Snorkel: Fast Training Set Generation for Information Extraction

Alexander J. Ratner, Stephen H. Bach, Henry R. Ehrenberg, Chris Ré
Computer Science Department, Stanford University
{ajratner, bach, henryre, chrismre}@cs.stanford.edu

ABSTRACT

State-of-the art machine learning methods such as deep learning rely on large sets of hand-labeled *training data*. Collecting training data is prohibitively slow and expensive, especially when technical domain expertise is required; even the largest technology companies struggle with this challenge[1]. We address this critical bottleneck with Snorkel[2], a new system for quickly creating, managing, and modeling training sets. Snorkel enables users to generate large volumes of training data by writing *labeling functions*, which are simple functions that express heuristics and other weak supervision strategies. These user-authored labeling functions may have low accuracies and may overlap and conflict, but Snorkel automatically learns their accuracies and synthesizes their output labels. Experiments and theory [3, 4] show that surprisingly, by modeling the labeling process in this way, we can train high-accuracy machine learning models even using potentially lower-accuracy inputs. Snorkel is currently used in production at top technology and consulting companies, and used by researchers to extract information from electronic health records, after-action combat reports, and the scientific literature. In this demonstration, we focus on the challenging task of information extraction, a common application of Snorkel in practice. Using the task of extracting corporate employment relationships from news articles, we will demonstrate and build intuition for a radically different way of developing machine learning systems which allows us to effectively bypass the bottleneck of hand-labeling training data.

1. INTRODUCTION

Traditionally, high-performance machine learning methods require custom *features* and hand-labeled *training data* for a given domain and task. In the last several years, *deep learning* models have removed much of the burden of manually engineering features, at the cost of requiring even larger labeled training data sets. For example, to apply a deep learning model to the problem of extracting mentions of company-employee relationships in news articles, we would need to label potentially tens of thousands of example mentions by hand. This labeling process is even slower and more expensive when special domain expertise is required. Users across the spectrum face this challenge. For example, industry and scientific consortia have spent tens of person-years setting annotation guidelines and labeling training data [2], and from our work with major consulting companies, we find that a range of companies of different sizes all struggle to collect and manage training data. Furthermore, once training datasets are created, real-world applications often evolve, necessitating expensive re-labeling of data.

In Snorkel, we seek to radically accelerate the process of training and deploying new machine learning systems by avoiding both traditional feature engineering and manual training data labeling. Instead, developers only write *labeling functions* which programmatically, but noisily, label large volumes of training data. These labeling functions can express heuristics, rules, and other popular strategies for *weak supervision* such as distant supervision, crowdsourcing, and ensembling of less accurate classifiers. This provides a unifying framework for approaches which allow us to generate training data in a variety of settings and with varying levels of supervision resources available. The resulting labeling functions will have unknown accuracies, and may overlap and conflict with each other. Snorkel automatically learns each labeling function's reliability, synthesizes their output labels, and uses the final denoised labels to train an end machine learning model.

Snorkel's workflow is fundamentally different from traditional machine learning approaches, and is based on the newly proposed data programming paradigm [4]. Rather than viewing training data as a perfectly correct input provided *a priori*, we view the labeling of training data as a stochastic process that we can model. Surprisingly, by learning this model, we can use potentially low-accuracy labeling functions to train high-accuracy end models. Data programming is already used to train models over text, images, and semi-structured data in a variety of application domains. For example, systems built with data programming are used to mine electronic health records in order to analyze critical factors in post-surgical outcomes, assist law enforcement efforts to fight human trafficking by extracting information from the dark web, and analyze after-action combat reports

[1] https://www.wired.com/2016/11/googles-search-engine-can-now-answer-questions-human-help
[2] http://snorkel.stanford.edu

SIGMOD'17, May 14-19, 2017, Chicago, IL, USA
 ISBN 978-1-4503-4197-4/17/05…$15.00
DOI: http://dx.doi.org/10.1145/3035918.3056442

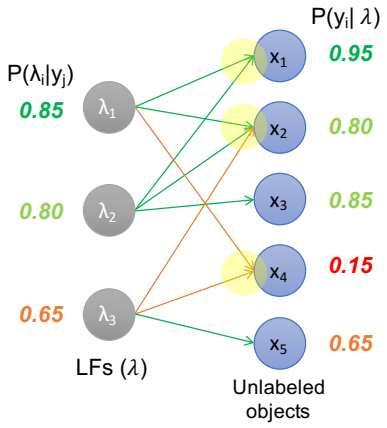

Figure 1: In data programming, we learn the accuracies of the labeling functions (LFs) by observing their agreements and disagreements with each other; we then use the predictions of this generative model as *noise-aware* training labels.

to study military conflicts.

