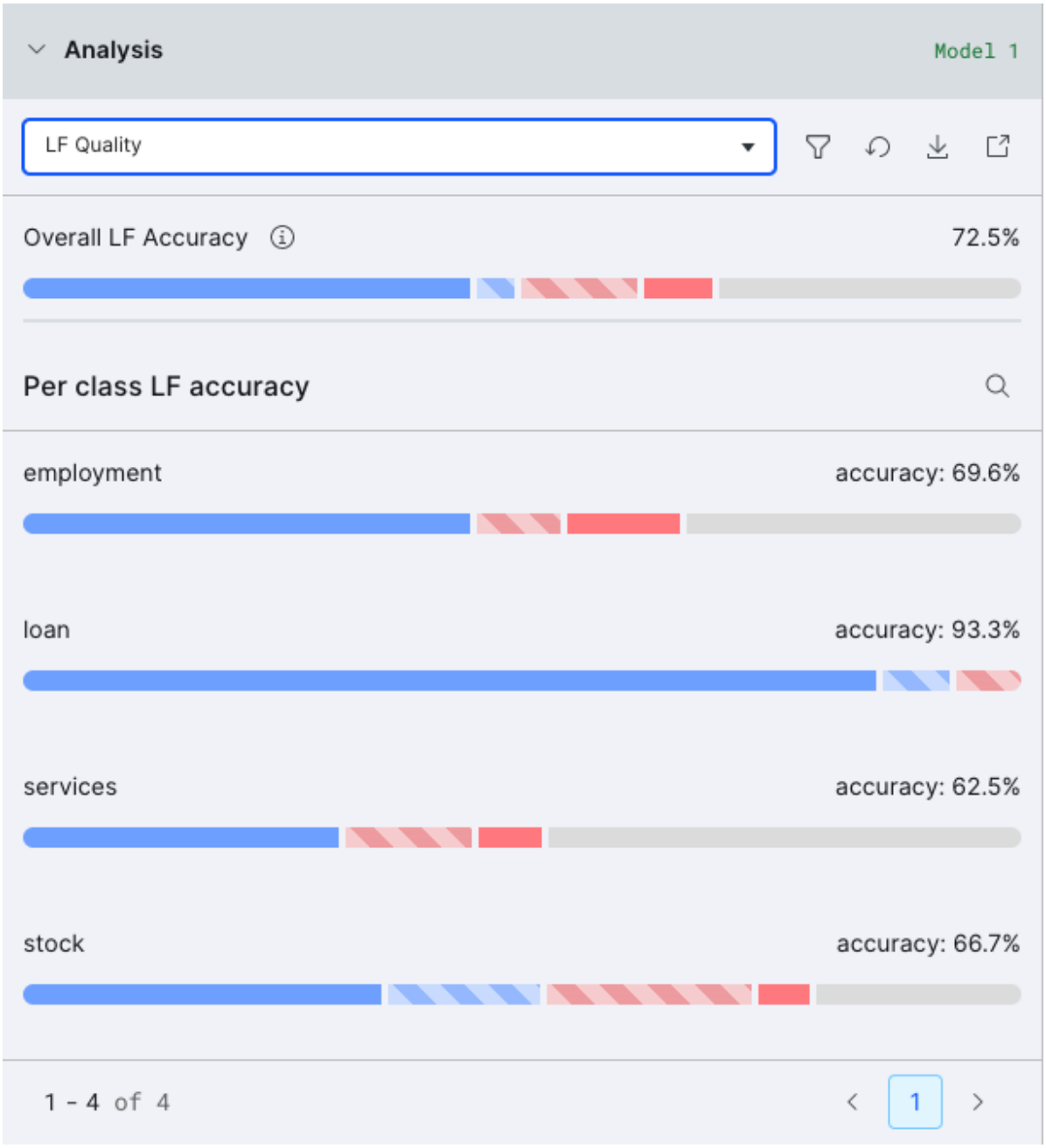
These visualizations display the behavior of your labeling functions over your ground truth data, and relates it to the downstream performance of the label model. You can use them to find which labeling functions should be refined. It groups each data point into one of five categories:

1. Correct/Correct (solid blue): all LFs voting on these points were correct, and so was the downstream label model

Copyright © 2025 Snorkel AI, Inc. All rights reserved

2. Conflicting/Correct (shaded blue): only some LFs voting were correct, but the label model was able to denoise the labeling functions and produce a correct training set label
3. Conflicting/Incorrect (shaded red): only some LFs voting were correct, and the label model was unable to denoise the labeling functions which then produced an incorrect training set label
4. Incorrect/Incorrect (solid red): all LFs voting on these points were incorrect, and so was the downstream label model
5. Uncovered (grey): no LFs voted on these points, and so no training set label was produced

By clicking on any of the colored bars, you can view the data points in each group. We recommend starting with the shaded red group and working your way down to the grey group in order to improve your training data.



Copyright © 2025 Snorkel AI, Inc. All rights reserved

Filter metrics by slices

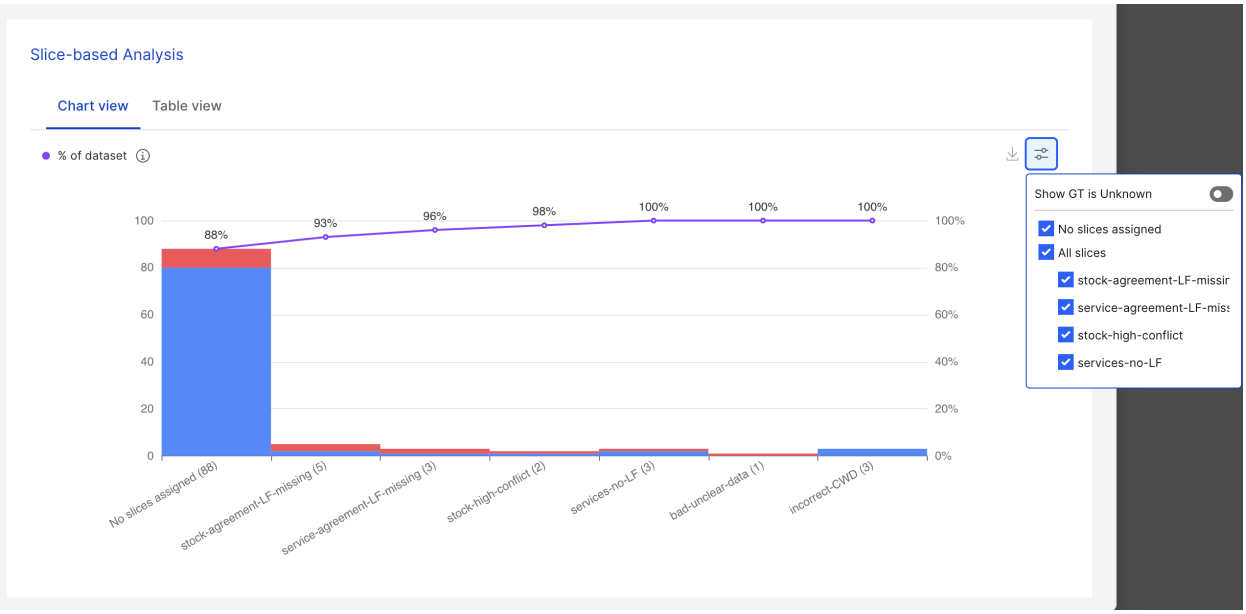
You can use the **Slices Filter** dropdown to view model metrics on subsets of the data. Here, you will see all of your error analysis slices as defined in Error Analysis Slices. You can then check or uncheck individual slices to include or exclude those examples from the analysis.

Error analysis by slices

You can use methods of error analysis like the slices-based analysis chart and tables to survey your data in the iterative development loop.

Slice-based analysis: Chart view

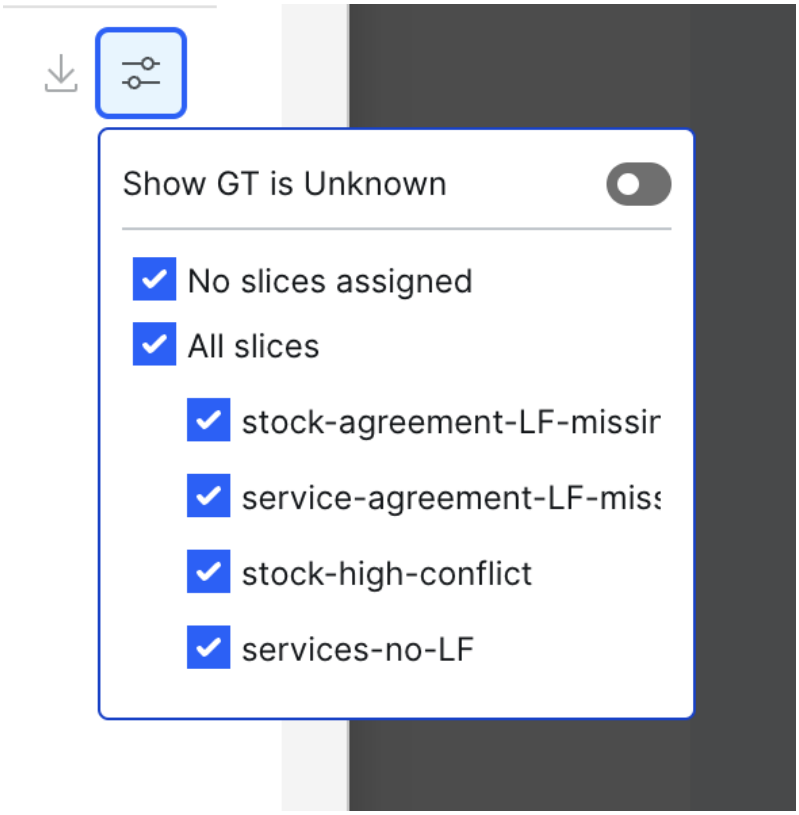
The slice-based analysis chart shows the correct (blue) and incorrect (red) data count for each slice. To also see the count for data without GT, toggle the **Show GT is Unknown** button.



Incorrect data signifies data where the ground truth is not equal to the prediction. You can go through incorrect data and categorize them into smaller error buckets by adding slices to the data based on the error types.

Filters can also be used to display only the slices you're interested in viewing. You can display all slices by checking that option, or you can display any number of them by deselecting the irrelevant ones in the filters.

Copyright © 2025 Snorkel AI, Inc. All rights reserved



The purple line above the bars represents the size of each error bucket accumulated from left to right. The values denoted by the purple line correspond to the percentages on the right y-axis.

If the user iterates through each error bucket from left to right and fixes all the errors in each of the error buckets, the % of correct data over the entire dataset will be the same as the number denoted on the purple line.

Slice-based analysis: Table view

The table view shows each of the slices and their respective **maximum F1 macro growths** of the entire selected dataset. The maximum F1 macro growth for each slice indicates the maximum increase of the F1 macro score (model performance) that we can achieve if all the errors in the slice get fixed.

Slice-based Analysis

Chart view | Table view

Slice	Max. F1 Macro growth	
incorrect-CWD	0.0%	Add LFs
stock-high-conflict	1.0%	Add LFs
services-no-LF	0.8%	Add LFs
service-agreement-LF-missing	1.6%	Add LFs
stock-agreement-LF-missing	3.0%	Add LFs
stock-no-LF	0.0%	Add LFs
stock-share-equity-keyword	0.0%	Add LFs
loan-mortgage-loan-keyword	0.0%	Add LFs
loan-no-LF	0.0%	Add LFs
bad-unclear-data	1.0%	Add LFs

Copyright © 2025 Snorkel AI, Inc. All rights reserved

One-click to add LFs from slices The Add LFs button (right column) allows you to add up to five suggested label functions (LFs) to their corresponding slices. Suggested LFs are based on the slices data and can be added using one click. A success notification appears if high quality LFs are added successfully.

A successful LF is one that has precision > 85%, and unsuccessful LFs cannot be added.

Postprocessors

Apply postprocessors when turned on will reflect in the metrics, i.e., all post processors for the model node will be applied before calculating the metrics. This is turned on by default.

Suggestions

In the right pane, you will find suggestions. Details on each message you may see:

- **Try tuning on valid set**: The Label Model's distribution has a class with distribution > 10% points different than the Ground Truth's or Model's output.
- **The following LFs are correlated with model errors...:** These LFs are causing a disproportionate amount of errors.
- **Try tuning or oversampling based on the valid set:** The train score is 5 points higher than the valid score.
- **Resample dev set iterate on LFs:** The valid score is 5 points higher than the train score.
- **Try improving precision of LFs in Label:** More than half the errors are from the LFs outputting incorrect predictions.
- **Try training a model using LFs as features:** More than half the errors are caused by the model incorrectly predicting a correct LF output.
- **Try writing more LFs:** The model predicts `ABSTAIN` on more than half of errors.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

Class level metrics

Class level metrics displays model performance on per-class basis to help decide where to best focus efforts.

This topic explains the class level metrics available in Snorkel Flow:

- Word precision
- Word recall
- Word F1
- Word Entity F1
- Word Entity precision
- Word Entity recall
- Word Entity F1

Word metrics

Word metrics are metrics that evaluate the performance of a model in identifying and labeling individual words in sequence tagging tasks, such as Named Entity Recognition (NER). To calculate word metrics, we consider the following:

- **True Positive (TP)**: A word that the model correctly labels as belonging to the target category.
- **False Negative (FN)**: A word that belongs to the target category but the model either misses or incorrectly labels.
- **False Positive (FP)**: A word that does not belong to the target category but the model incorrectly labels as belonging to the target category.

Word recall

Word recall evaluates how well a model correctly identifies all relevant words that belong to a specific category. It measures the proportion of ground truth positive words that the model correctly labels as positive.

This is the formula for word recall:

$$\text{Word Recall} = \frac{\text{True Positives (TP)}}{\text{True Positives (TP)} + \text{False Negatives (FN)}}$$

A high word recall indicates that the model is successfully identifying most or all of the relevant words for the target category, meaning it has few false negatives. However, word recall alone does not provide information on how accurately it labels those words, meaning it doesn't account for false positives. For a more comprehensive evaluation, use word recall alongside word precision and the word F1 score.

Word precision

Word precision evaluates the specificity of a model in sequence tagging tasks. Word precision measures how accurately the model labels words as belonging to a specific category.

This is the formula for word precision:

$$\text{Word Precision} = \frac{\text{True Positives (TP)}}{\text{True Positives (TP)} + \text{False Positives (FP)}}$$

Copyright © 2025 Snorkel AI, Inc. All rights reserved

A high word precision indicates that most of the words labeled as positive by the model are indeed correct, meaning there are few false positives. However, word precision alone does not account for missed words (false negatives), which is why it's often reported along with word recall and the word F1 score to provide a balanced view of the model's performance.

Word F1

Word F1 is a metric that combines word precision and word recall to provide a single measure of a model's performance. This metric balances the trade-off between precision, which is how many labeled words are correct, and recall, which is how many relevant words are found, to give a holistic view of the model's accuracy in identifying and labeling individual words.

The Word F1 score is the harmonic mean of word precision and word recall:

$$\text{Word F1} = \frac{2 \times \text{Word Precision} \times \text{Word Recall}}{\text{Word Precision} + \text{Word Recall}}$$

The Word F1 score ranges from 0 to 1:

- A score of **0** indicates the worst possible performance, meaning the model failed in both precision and recall.
- A score of **1** indicates perfect precision and recall, meaning the model has correctly identified and labeled all relevant words without any errors.

The Word F1 score is especially useful when both false positives and false negatives are important, as it provides a balanced measure of the model's overall ability to accurately identify and label individual words in the dataset.

Word Entity Metrics

Word Entity Metrics are metrics that evaluate the performance of a model in identifying and labeling multi-word entities (spans) in tasks like NER. To calculate word entity metrics, we consider the following:

- **True Positive (TP)**: A group of words that the model correctly identifies and labels as an entity, matching both the entity type and exact boundary.
- **False Positive (FP)**: A group of words that the model identifies and labels as an entity, but is either incorrect in type or boundary or does not match any true entity in the data. Even partial matches are considered false positives.
- **False Negative (FN)**: A group of words that the model misses, either because the model didn't identify it at all or didn't match the entity's boundaries or type correctly.

Word Entity Recall

Word Entity Recall evaluates how well a model correctly identifies all relevant groups of words that belong to a specific category. It measures the proportion of ground truth positive groups of words that the model correctly labels as positive.

This is the formula for word entity recall:

$$\text{Word Entity Recall} = \frac{\text{True Positives (TP)}}{\text{True Positives (TP)} + \text{False Negatives (FN)}}$$

A high word entity recall indicates that the model successfully identifies most or all relevant spans within the dataset, meaning it does not miss many entities. Like word recall, word entity recall is most informative when used alongside word entity precision and the word entity F1 score, as this provides a balanced view of the model's ability to find and accurately label all entities.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

Word Entity Precision

Word Entity Precision evaluates the specificity of a model in sequence tagging tasks. Word entity precision measures how accurately the model labels groups of words as belonging to a specific category.

This is the formula for word entity precision:

$$\text{Word Entity Precision} = \frac{\text{True Positives (TP)}}{\text{True Positives (TP)} + \text{False Positives (FP)}}$$

A high word entity precision indicates that most of the spans the model identifies as entities are accurate, meaning they correctly match the entities' boundaries and types as defined in the labeled data. However, word entity precision alone does not account for whether the model missed any actual spans, which is why it's often used in conjunction with word entity recall and word entity F1 score for a more complete evaluation.

Word Entity F1

Word Entity F1 is a metric that combines word entity precision and word entity recall to provide a single measure of a model's performance in identifying and labeling multi-word entities (spans) in tasks like NER. It is particularly useful because it balances the trade-off between word entity precision and word entity recall, making it an effective metric when both false positives and false negatives are of concern.

The Word Entity F1 score is calculated as the harmonic mean of word entity precision and word entity recall:

$$\text{Word Entity F1} = \frac{2 \times \text{Word Entity Precision} \times \text{Word Entity Recall}}{\text{Word Entity Precision} + \text{Word Entity Recall}}$$

A Word Entity F1 score ranges from 0 to 1:

- A score of 0 indicates that the model failed to correctly identify any spans.
- A score of 1 indicates perfect precision and recall, meaning the model correctly identifies all spans without any errors or omissions.

The Word Entity F1 score is especially valuable in scenarios where it is important to accurately identify spans (precision) and ensure that no spans are missed (recall), as it reflects the model's overall effectiveness in capturing the correct entities.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

Candidate extractor scoring

What you will learn:
How to evaluate performance of a candidate extractor against ground truth

In the text extraction application (see Information extraction: Extracting execution dates from contracts), the first step is to identify a high-recall set of candidate spans. However, in many cases, the candidate extractor may not be perfect. In this case, you should consider evaluating the performance of the candidate extractor against ground truth you collected externally.

In this example, let's assume that we have created a text extraction application and commited the candidate extractor as detailed in Information extraction: Extracting execution dates from contracts

The in-platform Notebook interface allows you to evaluate the recall of your extractor.

First, get the `node_uid` of the application DAG based on the application name.

```
import snorkelflow.client as sf
# APP_NAME = <insert your application name here>
extractor_node = sf.get_node_uid(APP_NAME, search_op_type="SpanExtractor")[0]
```

Second, get the output of the extractor node:

```
extracted_spans_df = sf.get_node_output_data(APP_NAME,
extractor_node).reset_index()
```

Next, add active datasources to the extractor node:

```
datasource_uids = [x['datasource_uid']  for x in
sf.get_datasources(DATASET_NAME)]
sf.add_active_datasources(extractor_node, datasource_uids)
```

Then, add the external document level ground truth labels to the extractor node. We will need `x_uids`, which is a list of `uid` of documents, formatted as `doc::{uid}`, and `labels`, which is a list of spans (tuples of `char_start`, `char_end`, and `_gt_label`) for the corresponding document. See example below:

```
# assume we have two documents, with 1 and 2 as the ``uid``.
x_uids = ["doc::1", "doc::2"]
labels = [
    [(0, 1, 0)], # GT labels for doc with uid of 1
    [(0, 2, 0), (4, 5, 1)], # GT labels for doc with uid of 2
]
sf.add_ground_truth(extractor_node, x_uids, labels)
```

Finally, use `sf.get_candidate_extractor_metrics` to evaluate the candidate extractor.

```
sf.get_candidate_extractor_metrics(extractor_node, extracted_spans_df)
```

Copyright © 2025 Snorkel AI, Inc. All rights reserved

Copyright © 2025 Snorkel AI, Inc. All rights reserved

Deploying Snorkel-built models

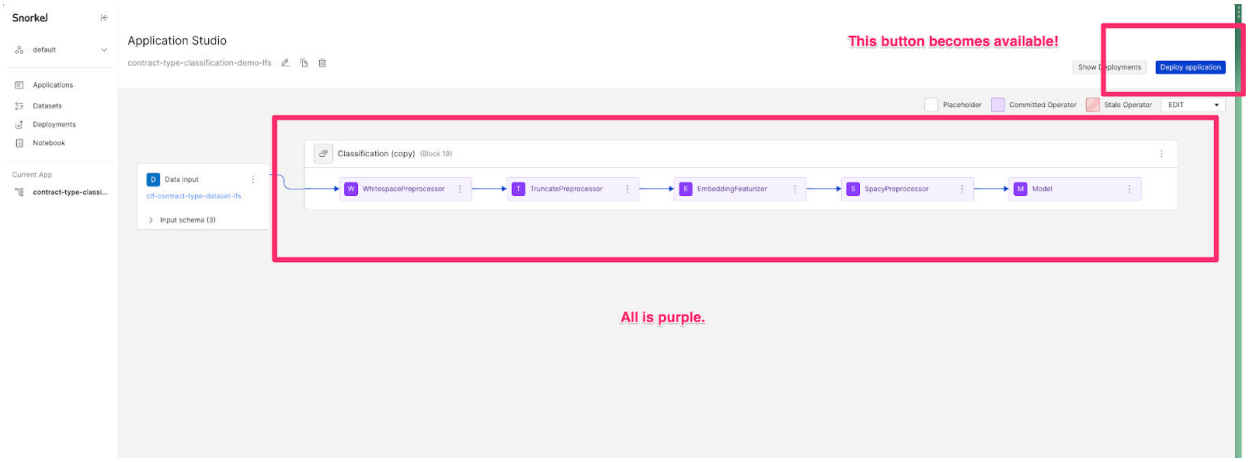
This page walks through how to deploy a Snorkel-built application, export said deployment, and stand it up in an external production environment for inference and scoring.

