An introduction to Snorkel and data-centric AI

The how, what, and why of Snorkel AI's programmatic data labeling approach

Snorkel

Introduction

Snorkel's journey began at the Stanford AI lab in 2015, where the Snorkel AI founding team started studying the then largely overlooked problem of labeling and managing the training data that machine learning models learn from. Most AI approaches work by learning from thousands or millions of examples of a task that have been labeled with the correct answer or action–known as “training data.” The not-so-hidden secret of AI is that even today, this training data requires vast volumes of painstaking manual labeling effort.

We began to see then that this ‘dark ages’ approach to labeling training data by hand would not scale, especially as state-of-the-art machine learning models became more accessible and increasingly data-hungry. This consequence is especially the case in verticals like healthcare, government, finance, etc., where data is incredibly difficult and costly to label due to privacy, expertise, and frequent relabeling requirements. Indeed, training data has arguably become the critical bottleneck in AI today.

Convinced that there had to be a better way than hand-labeling, we spent over five years developing new programmatic approaches to labeling, augmenting, structuring, and managing this training data called Snorkel. Snorkel was co-developed and deployed with some of the world’s leading organizations like Google, Intel, Apple, Stanford Medicine and represented in over fifty peer-reviewed publications. With Snorkel, you create massive amounts of labeled training data programmatically in a matter of hours instead of weeks or months of labeling data manually. Snorkel unlocks a fundamentally new, faster, and more practical way to develop AI.

To apply this revolutionary technology to the complete ML lifecycle, we spun out of the Stanford AI lab in 2019 to build Snorkel Flow, the data-centric AI application development platform powered by programmatic data labeling. Snorkel Flow enables data scientists, machine learning engineers, and subject matter experts to collaboratively create and manage training data rapidly, train custom ML models, analyze and iterate to drive systematic improvements, and adapt and deploy AI applications quickly.

In this ebook, we discuss the following topics.

- How manual labeling blocks enterprises from scaling AI
- Benefits of adopting data-centric AI approach
- What is Snorkel?
- How does Snorkel work?
- Case studies on Snorkel
- Accelerating AI with Snorkel Flow

The training data bottleneck

We are seeing one of the most significant transformations of enterprise software in our lifetimes—from software specified by code to a new wave of software systems that learn from data using AI and ML. These new software systems learn how to carry out nuanced tasks over complex data that otherwise would have been impossible to specify by manually written code, by instead learning directly from labeled examples–often called training data. This change unlocks new applications that were impossible to create before, with less engineering work needed than ever before–but relying on large volumes of this carefully and custom-curated training data.

Businesses stand to benefit significantly from this new wave of software systems.

- Boost operational efficiency and savings with automation.
- Grow top-line revenue with new products and better customer experiences
- Capture human knowledge and scale capacity with intelligent applications
- Respond to customer, competitive, or regulatory shifts faster.

This powerful, adaptive enterprise software that goes beyond the capabilities of hand-written code offers a way to solve business-critical problems faster, with more accuracy, and at a lower cost. Driven by this promise, enterprises are spending billions of dollars attempting to put AI and ML to use. IDC predicts that by 2024, the AI market expects to break the $500 billion mark.

Yet few enterprises are realizing value from their AI investments. In a global survey of 3000 business managers and executives, more than half of all respondents affirmed that their organizations are piloting or deploying AI, have an AI strategy, and understand how AI can generate business value. Yet, just 1 in 10 companies generates significant financial benefits with AI. (Source: MIT Sloan Management Review and BCG report)

Just 1 in 10 companies generates significant financial benefits with AI [today].

Study by MIT Sloan Management Review and Boston Consulting Group

With so much technical progress, and so much of it making it into commoditized and robustly supported open-source form, why is there so little real enterprise success? The answer all too often is that many enterprises continue to be bottlenecked by one key ingredient: the large amounts of labeled data to train these new systems.

Over the years at Stanford AI Lab and now Snorkel AI, we've talked to hundreds of organizations. We've seen that organizations in most verticals today face challenges around getting and maintaining training data. These challenges block even organizations with the world's largest technology budgets from using ML. In a recent poll, we found that 80% of AI practitioners reported that 1 out of 2 projects are blocked by lack of training data.

More than half of the AI projects are blocked by lack of training data for 80% of AI practitioners

Snorkel AI survey of 300 practitioners 2021

Most training data created today is manually labeled whether it is done internally or carried out crowdsourced services. However, organizations face the following key challenges with manually labeled training data:

1. Time to production is long. Manual labeling is painfully slow. Even with an unlimited labor force or budget, it can take person-months/years to deliver necessary training data and train models with production-grade quality.

2. Annotation eats up a significant portion of the budget. Manual labeling is inherently expensive as it scales linearly at best and is often error-prone. Data science teams rarely have adequate annotation budgets.

3. Complex data sets need subject matter expertise. Most training data requires highly trained experts, SMEs, to label, e.g., doctors, legal analysts, network technicians, etc., who often need to be well-versed in specific organization's goals and datasets. However, available SMEs are limited, and expensive to label each datapoint manually.