In this demonstration, we will highlight the use of Snorkel to power the rapid development of new structured information extraction applications, an increasingly critical part of many analysis pipelines. Specifically, we will consider the example of extracting corporate employee relationships–e.g. `Employs(CompanyA, PersonB)`–from news articles, and walk attendees through the process of writing labeling functions and modeling their accuracies in Snorkel. We will also be able to discuss a wide range of experiences with this fundamentally new approach to building machine learning systems, based on Snorkel deployments and hackathons at various technology and consulting companies, government agencies, and research labs.

2. DATA PROGRAMMING

Data programming [4] is a new machine learning paradigm for quickly and cheaply generating training sets for machine learning models. In data programming, instead of hand-labeling training data, the user writes *labeling functions*, which are just functions that label some subset of the available data, may have arbitrary unknown accuracies, and may overlap and conflict with each other.

The process of data programming is divided into two stages: (1) modeling the outputs of the labeling functions with a generative model, and (2) training an end model with the estimated labels.

Modeling the Labeling Functions. For concreteness, we consider the binary classification setting, as is relevant to our information extraction task. Our input is thus a set of candidate extractions $x \in \mathcal{X}$, with unknown true labels $y \in \{-1, 1\}$, and a set of m labeling functions $\lambda_i : \mathcal{X} \mapsto \{-1, 0, 1\}$ provided by the user. In data programming, these labeling functions are treated as implicitly specifying a generative model of the training set labeling process, $\mu_\alpha(\lambda, y)$, where α is the vector of the accuracies of the labeling functions that we estimate using maximum likelihood estimation (see Fig. 1).

Training the End Model . Given the estimated generative model, we produce a set of *noise-aware* training labels, which are just the predictions of the generative model $P_{\mu_\alpha}(y|\lambda)$. We then train a *noise aware* version of our end classification model by minimizing the expected loss with respect to these noisy training labels, i.e. we solve

$$\hat{\theta} = \text{argmin}_\theta \sum_x \mathbb{E}_{\mu_\alpha} \left[l(x, y; \theta) \mid \lambda \right]$$

where l is a loss function such as the logistic loss or a more complex loss, such as that of an LSTM or convolutional neural network.

One of the most exciting results around data programming is that given enough labeling functions of high enough average quality, the end performance of the model we are ultimately training scales with respect to the amount of *unlabeled* data used at the same asymptotic rate as in the fully-supervised case. Further details can be found in [4].

3. SYSTEM OVERVIEW

We describe the information extraction workflow in Snorkel, as shown in Fig. 2, using the task of extracting company employee relationship mentions from news articles as a running example.

3.1 Preprocessing

Snorkel provides an automated data preprocessing framework for the user, built around a flexible data model which automatically represents input data as a hierarchy of unstructured *contexts*. For example, news articles would be segmented into headline and body sections, and the body segmented into paragraphs and sentences.

As in most standard information extraction approaches, we then extract *candidate* relations, which reduces our problem to binary classification over candidates. In our company-employee relation problem, for example, we might consider all potential pairs of a company and a person mentioned in the same sentence as candidate company-employee relation mentions; our task is then to train a model which classifies each one as a true mention of this type or not. In Snorkel, we represent candidates as tuples of pointers to elements of the context hierarchy, a clean representation which enables easy user interaction when writing labeling functions. For example, a candidate relation `Employs(CompanyA, PersonB)` would be represented as a pair of pointers to the mentions of `CompanyA` and `PersonB`.

3.2 Writing Labeling Functions

The primary user interaction with Snorkel is the iterative development of labeling functions. In Snorkel, labeling functions are simply Python functions which take a candidate as input, and output a `True`, `False` or `None` label. Labeling functions are able to express and subsume a wide range of approaches commonly used in practice to generate noisy training data, including:

- **Heuristic pattern matching:** Labeling functions in Snorkel can express any pattern matching rule (e.g. regular expressions), or more complex heuristics, even utilizing external libraries. For example we might look for the phrase "works for" between two noun phrases in a sentence, or reference an external dictionary.