Requirements

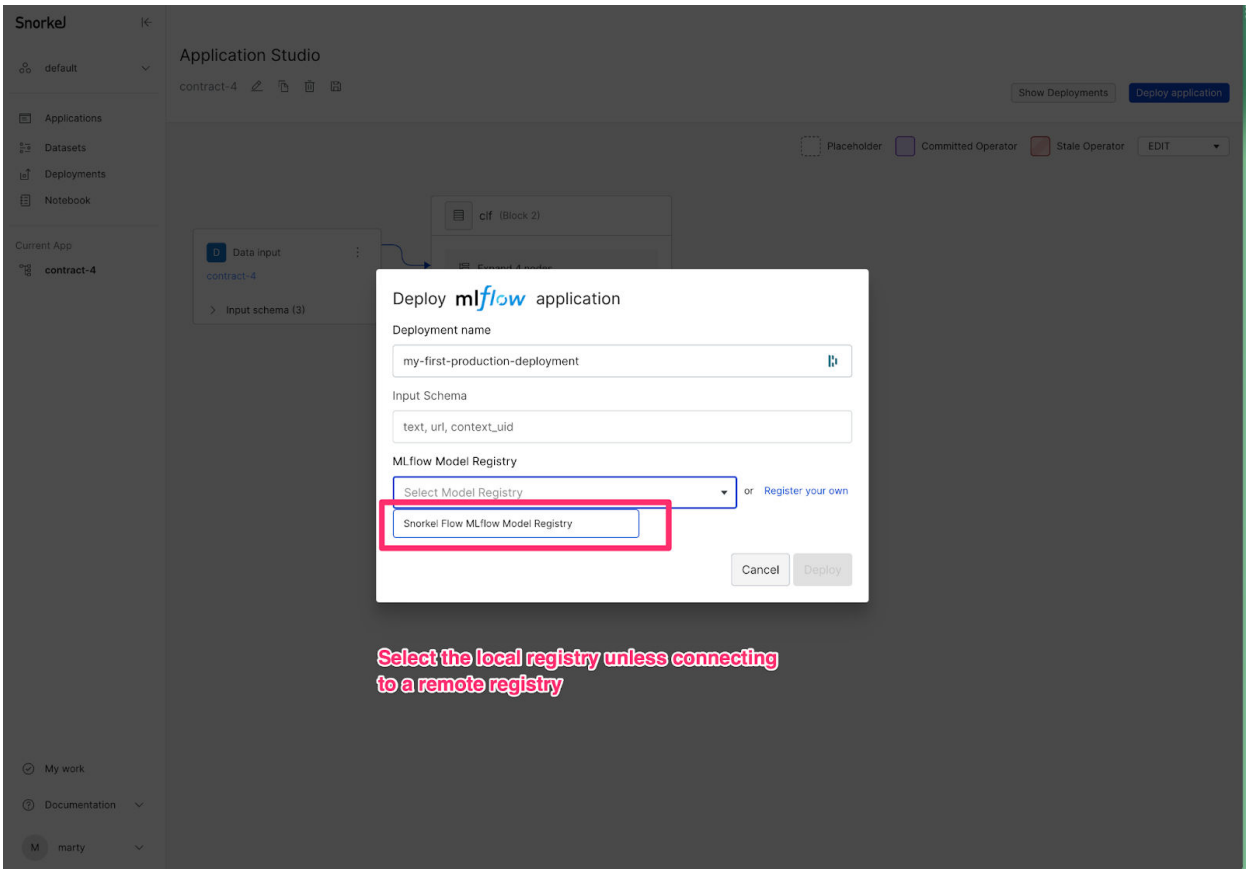
Requirements	Options
Supported operating systems	- Ubuntu 20.04+ - CentOS 7+ - RHEL 7+
Supported hardware	- CPU: amd64 (x86_64) - GPU: Nvidia Turing GPU or later generation supported by CUDA 11.8
Required software	- Docker - Python 3.8 - Python 3.10 - pyenv/virtualenv (or Conda) - MLflow 2.5.0+
Recommended minimum deployment environment specifications	- MLflow 2.5.0+ - 16 GB RAM - 100 GB disk space

Prepping and deploying a Snorkel Flow application with MLflow

Before deploying your application, ensure that each node in the DAG view is committed. Committed nodes are purple. Once all nodes are committed, the **Deploy application** button becomes available in the upper right hand side of the screen. A versioned application graph is bundled up and subsequently referred to as a **Deployment**.



Copyright © 2025 Snorkel AI, Inc. All rights reserved



Each deployment requires a unique name and input schema. The input schema is originally generated at dataset creation, but can be updated using the SDK.

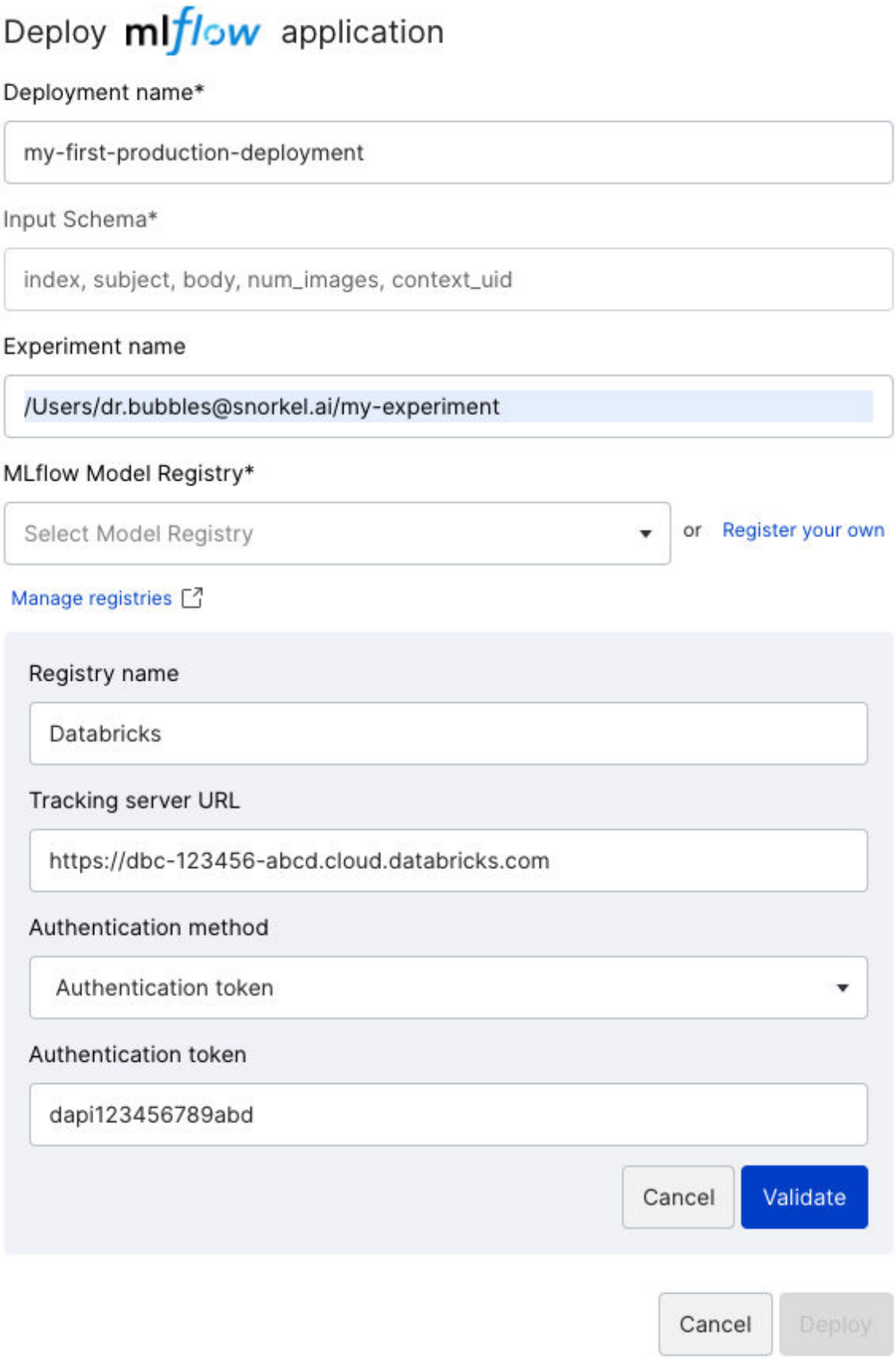
> **NOTE**
>
> `context_uid` is always included in the input schema, but is strictly an optional field.

Snorkel does not support long-running application deployments in the platform. Instead, Snorkel Flow bundles deployment artifacts into an MLflow package, an open source standard for model hosting. MLflow is supported by all major cloud providers and most major model serving vendors.

By default, you can deploy applications using Snorkel Flow's MLflow Model Registry. However, you can also register connections with remote MLflow registries by selecting **Register your own** as shown below.

For the Databricks Model Registry (including Azure Databricks), specify the **Experiment name** in the format `/Users/<username>/<experiment_name>`, where `<username>` is your Databricks username. See Model serving with Databricks for how to serve the registered model on Databricks.

Copyright © 2025 Snorkel AI, Inc. All rights reserved



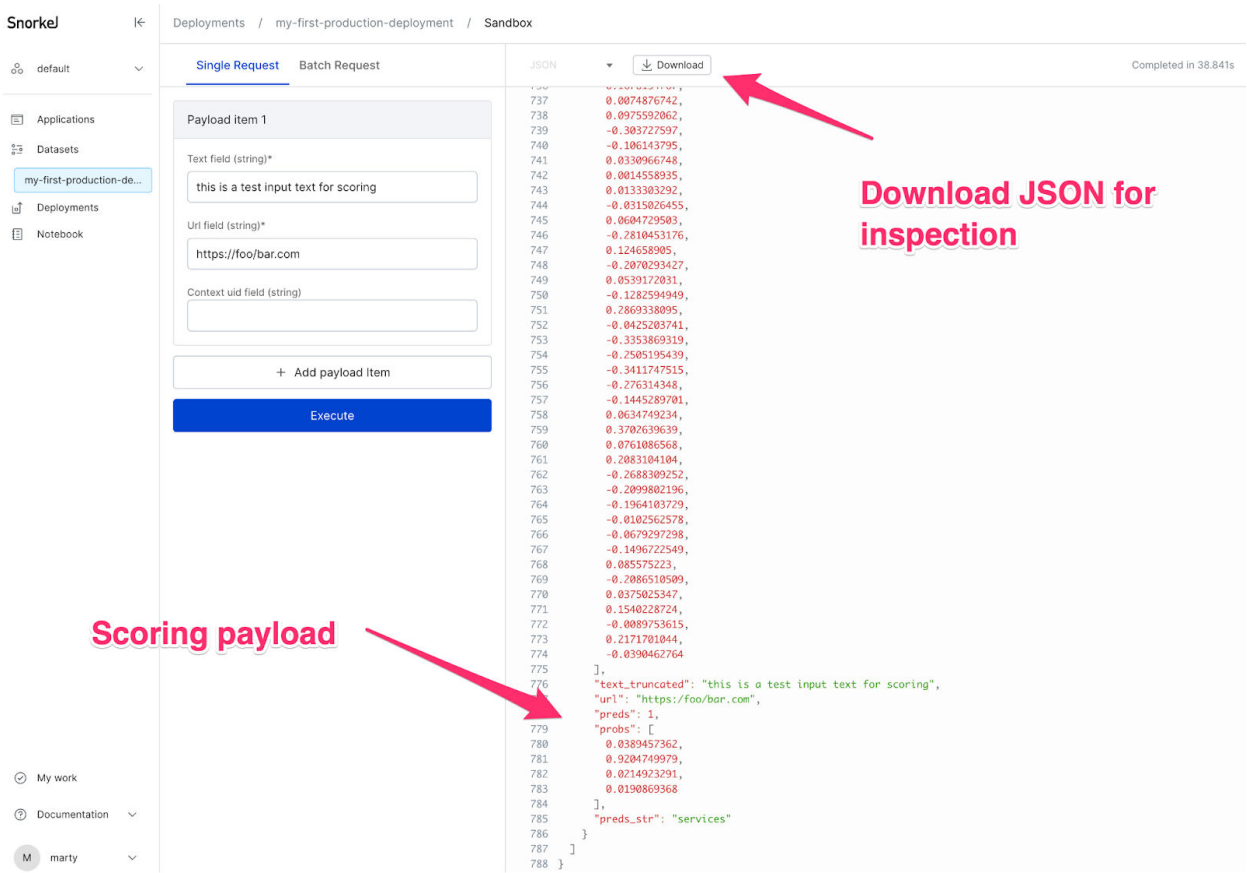
Copyright © 2025 Snorkel AI, Inc. All rights reserved

Once a workflow is deployed, you can select the **Deployments** option in the left-side menu to see your deployments.

Crawl: Testing deployments in Snorkel Flow's scoring sandbox

You can test your Snorkel Flow model deployment in a minimal sandbox environment that is intended for testing inference on a small number of records (< 5). To access the sandbox environment for a given deployment, select the inline associated icon .

> **TIP**
>
> Before exporting the deployment out of Snorkel Flow, we strongly recommend using the **Scoring Sandbox** to ensure that a test record can be properly sent through the pipeline. This allows you to clearly understand the input fields and types that are required to generate predictions.



Copyright © 2025 Snorkel AI, Inc. All rights reserved

Configuring and right-sizing a production deployment environment

Before exporting the deployment, it is important to consider the following questions. Discuss these questions with your Snorkel account team to determine the infrastructure that is required to host the deployment:

- Does the deployment contain any expensive operators? Examples include calculating document-level embeddings, or an expensive end model or model ensemble. This will affect resource allocation and inference latency.
- How often will inference be run? What is the interval of a single time window? Yearly/quarterly/monthly/weekly/daily/multiple times per day?
- How many records will be sent for scoring per time window?
- What are the CPU and RAM resources that are available for inference? Snorkel recommends 4 CPU / 16 GB RAM minimum, but this can vary depending on the construction of the application graph (DAG).
- Will additional business logic need to be performed post-scoring? For example, will predictions only be considered if model confidence is greater than 85%? Will extracted fields need to be combined before a database insert? Post-processing and business logic can be handled in the DAG or outside of the Snorkel deployment object.
- Does the business have inference throughput or latency requirements? This will affect resource allocation.

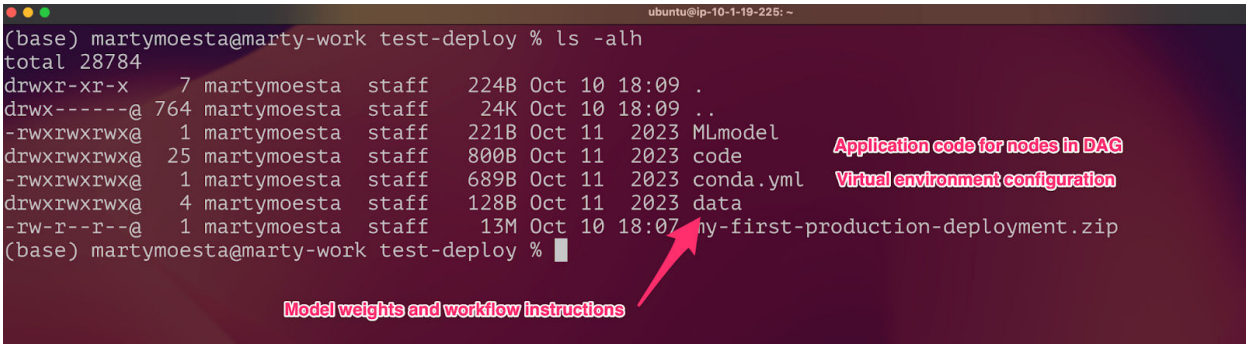
The following table has our recommendations for T-shirt sizing a deployment. Snorkel cannot guarantee performance or uptime SLA's surrounding deployments. Work with your Snorkel account team to right-size CPU/RAM allocations, given your business requirements.

T-Shirt Size	Model size	Frequency and Volume	Minimum Resourcing Recommendation
Small	Small	Batch inference < 10k records yearly or quarterly	4 CPU / 16 GB RAM
Medium	Small	Batch inference >10k records monthly	4 CPU / 16 GB RAM
Large	Medium/Large	Batch inference > 10k records daily, live inference	12 CPU/ 48 GB RAM
XL	Any	Batch inference > 1M records daily, live inference	24 CPU / 96 GB RAM

Walk: Testing a deployment in an external environment (local python)

To download your deployment to test locally, select the **Deployments** option in the left-side menu, then select the **Download deployment package** button. This will download the code and libraries that are required to run the exported deployment as a zip file to your local machine. An exported deployment can also be downloaded using the SDK. Once unzipped, the export will look something like the following:

Copyright © 2025 Snorkel AI, Inc. All rights reserved



For a full explanation of the contents of the export, see the MLflow models documentation.

> **TIP**
>
> We recommend hosting deployments through containerized application images, either with docker or kubernetes. However, before generating an image and standing up a container, we recommend that you first test inference in a local python execution environment.

Unzip the contents of the downloaded package. The following command will unzip the downloaded package into a directory called `mlflow_deployment`:

```
$ unzip -d mlflow_deployment <<insert_downloaded_deployment>>.zip
```

Next, gather example scoring data. MLflow accepts json exports of Pandas DataFrames, oriented in the `split` format. Convert a DataFrame to valid json using the following code snippet:

```
import pandas as pd
import json
df = pd.DataFrame("<<insert your data here>>")
with open("scoring_example.json", "w") as f:
    json.dump({"dataframe_split": df.to_dict(orient='split')}, f)
```

Now you can use the `mlflow models predict` command to score example records:

```
$ mlflow models predict -m mlflow_deployment -i scoring_example.json
####output
[{"text": "the quick brown fox jumped over the lazy log",
"url": "https://snorkel.ai", "preds": 2,
"probs": [0.0003, 0.0003, 0.998, 0.0004], "preds_str": "stock"}]
```

If your machine is equipped with GPU and want to take advantage of it, pass `IS_GPU_ENABLED=1` as an environment variable:

```
$ IS_GPU_ENABLED=1 mlflow models predict -m mlflow_deployment -i
scoring_example.json
```

Copyright © 2025 Snorkel AI, Inc. All rights reserved

Run: Generating a container-based deployment (Docker, Kubernetes, or Google Cloud Run)

We recommend using container-based approaches for hosting production inference workloads. Containers enable OS-agnostic workloads to run anywhere with variable resource allocations that can be easily spun up and down on-demand.

Before standing up a container hosting the deployment, an application image must first be generated. Snorkel supports Docker for management of application images and supports the MLflow Command-Line Interface (CLI) for generating the application image. To generate an application image:

1. Open a terminal.
2. Login to either Docker locally or Dockerhub through the docker login command.
3. Navigate to the unzipped deployment directory using the command line. Execute the following command:

```
$ mlflow models build-docker -m **{path to unzipped deployment directory}** -n
my-image-name
```

The `build-docker` command will convert the deployment export into an application image. See build-docker for more information. After the command completes, validate the creation of an application image through the docker images command.

Customizing a Dockerfile before building

If you run into issues with the `build-docker` command or you want to customize the Dockerfile before building the application image, you can use the `generate-dockerfile` command in MLFlow 2.5.0+ to generate a Dockerfile and artifacts that can be customized further before building the application image.

Common changes include:

- Changing Ubuntu versions
- Matching Python versions between the Dockerfile and the deployment

Use this simple shell script to generate a Dockerfile and modify it before building the image:

Copyright © 2025 Snorkel AI, Inc. All rights reserved

```
mlflow models generate-dockerfile -m ${MODEL_DIR} -d ${OUTPUT_DIR}
cd ${OUTPUT_DIR}
MODEL_FOLDER=$(basename ${MODEL_DIR})

# Update Dockerfile
sed -i 's/FROM ubuntu:20.04/FROM ubuntu:22.04/' Dockerfile

# Check Python version in conda.yaml
PYTHON_VERSION=$(grep 'python=' model_dir/${MODEL_FOLDER}/conda.yaml | cut -d
'=' -f 2 | cut -d '.' -f 1,2)

# Extract Python version from Dockerfile
DOCKERFILE_PYTHON_VERSION=$(grep -oP 'python3\.\d+' Dockerfile | head -1 | cut
-d 'n' -f 2)

echo "PYTHON_VERSION: $PYTHON_VERSION"
echo "DOCKERFILE_PYTHON_VERSION: $DOCKERFILE_PYTHON_VERSION"

if [[ -n $PYTHON_VERSION && $PYTHON_VERSION != $DOCKERFILE_PYTHON_VERSION ]];
then
    sed -i "s/python${DOCKERFILE_PYTHON_VERSION}/python${PYTHON_VERSION}/g"
Dockerfile
    sed -i "s/python${DOCKERFILE_PYTHON_VERSION}-
distutils/python${PYTHON_VERSION}-distutils/g" Dockerfile
fi

docker build -t ${DOCKER_TAG} .
```

Deploying the image and testing the container

Use the following command to run the recently-created image as a container using Docker:

```
$ docker run -p 5001:8080 my-image-name
```

To run a model on GPUs,

```
$ export IS_GPU_ENABLED=1
$ docker run --gpus all -e IS_GPU_ENABLED -p 5001:8080 my-image-name
```

This command will expose a REST API on all interfaces on port 5001. Validate this with the docker ps command. The **/invocations** API is used to score records (see Deploy MLflow models for more information). Open a separate terminal window and test record scoring with a simple curl command:

```
$ curl [http://127.0.0.1:5001/invocations ](http://127.0.0.1:5001/invocations)
  -H 'Content-Type: application/json' \
  -d @/path/to/example_scoring.json -v
```

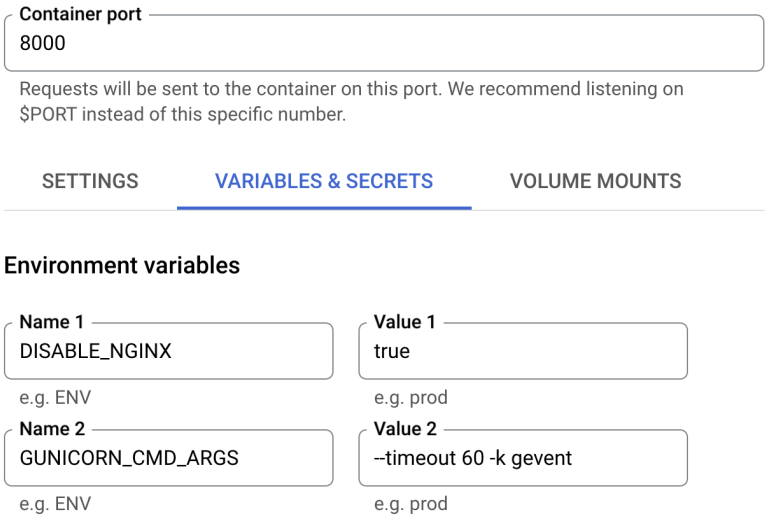
Copyright © 2025 Snorkel AI, Inc. All rights reserved

In this example, *example_scoring.json* is a Pandas DataFrame with column *input_schema* that was converted to a json object using the `orient = “split”` parameter in the Pandas `to_json()` function.

Deploying to Google Cloud Run

The Docker image that is built in the above section can be pushed to Artifact Registry (or Container Registry) in Google Cloud and can be deployed to Cloud Run. To do this, change the following configurations in Cloud Run.

1. Set the Container port to 8000.
2. Add the following environ variables:
   - `DISABLE_NGINX=true`
   - `GUNICORN_CMD_ARGS="--timeout 60 -k gevent"`
3. Set the Memory 2GB or higher.



Summary

This walkthrough demonstrated how to deploy a Snorkel-built application, export said deployment, and stand it up in an external production environment. Reach out to your Snorkel account team for assistance when architecting a long-term production deployment using Snorkel-built artifacts or contact support@snorkel.ai for assistance.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

Deploying Snorkel-built models to AWS SageMaker

This tutorial walks through the three steps that are required to deploy a Snorkel-built application to AWS SageMaker:

1. Deploy and export your model in Snorkel Flow.
2. Build and push a containerized runtime environment to AWS ECR, if none exists.
3. Push the exported model object to the AWS SageMaker model registry.

Requirements

Ensure your environment meets all the requirements for deploying a Snorkel-built model.

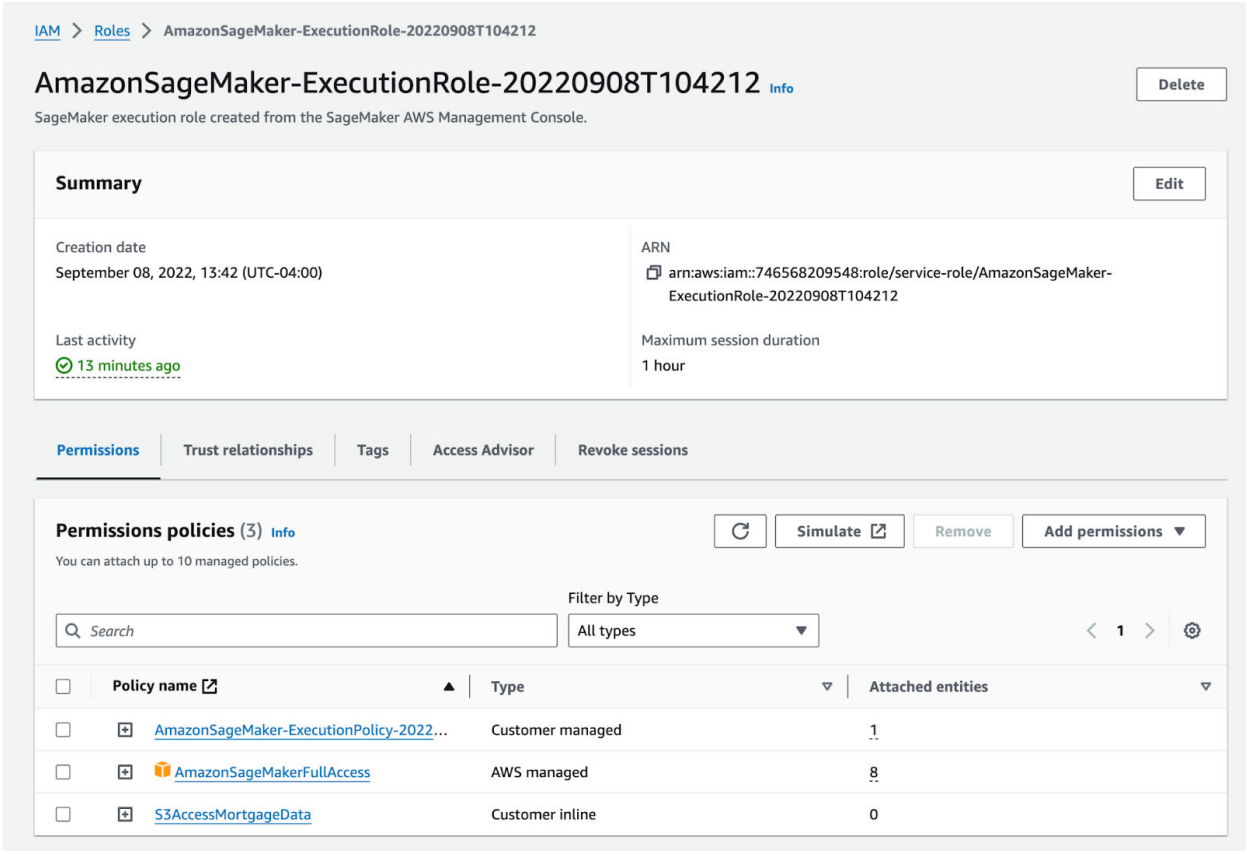
AWS CLI

Install and configure AWS CLI with the relevant credentials. For more, see the AWS CLI documentation.