4. Outsourcing labeling is not always an option. Most organizations cannot ship data off-prem to be labeled, making it impossible to use hand-labeling services. This challenge indicates that development teams are stuck for months waiting on building training datasets.

5. Adapting applications requires relabeling. Most organizations have to deal with constant change in input data and upstream systems and processes and downstream goals and business objectives—rendering existing training data obsolete. This challenge requires enterprises to relabel training data constantly.

6. AI Applications are hard to govern. Most organizations need to be able to audit how their data is being labeled, and consequently, what their AI systems are learning from. Even when outsourcing labeling is an option, performing essential audits on hand-labeled data is a near impossibility.

These challenges have resulted in AI practice being model-centric where iterating on the model is the only lever for performance improvement. Siloed, manual labeling process makes it hard if not impossible to iterate on data, leaving significant performance on the table. AI application development needs a practical solution to truly deliver on the promise of AI.

Snorkel AI survey of 300 practitioners 2021

The future of AI is data-centric

The Snorkel project started at the Stanford AI Lab in 2015 around two core hypotheses:

1. As models became increasingly powerful and commoditized, success or failure in AI was going to be all about the training data, and as a result, AI development was going to shift from being model-centric to data-centric.
2. If AI development was going to be data-centric, tasks like labeling, augmenting, slicing, cleaning, and monitoring data would all have to be increasingly programmatic for AI to be as practical and accessible as any other type of software development.

Over half a decade later, and two years since Snorkel AI's founding, we see the thesis on data-centric AI has become so relevant and resonant. Today, an increasing number of organizations see that data is the arbiter of their AI success or failure, as well as their risk, governance, and privacy compliance, fairness and equitability, and agility. And, as data moves to the forefront, so too does the pain of labeling and managing it all by hand. Even the largest organizations in the world are blocked from using AI when person-months of manual effort are needed every time a model needs to be built or updated.

At the forefront of this shift from model-centric to data-centric AI, Snorkel Flow, has enabled some of the world's largest and most sophisticated organizations to bridge this gap between the challenges of real-world data and the power of modern AI. **Snorkel Flow, a data-centric AI development platform based on *programmatic labeling*, is used by Fortune 500 enterprises such as Chubb and BNY Mellon and government agencies to accelerate AI development by 10-100x speedups, unlock seven-to-eight figure ROI, and scale AI.**

Snorkel Flow productionalizes over four years of research carried out at Stanford AI Lab. Snorkel Research Project was sponsored by Google, Intel, DARPA, and several other leading organizations and the research was represented in over 50 academic conferences such as ACL, NeurIPS, Nature and more.

Next, learn about Snorkel's core concepts, how it works, and how Snorkel Flow changes AI development workflow from a model-centric to a data-centric approach.

What Is Snorkel?

Is Snorkel an algorithm? An AI platform? A company? Let’s find out.

Snorkel is a machine learning approach that utilizes programmatic labeling and statistical modeling to build and iteratively improve training datasets. It is not limited to a specific algorithm or modeling technique, as many variants of each have been utilized in various iterations of Snorkel over the years.

Snorkel started as a research project at Stanford AI Lab in 2015, where Snorkel AI’s founding team set out to explore a higher-level interface to machine learning through training data. This project was sponsored by Google, Intel, DARPA, and several other leading organizations and the research has been represented in over fifty academic papers presented at conferences such as ACL, NeurIPS, Nature and more.

Snorkel Open Source Research Library was developed from 2015 to 2017 as a prototyping tool. It is a Python library that contains a legacy base class for defining code-based labeling functions (LFs) and some early algorithms for combining LF votes, rather than a comprehensive platform supporting the AI development lifecycle.

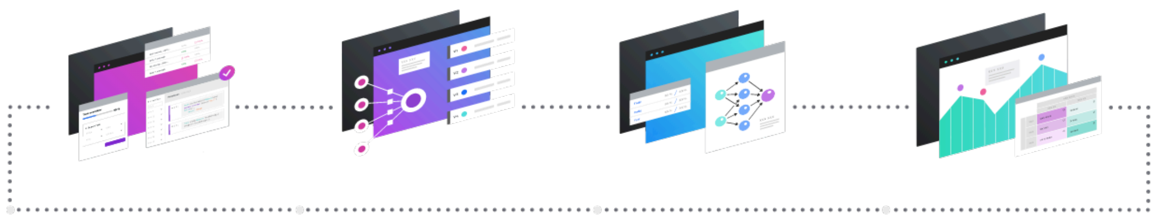
Snorkel AI was founded in 2019 by the original creators of Snorkel. The company’s mission is to unlock a better, faster, data-centric way to build AI applications.

Snorkel Flow is a one-of-its-kind data-centric platform for building AI applications, powered by the Snorkel approach to machine learning and incorporating years of experience by the Snorkel team in applying it to real-world problems. Today, it is used by numerous Fortune 500 companies to build AI accurate and adaptable applications fast.