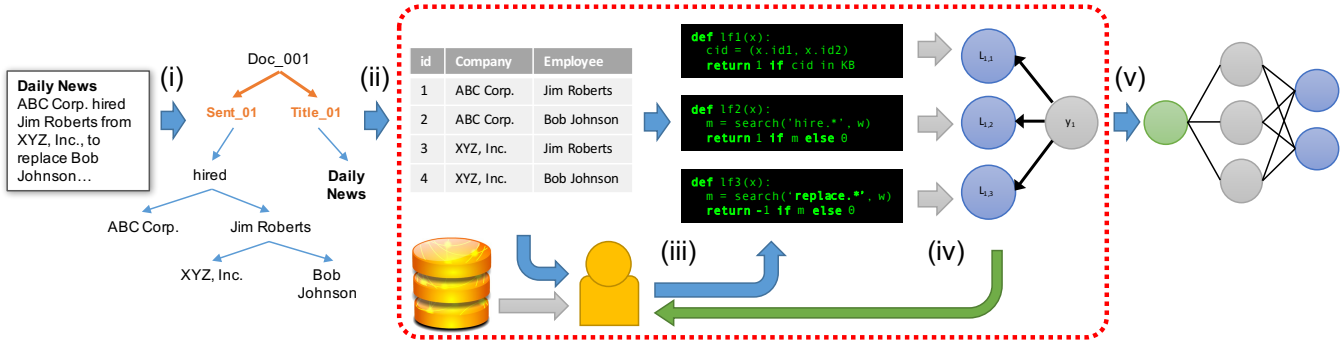

Figure 2: The information extraction workflow in Snorkel: (i) An unstructured input document is parsed into a *context hierarchy*, including a grammatical dependency parse of the sentences; (ii) *candidates* are extracted; (iii) the user inspects sample candidates, and writes *labeling functions*; these labeling functions may potentially integrate external sources such as existing knowledge bases, crowd labels or weak classifiers; (iv) the accuracies of these labeling functions is automatically modeled, which may also provide useful feedback to the user; (v) finally, an end extraction model is *trained*. In the demonstration, our primary focus will be on the iterative labeling function development process and environment in steps (iii) and (iv).

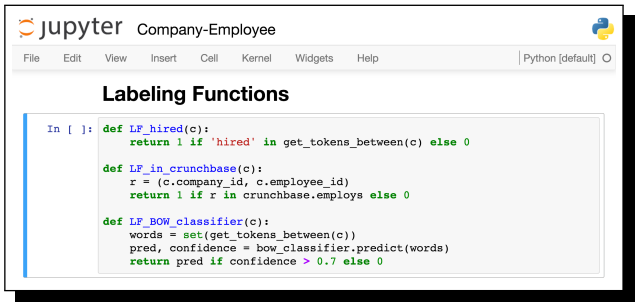

Figure 3: Labeling functions which express pattern-matching, distant supervision, and weak classifier heuristics, respectively, in Snorkel's Jupyter notebook interface.

- **Distant supervision:** Labeling functions can easily allow users to leverage external knowledge bases and other data resources to label data points. For example, we can label as true any mention of a company and person that co-occur in a sentence, and that are known to have a relationship based on some external knowledge base (such as Crunchbase[3]).
- **Ensembling of weak classifiers:** In some scenarios, we may have one or more pretrained weak classifiers, which may be biased or low-quality. These can be incorporated as labeling functions in Snorkel.

Figure 3 shows examples of these approaches. In our experience working on information extraction tasks, we have found that developers can have access to varying levels of these supervision *resources*, which can all be handled by Snorkel; for example:

- **Low resource:** A user may have no external knowledgebases or other such resources, and can instead rely on Snorkel's notebook-based interface and Viewer utility to rapidly generate pattern-based labeling functions, which our end extraction model will learn to generalize.
- **Medium resource:** A user might have access to various resources such as external knowledge bases, weak classifiers, or unreliable crowd labels that can be integrated as labeling functions.
- **High resource:** A user might additionally have access to labeled training data, which can also be easily integrated into the Snorkel framework.

In the demonstration, we will give examples to illustrate labeling function development in a range of these regimes.