Ensure you have permissions to push containers to Amazon ECR from the AWS CLI.

Verify that you have AWS CLI access with the `aws sts get-caller-identity` command. If this command does not return an JSON IAM profile, see the AWS CLI documentation for more information about getting access configured.

Configure and assign the SageMaker execution role through the AWS CLI. The SageMaker execution role includes full access to the SageMaker service. Configure the role here, and then refined for production use:



Copyright © 2025 Snorkel AI, Inc. All rights reserved

Deploy and export your model in Snorkel Flow

Follow the instructions in Deploying Snorkel-built models to deploy your model in Snorkel Flow. Next, download the deployment artifacts. To do so, select the **Deployments** option in the left-side menu, then select the **Download deployment package** button. This button will download the code and libraries that are required to run the exported deployment as a zip file to your local machine. Create a new directory on your development system and unzip the file into said directory.

Now you're ready to deploy your model to AWS SageMaker.

Deploy to AWS SageMaker

There are two options for deploying your model to AWS SageMaker:

- Built-in SageMaker support in MLflow
- AWS CLI

Deploy to AWS using MLFlow built-in SageMaker support

This option uses the built-in SageMaker support in MLflow to deploy your model. It's less flexible than the AWS CLI option, but it's easier to use for simpler deployments.

Building a model runtime environment

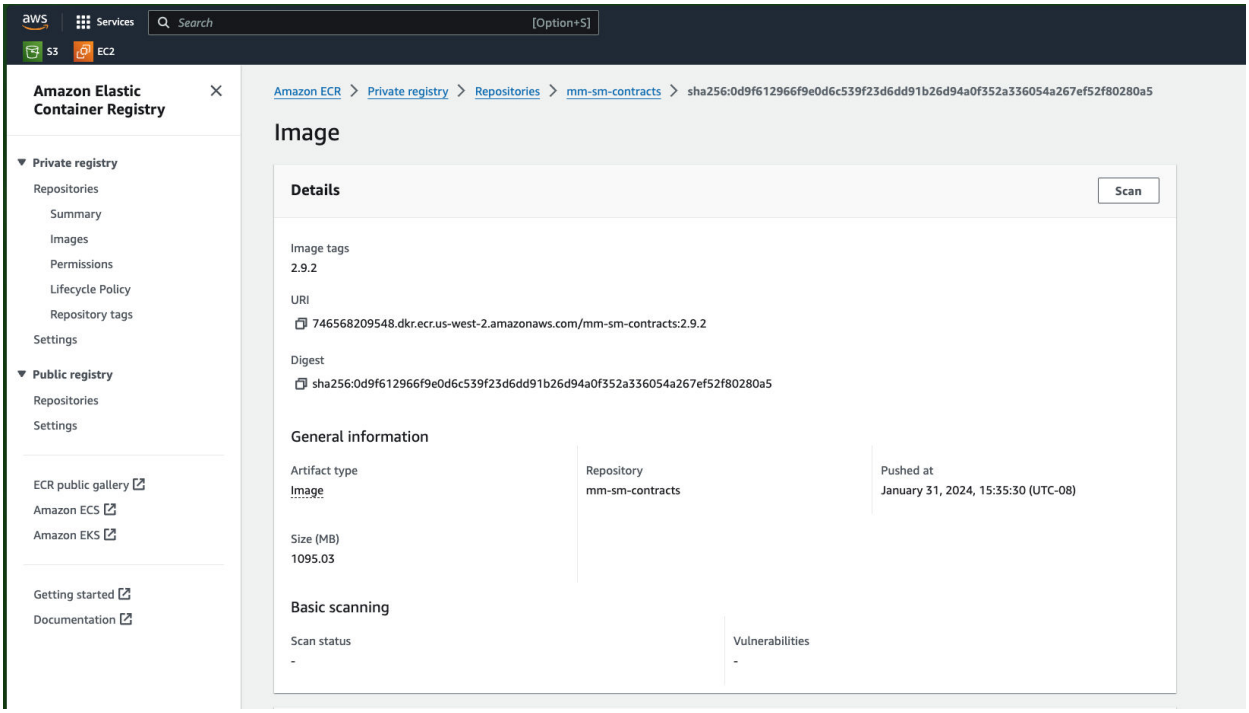
Copy the ARN of the Sagemaker role that will be used to deploy the model and save it as an environment variable in your terminal session (`ARN="{INSERT ARN}"`). Next, navigate to the local model directory in the terminal and run the following command:

```
mlflow sagemaker build-and-push-container
```

This will create a containerized environment for the MLflow deployment and push this environment to Amazon ECR. This takes approximately 10 minutes.

After the command completes, log into the AWS console or use the AWS CLI to view the new container. This container will be used later when creating the Sagemaker deployment and endpoint.

Copyright © 2025 Snorkel AI, Inc. All rights reserved



Deploy an AWS SageMaker endpoint

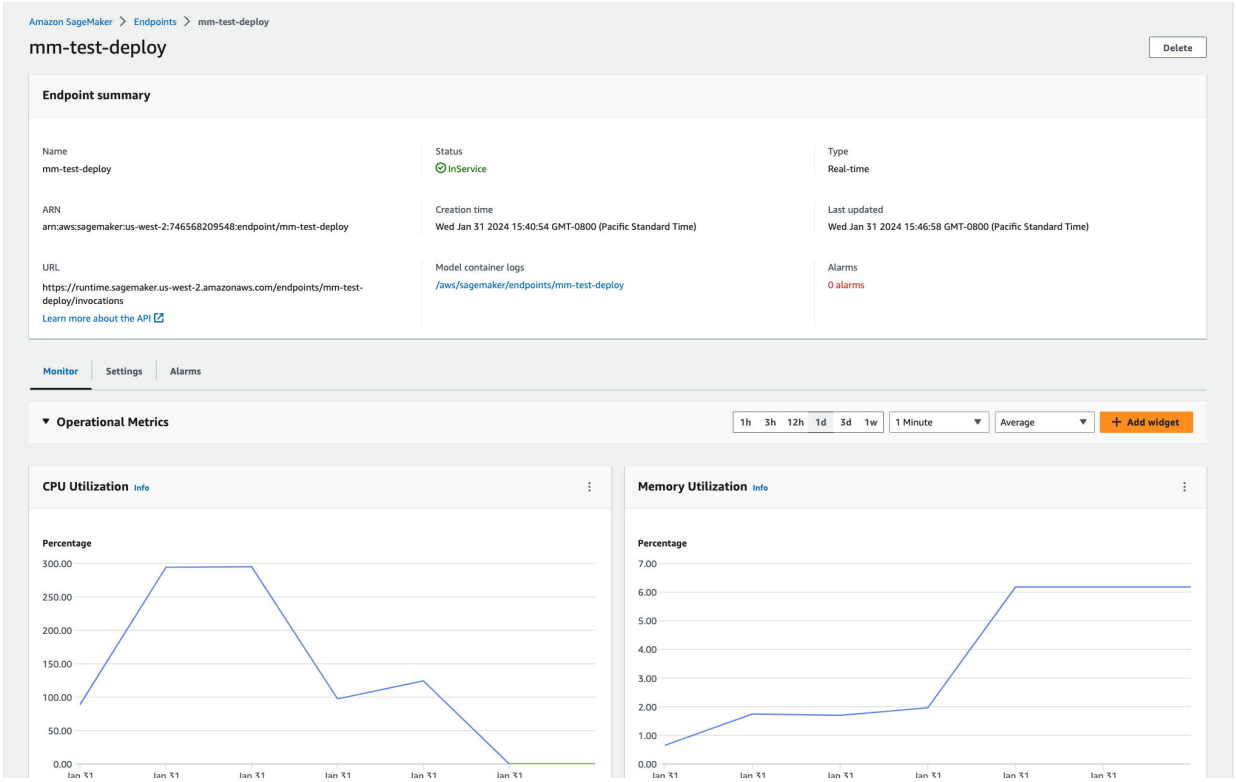
Now that a container runtime environment has been created, it can be combined with the local model directory and pushed to Sagemaker as an endpoint. The endpoint is then used to directly perform model scoring. To create an endpoint, run the following command:

```
mlflow deployments create -t sagemaker -m {local model directory path}
--name {deployment name}
-C region_name={region name}
-C instance_count={instance count}
-C execution_role_arn=$ARN
-C env='{"DISABLE_NGINX": "true"}'
```

For example, you can specify `region_name="us-west-2"` and `instance_count=1`.

You can view the newly created endpoint in the AWS console or CLI.

Copyright © 2025 Snorkel AI, Inc. All rights reserved



Inference (scoring new data)

For simplicity, this tutorial will focus on batch deployments for **inService** endpoints. To query, create a deployment client and pass the DataFrame that needs scoring to the deployed endpoint.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

```
ubuntu@ip-10-1-27-195:~/mlflow$ python
Python 3.8.10 (default, Nov 22 2023, 10:22:35)
[GCC 9.4.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> region = "us-west-2"
>>> import mlflow.deployments
>>> deployment_client = mlflow.deployments.get_deploy_client("sagemaker:/" +
region)
>>> deployment_name = "mm-test-deploy"
>>> deployment_info = deployment_client.get_deployment(name=deployment_name)
>>> print(f"MLflow SageMaker Deployment status is:
{deployment_info['EndpointStatus']}")
MLflow SageMaker Deployment status is: InService
>>>
>>> import pandas as pd
>>> d = {'context_uid': [1, 2], 'subject': ["test", "test"], 'body': ["test",
"test"], 'num_images': [2, 3]}
>>> df = pd.DataFrame(data=d)
>>> df
   context_uid subject  body  num_images
0            1    test  test           2
1            2    test  test           3
>>> prediction1 = deployment_client.predict(deployment_name, df)
>>> prediction1
{'predictions': [{'body': 'test', 'context_uid': 1, 'num_images': 2.0,
'subject': 'test', 'preds': 0, 'probs': [0.379504032603055,
0.620495967396945], 'preds_str': 'SPAM'}, {'body': 'test', 'context_uid': 2,
'num_images': 3.0, 'subject': 'test', 'preds': 0, 'probs':
[0.42396641921101064, 0.5760335807889894], 'preds_str': 'SPAM'}]}
```

Deploy to AWS using the AWS CLI

This option uses the AWS CLI to deploy your model to SageMaker. It's more flexible than the MLflow built-in SageMaker support, but it takes a few more steps. Use this option if the first option doesn't work for your use case.

1. Prepare your environment. Ensure you have the necessary AWS credentials configured and gather the required parameters:
   - AWS region
   - ECR repository name
   - SageMaker role ARN
   - Endpoint name

2. Log in to Amazon ECR. Authenticate Docker with Amazon ECR to allow image uploads.

   ```
   aws ecr get-login-password --region <region> | docker login --username AWS
   --password-stdin <account_id>.dkr.ecr.<region>.amazonaws.com
   ```

Copyright © 2025 Snorkel AI, Inc. All rights reserved

3. Create ECR repository for the runtime environment of the model.

```
aws ecr create-repository --repository-name <ecr_repo_name> --region
<region>
```

4. Build and test your Docker image locally prior to pushing to ECR. See the instructions in the Deploying Snorkel-built models tutorial.

5. Once you have built and tested your Docker image locally, tag it with the ECR image URI:

```
docker tag <local_image_name>:<image_tag> <account_id>.dkr.ecr.
<region>.amazonaws.com/<ecr_repo_name>:<image_tag>
```

6. Push the tagged Docker image to ECR repository:

```
docker push <account_id>.dkr.ecr.<region>.amazonaws.com/<ecr_repo_name>:
<image_tag>
```

7. Create a SageMaker model using the ECR image URI and the SageMaker execution role ARN:

```
aws sagemaker create-model \
    --model-name <model_name> \
    --primary-container Image=<ecr_image_uri> \
    --execution-role-arn <sagemaker_role_arn> \
    --region <region>
```

8. Create a SageMaker endpoint configuration. Here's a basic example:

```
aws sagemaker create-endpoint-config \
    --endpoint-config-name <endpoint_name> \
    --production-variants VariantName=AllTraffic,ModelName=
<model_name>,InitialInstanceCount=1,InstanceType=ml.m5.large,InitialVariantWe
\
    --region <region>
```

9. Deploy the SageMaker endpoint to serve your model:

```
aws sagemaker create-endpoint \
    --endpoint-name <endpoint_name> \
    --endpoint-config-name <endpoint_name> \
    --region <region>
```

Copyright © 2025 Snorkel AI, Inc. All rights reserved

Inference

```
aws sagemaker-runtime invoke-endpoint \
    --endpoint-name <endpoint_name> \
    --body fileb://<data_file> \
    --content-type application/json \
    <output_json_file>
```

Conclusion

This tutorial demonstrates how an exported Snorkel workflow can be configured in AWS SageMaker to score production data. If you have any questions about this technique, please reach out to support@snorkel.ai.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

Deploying Snorkel-built models to Azure Machine Learning

This tutorial walks through two options for deploying a Snorkel-built model on Azure Machine Learning (Azure ML):

- Using Azure CLI
  1. Configure a Dockerfile to be compatible with Azure ML.
  2. Build and push the Dockerfile to Azure Container Registry.
  3. Create an endpoint and deploy your model.
- Using Azure ML Studio UI
  1. Modify your MLflow model so that it is compatible with Azure ML.
  2. Upload your model to Azure ML.
  3. Create an endpoint and deploy your model.

Using the Azure CLI requires more configuration upfront, but is easier to debug and much more flexible for most use cases so is therefore the recommended method. Using the Azure ML Studio UI is easier to set up, but oftentimes has issues with external dependencies, so for most use cases it will not be the best option.

Requirements

- Ensure your environment meets all the requirements for deploying a Snorkel-built model.
- Azure CLI installed on your machine. For more, see the Azure CLI documentation.
- An Azure account with access to an existing Azure ML workspace.
- An MLflow model that was downloaded from Snorkel Flow.

Using Azure CLI

Configure a Dockerfile

Here's a standard Dockerfile that you can use as a starting point. You may need to make some modifications based on Model dependencies. The key components are:

- Install system dependencies
- Install Python & model dependencies
- Start MLflow serving with the explicit model path

Copyright © 2025 Snorkel AI, Inc. All rights reserved

```
# Build an image that can serve mlflow models.
FROM ubuntu:22.04

ARG MLFLOW_MODEL_NAME
ARG DEPLOY_PORT

ENV MLFLOW_MODEL_NAME=${MLFLOW_MODEL_NAME}
ENV DEPLOY_PORT=${DEPLOY_PORT}

# Install system dependencies
RUN apt-get -y update && DEBIAN_FRONTEND=noninteractive TZ=Etc/UTC apt-get
install -y --no-install-recommends wget curl nginx ca-certificates bzip2
build-essential cmake git-core

RUN apt install -y python3.10 python3.10-distutils python3-dev\
    && ln -s -f $(which python3.10) /usr/bin/python \
    && wget https://bootstrap.pypa.io/get-pip.py -O /tmp/get-pip.py \
    && python /tmp/get-pip.py
RUN pip install gevent mlflow==2.16.0

ENV MLFLOW_DISABLE_ENV_CREATION=True
ENV ENABLE_MLSERVER=False
ENV GUNICORN_CMD_ARGS="--timeout 60 -k gevent"

# install dependencies for the MLflow model
COPY ${MLFLOW_MODEL_NAME} /opt/azureml/${MLFLOW_MODEL_NAME}
WORKDIR /opt/azureml/${MLFLOW_MODEL_NAME}
RUN pip install -r /opt/azureml/${MLFLOW_MODEL_NAME}/requirements.txt

# clean up apt cache to reduce image size
RUN rm -rf /var/lib/apt/lists/*

# Set the model path environment variable
ENV MODEL_PATH="/var/azureml-app/azureml-
models/${MLFLOW_MODEL_NAME}/1/${MLFLOW_MODEL_NAME}"

# Start MLflow serving with the explicit model path
CMD ["sh", "-c", "mlflow models serve --model-uri \"$MODEL_PATH\" --no-conda -
-port $DEPLOY_PORT --host 0.0.0.0"]
```

Create deployment and endpoint configuration files

Azure ML requires two configuration files to deploy a model:

- `mlflow-deployment.yml`, a deployment configuration file

Copyright © 2025 Snorkel AI, Inc. All rights reserved

```
# mlflow-deployment.yml
$schema:
https://azuremlschemas.azureedge.net/latest/managedOnlineDeployment.schema.jso
name: {{DEPLOYMENT_NAME}}
endpoint_name: "{{ENDPOINT_NAME}}"
model:
  name: {{MODEL_NAME}}
  version: 1
  path: ./{{MODEL_NAME}}
environment_variables:
  MODEL_NAME: {{MODEL_NAME}}
environment:
  name: {{ENVIRONMENT_NAME}}
  image: {{ACR_NAME}}.azurecr.io/{{IMAGE_NAME}}:latest
  inference_config:
    liveness_route:
      path: /ping
      port: {{DEPLOY_PORT}}
    readiness_route:
      path: /ping
      port: {{DEPLOY_PORT}}
    scoring_route:
      path: /invocations
      port: {{DEPLOY_PORT}}
instance_type: Standard_E2s_v3
instance_count: 1
```

- `mlflow-endpoint.yml`, an endpoint configuration file

```
# mlflow-endpoint.yml
$schema:
https://azuremlschemas.azureedge.net/latest/managedOnlineEndpoint.schema.json
name: "{{ENDPOINT_NAME}}"
auth_mode: key
```

For more, see the Azure ML docs on endpoint configuration files.

Create requirements file

Snorkel Flow MLFlow models include a `conda.yaml` file that specifies the dependencies for the model. For Azure ML, we recommend not using `conda` to manage dependencies, but instead using `pip`. To do this, create a `requirements.txt` file in the root of your MLFlow model directory and copy the python packages from the `conda.yaml` file to the `requirements.txt` file.

For convenience, here's a bash command that will do this for you if run from the root of your MLFlow model directory:

Copyright © 2025 Snorkel AI, Inc. All rights reserved

```
awk '/- pip:/ {flag=1; next} /^  - / && flag {print substr($0, 5)}'
"conda.yaml" > "requirements.txt"
```

Set up local directory structure

The recommended directory structure for your MLFlow model, dockerfile, and deployment artifacts is as follows:

```
.
├── Dockerfile
├── mlflow-endpoint.yaml
├── mlflow-deployment.yaml
├── mlflow_model
│   ├── <model_files>
│   ├── requirements.txt
│   └── ...
```

The rest of the steps should be run from the root of this directory.

Build and push the Dockerfile

Once the dockerfile is configured, build and push the dockerfile to Azure Container Registry (ACR). Azure ML workspaces come with a default ACR, so you can use that. Other ACRs can be used as well, but permission will need to be configured so that the Azure ML workspace and the ACR can access each other.

The ID of the connected ACR is found in the Azure ML workspace's **Overview** page along with its other resources.

- `MODEL_NAME`: The name of the model you are deploying. It should be the same as the name of the directory containing the model files.
- `DEPLOY_PORT`: The port you want to expose on the deployment to use for inference.
- `IMAGE_NAME`: The name of the image you want to build.
- `IMAGE_TAG`: The tag of the image you want to build.
- `ACR_ID`: The ID of the connected ACR.
- `DOCKERFILE_PATH`: The path to the dockerfile you want to build.

```
az acr build --build-arg MLFLOW_MODEL_NAME=$MODEL_NAME --build-arg
DEPLOY_PORT=$DEPLOY_PORT -t $IMAGE_NAME:$IMAGE_TAG -r $ACR_ID $DOCKERFILE_PATH
```

Create an endpoint

- `RESOURCE_GROUP`: The name of the resource group for your Azure ML workspace.
- `WORKSPACE`: The name of your Azure ML workspace.

```
az ml online-endpoint create -f mlflow-endpoint.yaml --resource-group
$RESOURCE_GROUP --workspace-name $WORKSPACE
```

Copyright © 2025 Snorkel AI, Inc. All rights reserved

Deploy the model

- `ENDPOINT_NAME`: The name of the endpoint you want to deploy the model to, which should match the name in the endpoint configuration file:

```
az ml online-deployment create -f mlflow-deployment.yaml --resource-group
$RESOURCE_GROUP --workspace-name $WORKSPACE --endpoint-name $ENDPOINT_NAME
```

Inference

Once the model is deployed, you can test it by sending a request to the endpoint.

- `DEPLOYMENT_NAME`: The name of the deployment you want to test, which should match the name in the deployment configuration file.
- `PATH_TO_DATA_FILE`: The path to the data file you want to send to the endpoint.

```
az ml online-endpoint invoke -n $ENDPOINT_NAME --resource-group
$RESOURCE_GROUP --workspace-name $WORKSPACE --deployment-name $DEPLOYMENT_NAME
--request-file $PATH_TO_DATA_FILE
```

Using Azure ML Studio UI

Modify your MLflow model

Snorkel Flow leverages MLflow's `[add_libraries_to_model](https://mlflow.org/docs/latest/python_api/mlflow.models.html#mlflow.models.add_libraries_to_model)` function to package Snorkel Flow's proprietary source code as a wheel file alongside the model. This action allows the model to run outside of the Snorkel Flow platform. Azure ML natively supports the MLflow model format, but does not support this pre-packaged wheel file. Because of this, you'll first need to modify your MLflow model to be compatible to Azure ML.