With terminology out of the way, let’s dive into how Snorkel works! As a toy running example, we’ll pretend we’re building a binary classifier for identifying spam emails.

How does Snorkel work?

The Snorkel approach consists of four primary steps, which a user iterates through to systematically improve the quality of an ML model until it's ready to deploy:



01

Label & build
Label and build training data programmatically in hours without months of hand-labeling.

02

Integrate & manage
Automatically clean, integrate, and manage programmatic training data from all sources.

03

Train & deploy
Train and deploy state-of-the-art machine learning models in-platform or via Python SDK.

04

Analyze & monitor
Analyze and monitor model performance to rapidly identify and correct error modes in the data.

At first glance, this looks very much like a traditional ML pipeline. But you'll see as we walk through each step below in greater detail that:

1. When powered by Snorkel, each step has more flexibility and control than its counterpart in a traditional pipeline.
2. In a Snorkel application, it is much easier to iterate through these steps repeatedly, allowing for systematic improvement in hours or days instead of weeks or months.

Let's dive into each of the four stages of the Snorkel's approach next.

01. Label and build

With a legacy hand-labeling approach to ML, you may begin by labeling anywhere from thousands to millions of individual data points one-by-one. Some are easy to label, some are hard, and some may feel redundant because of how similar they are to ones you've already labeled, but it doesn't matter—you label them all, one-by-one. For most examples that you label, you *could* explain why you're labeling it that way. But there's nowhere to incorporate that reasoning in a legacy approach, so you toss it away and just give a label instead.

With Snorkel, you don't have to throw your rich domain knowledge away—instead, you capture it in the form of **labeling functions**.

Labeling Functions (LFs) are a core abstraction of the Snorkel approach to ML. An LF is simply an arbitrary function that takes in a data point and either outputs a label or abstains. Importantly, this interface assumes nothing about *how* your LF arrives at that label. In practice, this flexible definition allows for incorporating a huge variety of signals in your training data creation process (the "kitchen sink" approach)—essentially, if you can think of a programmatic way to label some subset of your data with a precision that's better than random, toss it in!

Running Example. As an example, imagine that you are trying to train an email spam detector for your company's email server. One approach for creating a training set would be to label tens or hundreds of thousands of individual emails by hand. Another approach would be to encode your domain knowledge in labeling functions. For example, you wouldn't have to look through many data samples to realize that certain words ("viagra", "wire transfer", etc.) are strongly correlated with spam. In contrast, others ("spreadsheet", "OKR", etc.) tend to occur in valid business-relevant emails. By converting these keywords into labeling functions, you can potentially label thousands of examples at once, based on your own domain knowledge, but applied now at a much greater scale.

Pattern Matching	If a phrase like "send money" is in a email
Boolean Search	If unknown_sender AND foreign_source
DB Lookup	If sender is in our Blacklist.db
Heuristics	If SpellChecker finds 3+ spelling errors
Legacy System	If LegacySytem votes spam
Third Party Model	If TweetSpamDetector votes spam
Crowd Labels	If Worker #23 votes spam

Labeling Functions (LFs) can come from a huge variety of sources, including writing new heuristics (rules, patterns, etc.) or wrapping existing knowledge resources (e.g., models, crowdworker labels, ontologies, etc.). Here are some simple pseudocode examples for the kinds of LFs that you could write for an email spam detector.

Frequently asked questions

- **How do I know what LFs to write?**

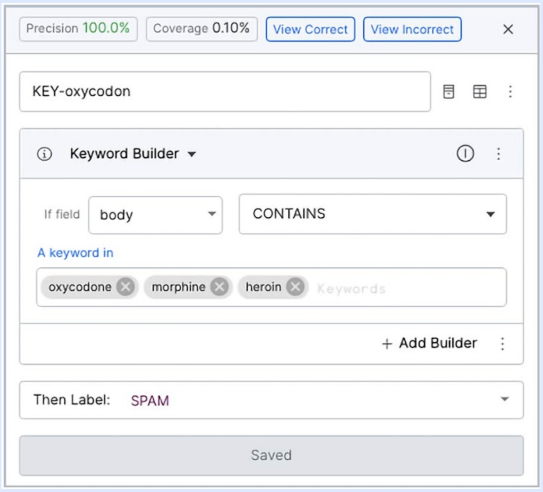
  You may be wondering: "That's great that I can create a wide variety of LFs—but how do I know which ones I should write?" The answer comes from focusing on your ultimate goal: you're not trying to create a training set—you're trying to train a high-quality ML model! So rather than trying to come up with the long list of LFs that you could write, in Snorkel Flow, you start by creating some minimal initial set of LFs (e.g., one LF per class) and then iterating. As you'll see in Step 4: Analyze, in Snorkel Flow, you get feedback on where your model is making mistakes and how that's correlated with your training data and LF outputs. As you look at those specific examples and think about how you as a human would label them, that can guide you to what type of LF you should write next, whether that's bringing in an existing resource or writing a new heuristic. If you have a small amount of labeled data in your training set, Snorkel Flow can also auto-suggest entire LFs or auto-tune thresholds for LFs whose structure you define.