To modify your MLflow model:

1. Unzip the downloaded zip file. Unzip the wheel file to the `my-model/code` folder.

   ```
   $ unzip -d my-model my-model-downloaded-from-snorkelflow.zip
   $ cd model
   $ unzip -d code wheels/snorkelflowmlflow-0.XX.Y-py3-none-any.whl
   ```

2. Open `my-model/conda.yaml` and delete this line of code: `wheels/snorkelflowmlflow-0.XX.Y-py3-none-any.whl`.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

```
channels:
- conda-forge
dependencies:
- python=3.8.10
- pip<=20.0.2
- pip:
...
  - pydantic==1.10.13
  **- wheels/snorkelflowmlflow-0.XX.Y-py3-none-any.whl         <- delete
this line**
  - llvmlite==0.41.1
  - cloudpickle==1.6.0
...
```

3. Open `my-model/MLmodel` and add this line of code: `code: code`.

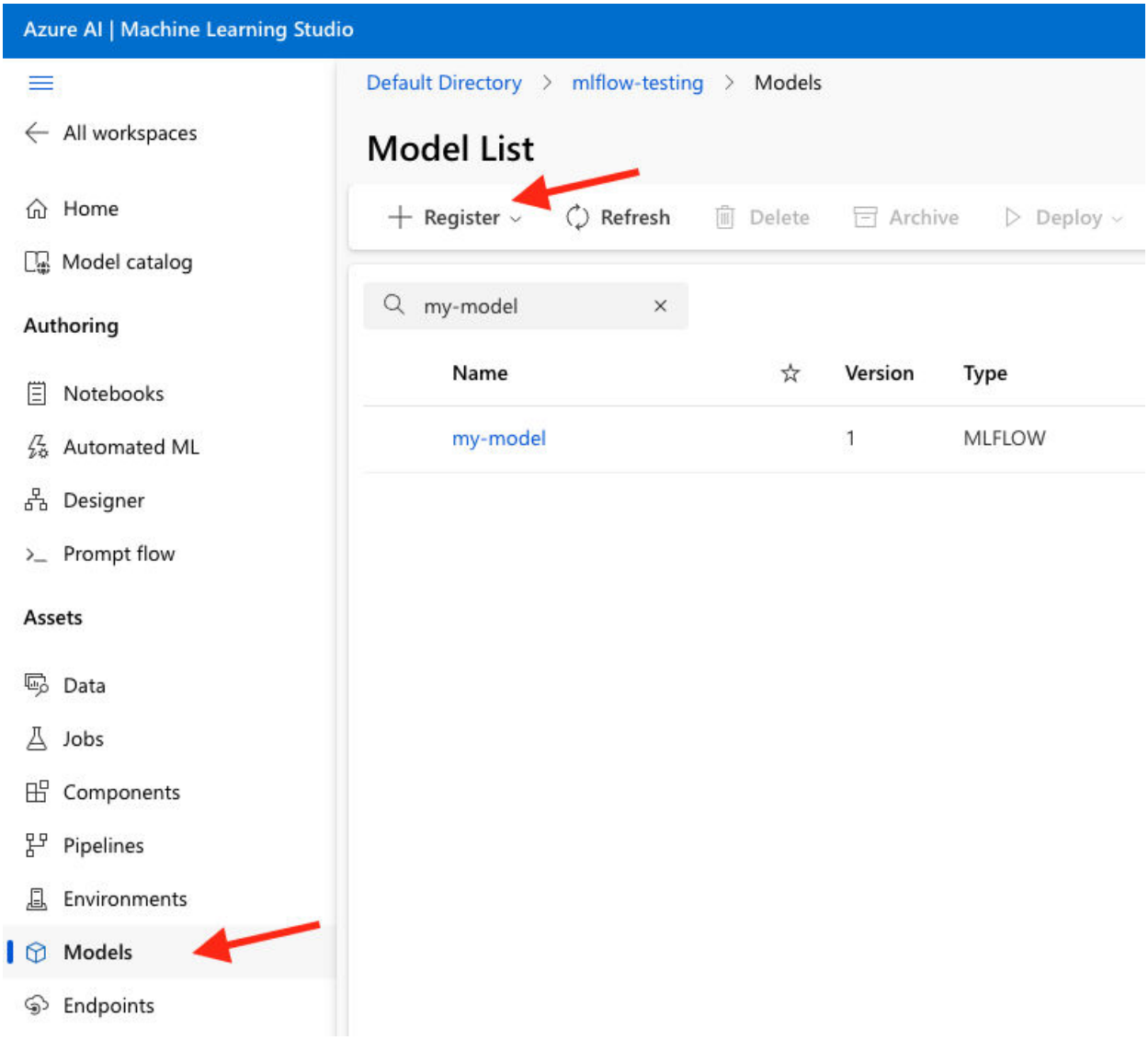
```
flavors:
  python_function:
    data: data
    **code: code                                                <- Add this
line**
    env: conda.yaml
    loader_module: application_package.mlflow_utils
mlflow_version: 2.10.2
model_uuid: 7bf8f4cb4a7e4a5e998a10f3c92ea193
...
```

Upload the model to Azure ML

Once you've made the necessary modifications to your MLflow model, you can upload it to Azure ML!

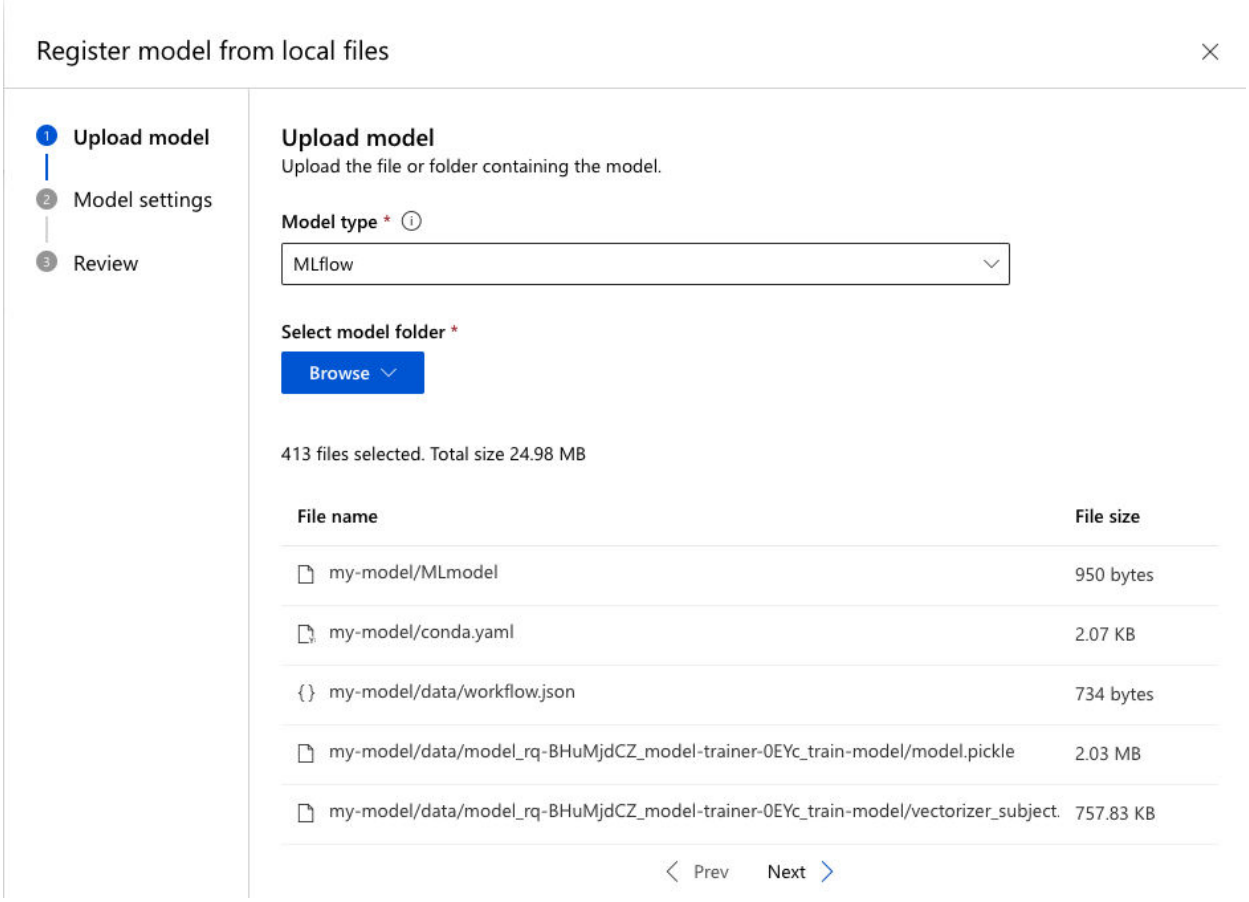
1. Using an Azure Machine learning studio session, select **Models**, and then select **Register**.

Copyright © 2025 Snorkel AI, Inc. All rights reserved



1. Adjust the following settings, then register the model to Azure ML:
2. Select **MLflow** as Model type.
3. Select the **my-model** folder.
4. Use the defaults for all other settings.

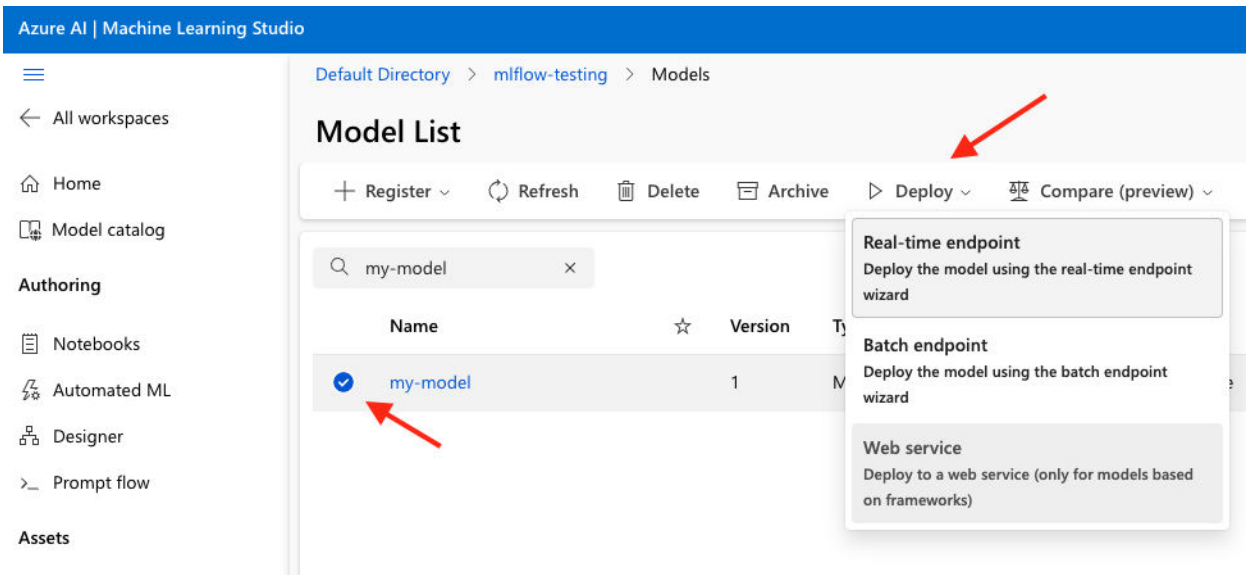
Copyright © 2025 Snorkel AI, Inc. All rights reserved



Create an endpoint and deploy the model

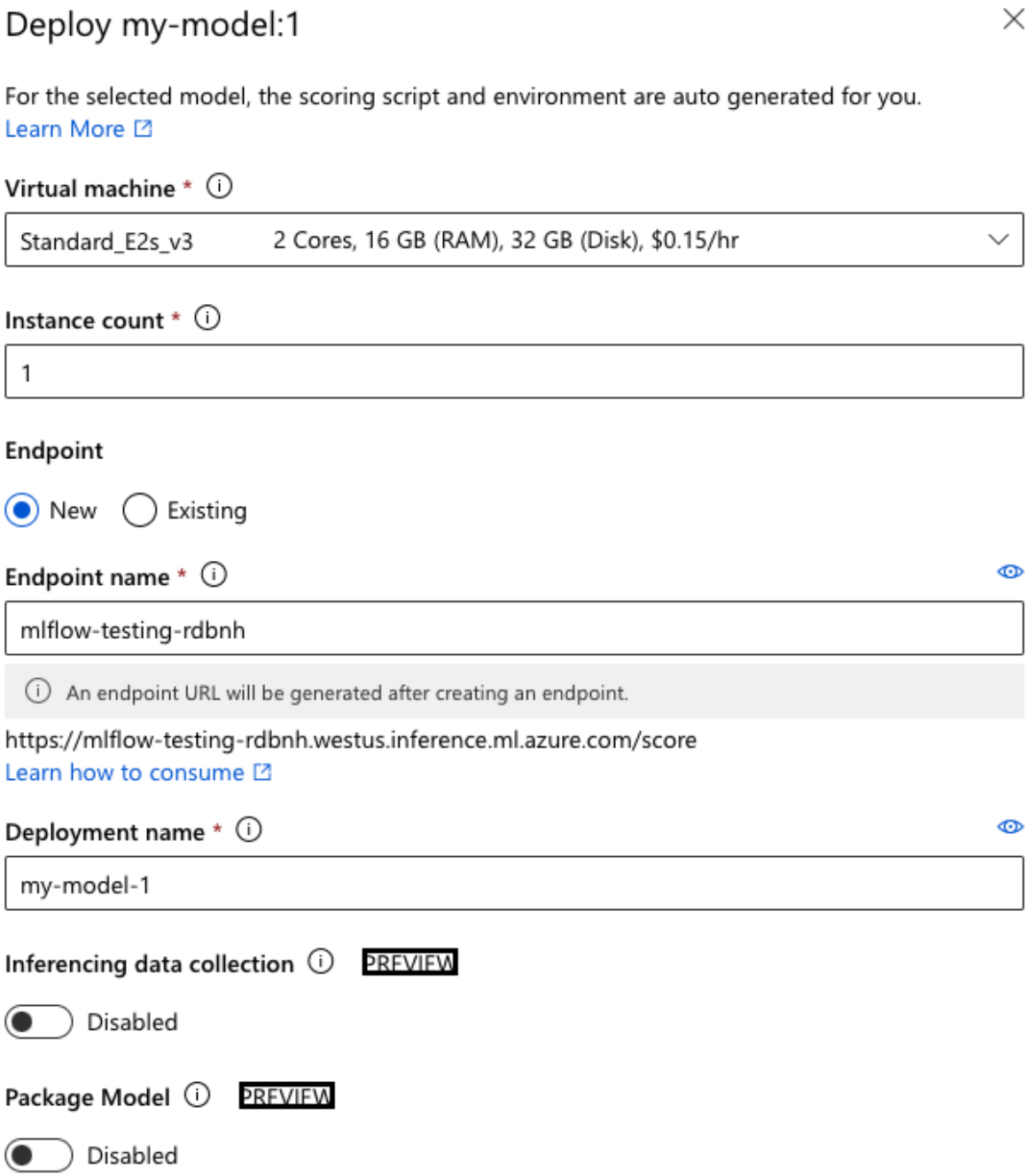
Once your model is registered to Azure ML, you can create an endpoint to deploy your model to.

1. Select the model name in the model list, select **Deploy**, and then select **Real-time endpoint**.



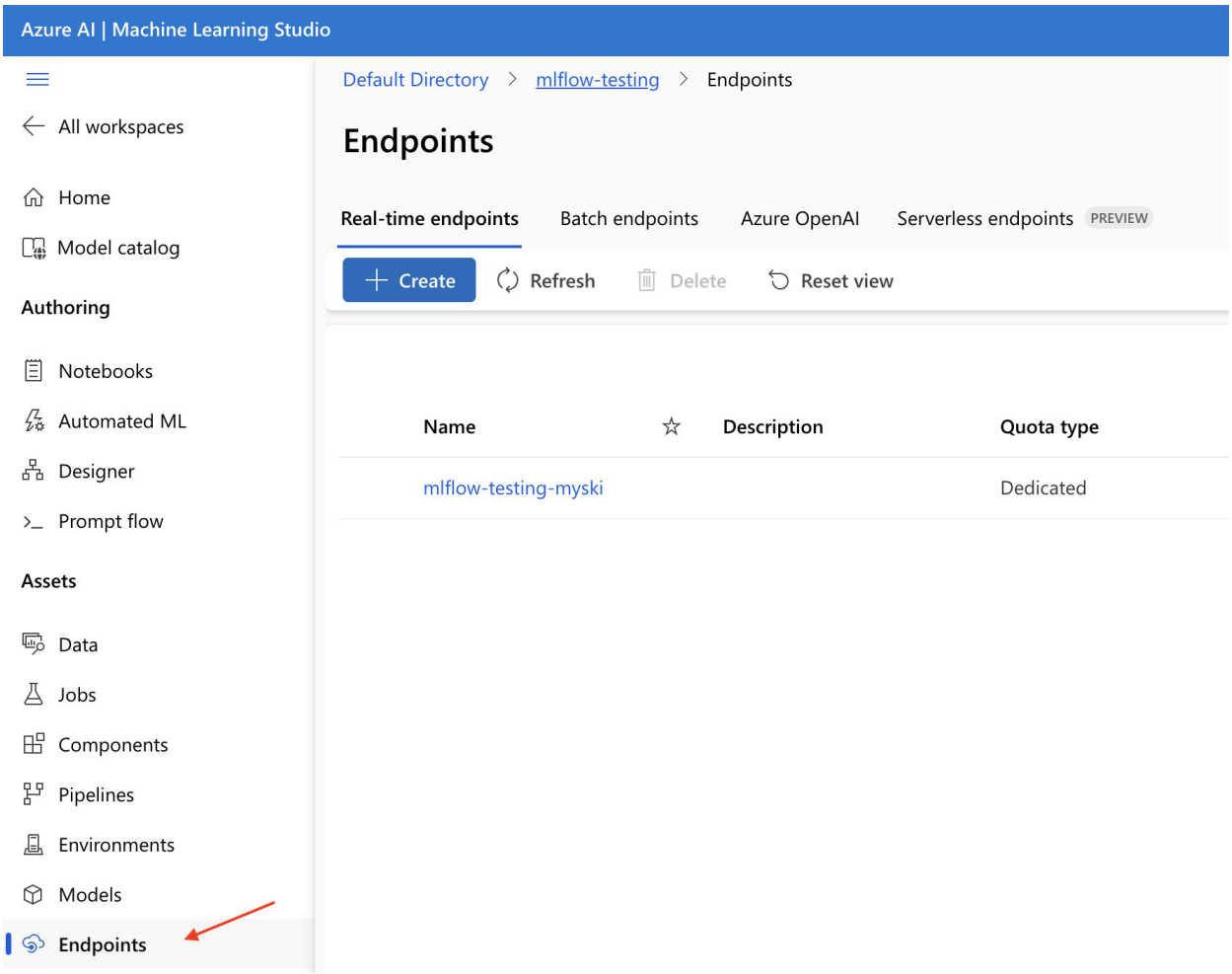
1. Choose a virtual machine with enough memory, then select **Deploy**. If you are having issues at this stage, see Azure's documentation for troubleshooting online endpoint deployments for more information.

Copyright © 2025 Snorkel AI, Inc. All rights reserved



1. Select **Endpoints** in the left-side menu to see the created endpoint.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

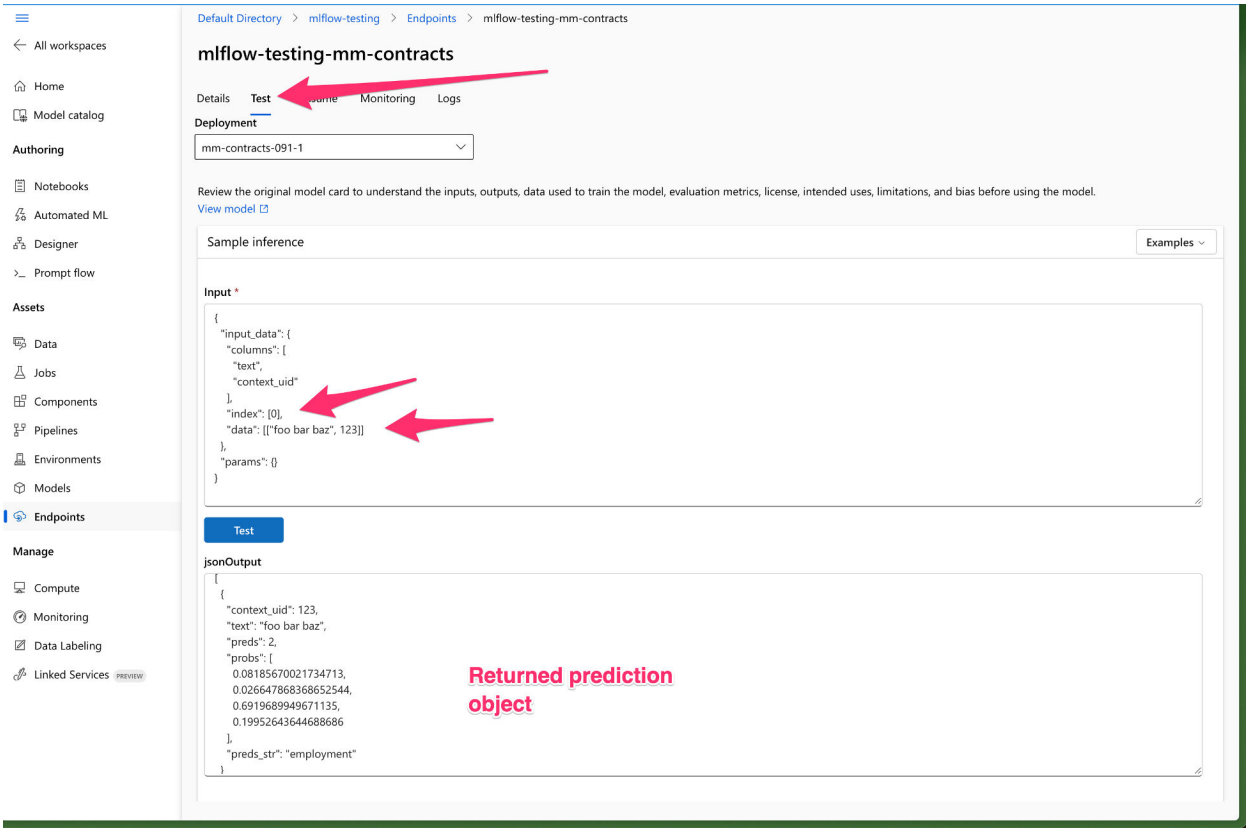


Once the endpoint boots up, you can start testing and running it!

Testing deployed endpoints

From the **Endpoints** home page, select **Test**. Use an example record to ensure the returned prediction is returned as expected. Use the **Logs** section to debug any errors that arise while performing inference.

Copyright © 2025 Snorkel AI, Inc. All rights reserved



Conclusion

This article has demonstrated how to export a Snorkel-built model from Snorkel Flow, onboard it to Azure ML, create a deployment endpoint and validate the endpoint's results. If you encounter issues during this process, please reach out to support@snorkel.ai.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

Deploying Snorkel-built models to the Databricks Workspace Model Registry and Unity Catalog

Snorkel Flow supports integrations for Databricks Workspace Model Registry and Unity Catalog, which is a unified governance solution for managing data and AI assets in the Databricks platform.

This tutorial demonstrates how to connect Snorkel-built models to the Databricks platform in three steps:

1. Connect to your organization's Databricks Model Registry.
2. Push the model object from Snorkel to the Model Registry and optional Unity Catalog.
3. Deploy the model in Databricks.

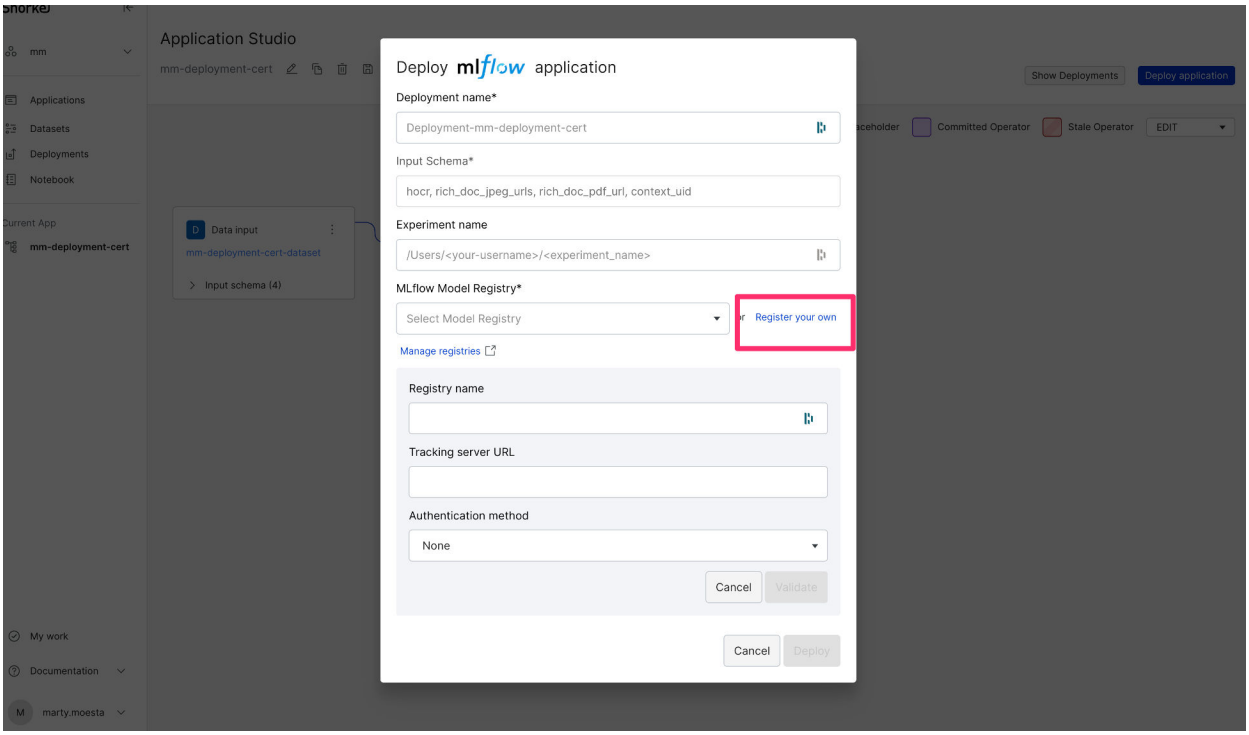
Prerequisites

Ensure you have the `CREATE_MODEL` privilege for the catalog or schema in Databricks. For more about registry and catalog permissions, see Unity Catalog privileges and securable objects.

Connect to your Databricks Model Registry

Using the Deploying Snorkel-built models tutorial, bundle the Snorkel workflow for deployment.

Instead of selecting Snorkel's default model registry, choose the **Register your own** option:



- **Registry name:** The name of the project. This value shows up as the *Name* in the *Registered Models Section* in Databricks' Model Registry.
- **Tracking server URL:** Your organization's base Databricks URL, For example, `https://dbc-8db1117a-9cc7.cloud.databricks.com`.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

- **Authentication Method:** Username/password or authentication token. Snorkel recommends using authentication tokens to comply with security best practices. For more information, see Authentication for Databricks automation - overview.

Push the model object from Snorkel to the Databricks Unity Catalog

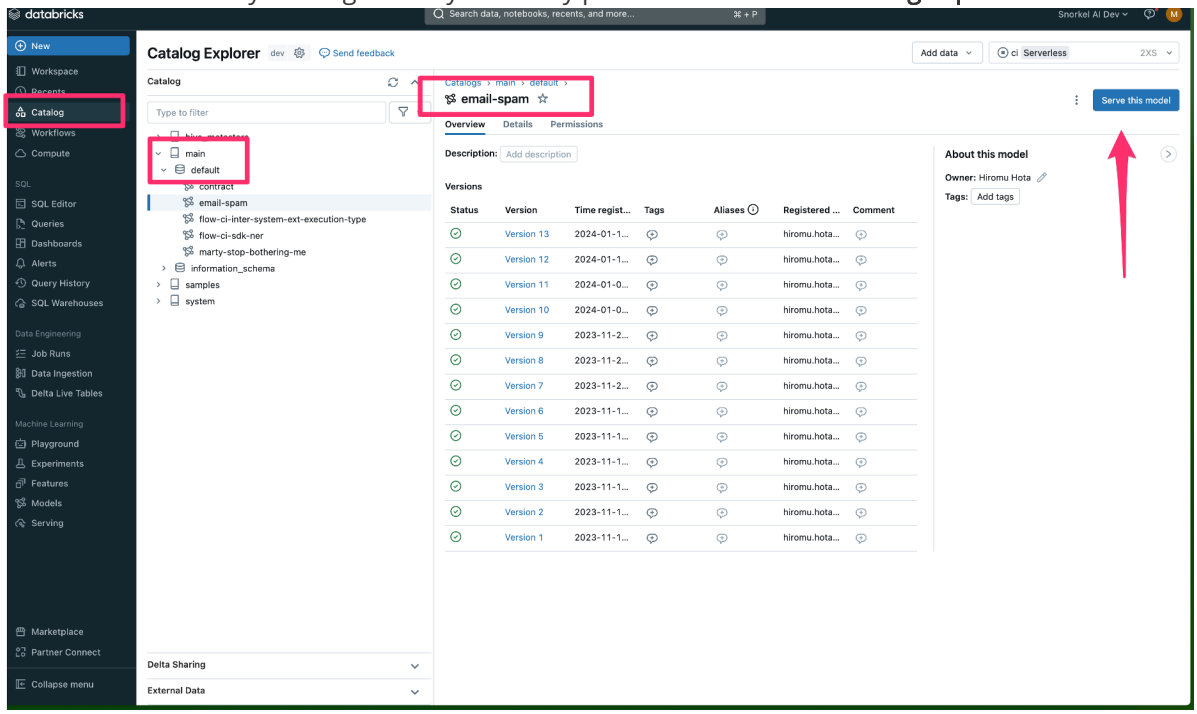
1. Enter in the relevant fields, and then **Validate** your credentials and connectivity to your new registry.

> NOTE
>
> To register the asset with the Databricks Unity Catalog, the **Deployment Name** must be in the form of `<catalog>.<schema>.<model>`.

2. If you'd like to track each model as an **Experiment** in Databricks, ensure the **Experiment Name** follows Databricks naming convention pattern as an absolute path. For example, `/Users/<username>/<experiment_name>`.
3. After configuring the required fields, select **Deploy** to push the model asset to Databricks Model Registry and Databricks Unity Catalog.

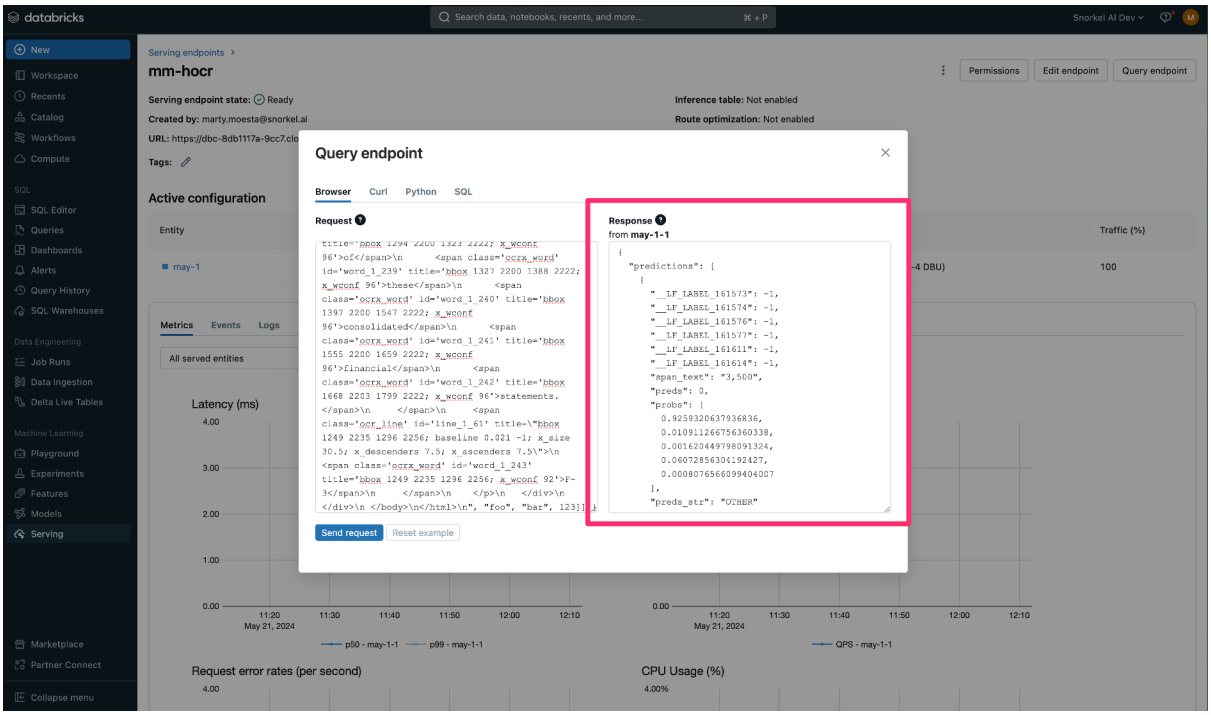
Deploy the model in Databricks

1. After successfully pushing the model to Databricks, login to the Databricks portal.
2. Connected to the Unity Catalog to find your newly pushed model in the **Catalog Explorer.**



3. Select **Serve this model** to deploy the model for testing or production.
4. Configure the remaining model serving settings for your use case.
5. After creating a serving endpoint, test the endpoint to confirm a valid model response.

Copyright © 2025 Snorkel AI, Inc. All rights reserved



Conclusion

In this tutorial, you connected to the Databricks Model Registries and the Databricks Unity Catalog, pushed Snorkel-built models to Databricks, and deployed them for production workloads.

If you have any questions about this workflow, contact the Snorkel support team.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

Deploying Snorkel-built models to Vertex AI

This tutorial walks through the four steps that are required to deploy a Snorkel-built application to Vertex AI:

1. Upload your image to Artifact Registry.
2. Upload your image to Vertex AI's Model Registry.
3. Create an endpoint.
4. Deploy the model to the endpoint.

Requirements

Ensure your environment meets all the requirements for deploying a Snorkel-built model.

- Google Cloud CLI
- Vertex AI SDK for Python
- A Google Cloud account with privileges of accessing:
  - Artifact Registry
  - Vertex AI
- A containerized MLflow model

If you need to containerize your MLflow model, see Deploying Snorkel-built models for instructions on how to do so. You also need the value of the **Project ID** field from the GCP Console home screen.

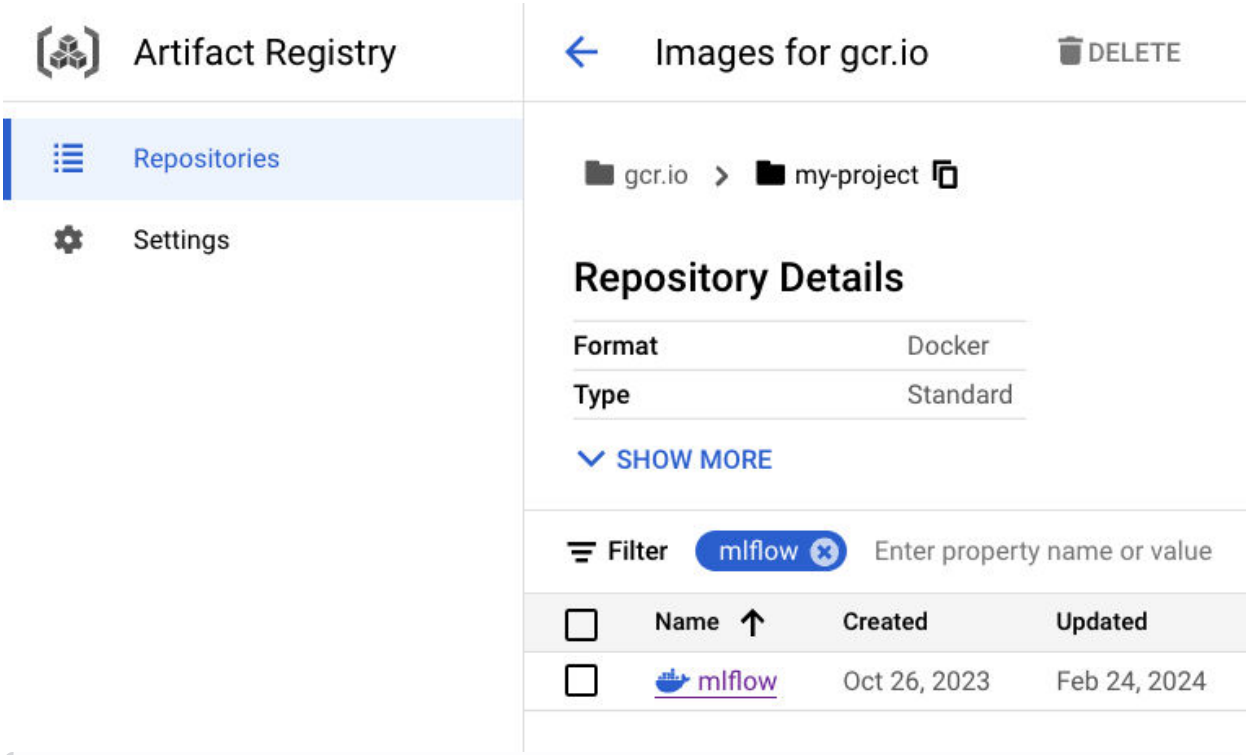
Upload your image to Artifact Registry

You can use the following code to upload your image to Artifact Registry. This example assumes that your containerized image is tagged as **MY_IMAGE_NAME**. See the documentation for more details about Artifact Registry.

```
$ gcloud auth configure-docker gcr.io
$ docker tag <MY_IMAGE_NAME> gcr.io/<GCP_PROJECT_ID>/<MY_IMAGE_NAME>
$ docker push gcr.io/<GCP_PROJECT_ID>/<MY_IMAGE_NAME>
```

You'll then be able to view your uploaded image on Google Cloud console in your browser.

Copyright © 2025 Snorkel AI, Inc. All rights reserved



> NOTE
> We are using Artifact Registry because Container Registry is deprecated.

Upload your image to Vertex AI's Model Registry

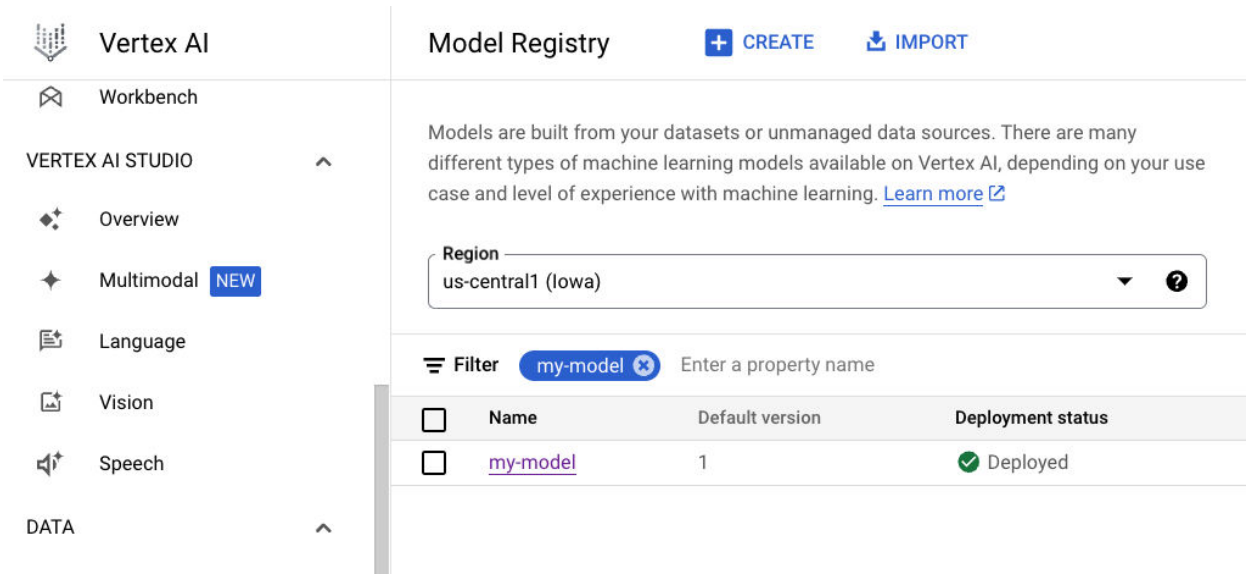
Now you can upload your image to Vertex AI's Model Registry and check its ID.

```
$ gcloud ai models upload \
  --container-image-uri="gcr.io/<GCP_PROJECT_ID>/<MY_IMAGE_NAME>" \
  --description=<DESCRIPTION> \
  --display-name=<MODEL_NAME> \
  --project=<GCP_PROJECT_ID> \
  --container-env-vars=PREDICTIONS_WRAPPER_ATTR_NAME=predictions \
  --container-ports=8080 \
  --container-predict-route=/invocations \
  --container-health-route=/ping

# check the model ID
$ gcloud ai models list --filter DISPLAY_NAME=<MODEL_NAME>
```

You'll then be able to view your uploaded image on Google Cloud console in your browser.

Copyright © 2025 Snorkel AI, Inc. All rights reserved



Create an endpoint

After you've uploaded your image to Vertex AI's Model Registry, you can create an endpoint. Make note of the ID of the endpoint.

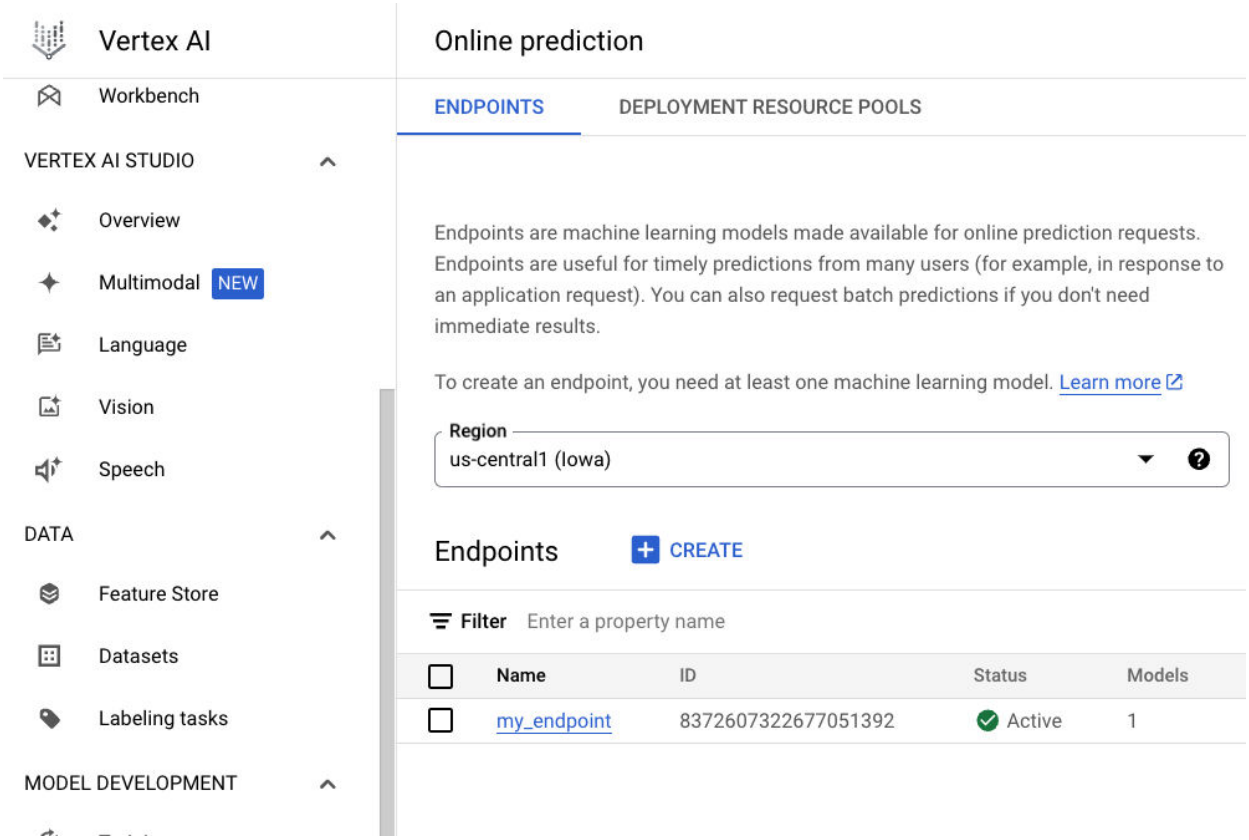
```
$ gcloud ai endpoints create \
  --project=<GCP_PROJECT_ID> \
  --display-name=<ENDPOINT_NAME>
```

Deploy your model to the endpoint

Once you have created an endpoint, you can deploy your model to the endpoint. Use the Endpoint ID returned by the endpoint create command:

```
$ gcloud ai endpoints deploy-model <ENDPOINT_ID> \
  --project=<GCP_PROJECT_ID> \
  --model=<MODEL_ID> \
  --display-name=<DEPLOYMENT_NAME>
```

Copyright © 2025 Snorkel AI, Inc. All rights reserved



Send an inference request

Once the model successfully is deployed to the endpoint, you can start sending an inference request.

Format the input JSON as described in TF Serving's API docs. Here's an example of this format for a basic classification model:

```
{
  "instances": [
    {
      "is_spam": true,
      "txt": "A royal prince needs your help! Please Venmo: 123-456-7890",
      "context_uid": 1
    },
    {
      "is_spam": false,
      "txt": "The meeting is scheduled for 10 am tomorrow",
      "context_uid": 4
    }
  ]
}
```

Here's an example CURL request to send an inference request to the endpoint:

Copyright © 2025 Snorkel AI, Inc. All rights reserved

```
$ curl -X POST -H "Authorization: Bearer $(gcloud auth print-access-token)" \
  -H "Content-Type: application/json" \
  -d @<PATH_TO_INPUT_JSON_FILE> \
  https://<REGION>-
aiplatform.googleapis.com/v1/projects/<GCP_PROJECT_ID>/locations/<REGION>/endpoi
```

Copyright © 2025 Snorkel AI, Inc. All rights reserved

MLflowModel format: MLflow-compatible application deployments

Application deployments (see Deploying Snorkel-built models) can be packaged in the `MLflowModel` format. The `MLflowModel` package format is compatible with the standard model format defined by the MLflow project. Deployments packaged in this format will be compatible with all systems that use MLflow models.

Deployment structure

Deployments of the `MLflowModel` format will be of the `pyfunc` model flavor (see MLflowPyfunc). As of v0.84.0, these deployments will have the following directory structure:

- `workflow_<ID>`
  - `MLmodel`: The model metadata `yaml` file.
  - `conda.yml`: The conda environment file that describes the packages that are required to run the model.
  - `data/`: The directory that contains a description of the application DAG and any required model data (e.g. weights).
  - `wheels/snorkelflowmlflow-XXX-py3-none-any.whl`: A wheel file that contains the Snorkel proprietary source code that is needed to run the model.

Python environment and dependencies

`MLflowModel` deployments are only compatible with `Python 3.8`.

In order to run an application deployment, the packages listed in `conda.yml` need to be installed. Depending on the application template, operator configuration, model architecture, etc., additional packages are also needed. Please contact your Snorkel representative for more details.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

Python SDK

The Snorkel Flow Python SDK lets you drive the development process programmatically and gives you access to advanced customization features.

Snorkel Flow's Python SDK comes in two different distributions: the full SDK and the basic SDK.

- The full SDK is pre-installed in the in-platform Notebook server.
- The basic SDK, a subset of the full SDK, can be installed on any Python environment. Installation instructions can be found in Installing the SDK.
- SDK usage from outside of the Snorkel Flow platform requires authentication via an API key. Instructions for authenticating your SDK client can be found in Authentication.

> NOTE
>
> The full SDK includes extra modules and functionality, such as custom operator registration.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

SDK quickstart

This guide shows you how to access or install Snorkel's SDK, and then use it to train a simple document classification model, mirroring the Document classification GUI tutorial.

- Install the SDK locally
- Access the SDK from a Snorkel instance

Install the SDK locally

Requirements

- Python version 3.9-3.11
- A Snorkel account with admin or developer access
- Your Snorkel API key
- Your Snorkel instance host name or IP address. You can copy the host name from your Snorkel instance's URL.

Install the SDK

Run the following command to install the basic Snorkel SDK in your Python environment:

```
pip install "snorkelflow[sdk]" "aiobotocore==2.17.0" \
--extra-index-url https://{SNORKELFLOW_API_KEY}@{your-snorkel-
hostname}/sdk/repo
```

> NOTE
>
> Special characters in the API key, such as /, have to be percent-encoded (see this explanation for details).

Use the code snippet below to percent-encode your API key.

```
import urllib.parse
print(urllib.parse.quote("your-api-key", safe=""))
```

Known issues

- `aiobotocore==2.17.0` must be included explicitly in the installation

Copyright © 2025 Snorkel AI, Inc. All rights reserved

Access the SDK from a Snorkel instance

Requirements

- A Snorkel account with admin or developer access
- Your Snorkel API key

You can call the Snorkel SDK from a Jupyter notebook hosted on your Snorkel instance.

1. Select **Notebook** from the left navigation.
2. Select **+** to create a new notebook from scratch, or upload one.
3. Select **Python 3 (ipykernel)** from the **Notebook** section. A new Jupyter notebook opens.
4. Authenticate with Snorkel. To create a new API key, follow the steps for generating an API key.

```
import snorkelflow.client as sf

# Use your Admin or Developer Snorkel API key
api_key = 'abcdefg'  #INPUT - Replace with your API key

# Set your workspace
workspace_name = 'YOUR_WORKSPACE_NAME' #INPUT - Replace with your workspace
name

# Authenticate with Snorkel Flow
ctx = sf.SnorkelFlowContext.from_kwargs(
    api_key=api_key,
    workspace_name=workspace_name,
)
```

Now you can use Snorkel's SDK.

Authentication

All API requests to Snorkel must be authenticated. The Python SDK client needs to use an authenticated `SnorkeflFlowContext` object to make API requests to Snorkel services.