- **Do my LFs need to have human-level precision?**

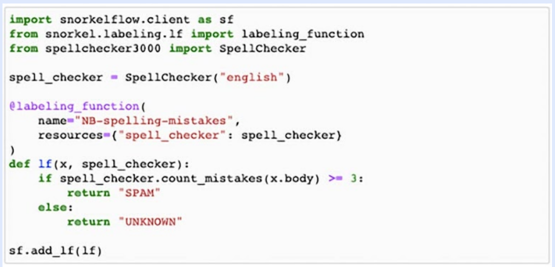
  Nope! It's called "weak supervision" for a reason—Snorkel expects "weak" labels. From a theoretical standpoint, with enough raw data to label, all Snorkel requires is that your LFs are better than random *on average*. But in practice, the more precise your LFs are (and the higher their coverage on your dataset), the quicker you'll get to a high-quality training set of sufficient size for your model, so Snorkel Flow color-codes LFs based on their precision and coverage to help you identify which ones are good to go, and which may need additional attention. Your LFs also don't need to cover the entire dataset—more coverage means a larger dataset, but we'll ultimately be training a classifier over a much more comprehensive feature set than just our relatively small set of LFs.

- **How should I express my LFs?**

  For many common types of LFs, Snorkel Flow includes a library of no-code templates where all you need to provide is the nugget of domain knowledge to complete it. For example, provide the specific keyword that you're looking for within the first three sentences of your document and toggle whether or not you want it to be case-sensitive, but let the template handle compiling that down into an optimized executable function. However, in some cases, you may want to express a very specific type of signal that doesn't have a corresponding template yet or which uses a closed source library that only you have access to—in that case, you can use the Python SDK to define a custom LF in the Snorkel Flow integrated notebook.



Precision 100.0%
Coverage 0.10%
KEY-oxycodon
Keyword Builder
If field: body
CONTAINS
A keyword in: oxycodone, morphine, heroin
Then Label: SPAM



```python
import snorkelflow.client as sf
from snorkel.labeling.lf import labeling_function
from spellchecker3000 import SpellChecker

spell_checker = SpellChecker("english")

@labeling_function(
    name="NB-spelling-mistakes",
    resources={"spell_checker": spell_checker}
)
def lf(x, spell_checker):
    if spell_checker.count_mistakes(x.body) >= 3:
        return "SPAM"
    else:
        return "UNKNOWN"

sf.add_lf(lf)
```

- **How is writing LFs different from building a rules-based classifier?**

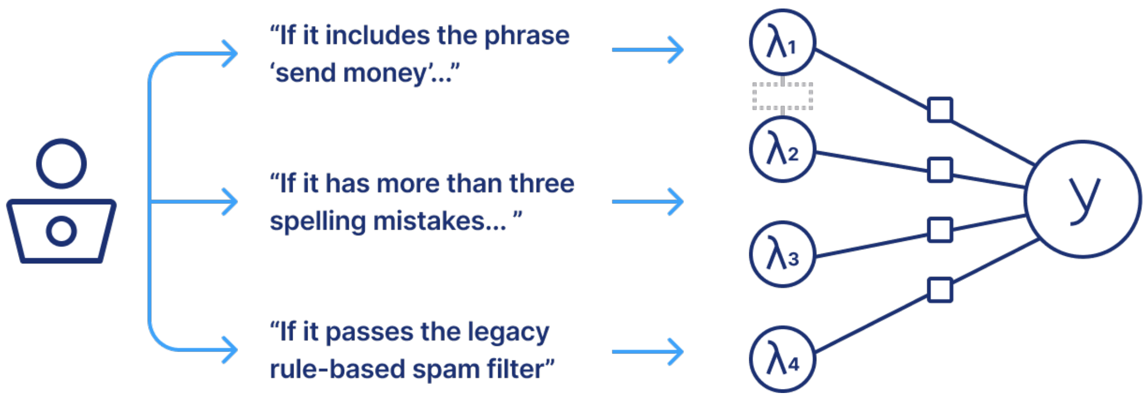
  Read on, my friend! We answer that precise question in Step 3: Train.

02. Integrate and manage

Once you've got a set of LFs, how do you turn these into training labels? As mentioned in the previous section, these LFs can be noisy—they may conflict with each other at times, make correlated errors, not cover all data points, cover certain classes better than others, etc.—and in most cases, we don't actually have the ground truth label for our training examples. But we can use theoretically grounded and empirically proven mathematical methods to identify optimal ways of combining these noisy supervision sources to create high-quality labels nonetheless, and now at a much greater scale and with much lower cost than doing so manually.