You will need an API key to authenticate.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

Generate an API Key

Follow these instructions to generate an API key:

- User and Admin settings

Authenticate from an external system

When you connect to Snorkel locally or from another external system, you must provide additional settings and authentication secrets. Use the following connection template:

```
import os
import snorkelflow.client as sf

# Snorkel Flow SDK configuration
SF_CONFIG = {
    "endpoint": "https://<your-snorkel-hostname>",
    "minio_endpoint": "https://<your-minio-endpoint>",
    "api_key": "<your-api-key>",
    "minio_access_key": "<your-minio-access-key>",
    "minio_secret_key": "<your-minio-secret-key>",
    "workspace_name": "<your-workspace-name>",
    "debug": True  # Optional: set to False to disable debug logging
}

# Set MinIO-related environment variables
os.environ["MINIO_URL"] = "https://<your-minio-api-url>"
os.environ["MINIO_ACCESS_KEY"] = "<your-minio-access-key>"
os.environ["MINIO_SECRET_KEY"] = "<your-minio-secret-key>"

# Initialize Snorkel Flow context
ctx = sf.SnorkelFlowContext.from_endpoint_url(**SF_CONFIG)
```

- `endpoint`: You can copy the host name from your Snorkel instance's URL.
- `api_key`: Your Snorkel API key
- Contact your Snorkel admin or support representative to obtain your `minio_endpoint`, `minio_access_key` and `minio_secret_key`.

Authenticate from a Snorkel-hosted instance

When you call the SDK from a Snorkel instance (through a Jupyter notebook), you can provide your API key from an environment variable, or explicitly.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

If you set your API key as the `SNORKELFLOW_API_KEY` environment variable, the SDK will find and use this API key automatically.

`bash $ export SNORKELFLOW_API_KEY="abcdefg"`

You can also pass an API key through the `api_key` argument while constructing the `SnorkelFlowContext` object.

```
import snorkelflow.client as sf

# Use your Admin or Developer Snorkel API key
api_key = 'abcdefg'  #INPUT - Replace with your API key

# Set your workspace
workspace_name = 'YOUR_WORKSPACE_NAME' #INPUT - Replace with your workspace
name

# Authenticate with Snorkel Flow
ctx = sf.SnorkelFlowContext.from_kwargs(
    api_key=api_key,
    workspace_name=workspace_name,
)
```

Quick Start: End-to-End Document Classification

This quick start guide will walk you through a full end-to-end example of using Snorkel Flow to build a document classification model. In practice, your Snorkel Flow journey will involve heavy interplay between the UI and the SDK, but for the purposes of this guide, we'll stick to the SDK.

In our task, we will be classifying 20,000 contracts into one of the four following categories:

- **Loan:** A loan agreement
- **Services:** A services agreement
- **Stock:** A stock purchase agreement
- **Employment:** An employment agreement

The majority of these contracts will not have any labels for us to train a fully supervised model. We will use the Snorkel Flow SDK to ingest our dataset, write code-based labeling functions, train a model, and deploy that model for external use.

Begin by importing the necessary packages

Copyright © 2025 Snorkel AI, Inc. All rights reserved

```
import pandas as pd
import snorkelflow.client as sf
from snorkelflow.models.model_configs import
SKLEARN_LOGISTIC_REGRESSION_CONFIG
from snorkelflow.sdk import Dataset, MLflowDeployment
from snorkel.labeling.lf import labeling_function

# Connect to Snorkel. Uncomment either the hosted or external option
# And replace with the full connection block including API key
# ctx = sf.SnorkelFlowContext.from_endpoint_url( ... ) # External connection
# ctx = sf.SnorkelFlowContext.from_kwargs( ... ) # Snorkel-hosted connection
```

For the connection, uncomment the `ctx` line and replace it with the full authentication block from the appropriate section above.

Creating a Dataset

We will use the `snorkelflow.sdk` module to create a Dataset. Snorkel Flow Datasets are the main mechanism for ingesting and interacting with source data. Upstream of Snorkel, we have already split up our data into several **data splits** (e.g. "train", "valid", "test") for our model development pipeline. Our data contains some null/missing values, which might cause some issues in our modeling pipeline. Our data upload will then consist of imputing our null values, then uploading the files to Snorkel Flow.

```
contracts_dataset = Dataset.create("contracts-clf-dataset")
# Upload the train splits
train_df = pd.read_parquet("s3://snorkel-contracts-dataset/train.parquet")
train_df = train_df.fillna("")
contracts_dataset.create_datasource(train_df, uid_col="uid", split="train")

dev_df = pd.read_parquet("s3://snorkel-contracts-dataset/dev.parquet")
dev_df = dev_df.fillna("")
contracts_dataset.create_datasource(dev_df, uid_col="uid", split="train")

valid_df = pd.read_parquet("s3://snorkel-contracts-dataset/valid.parquet")
valid_df = valid_df.fillna("")
contracts_dataset.create_datasource(valid_df, uid_col="uid", split="valid")

test_df = pd.read_parquet("s3://snorkel-contracts-dataset/test.parquet")
test_df = test_df.fillna("")
contracts_dataset.create_datasource(test_df, uid_col="uid", split="test")
```

You'll notice that the second file we uploaded ends in `dev.parquet`. In the Snorkel workflow, it's useful to have some very small selection of manually-labeled data in your `train` split to

Copyright © 2025 Snorkel AI, Inc. All rights reserved

help you write labeling functions. You will sometimes see this called the "dev split". If we so desire, we can view the data in our dataset by running `contracts_dataset.get_dataframe(...)`.

Creating an Application

Now that we have our data uploaded, we can create an application. Applications string together data processing logic, programmatic labeling, and model training into a single interface that can later be deployed end-to-end. To create an application, we will use the Snorkel Flow client `snorkelflow.client` that we imported earlier.

```
application_uid = sf.create_classification_application(
    "contracts-clf",
    dataset="contracts-clf-dataset",
    input_schema=["text"],
    labels = ["services", "stock", "loan", "employment"],
    label_col = "label"
)["application_uid"]
```

An application is set up as a "directed acyclic graph" (DAG), composed of a series of data transformations. This means that data moves from one transformation to the next, and is modified by one transformation at a time, but the data never flows backwards or loops.

The individual elements that make up the DAG are called nodes. Some nodes are Operators, transformations that manipulate the data by changing its rows and columns. Other nodes are **Model Nodes**, which are environments for iterating on training set development and model training. Most of the core programmatic labeling loop will happen in these "Model nodes". To interact with a model node through the SDK, we will need its UID. Additionally, we need to signal to our application that we are done with set up and want to start developing our training set. To do this, we need to activate the node with the `sf.add_active_datasources` function.

```
model_node_uid = sf.get_model_node(application_uid)
activate_datasources_job = sf.add_active_datasources(model_node_uid,
sync=True)
```

We will use this model node UID to write labeling functions and train our models.

Writing Labeling Functions

The best way to view, filter, slice, and comment on your data is through the Snorkel Flow UI. In this way, the UI and SDK have highly complementary roles in the Snorkel Flow workflow. The

Copyright © 2025 Snorkel AI, Inc. All rights reserved

SDK can be used to write custom, high-quality labeling functions to programmatically label unlabeled data. Here, we will create 4 labeling functions to label each of our 4 classes. In practice, you would come up with these labeling functions based on your interactions with the data in the UI.

To view how performant our labeling functions are, we can run `sf.get_lf_summary`. This will return a Pandas DataFrame with evaluation metrics for each labeling function.

```
@labeling_function(name="employment_lf")
def lf_emp(x):
    if "employment" in x.text:
        return "employment"
    return "UNKNOWN"
sf.add_code_lf(model_node_uid, lf_emp, label="employment")

@labeling_function(name="stock_lf")
def lf_stock(x):
    if "STOCK PURCHASE AGREEMENT" in x.text:
        return "stock"
    return "UNKNOWN"
sf.add_code_lf(model_node_uid, lf_stock, label="stock")

@labeling_function(name="services_lf")
def lf_services(x):
    import re
    if re.search(r"This.{1,50}Service Agreement", x.text, re.IGNORECASE):
        return "services"
    return "UNKNOWN"
sf.add_code_lf(model_node_uid, lf_services, label="services")

@labeling_function(name="loan_lf")
def lf_loan(x):
    import re
    if re.search(r"loan agreement", x.text, re.IGNORECASE):
        return "loan"
    return "UNKNOWN"
sf.add_code_lf(model_node_uid, lf_loan, label="loan")

sf.get_lf_summary(model_node_uid)
```

Code LFs can incorporate external libraries, algorithms, external databases, and more – they are extremely flexible tools for programmatically labeling data. Snorkel users have used Code LFs to create rich embeddings-based labeling functions, built LFs on top of external deep learning models, and more.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

Training a Model

Once we have our labeling functions, we can begin to create a training set and train a model. There are three parts to this process – we aggregate our LFs as an **LF Package**, use the Snorkel Flow *label model* to aggregate LF votes into a **training set**, and then train a machine learning classifier on this training set. After that, we will view model metrics to make sure our model is performing how we'd expect. We'll check out our metrics against a previously held-out validation split to see if we need to do any more tuning.

```
sf.package_lfs(model_node_uid)
sf.add_training_set(model_node_uid)
model_uid = sf.train_model(
    model_node_uid,
    feature_fields=["text"],
    model_description="My first model",
    model_config = SKLEARN_LOGISTIC_REGRESSION_CONFIG.dict()
)['model_uid']

sf.get_model_metrics(model_node_uid, model_uid, split="valid")
```

In a real-world setting, you would iterate on this process until you are satisfied with your model's performance. You can use Snorkel Flow's in-UI built-in error analysis tools to find slices of data that need more coverage or need more accurate labeling functions. You can also create your own custom error analysis tools, and use them to iterate on your LFs and improve your models!

Deployment

Once you are satisfied with your score, you can deploy your model as an MLFlow package to use it in your external data pipelines. In this example, we'll create an MLFlow deployment and then run our test dataset through it before downloading it for use elsewhere in our data pipeline.

```
sf.commit_model_to_node(model_node_uid, model_uid)
deployment = MLflowDeployment.create("contracts-clf", name="contracts-clf-
deployment")
test_df = contracts_dataset.get_dataframe(split="test", max_rows=5)
results = deployment.execute(df=test_df)
# Examine the "results" DataFrame to make sure it looks good
deployment.download('./deployment-path/')
```

That's it! In this tutorial, we have uploaded a messy dataset, cleaned it up, ingested it into Snorkel, written labeling functions, viewed LF stats, created a programmatically labeled training

Copyright © 2025 Snorkel AI, Inc. All rights reserved

set, trained a model, and deployed it for external use. This is just a snippet of what is possible with Snorkel Flow and the SDK – the SDK is a powerful tool for interacting with the core Snorkel workflow programmatically, and can extend and empower you to create more robust and flexible data pipelines.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

Example notebooks

The in-platform Notebook server, accessible via the Notebook button in the sidebar of the web UI, has a number of example notebooks under the SampleNotebooks directory in the File Browser.

Create a new dataset and add datasources

The following snippet is an example workflow using `snorkelflow.client` to create a new dataset and add datasources for different splits.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

```
import os
import snorkelflow.client as sf

# Configure client context for Snorkel Flow instance
ctx = sf.SnorkelFlowContext.from_kwargs()

# Create a dataset called ``contracts``
DATASET_NAME = "contracts"
sf.create_dataset(DATASET_NAME)

# Add a datasource for the train, valid, and test splits which have been
uploaded to
# MinIO in the contracts_data bucket as Parquet files with UID columns called
uid
sf.create_datasource(
    dataset=DATASET_NAME,
    path="s3://snorkel-contracts-dataset/train.parquet",
    file_type="parquet",
    uid_col="uid",
    split="train",
)
sf.create_datasource(
    dataset=DATASET_NAME,
    path="s3://snorkel-contracts-dataset/dev.parquet",
    file_type="parquet",
    uid_col="uid",
    split="train",
)
sf.create_datasource(
    dataset=DATASET_NAME,
    path="s3://snorkel-contracts-dataset/valid.parquet",
    file_type="parquet",
    uid_col="uid",
    split="valid",
)
sf.create_datasource(
    dataset=DATASET_NAME,
    path="s3://snorkel-contracts-dataset/test.parquet",
    file_type="parquet",
    uid_col="uid",
    split="test",
)
```

Copyright © 2025 Snorkel AI, Inc. All rights reserved

Create an application

Now that we have a dataset with datasources, let's create an application. Here we create a text extraction application to extract dates from the "text" field of the data.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

```
import snorkelflow.client as sf
APP_NAME = "date_extraction"

# Create a text extraction application.
# This will create an application with a block of operators suited for text
extraction.
app = sf.create_text_extraction_application(
    application_name=APP_NAME,
    dataset=DATASET_NAME,
    input_schema=["url", "text"],
    extraction_field="text",  # specify the field from which spans will be
extracted.
    labels=["label1", "label2"],
)

# Get the UID of the node "SpanExtractor" in the DAG.
extractor_node = sf.get_node_uid(APP_NAME, search_op_type="SpanExtractor")[0]

# Commit "DateSpanExtractor" to the node
sf.commit_builtin_operator(
    node=extractor_node,
    op_type="DateSpanExtractor",
    op_config={"field": "text"}
)

# Get the UID of the node "SpanReducer" in the DAG.
reducer_node = sf.get_node_uid(APP_NAME, search_op_type="SpanReducer")[0]

# Commit "IdentityReducer" to the node
sf.commit_builtin_operator(
    node=reducer_node,
    op_type="IdentityReducer",
    op_config={}
)

# Optionally, add "DateSpanNormalizer" after "DateSpanExtractor" node.
preview_node = sf.get_node_uid(APP_NAME,
search_op_type="SpanPreviewPreprocessor")[0]

sf.add_node(
    application=APP_NAME,
    input_node_uids=[extractor_node],
    output_node_uid=preview_node,
    expected_op_type="Featurizer",
    op_type="DateSpanNormalizer",
)
```

Alternatively, you can create the same application using lower-level methods.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

```
import snorkelflow.client as sf
# Create an application without any block.
sf.create_application(
    application_name=APP_NAME,
    dataset=DATASET_NAME,
    input_schema=["url", "text"],
)

# Add a block of operators using the application template "te" (Text
Extraction).
from snorkelflow.types.workflows import INPUT_NODE_ID
sf.add_block_to_application(
    application=APP_NAME,
    template_id="te",
    input_node=INPUT_NODE_ID,
    block_config=dict(
        label_map=dict(UNKNOWN=-1, NEGATIVE=0, POSITIVE=1),
        extraction_field="text",  # specify the field from which spans will be
extracted.
    ),
)

# Get the application info
app = sf.get_application(APP_NAME)

# Run the same commands as above to commit "DateSpanExtractor" and
"IdentityReducer"
```

Train a custom model using a Snorkel Flow training dataset

The following snippet is an example workflow to integrate a custom machine learning pipeline with the Snorkel Flow platform. Using the application in Quick Start: End-to-End Document Classification as an example, we will first get a training dataset created in Snorkel Flow, then train an ML model, and finally upload the predictions to Snorkel Flow in order to use the integrated analysis tools. Here, we use LogisticRegression and CountVectorizer from scikit-learn as an example, but you can use any model and featurizer of your choice.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

```
from sklearn.linear_model import LogisticRegression
from sklearn.feature_extraction.text import CountVectorizer

APP_NAME = "YOUR_APP_NAME" # Replace with your application name

feature_column = "text"
training_set_uid = 1
application_uid = sf.get_application_uid(name=APP_NAME)
node = sf.get_model_node(application_uid)

# Get the dataframe of each split and set aside the training set labels of the
train split
df_train = sf.get_node_data(
    node=node,
    split="train",
    training_set_labels=True,
    training_set_uid=training_set_uid,
    training_set_filter_unlabeled=True,
)
df_dev = sf.get_node_data(node=node, split="dev")
df_valid = sf.get_node_data(node=node, split="valid")
df_test = sf.get_node_data(node=node, split="test")
y_train = df_train.pop("training_set_labels")

# Train a featurizer on the feature column
vectorizer = CountVectorizer()
X_train = vectorizer.fit_transform(df_train[feature_column])

# Train a custom model on the training dataset
model = LogisticRegression().fit(X_train, y_train)

# Register the custom model with Snorkel Flow
# training_set argument is the training_set_uid as defined above
model_uid = sf.register_model(
    node=node, description="My custom model",
    training_set=training_set_uid
)

# For each split, get a feature matrix, get predictions, and add them to
Snorkel Flow
for df in (df_dev, df_valid, df_test):
    X = vectorizer.transform(df[feature_column])
    preds = model.predict(X)
    x_uids = df.index.tolist()
    sf.add_predictions(
        node=node,
        x_uids=x_uids,
        predicted_labels=preds,
```

Copyright © 2025 Snorkel AI, Inc. All rights reserved

```
        model_uid=model_uid,
    )
```

Load data with labels/annotations

In the above example, we have just seen that data can be loaded with training labels. Here, we show how one might load other metadata (i.e., ground truth labels, training labels, predictions, and/or tags) into the dataframe.

```
import snorkelflow.client as sf

# Load data with training labels
df_train = sf.get_node_data(
    node=node,
    split="train",
    training_set_labels=True,
    training_set_uid=1,
    training_set_sampler_config={"strategy": "auto"},
)

# Load data
df_dev = sf.get_node_data(
    node=node,
    split="dev",
    tags=True,
)

# Load data with ground truth labels and predictions
df_test = sf.get_node_data(
    node=node,
    split="test",
    ground_truth=True,  # True by default but explicitly set for illustration
purposes.
    model_uid=model_uid,
    model_predictions=True,
)
```

Generating labeling functions in the notebook

Labeling functions are defined using the `@labeling_function` decorator, which wraps a Python function that takes in a Pandas Series object representing an individual data point as

Copyright © 2025 Snorkel AI, Inc. All rights reserved

input and returns a string representing the label.

The following example shows a labeling function looking for the substring `"xyz"` in a field of the data. It returns the label `"LABEL"` if it finds the substring, and otherwise returns the special label `"UNKNOWN"`.

```
from snorkel.labeling.lf import labeling_function

@labeling_function(name="my_lf")
def lf(x):
    if "xyz" in x.field_name:
        return "LABEL"
    return "UNKNOWN"

# Save to an application
sf.add_code_lf(node, lf, label="LABEL")  # Default applied to split: "dev"
```

In a real LF, the label string (here, `"LABEL"`) needs to be replaced with the string of the label configured for the application.

> NOTE
>
> Once you've saved your LF, you will immediately see returned statistics from applying it to the `dev` set. If you click the **Update Statistics** button on the left pane of the Label page, you can also see the LF in the **LF Summary** pane.

Use variables or imports from outside the LF's scope

Labeling functions can also rely on external resources, like functions or objects defined outside of the labeling function as well as external libraries. Functions and objects can be passed to the labeling function using the `resources` variable, and add a matching keyword argument to the function signature (`find_word_index` in the example below). The recommended way to use additional libraries is to import them in the function body.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

```
from snorkel.labeling.lf import labeling_function

def find_word_index(text):
    import numpy
    try:
        idx = numpy.where(text.split(" ").index("employee"))
    except:
        idx = numpy.array([])
    return idx

@labeling_function(name="my_lf",
resources=dict(find_word_index=find_word_index))
def lf(x, find_word_index):
    import numpy
    idx = find_word_index(x.text)
    if numpy.mean(idx) <= 1000:
        return "employment"
    else:
        return "UNKNOWN"

sf.add_code_lf(node, lf, label="employment")
```

> NOTE
>
> While `resources` can be a nested dictionary, functions should be its direct children otherwise they would break when Snorkel Flow upgrades to a newer version of Python. See `@labeling_function` for more details.

Installing additional NLTK packages for use in an LF or operator

Snorkel Flow comes with popular and vader_lexicon NLTK packages pre-installed. If additional NLTK packages are needed, you can install into the CUSTOM_NLTK_PATH directory. After installation, all code LFs can make use of them. We show a simple example below.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

```
from snorkelflow.utils.file import CUSTOM_NLTK_PATH
from snorkelflow.studio import resources_fn_labeling_function

import nltk
nltk.download("punkt", download_dir=CUSTOM_NLTK_PATH)
nltk.download("averaged_perceptron_tagger", download_dir=CUSTOM_NLTK_PATH)

def get_nltk_tokenizer_parser() -> Dict:
    import nltk
    return {"tokenize": nltk.word_tokenize, "parse": nltk.pos_tag}

@resources_fn_labeling_function(
    name="lf_1", resources_fn=get_nltk_tokenizer_parser,
)
def lf_1(x: pd.Series, tokenize: Callable, parse: Callable) -> str:
    pos_pairs = parse(tokenize(x.span_preview))
    for token, pos in pos_pairs:
        if pos == "NNP":
            return "employment"
    return "UNKNOWN"

sf.add_code_lf(node, lf_1, label="employment")
```

Sampler configs for data loading and model training

Snorkel Flow offers a variety of options for data sampling when loading datasets or training models. This is particularly useful when loading large or unbalanced datasets. Sampling options are passed via a sampler configuration in the Python SDK for functions like `get_node_data` and `train_model`. The sampler configuration is a dictionary with the following three keys: `strategy` (required), `params` (optional), and `class_counts` (optional). The Options for `params` and `class_counts` vary by sampling strategy.

`none`

> No sampling is performed.
>
> *Example*
>
> ```
> sampler_config = {"strategy": "none"}
> ```

`fixed`

> Sample a fixed fraction of each class.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

> *Params*
>
> - `p`: fraction of each class to sample (must be > 0 and <= 1)
>
> *Example*

```
sampler_config = {"strategy": "fixed", "params": {"p": 0.5}}
```

`class_count`

> Sample the provided number for each class. Classes are upsampled or downsampled to matche the target counts if needed.
>
> *Class counts:* the number of samples per class
>
> *Example*

```
sampler_config = {"strategy": "class_count", "class_counts": [100, 200, 100]}
```

`target_proportion`

> Attempt to match the provided class proportions. Upsample classes to match the target proportions if needed.
>
> *Class counts:* unnormalized proportions for each class
>
> *Params*
>
> - `max_sample_multiple`: the sampled dataset will be no more than `max_sample_multiple` -times the size of the original dataset
>
> *Example*

```
sampler_config = {
    "strategy": "target_proportion",
    "params": {"max_sample_multiple": 5},
    "class_counts": [0.333, 0.333, 0.333],
}
```

`auto`

> Attempt to match the ground truth class proportions of the valid set. Upsample classes to match the target proportions if needed.
>
> *Params*
>
> - `max_sample_multiple`: the sampled dataset will be no more than max_sample_multiple-times the size of the original dataset
> - `min_total`: minimum total number of valid set labels to perform sampling; if not met, no sampling occurs
> - `min_per_class``: minimum number of valid set labels for each class to perform sampling; if not met, no sampling occurs
>
> *Example*

Copyright © 2025 Snorkel AI, Inc. All rights reserved

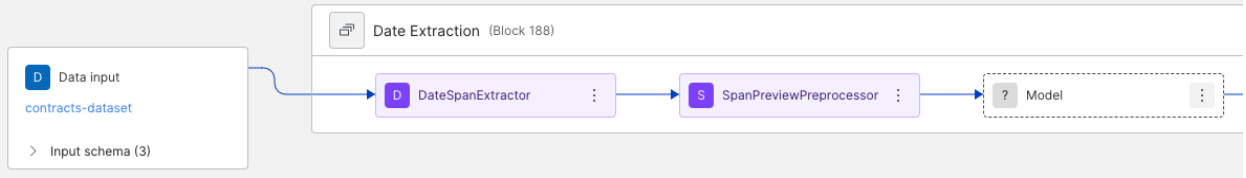
```
sampler_config = {
    "strategy": "auto",
    "params": {"max_sample_multiple": 5, "min_total": 100},
}
```

Copyright © 2025 Snorkel AI, Inc. All rights reserved

Developing and registering custom operators

In addition to Built-in Operators, one can develop a custom operator and use it as part of an application. In a nutshell, a custom operator is a Python function that accepts and outputs dataframes, decorated by one of Snorkel Flow operator decorators.

In this example, let's assume that we want to add an operator that adds a random number after `DateSpanExtractor`.



Develop a custom operator

First, get the (output) DataFrame from `DateSpanExtractor`:

```
import snorkelflow.client as sf
DATASET_NAME = "contracts"
APP_NAME = "date_extraction"
# Get the application info
app = sf.get_application(APP_NAME)

# Get the node UID of DateSpanExtractor
node = sf.get_node_uid(APP_NAME, search_op_type="DateSpanExtractor")[0]

# Get the first 10 rows of the output DataFrame from DateSpanExtractor,
# which will become the **input** for the operator you develop
df = sf.get_node_output_data(application=APP_NAME, node=node,
max_input_rows=10)
```

Note that the uid column of your data has been renamed to context_uid, and is now the index of the dataframe.