The problem we have to solve can be modeled as follows:

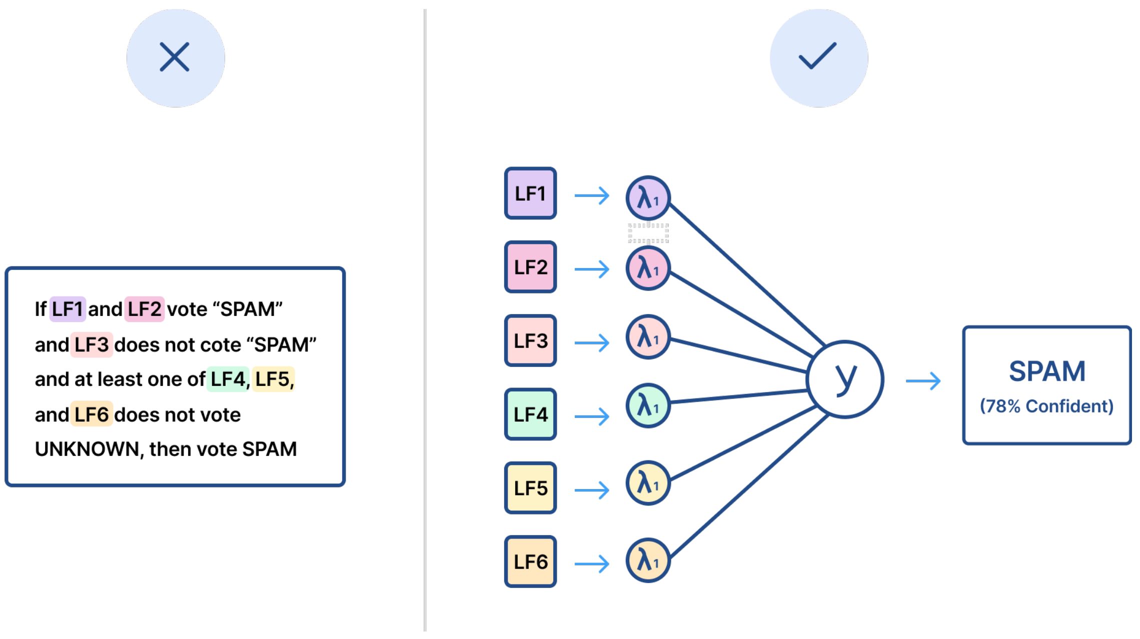
- Every data point has some "true" (hidden or latent) label Y—what the world's foremost expert on your problem would label it.
- We don't have that, but we do have a bunch of weak labels λ that, in general, are correlated with the true label.
- We now need a process for inferring the most likely true label for each data point, given all these noisy labels across our dataset.


"If it includes the phrase 'send money'..."
"If it has more than three spelling mistakes... "
"If it passes the legacy rule-based spam filter"
λ1
λ2
λ3
λ4
y

The model that we use to represent this problem is called a **label model**. It's important to note that there are many such models that we could use for this problem. To quote a famous statistician, "All models are wrong, but some are useful." Over the years, the Snorkel team has proposed different models and algorithms that make different assumptions and work best in different scenarios. Some of our early work is available in papers (e.g., NeurIPS 2016, ICML 2017, AAAI 2019, etc.), but none of these "is Snorkel" any more than the final model architecture we choose or the LF templates we use. They are each just different choices for a single component in the framework. Furthermore, we've never stopped innovating here—**Snorkel Flow includes the world's most comprehensive library of label model algorithms**, with all the above plus additional work from the past couple of years, including algorithmic, speed, and stability improvements.

After an appropriate label model has been selected and applied to our inputs—LF votes per data point, any ground truth labels that exist in the training set [2], priors about class balance, etc.—the model's output is a probabilistic label distribution per data point. In other words, for each data point in the training set, the model estimates how likely it is that its true label belongs to each class. We can then use these labels as-is for models that accept label distributions or keep only the maximum likelihood label for each. This becomes our training set.

Running Example. Returning to our spam detector mentioned above, many of our rules will have less than perfect accuracy. For example, an LF that looks for prescription drug names as a sign of spam email may vote incorrectly on emails from a valid customer that sells prescription drugs. On the other hand, an LF that looks at the historical response rate to emails sent from this sender may have high confidence that this is, in fact, an email worth responding to. The key is that we don't need to negotiate every area of conflict manually. Because the output of this step is a training label, not a final prediction, we can combine multiple LF votes (even conflicting ones) into a single confidence-weighted label per data point (e.g., I'm only 78% sure that this email is spam), and train on that.



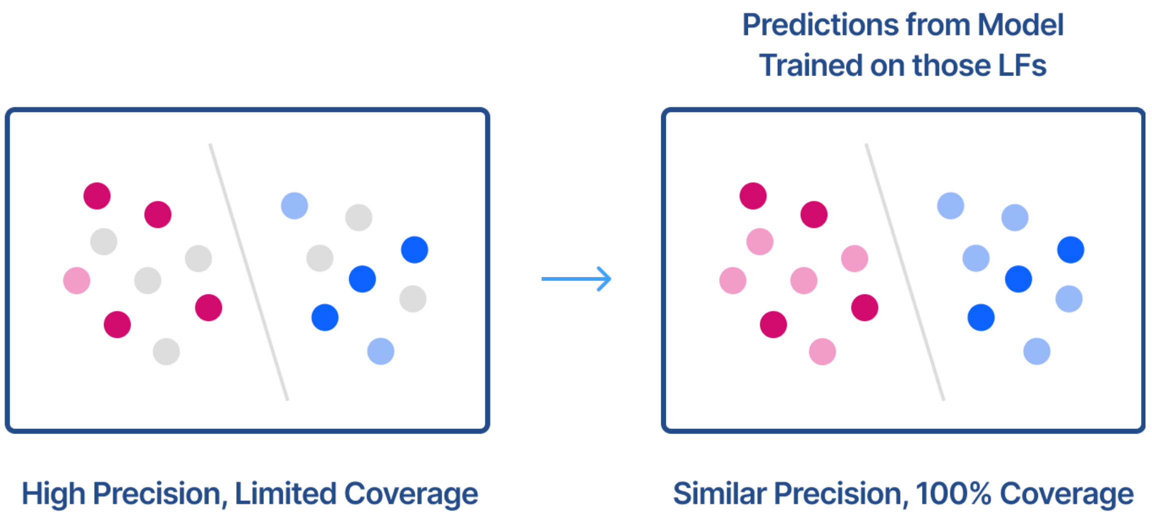
If LF1 and LF2 vote "SPAM" and LF3 does not cote "SPAM" and at least one of LF4, LF5, and LF6 does not vote UNKNOWN, then vote SPAM

Importantly, once a set of LFs has been developed, creating a new training set—even one with hundreds of thousands or millions of data points—now takes on the order of seconds or minutes, not person-weeks or person-months! As a result, the barrier to updating or adapting your training set to respond to emerging needs, data drift, etc., is significantly lowered, and you end up being able to iterate on your models' orders of magnitude more quickly than with legacy approaches driven by hand-labeling.

03. Train and deploy

With a large, freshly generated, domain-specific training dataset, the third step is to train a model appropriate for your data modality and task type. Here we recommend taking advantage of the fantastic progress that has been made in recent years in accessible, user-friendly, open-source model libraries. For example, in Snorkel Flow, we provide a model training interface compatible with Scikit-Learn, XGBoost, Transformers, and Flair, to name a few. You can also export training labels to train a custom model offline and then upload the predictions via the SDK.

The natural question to ask at this point is: Why do we need a model if we already have a bunch of LFs ("rules") capable of creating training labels (and therefore predictions)? There are a few good reasons for this:


Predictions from Model Trained on those LFs
High Precision, Limited Coverage
Similar Precision, 100% Coverage

While your LFs may only label a subset of the data points on the left in this toy 2D problem, a classifier trained on those data points can learn a "smoother" decision boundary that correctly classifies similar data points nearby with no LF labels.

1. **Generalization**: Rules are nice—they're direct and interpretable. However, they also tend to be brittle. On the other hand, machine learning models tend to do a much better job of dealing with minor variations between data points. For example, imagine two emails identical in every way except for one word being swapped out for a synonym. A rule-based system looking for that keyword may fail on one and not the other, while an ML model will almost certainly classify the two similarly. This disparity is because the model can utilize a more comprehensive feature set. In other words, rather than depending on one keyword to make its decision, the model can factor in the presence (or absence) of thousands of keywords all at once.

2. **Non-Servable Features:** In a rule-based classifier, all rules must be able to be executed at inference time. With Snorkel, on the other hand, it is possible to create your training set using features that won't be available at test time—we call these types of features "non-servable." For example, you can use the historical response rate to emails from certain senders to add weak labels to thousands of emails in your dataset for which you have that information, and then train a model only on the content of the email (all "servable" features) so that it can make predictions on new incoming emails as they arrive.

3. **Transfer Learning**: One reason machine learning (and more specifically, deep learning) has taken off in recent years is the evolution of representation learning, or learning rich features for your data automatically *from the data* rather than feature engineering by hand. Pre-trained embeddings and models like the ones available in Snorkel Flow bring rich background information to the table—e.g., for NLP tasks, synonymous words may share no similarities in their surface forms, while having nearly identical representations in a model. Utilizing these techniques can significantly improve the ability of a model to generalize to unseen examples.