Next, define your operator. Custom operators take a dataframe as input and must return a dataframe. As an example, we will create an operator that adds a column of random numbers.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

```
import pandas as pd
from snorkelflow.operators import pandas_operator

@pandas_operator(
    name="add_rand_num", input_schema={}, output_schema={"rand_num": float},
)
def add_rand_num(df: pd.DataFrame) -> pd.DataFrame:
    import random
    df["rand_num"] = [random.random() for _ in range(len(df))]
    return df
```

While `@pandas_operator` lets you define a `Operator` using simple Pandas syntax, `Operator`s are executed over Dask DataFrames. To test our operator, we will first need to convert df to a Dask DataFrame. Then we will run our operator over the converted DataFrame and view the first rows. Next, we may want to test our operator against a dataframe in our notebook. We can do this by calling `my_function.execute_df(df)` on a Pandas DataFrame.

```
import dask.dataframe as dd
ddf = dd.from_pandas(df, npartitions=1)
df_processed_in_nb = add_rand_num.execute([ddf]).compute()
print(df_processed_in_nb.head())
operator_test_in_notebook = add_rand_num.execute_df(df)
```

If we're happy with the results, we can register our operator with Snorkel Flow:

```
operator_uid = sf.add_operator(
    operator=add_rand_num,
    overwrite=False
)["op_uid"]
```

Choosing the right custom decorator

Snorkel Flow provides several decorators for wrapping custom functions to produce an operator, depending on the operator's purpose and properties. We describe some of the more common ones below.

1. pandas_featurizer: A featurizer is an operator that only adds one or more columns (features) to a DataFrame, but does not add or delete any rows. Use this decorator when creating a featurizer. Most operators you end up creating will use this decorator. Note that when using `pandas_featurizer`, you must specify the `input_schema` (columns required by the operator) and `output_schema` (new columns created by the operator).
2. span_normalizer: A normalizer is an operator that converts the `span_text` column into a canonical form, saved in a new `normalized_span` column. Use this decorator if you want to produce a

Copyright © 2025 Snorkel AI, Inc. All rights reserved

normalizer, by decorating the function that maps a string (span text) to another string (normalized span).

3. **span_reducer**: A reducer keeps a subset of rows of the original DataFrame, by e.g. choosing one span to keep per document. Use this decorator to create a reducer by defining the function that maps a Pandas DataFrame to a reduced DataFrame (with a subset of the rows).
4. **row_filter**: A row filter selects certain rows from the original DataFrame to enable conditional processing.
5. **field_extractor**: An extractor operates on a dataframe with one row per document, and produces a dataframe with one row per span. Use this decorator to write a custom extractor by decorating a function that maps the document string to a list of spans (tuples of `char_start`, `char_end`, and an optional `span_entity`).
6. **pandas_operator**: Use this decorator when you have a function that maps a pandas DataFrame to another Pandas DataFrame, but is not of any of the above types (featurizer, normalizer, reducer, or extractor).
7. **page_splitter**: (PDF apps only) This operator takes in a row corresponding to a document and defines how to split the doc up into pages.

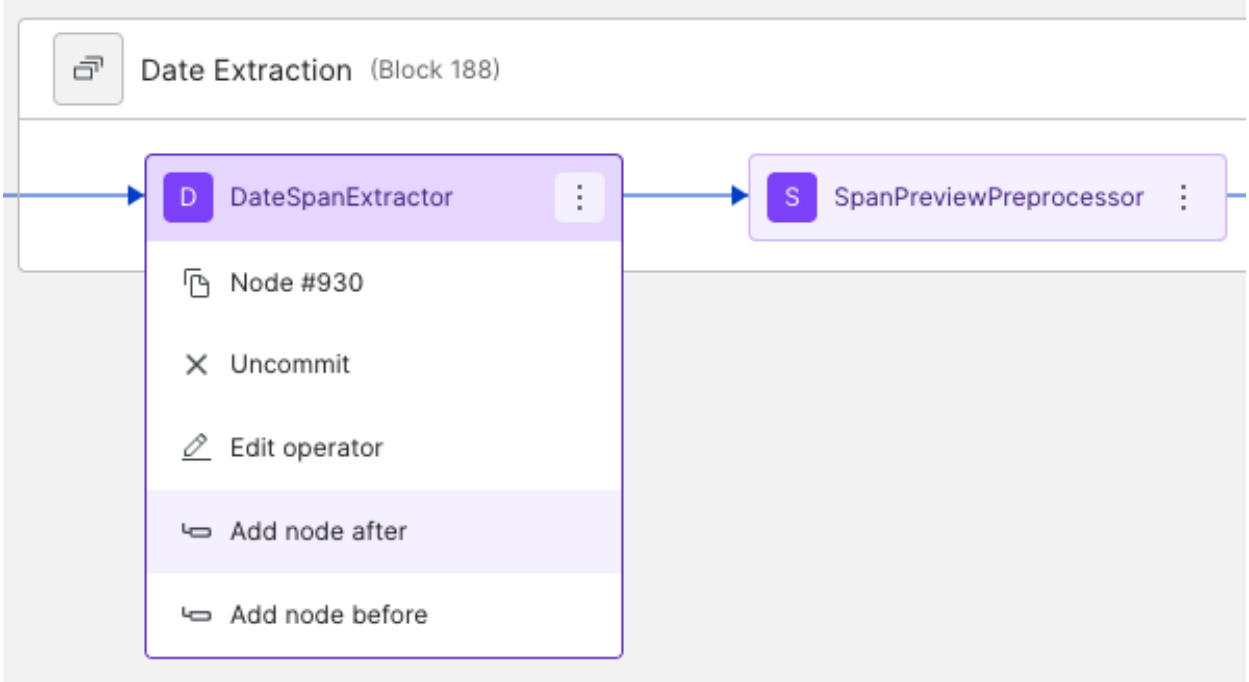
See snorkelflow.operators for the exhaustive list of available decorators.

Use a custom operator

In Application Studio

Click on the hamburger icon on the `DateSpanExtractor`, select **Add node after**, then select **ChangeColumn**. Next, click on the added Featurizer node to select **add_rand_num** that we just added. If you do not see the **Add node after** option, see Compose complex operator graphs to enable this functionality.

Copyright © 2025 Snorkel AI, Inc. All rights reserved



In Notebook

Inserting an operator requires both `input_node_uids` and `output_node_uid`. You can check them either manually in Application Studio or programmatically in a notebook. The snippet below retrieves those node uids, adds an uncommitted node, finally commits the custom operator to this node.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

```
import snorkelflow.client as sf
# Get the node uid of DateSpanExtractor, which will become the custom
operator's input node
input_node = sf.get_node_uid(APP_NAME, search_op_type="DateSpanExtractor")[0]

# Get the node uid of SpanPreviewPreprocessor, which will become the custom
operator's output node
output_node = sf.get_node_uid(APP_NAME,
search_op_type="SpanPreviewPreprocessor")[0]

# First, add a node
node = sf.add_node(
    application=APP_NAME,
    input_node_uids=[input_node],
    expected_op_type="Featurizer",
    output_node_uid=output_node,
)["node_uid"]

# Then, commit the custom operator
sf.commit_custom_operator(
    node=node,
    operator_uid=operator_uid,
)
```

You can quickly check if the custom operator behaves as expected in Notebook.

```
# Get the first 10 rows of the output DataFrame from the added node
df_processed_in_pf = sf.get_node_output_data(application=APP_NAME, node=node,
max_input_rows=10)
print(df_processed_in_pf.head())
```

> NOTE
>
> When a node is added to the DAG, the active data sources at the model node become stale. To check that the model node receives the correct dataframe from the upstream, the model node's active data sources should be refreshed.

```
# Get the UID of the model node and refresh its active data sources
model_node = int(sf.get_model_nodes(APP_NAME)[0])
sf.refresh_active_datasources(model_node)

# Get the (input) DataFrame at the model node
df = sf.get_node_data(model_node)
print(df.head())
```

Copyright © 2025 Snorkel AI, Inc. All rights reserved

Developing a custom page splitter

Page splitters are specifically for PDF apps and are useful in making large documents easier to label and process on the platform. A user can define custom logic for splitting up a PDF by using the `@page_splitter` decorator. The function takes in a single row of a DataFrame (corresponding to a single document) and outputs a list of page indices in each split. The snippet below separates PDF documents into odd and even pages

```
from snorkelflow.operators import page_splitter
import pandas as pd

@page_splitter(name="even_odd_pages")
def even_odd_pages(row: pd.Series) -> List[List[int]]:
    from snorkelflow.rich_docs.rich_doc import RichDocCols

    page_docs = row[RichDocCols.PAGE_DOCS]
    even_splits = list(range(0, len(page_docs), 2))
    odd_splits = list(range(1, len(page_docs), 2))
    return [even_splits, odd_splits]
```

Developing a custom row filter

A user can define custom logic for filtering data by using the `@row_filter` decorator. The function takes in the entire DataFrame and outputs a Pandas Series that is used as the filter. The snippet below filters rows of the DataFrame by looking at a specific regex pattern:

```
import re
from snorkelflow.operators import row_filter
import pandas as pd
pattern = re.compile(r"(?:hello|good morning)", flags=re.IGNORECASE)
@row_filter(
    name="greetings_filter",
    resources=dict(compiled_rgx=pattern),
)
def greetings_filter(df: pd.DataFrame, compiled_rgx: re.Pattern) -> pd.Series:
    text_col = df["text"]
    return text_col.apply(lambda x: bool(compiled_rgx.match(x)))
```

Copyright © 2025 Snorkel AI, Inc. All rights reserved

Advanced options

Including other functions and objects as resources

You may want to define an external helper function or rely on a previously constructed object within your operator. This is supported by bundling these objects into the operator before registering with Snorkel Flow. This can be done within the Operator function's scope, or by passing it in via the `resources` key (and adding a corresponding keyword argument to the function signature).

For instance, you may want to reference a list of user ids within a `PandasOperator`:

```
from snorkelflow.operators import pandas_operator

@pandas_operator(name="Is allowlisted", input_schema={},
resources=dict(allowlist=[5, 20, 100]))
def is_allowlisted(df: pd.DataFrame, allowlist: List[int]) -> pd.DataFrame:
    df["is_allowlisted"] = [user_id in allowlist for user_id in df["user_id"]]
    return df
```

When using resources that are large (e.g. a spacy model), use the `resources_fn` key to pass in a function that generates the resources instead of passing the resources themselves.

```
from snorkelflow.operators import pandas_featurizer
def spacy_fn() -> Dict:
    import spacy
    nlp = spacy.load('en_core_web_sm')
    return {'nlp': nlp}

@pandas_featurizer(name="Tokenize", input_schema={'text': str}, output_schema=
{'tokens': str}, resources_fn=spacy_fn)
def tokenize(df: pd.DataFrame, nlp: Any) -> pd.DataFrame:
    df["tokens"] = df.text.apply(lambda x: nlp.tokenizer(x).to_json())
    return df
```

> NOTE
>
> Using custom or complex Python objects as resources is in beta and may lead to incompatibilities when executing the operator via the web UI.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

Using other libraries in operators

The recommended way to use additional libraries is to import them into the function body.

```
import pandas as pd
from snorkelflow.operators import pandas_operator

@pandas_operator(
    name="add_random_num", input_schema={}, output_schema={"rand_num": float},
)
def add_random_num(df: pd.DataFrame) -> pd.DataFrame:
    import random                                                    # import here!
    df["rand_num"] = [random.random() for _ in range(len(df))]
    return df
```

Specifying input/output schemas for additional validation

For the `@pandas_operator` and `@dask_operator` decorators, an `input_schema` and/or `output_schema` can be specified to explicitly state the expected input/output types of the operator. Custom operators take a dataframe as input and must return a dataframe.

In this example, the input DataFrame is expected to have a column `num_images` with `dtype int`, and this operator will output a new DataFrame with a `special_number` column with `dtype float`.

```
from snorkelflow.operators import pandas_operator
@pandas_operator(
    name="special number generator",
    input_schema={"num_images": int},
    output_schema={"special_number": float},
)
def new_columns(df: pd.DataFrame) -> pd.DataFrame:
    df["special_number"] = df["num_images"].fillna(1) * 1.5
    return df
```

Documentation on additional advanced options can be found in snorkelflow.operators.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

Investigating issues

If you encounter any issues and want to understand the details, you could enable the debug mode of the Snorkel Flow SDK. For example, if you want to investigate issues when using the operator, you can do the following:

```
ctx.set_debug(True)
df_processed_in_pf=sf.get_node_output_data(application=APP_NAME, node=node,
max_input_rows=10)
```

It will show the detailed error message and stacktrace which can help you fix the custom operator.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

Developing and registering custom extractors

In addition to Custom Operators, one can develop a custom Span Extractor for Information Extraction applications. A Span Extractor is an operator that runs over a dataframe of documents, with each row representing a single document, and produces a new dataframe where each row represents a span of text and the document in which it was extracted from. A custom extractor provides additional flexibility to SnorkelFlow built-in application templates.

A custom extractor is a Python function decorated by the `@field_extractor` decorator which returns a list of spans represented by a tuple of three values:

- `char_start`: Index of the starting character of the span
- `char_end`: Index of the ending character of the span. This represents the inclusive bounds of the span (the index of the last character in the span). In Python syntax, the span substring is content[char_start: char_end + 1].
- `span_entity`: A string that serves as a canonical identifier for the span for Entity Classification applications (for most Extraction applications this should be simply set to `None`)

Develop a custom extractor

The following example demonstrates how to use a custom regex to extract candidates.

First, we define the custom extractor. For example, we will create an operator that extracts greeting spans from field body.

```
import re
from snorkelflow.operators import field_extractor

regex = re.compile(r"(?:hello|good morning)", flags=re.IGNORECASE)

@field_extractor(
    name="greetings_extractor",
    field="body",
    resources=dict(compiled_regex=regex),
)
def greetings_extractor(content: str, compiled_regex: re.Pattern) ->
List[Tuple[int, int, Optional[str]]]:
    spans: List[Tuple[int, int, Optional[str]]] = []
    for match in compiled_regex.finditer(content):
        char_start, char_end = match.span(0)
        spans.append((char_start, char_end - 1, None))
    return spans
```

Copyright © 2025 Snorkel AI, Inc. All rights reserved

While `@field_extractor` lets you define an `Operator` using simple Python string syntax, `Operator` s are executed over Dask DataFrames. To test our extractor, we will first need to convert the data to a Dask DataFrame. Then we will run our extractor over the converted DataFrame and view the first rows.

```
import dask.dataframe as dd
# the input node (previous node) to our custom extractor
# since this extractor is the first node in our app we just set this value to
-1
PREVIOUS_EXT_NODE = -1
APP_NAME = "name-of-your-application"

df = sf.get_node_output_data(
    application=APP_NAME,
    node=PREVIOUS_EXT_NODE,
    max_input_rows=10
)
ddf = dd.from_pandas(df, npartitions=1)
df_processed_in_nb = greetings_extractor.execute([ddf]).compute()
print(df_processed_in_nb.head())
```

Ensure that you're extractor is working properly on this subset by evaluating the newly created `span_text` column in the resulting DataFrame. If you are happy with the results, add the extractor:

```
sf.add_operator(greetings_extractor)
```

Go to Application Studio (aka "the DAG") and click on the extractor node and search for the **greetings_extractor** operator that we just added to commit it in our application. Note, if the model node is in red you will also need to refresh the its datasources by clicking the curved arrow icon.

Click on the model node to view the output of the extractor in the data viewer.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

Developing and registering custom operator classes

Generally, parameters for custom operators are hard-coded within the user-defined function. This is used for simple and quick transformations. To create custom operators that have parameters specified in the Application Studio overview, we instead create custom operator classes.

As an example, let's assume that we want to add an operator that scales a numeric column. That is, we want the mean of the column to be 0, and the standard deviation of the column to be 1. The Scaler customer operator we define should do so by subtracting a specified mean and dividing by a specified standard deviation. This operator takes a number of arguments that should be configurable in the Application Studio overview, such as field (the name of column to scale), target_field (the new column name), mean, and std. To enable these configurable operators, we define an operator class instead.

Defining the class

To develop an operator class, we write a class in Notebook that inherits from the `Featurizer` class. We need to define several necessary methods as part of this new operator class: `__init__`, `input_schema`, `output_schema`, `_compute_features`, and `_fit` (optional).

Let's take a look at each of these methods individually as they are added to the Notebook.

`__init__` defines the arguments that will be configured as part of our newly created operator, which will be called `CustomStandardScaler`. These values will be customizable in the Application Studio overview page. As such, we place `field`, `target_field`, `mean`, `std` as the parameters of `__init__` and initialize these as instance variables.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

```
from typing import Any, Dict, Optional

from snorkelflow.operators.featurizer import Featurizer
from snorkelflow.operators.operator import ColSchema

class CustomStandardScaler(Featurizer):
    """Preprocessor that scales a numerical column to mean=0 and std=1."""

    def __init__(
        self,
        field: str,
        target_field: Optional[str] = None,
        mean: float = 0.0,
        std: float = 1.0,
    ) -> None:
        self.field = field
        self.target_field = target_field or f"{field}_scaled"
        if std is not None and std < 0:
            raise ValueError("std must be >= 0")
        self.mean = mean
        self.std = std
```

Custom operator classes enable us to define an optional `_fit` method for trainable operators. Fit functionality is very useful for operators with parameters that depend on the dataset itself. In this example, mean and std are configurable parameters that need to be calculated on the field column.

The class method `_fit` is generally used to automatically fit the operator arguments based on training data. This method takes in the dataframe `df` and additional parameters for the fit, in this case `field` and `target_field`. `_fit` should return a dictionary of kwargs matching those specified in `__init__`. In this example, the necessary `mean` and `std` are calculated using the numpy library and added as arguments along with `field` and `target_field`.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

```
class CustomStandardScaler(Featurizer):
    ...
    @classmethod
    def _fit(cls, df: pd.DataFrame, field: str, target_field: str) ->
Dict[str, Any]:  # type: ignore
        import numpy as np
        kwargs = {
            "field": field,
            "target_field": target_field,
            "mean": np.mean(df[field]),
            "std": np.std(df[field]),
        }
        return kwargs
```

`input_schema` and `output_schema` are properties that define the input and output dataframes format of this operator. Specifically, `CustomStandardScaler` will operate on a numerical(float) column with the passed-in name from `self.field`. Additionally the output column will be numerically(float) valued and have the passed in name from `self.target_field`.

```
class CustomStandardScaler(Featurizer):
    field: str
    target_field: str

    @property
    def input_schema(self) -> ColSchema:
        return {self.field: float}

    @property
    def output_schema(self) -> ColSchema:
        return {self.target_field: float}
```

`_compute_features` will contain the actual scaling logic and dictates how the transformation is calculated on the input dataframe `input_df`. As mentioned before, the scaling logic will simply subtract the mean from the column and divide it by the std. Finally, we return the resulting dataframe.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

```
class CustomStandardScaler(Featurizer):
    field: str
    target_field: str
    std: float
    mean: float

    def compute_features(self, input_df: pd.DataFrame, callback:
Optional[OpProgressCallback] = None) -> pd.DataFrame:
        """Override base class method"""
        input_df[self.target_field] = (input_df[self.field] - self.mean) /
self.std
        return input_df
```

Here is the full code block which defines our `CustomStandardScaler` operator class. Note that custom operators compute on dataframes as inputs and outputs.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

```
from typing import Any, Dict, Optional

from snorkelflow.operators.featurizer import Featurizer, OpProgressCallback
from snorkelflow.operators.operator import ColSchema


class CustomStandardScaler(Featurizer):
    """Preprocessor that scales a numerical column to mean=0 and std=1."""

    def __init__(
        self,
        field: str,
        target_field: Optional[str] = None,
        mean: float = 0.0,
        std: float = 1.0,
    ) -> None:
        self.field = field
        self.target_field = target_field or f"{field}_scaled"
        if std is not None and std < 0:
            raise ValueError("std must be >= 0")
        self.mean = mean
        self.std = std

    @classmethod
    def _fit(cls, df: pd.DataFrame, field: str, target_field: str) ->
Dict[str, Any]:  # type: ignore
        import numpy as np
        kwargs = {
            "field": field,
            "target_field": target_field,
            "mean": np.mean(df[field]),
            "std": np.std(df[field]),
        }
        return kwargs

    @property
    def input_schema(self) -> ColSchema:
        return {self.field: float}

    @property
    def output_schema(self) -> ColSchema:
        return {self.target_field: float}

    def _compute_features(self, input_df: pd.DataFrame, callback:
Optional[OpProgressCallback] = None) -> pd.DataFrame:
        """Override base class method"""
        input_df[self.target_field] = (input_df[self.field] - self.mean) /
```

Copyright © 2025 Snorkel AI, Inc. All rights reserved

```
self.std
        return input_df
```

Registering the class

After defining the operator class, we are now ready to register this class for use in Snorkel Flow.