04. Analyze and monitor

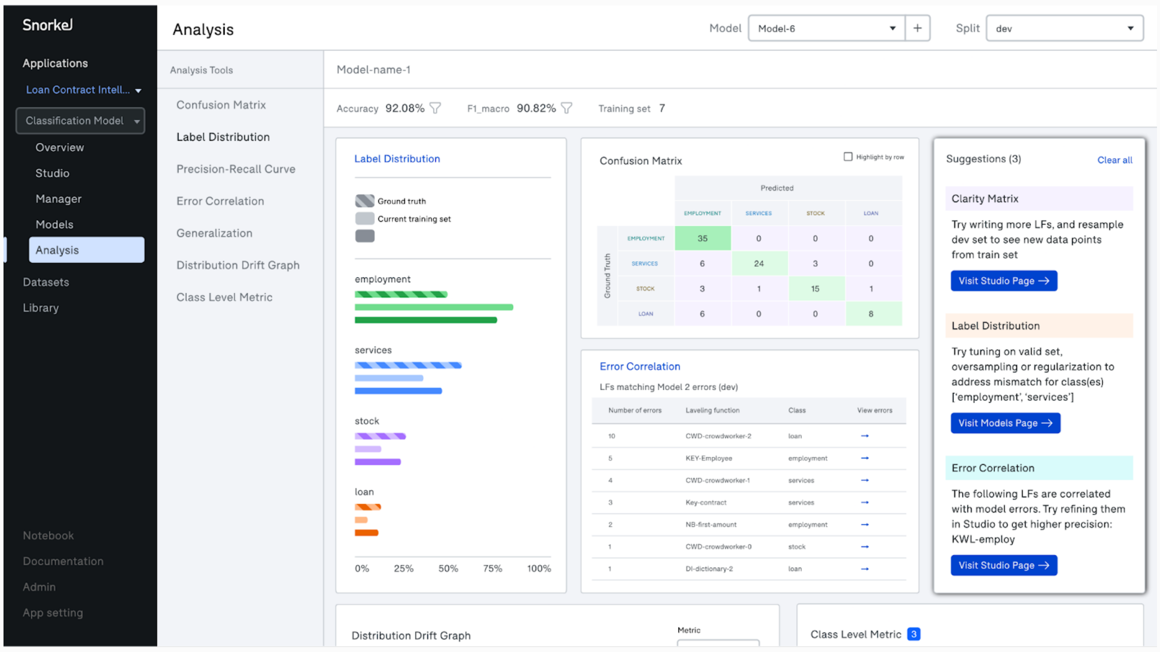
Once an ML model has been trained, what are your options? First, you'll want to evaluate its performance with metrics appropriate for your problem (e.g., accuracy, F1, etc.). If its quality is sufficient, you can deploy it as part of an AI application and begin to monitor its performance over time (more on that in the next section). On the other hand, if the quality is insufficient (which it nearly always is on your first try), **you begin a targeted, iterative improvement process, improving the model by improving the data**.

What do we mean by targeted improvement? When a model predicts the wrong label for an example, there are multiple possibilities why:

1. There's an issue with the model. For example, you may have a model with insufficient learning capacity, or your regularization parameter may be set too high.
2. There's an issue with the data. For example, you may have too many incorrect training labels, or there may be "blind spots" in your dataset, where examples in your test set have no similar examples in your train set.

In our experience, too many practitioners assume (implicitly or explicitly) that their dataset is complete, correct, and untouchable, so they spend all their time iterating on the model. **In reality, most model mistakes are the fault of the data, not the model architecture or training routine!** And if the issue is in the data, then that's where you should spend your time, rather than blindly hyperparameter tuning (hoping to land in a better local optimum somewhere) or collecting more data in general with no attention to where the issues actually lie within it.

Running Example. Returning to our spam detector example, our training set may include many labeled examples of spam emails trying to sell us prescription drugs or get us to wire them money. It may not have any labeled examples of social engineering attacks, where the sender is pretending to be someone we know in order to get us to share sensitive information. If this new type of attack doesn't have enough in common with the other types of spam emails where we have more examples, our model may perform poorly on most of them—not because we need a different model or more data in general, but because we have a blind spot. One new LF that covers even just a small portion of this new class of errors could easily be enough signal to squash that whole bucket, and the analysis page can guide us right to those examples.



Model-name-1

Accuracy 92.08% F1_macro 90.82% Training set 7

Label Distribution

Confusion Matrix

		Predicted			
		EMPLOYMENT	SERVICES	STOCK	LOAN
Ground Truth	EMPLOYMENT	35	0	0	0
	SERVICES	6	24	3	0
	STOCK	3	1	15	1
	LOAN	6	0	0	8

Error Correlation

LFs matching Model 2 errors (dev)

Number of errors	Laveling function	Class	View errors
10	CWD-crowdworker-2	loan	→
5	KEY-Employee	employment	→
4	CWD-crowdworker-1	services	→
3	Key-contract	services	→
2	NB-first-amount	employment	→
1	CWD-crowdworker-0	stock	→
1	DI-dictionary-2	loan	→

Suggestions (3) Clear all

Clarity Matrix

Try writing more LFs, and resample dev set to see new data points from train set

Visit Studio Page →

Label Distribution

Try tuning on valid set, oversampling or regularization to address mismatch for class(es) ['employment', 'services']

Visit Models Page →

Error Correlation

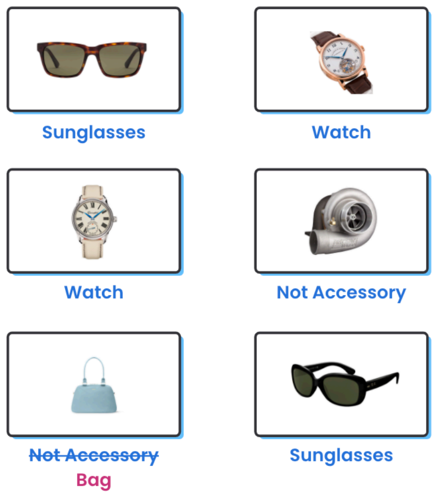
The following LFs are correlated with model errors. Try refining them in Studio to get higher precision: KWL-employ

Visit Studio Page →

Importantly, with the Snorkel approach, you can update a large training set and retrain a powerful model in minutes, in one sitting, by one individual or small team, rather than over weeks or months of coordination for large manual labeling jobs. The result is a more rapid, iterative, and adaptive process for creating and maintaining high-quality AI applications.

Case study: Google

Google built & maintained hundreds models for content classification with Snorkel



Challenge

The Google teams oversaw 100s of content classifiers, each with its training data set. Developing a new classifier required significant manual data labeling effort. The team also struggled to adapt classifiers when business decisions necessitated modifying an existing application.

Google teams spent a considerable amount of time and resources labeling and relabeling thousands of data points manually for months for each classifier.

Solution

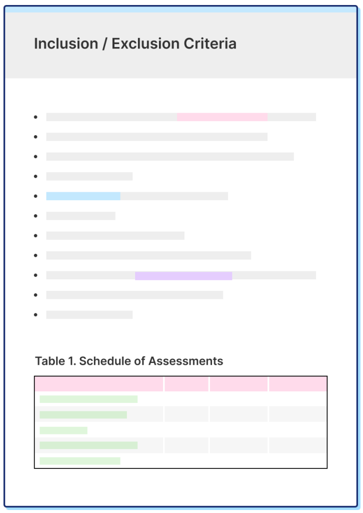
Google developed topic and product—content classifiers using Snorkel. They wrote labeling functions that expressed heuristics (based on linked URL, topics, entities) and used organizational resources such as existing semantic topic models and knowledge graphs. **These labeling functions labeled 684,000 data points in a few minutes for topic classification and 6.5 million data points for product classification in 30 minutes.**

With Snorkel, the Google team built classifiers of comparable quality to ones trained with tens of thousands of hand-labeled examples. They converted non-servable organizational resources to servable models for **an average 52% performance improvement while generating millions of data points in tens of minutes.**

6 Months	52%	6.5M
of hand-labeling data replaced in 30 mins	performance improvement	Programmatically labeled data points

Case study: Genentech

Genentech extracted & structured information from clinical trial PDFs with Snorkel Flow


Inclusion / Exclusion Criteria
Table 1. Schedule of Assessments

Challenge

Genentech needed to extract valuable information from Clinical Trial Protocol PDFs. They had tried multiple approaches, including domain adaptation, that it did not yield meaningful results.

Instead they wanted to focus on iterating and augmenting the training data. **But it would take 140 subject matter experts a month to label their training set.**

Solution

Using Snorkel Flow, Genentech built applications with NER, entity linking, and classification models to accurately determine inclusion/exclusion criteria and the Schedule of Assessment analysis from clinical trial PDFs. **The output data was used to harmonize clinical trial protocols terminology across the organization.**

Genentech estimates that AI applications built using Snorkel Flow will increase diverse populations' recruitment and reduce trial times, costs, and patient dropout rates. This in turn will **dramatically reduce the drug development costs** and increase the number of drugs in the pipeline leading to better evaluation of treatments, more cures, and treatments for society

1 day	99%	350K
Time to change label schema and relabel all training data	Accuracy achieved for NER + entity linking	Programmatically labeled clinical trial PDFs

Accelerate AI with Snorkel Flow

It's clear that adopting Snorkel's data-centric approach can be instrumental to not only automating the labeling process while keeping human-in-the-loop but also to accelerating AI development.

But where do you start? Rather than dealing with annotation guides and contracts for crowdsourced labeling, put Snorkel Flow to use.

With Snorkel Flow, Fortune 500 organizations such as Chubb, BNY Mellon, Genentech, and more have built accurate and adaptable AI applications fast by putting the power of programmatic labeling to use.

Beyond realizing Snorkel's data-centric approach, Snorkel Flow has many other features that elevate it from a training set or model creation tool to a complete enterprise platform for building AI applications. Some of those features include:

- **Application studio:** Support for multiple data modalities (unstructured text, structured text, time series, etc.), problem types (NER, info extraction, multi-label classification, etc.), and custom application graphs combining multiple models, pre-/post-processors, and custom business logic
- **Programmatic labeling IDE**: no-code LF templates, auto-suggest and auto-tuning of LFs, interactive data visualization, automated management and versioning of LFs, etc.
- **Training data management**: advanced label model algorithms, automated parallelization, and hyperparameter selection, dataset versioning and comparisons, etc.
- **Model training and serving**: one-click model training or fine-tuning, hyperparameter search, endpoint creation for mode/application serving, export for serving at scale, etc.
- **Deployment and security**: REST API, monitoring services, and managed workers for job execution; encryption, authentication, and role-based access control (RBAC), managed SSO integration; support for hosted, hybrid, or on-premises deployments, etc.

To see Snorkel Flow in action: Request a demo.

To stay in touch, follow us on Twitter, LinkedIn, Youtube.