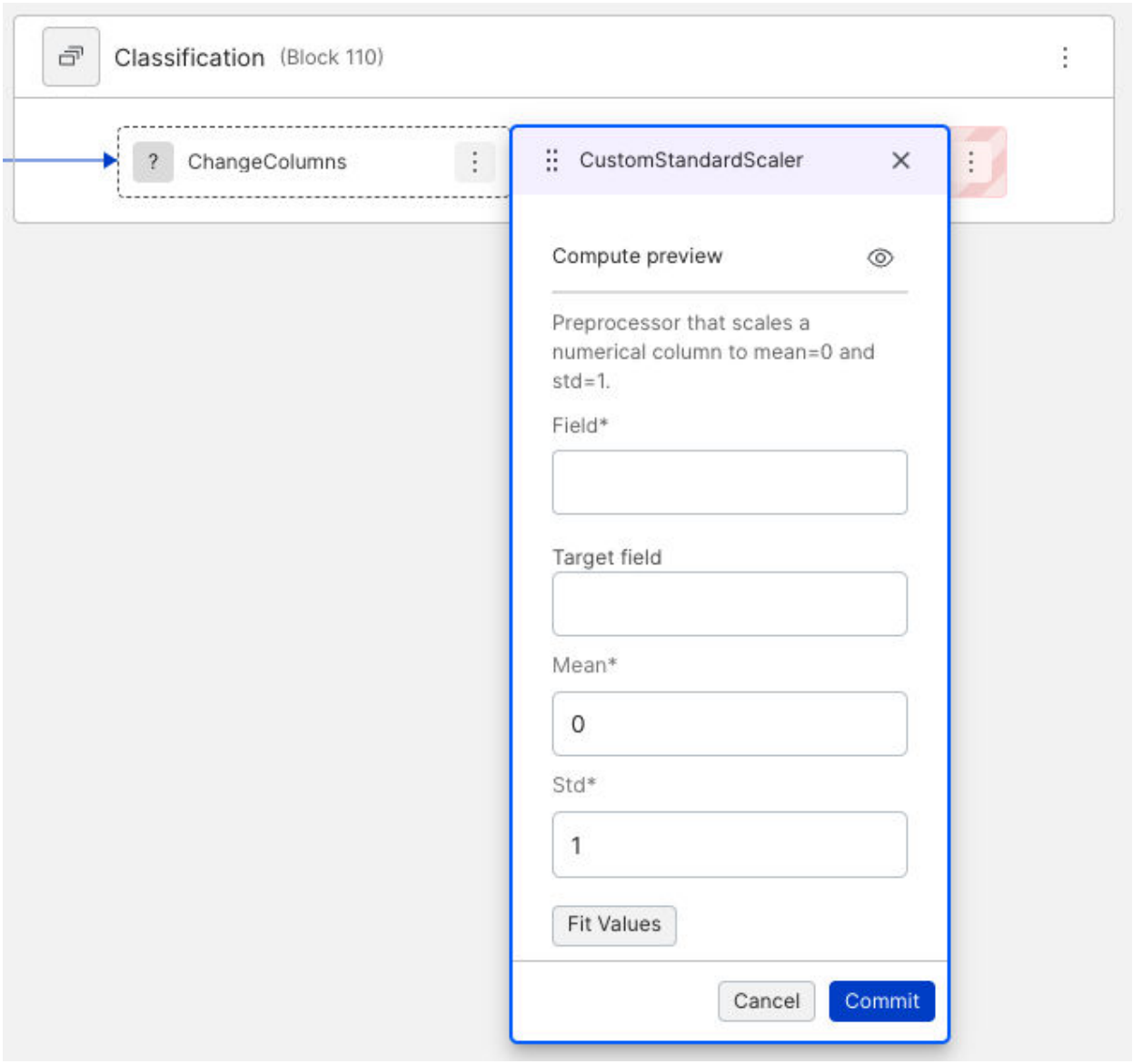
We do so by calling the `add_operator_class` SDK function and pass in our class definition `CustomStandardScaler`.

```
sf.add_operator_class(
    CustomStandardScaler, overwrite=True
)
```

Use a custom operator class

In the Application Studio overview, click on the hamburger icon on the `Model` node, select **Add node before**, then select **ChangeColumn**. Next, click on the added Featurizer node to select **CustomStandardScaler** that we just added. Since we defined a class, the custom operator now acts as a built-in operator with fully configurable parameters that were defined in our `__init__`. Simply enter the necessary parameters and commit the custom operator like any other built-in operator.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

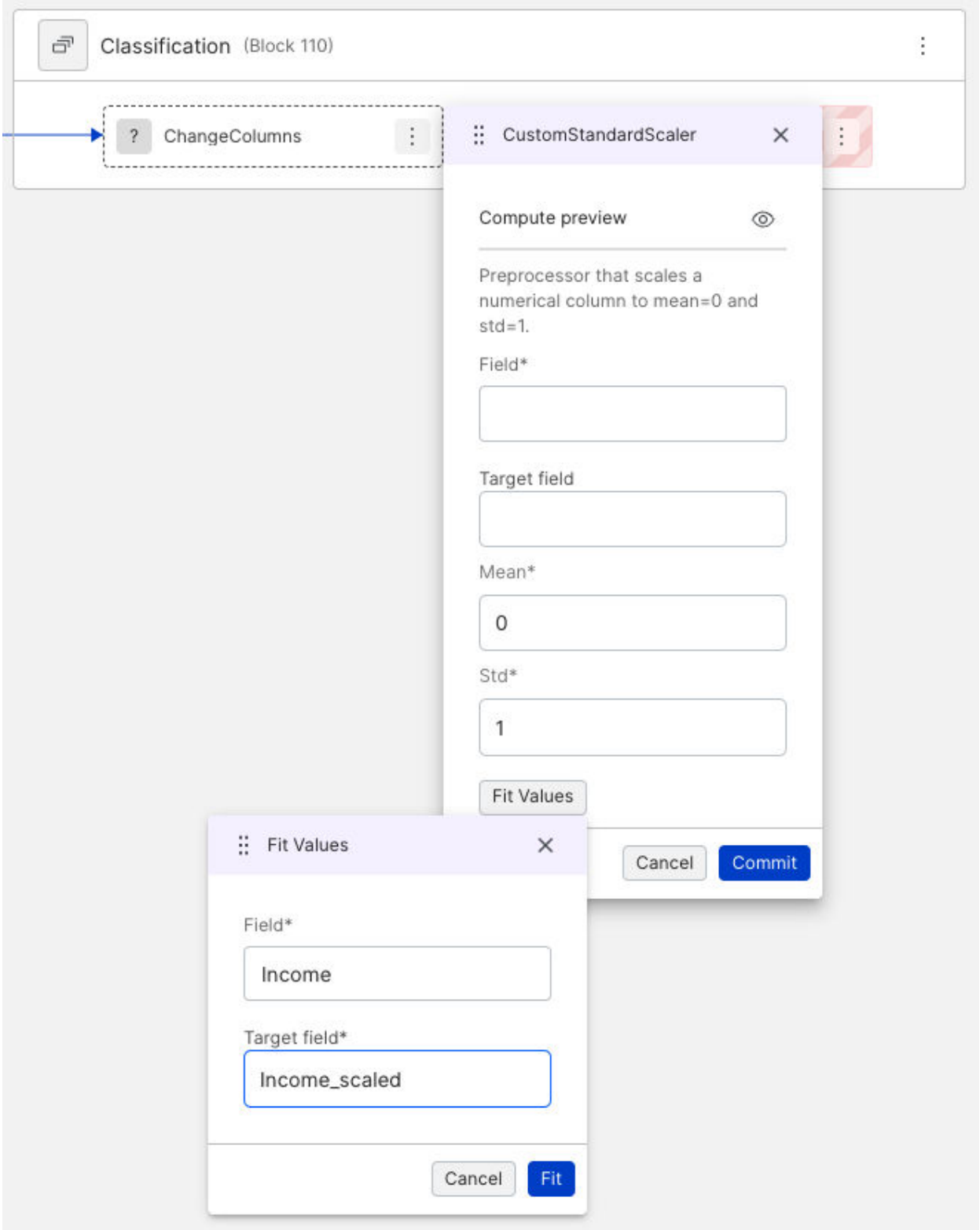


Fit an operator in Snorkel Flow

Once we have a trainable operator with fit functionality. We can revisit the Application Studio overview and examine the operator config of our newly created operator `CustomStandardScaler`.

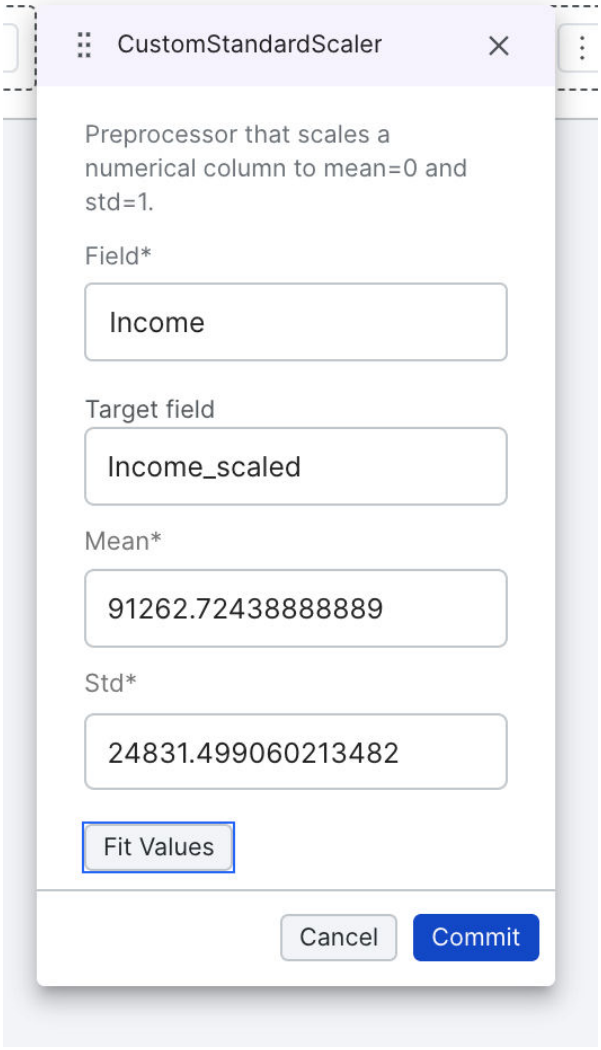
For operators with `_fit` defined, a `Fit Values` button will appear on the bottom left. Clicking on that will lead to a separate modal that takes in the necessary arguments. In this example, if we wanted to scale the `Income` and place it in a new column `Income_scaled`, we simply select those fields and click `Fit`.

Copyright © 2025 Snorkel AI, Inc. All rights reserved



After the operator has finished fitting to the data, the previous operator config modal will be automatically populated with the result. The operator can then be committed just like any other operator.

Copyright © 2025 Snorkel AI, Inc. All rights reserved



Investigating issues

If you encounter any issues and want to understand the details, you could enable the debug mode of the Snorkel Flow SDK. For example, if you want to investigate issues when registering the operator class, you can do:

```
ctx.set_debug(True)
sf.add_operator_class(CustomStandardScaler, overwrite=True)
```

It will show the detailed error message and stacktrace which can help you fix the custom operator class.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

Using custom data points

Snorkel Flow supports the usage of custom data points (such as paragraphs, bounding box, time range, etc.).

If you want to change the data point used to identify each row without changing the dataframe itself, you can use the ChangeDatapoint operator. It lets you specify the desired data point type and columns, but it won't work if the set of desired data point columns doesn't lead to a unique data point per row.

For that, you'll have to write a custom operator (Developing and registering custom operator classes) whose output dataframe will be a data point with the desired type and columns.

As an example, we'll use custom operators to

1. split all the documents into paragraphs,
2. and reduce the separate paragraph predictions from our ML model into one document prediction.

Data point format

In Snorkel, data point UIDs have the following format:

```
<datapoint_type>::<comma-separated list of values from the columns selected>
```

As an example, here's an example dataframe with the data point UID if the type is `page` and the columns selected are `doc_uid`, `page_uid`.

data point UID/index	doc_uid	page_uid	text
page::12,2	12	2	random text
page::15,4	15	4	also random

Adding a data source

In this example, we'll use the contract classification dataset. Follow the instructions under Upload Data in Document Classification and Classifying Contract Types.

Creating an application

Create a new application and select as type "Classification".

Copyright © 2025 Snorkel AI, Inc. All rights reserved

1. Choose the dataset you created in the previous step.
2. Give the application name.
3. “Select all” for the input schema.
4. Choose “label” for the ground truth column.
5. Click “Autofill Labels” to get the unique values from the label column.
6. “Save” the application.

You should now see the Directed Acyclic Graph (DAG) of a basic classification application.

Writing a custom operator class

For more background on how to write custom operators, take a look at Developing and registering custom operator classes and in particular the **Using custom data points** section. To add the custom operators, open the in-platform Notebook and execute the following code block. It adds a custom paragraph splitter, where a paragraph is defined as text separated by two new lines.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

```
from typing import List, Type
import dask.dataframe as dd
from snorkelflow.operators import Operator
from snorkelflow.operators.operator import ColSchema
from snorkelflow.utils.datapoint import DatapointType

class CustomParagraphSplitter(Operator):

    def __init__(
        self,
        field: str,
        paragraph_idx_field: str = "paragraph_idx",
    ) -> None:
        self.field = field
        self.paragraph_idx_field = paragraph_idx_field
        self.paragraph_text_field = f'{field}_paragraph'
        self.new_datapoint_cols = [self.paragraph_idx_field]

    @property
    def input_schema(self) -> ColSchema:
        return {self.field: str, 'context_uid': int}

    @property
    def output_schema(self) -> ColSchema:
        return {self.paragraph_idx_field: int, self.paragraph_text_field: str}

    def get_datapoint_type(
        self, input_datapoint_types: List[Type[DatapointType]]
    ) -> Type[DatapointType]:
        from snorkelflow.utils.datapoint import Datapoint
        """Get datapoint_type for output DataFrame of the operator, given
types for inputs."""
        return Datapoint

    def _execute(self, input_ddfs: List[dd.DataFrame]) -> dd.DataFrame:
        from snorkelflow.utils.datapoint import Datapoint

        def map_fn(input_df: pd.DataFrame) -> pd.DataFrame:
            from snorkelflow.types.load import DATAPOINT_UID_COL
            import pandas as pd

            def apply_fn(row: pd.Series) -> pd.Series:
                paragraphs = row[self.field].split('\n\n')
                row[self.paragraph_idx_field] = list(range(len(paragraphs)))
                row[self.paragraph_text_field] = paragraphs
                return row

            df = input_df.apply(apply_fn, 1)
```

Copyright © 2025 Snorkel AI, Inc. All rights reserved

```
            df_paragraphs = df[[self.paragraph_idx_field,
self.paragraph_text_field]].apply(pd.Series.explode).dropna()
            df_paragraphs = df_paragraphs.astype({self.paragraph_idx_field:
int})
            df = df_paragraphs.join(input_df)

            # Set the new index and sort -- note that we're doing this on the
pandas level
            datapoint_instance = Datapoint(["context_uid",
self.paragraph_idx_field])
            df[DATAPOINT_UID_COL] =
datapoint_instance.get_datapoint_uid_col_pandas(df)
            return df.set_index(DATAPOINT_UID_COL).sort_index()

        return self._execute_avoiding_set_index(input_ddfs[0], map_fn,
Datapoint)
```

Note that in `_execute`, we call `_execute_avoiding_set_index` which takes care of passing the correct meta to dask and setting the divisions. We'll see an example of doing it manually below when we write the reducer. `_execute` takes a dataframe as input and outputs a dataframe.

The `map_fn` sets the index and sorts on each dask dataframe partition. Because the input dataframe is sorted and we're adding to the index, sorting within a partition guarantees that the resulting dataframe is also sorted. This allows us to avoid setting the index on dask, which is computationally expensive as it triggers a reshuffling of the data.

Now that we've defined the operator, we need to add it to the DAG.

```
import snorkelflow.client as sf

DATASET_NAME = ""
APP_NAME = ""

# register the operator
sf.add_operator_class(
    CustomParagraphSplitter, overwrite=True
)

# add a node with the source as parent, and commit the CustomParagraphSplitter
to it
ps_node_uid = sf.add_node(APP_NAME, [-1], None, add_to_parent_block=True)
['node_uid']
sf.commit_builtin_operator(
    ps_node_uid, op_type='CustomParagraphSplitter', op_config={'field':
'text'}
)
```

Copyright © 2025 Snorkel AI, Inc. All rights reserved

Adding a model node

Next, add a model node that will take the output of the CustomParagraphSplitter and predict on it. This can be done by executing:

```
# add a model node after the CustomParagraphSplitter
model_node_uid = sf.add_block_to_application(
    application=APP_NAME,
    template_id="clf",
    input_node=ps_node_uid,
    block_config=dict(label_map={'POS': 1, 'NEG': 0, 'UNKNOWN': -1},
label_col=None),
)['node_uids'][0]
```

Writing a custom reducer

We'll now add a reducer that will output a ground truth for a document based on the ground truths for the paragraphs. In this example, we'll use the most common paragraph prediction in the document as that document's prediction.

Similarly, as before, we first define the custom reducer:

Copyright © 2025 Snorkel AI, Inc. All rights reserved

```
from snorkelflow.operators import Operator
from snorkelflow.operators.operator import ColSchema

class CustomParagraphReducer(Operator):
    def __init__(
        self,
        field: str,
        paragraph_idx_field: str = "paragraph_idx",
    ) -> None:
        self.paragraph_idx_field = paragraph_idx_field
        self.paragraph_text_field = f'{field}_paragraph'

    @property
    def input_schema(self) -> ColSchema:
        return {self.paragraph_idx_field: int, self.paragraph_text_field: str,
'preds': int}

    @property
    def output_schema(self) -> ColSchema:
        return {}

    @property
    def drop_schema(self) -> Optional[List[str]]:
        return [self.paragraph_idx_field, self.paragraph_text_field]

    def get_datapoint_instance(
        self, input_datapoint_instances: List[DatapointType]
    ) -> DatapointType:
        from snorkelflow.utils.datapoint import DocDatapoint
        return DocDatapoint()  # by default DocDatapoint columns are
["context_uid"]

    def _execute(self, input_ddfs: List[dd.DataFrame]) -> dd.DataFrame:
        from snorkelflow.utils.datapoint import Datapoint
        from snorkelflow.types.load import DATAPOINT_UID_COL
        from snorkelflow.utils.datapoint import DocDatapoint

        def map_fn(input_df: pd.DataFrame) -> pd.DataFrame:
            import pandas as pd

            # Calculate the doc level prediction as the paragraph prediction
that appears most often
            df_scores = input_df[['context_uid',
'preds']].groupby(['context_uid']).agg(lambda l:
pd.Series.mode(l).iloc[0]).reset_index()
            df_rest = input_df[[col for col in input_df.columns if col not in
[self.paragraph_idx_field, self.paragraph_text_field, 'preds']]]
            df_rest = df_rest.groupby(['context_uid']).agg(lambda x:
```

Copyright © 2025 Snorkel AI, Inc. All rights reserved

```
x.iloc[0])
            df = df_rest.merge(df_scores, on='context_uid')

            # Set the new index and sort
            datapoint_instance = DocDatapoint()
            df[DATAPOINT_UID_COL] =
datapoint_instance.get_datapoint_uid_col_pandas(df)
            return df.set_index(DATAPOINT_UID_COL).sort_index()

        ddf = input_ddfs[0]
        # The meta has to contain the rows in same order as the result of
map_partitions
        meta: Dict[str, type] = {'context_uid': int}
        meta.update(**{k:v for k,v in ddf.dtypes.to_dict().items() if k not in
[self.paragraph_idx_field, self.paragraph_text_field, 'preds']})
        meta['preds'] = object
        ddf = ddf.map_partitions(map_fn, meta=meta)
        # Set the divisions (allows us to avoid setting index on dask level)
        ddf.divisions = DocDatapoint.get_new_divisions(ddf.divisions, 1)
        return ddf
```

Few things to note:

- Only dropping data point columns from the end is permitted, as the resulting dataframe will still be sorted in that case. In this case, the input dataframe contains `["context_uid", "paragraph_idx"]` as data point columns, and we dropped `paragraph_idx`. We couldn't drop `context_uid`.
- In `_execute` when calling `map_partitions` on the dask dataframe, we have to pass in the `meta` of the output dataframe. The order of the columns is important and has to match the order of columns as the result of applying `map_partitions`.
- If dropping data point columns, ensure it doesn't result in duplicate values in the index. Here, the `groupby` guarantees that.

Then we add it to the DAG after the model node:

```
# register the reducer
sf.add_operator_class(
    CustomParagraphReducer, overwrite=True
)

# add a node after the model one and commit the CustomParagraphReducer to it
pr_node_uid = sf.add_node(APP_NAME, [model_node_uid], None,
add_to_parent_block=True)['node_uid']
sf.commit_builtin_operator(
    pr_node_uid, op_type='CustomParagraphReducer', op_config={'field': 'text'}
)
```

Copyright © 2025 Snorkel AI, Inc. All rights reserved

Iteratively develop ML model

If you go back to the DAG now, you will see the new custom operators that we added:

Click on the model node that comes after the `CustomParagraphSplitter` to go to the **Label** page. From here, you can follow the standard process of adding labeling functions (LFs), programmatically labelling the dataset, and training a model. If you're unsure how to do that, you can refer to those steps in the Document Classification and Classifying Contract Types tutorial.

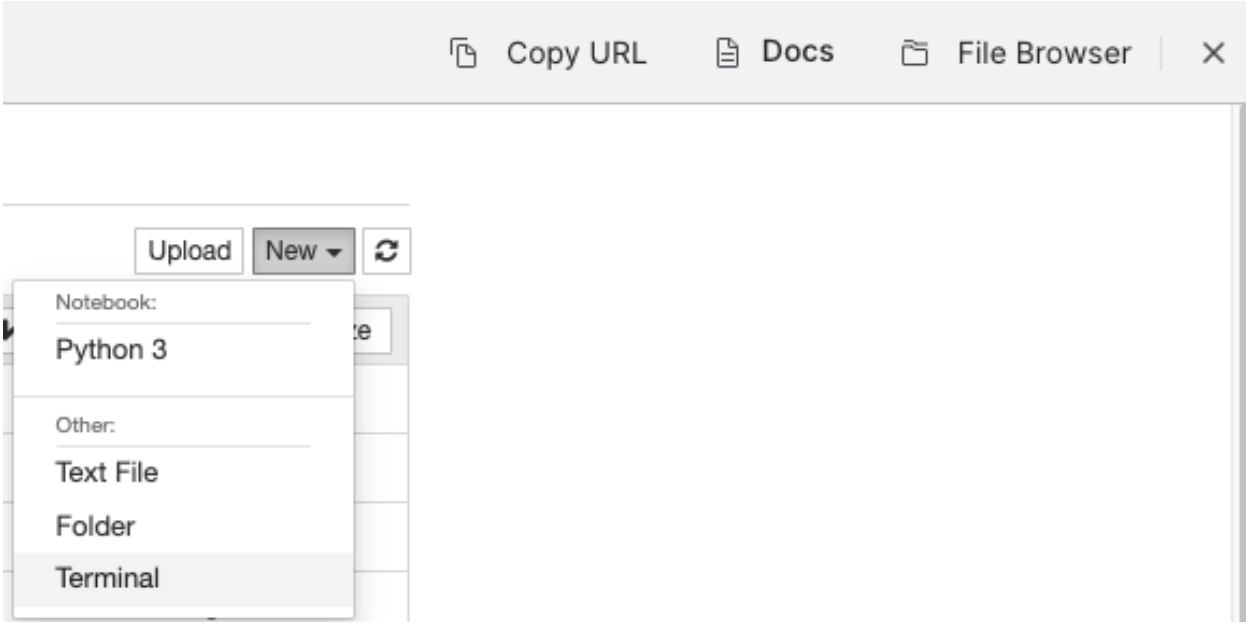
Copyright © 2025 Snorkel AI, Inc. All rights reserved

Tips

How to use Git CLI

The in-platform notebook server comes with Git CLI, which allows users to version controll notebooks, data files, and more. While Git CLI is available through the Notebook via system shell access (i.e., the magic command `!git`), the built-in terminal works better with commands that prompt user input.

First, click “File Browser” at the top-right corner, followed by “New” -> “Terminal” as below:



Then paste the HTTPS URL for your repository, like https://github.com/USERNAME/REPO.git, to the terminal:

```
$ git clone https://github.com/USERNAME/REPO.git
Username: YOUR_USERNAME
Password: YOUR_PERSONAL_ACCESS_TOKEN
```

As of today, you can only use your personal access token. See here for more details.

> NOTE
>
> Note that SSH URL, like `git@github.com:user/repo.git`, is not supported.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

> NOTE
>
> Snorkel Flow does not come with its own internal Git repository. Please create or provide your own from GitHub or equivalent.

Copyright © 2025 Snorkel AI, Inc. All rights reserved

Model zoo

The Snorkel Flow Model Zoo supports models from three major modeling libraries: Scikit-Learn, XGBoost, and HuggingFace. To see the list of models available, see model zoo.

> NOTE
>
> The Snorkel Flow Model Zoo is only available from the in-platform notebook server.

If you want to use non-sklearn models, make sure you've installed relevant frameworks such as `torch`, `transformers`, and `xgboost`.

The model zoo contains model configurations for several commonly used models, which provide a good starting point for model training. To create a model from a model configurations, first make a deep copy of the default config for that class to avoid modifying the constant. Then, make necessary modifications to the config dictionary, such as specifying the number of classes and the fields to use as input.

If you're not familiar with the default config, you can simply print it out to see what options are available.

```
# Import SKLearn logistic regression default config
from snorkelflow.models.model_configs import
SKLEARN_LOGISTIC_REGRESSION_CONFIG

# Import default model options
from snorkelflow.models.utils import (
    MODEL_CONFIG_OPTION_FIELDS,
    MODEL_CONFIG_OPTION_N_CLASSES,
)

# Make a copy of the SKLearn logistic regression config
model_config = copy.deepcopy(SKLEARN_LOGISTIC_REGRESSION_CONFIG)

# Modify the model config to specify the fields used and number of classes
model_config.options[MODEL_CONFIG_OPTION_FIELDS] = ["text"]
model_config.options[MODEL_CONFIG_OPTION_N_CLASSES] = N_CLASSES
```

Use the model registry to instantiate a model from a model config.

```
# Import model registry
from snorkelflow.models.model_registry import get_model_from_config

model = get_model_from_config(model_config)
```

Copyright © 2025 Snorkel AI, Inc. All rights reserved

Next, load the Pandas DataFrame and Snorkel labels from Snorkel Flow using the appropriate training set. For more details on training data loading from Snorkel Flow, see Python SDK.

```
import snorkelflow.client as sf
node = sf.get_model_node(APP_NAME)
train_df = sf.get_node_data(
    node=node,
    split="train",
    training_set_uid=1,
    training_set_labels=True,
    training_set_sampler_config={"strategy": "auto"},
)
```

Finally, you can train the instantiated model and make predictions on Pandas DataFrames using the train and predict methods in the model.

```
# To train a model, call `train` with the train dataframe (`df`),
# and training set labels (`Y`).

# Train model
model.train(df=train_df, Y=train_df["training_set_labels"])

# Predict on the valid set
preds, probs = model.predict(df=valid_df)

# Score on the valid set
print("Accuracy Score on Valid Set: ",
accuracy_score(y_true=valid_Y["ground_truth"], y_pred=preds))
```

For details on registering models and predictions with Snorkel Flow, see Python SDK.

Copyright © 2025 Snorkel AI, Inc. All rights reserved