Snorkel makes it easy for users to quickly iterate through the typical label function development cycle. The construction of an initial set of labeling functions is accelerated by the object-relational mapping (ORM) layer, implemented in SQLAlchemy [4]. Users can write labeling functions that traverse the hierarchy of input data (see Fig. 2) to access information without writing any SQL. Data exploration over the data model is simplified by the Snorkel's Jupyter notebook front-end.

Snorkel also computes a range of performance evaluation metrics for labeling function sets. These metrics measure key labeling function attributes such as empirical accuracy on any available labeled data, degrees of overlaps and conflicts, and distributions of labels.

Users can also use the output of the generative model over the labeling functions (as described in Section 3.3) to improve the labeling function set. When combined with Snorkel's utilities for viewing example candidates, this empowers users to quickly identify error modes (see Fig. 4). For example, this could enable a user to notice that customers or clients are being incorrectly labeled as employees; they could then write a labeling function to correct this. Using all of these feedback signals, users can rapidly refine and debug their existing labeling functions, add new ones, or remove unhelpful ones [3].

3.3 Automated Modeling and Training

In the *modeling* phase, Snorkel automatically learns a generative model of the labeling process (see Sec. 2), recovering the accuracy parameters of the user-authored labeling

[3]https://www.crunchbase.com

[4]http://www.sqlalchemy.org

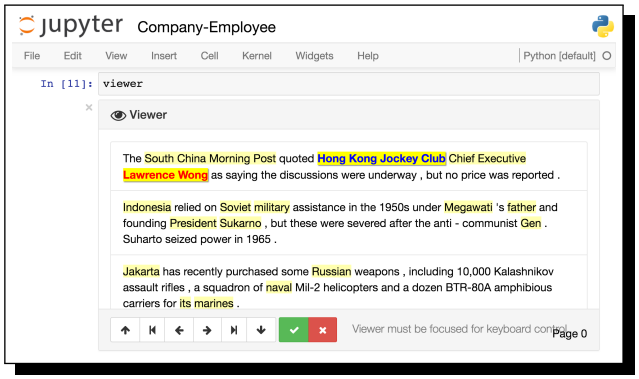

Figure 4: The Viewer utility in Snorkel, showing candidate company-employee relation mentions, comprised of candidate person and company mention pairs.

functions based on their overlapping output. In Snorkel, we can also automatically learn dependencies that may exist between the labeling functions, further improving the accuracy of our model [1].

Snorkel then uses the predictions of this generative model as *noise-aware* labels to train any generic machine learning model as an end extractor. By default, Snorkel has plugins for TensorFlow[5], which are adapted to take advantage of Snorkel's flexible context hierarchy. For example, a document having sections, subsections, and paragraphs with headers would be parsed into a hierarchy of these constituent parts, and this would trigger a specific way that this data was then passed on to the end extraction model. The context hierarchy helps to automate this representation step and speeds up the featurization and training processes for an end machine learning model, which could be one of Snorkel's default models or one implemented by the user.

4. DEMONSTRATION OVERVIEW

We will interactively walk attendees through the process of building an information extraction application with Snorkel, focusing on the labeling function development cycle (see Fig. 2). The demonstration will center on the representative task of extracting mentions of people being employed by companies from real world news articles. The demonstration will be completely interactive, allowing users to experiment with making changes to any stage of the pipeline, in particular adding, modifying, and removing labeling functions, and then rerunning modeling and training to see the results.

Using our past experience hosting Snorkel-based hackathons, we designed the demonstration to showcase critical user interaction points. The preprocessing steps will be completed ahead of time (but can be re-run for interested attendees), and we will walk through the following stages:

1. **Data exploration:** Starting with a random sample of candidate extractions of type `Employs(CompanyA, PersonB)`, we will use the Snorkel Viewer utility to (a) get a sense of the difficulty of the problem (for attendees new to information extraction) and (b) develop ideas for pattern- and heuristic-based labeling functions.

2. **Writing labeling functions:** Attendees will write a few simple labeling functions, using helper methods to accelerate this process. We will also have several prewritten examples for inspiration and for jumpstarting application performance. Most labeling functions are simple Python functions in a Jupyter notebook, so it will be easy for attendees to modify and write their own labeling functions.

3. **Advanced labeling function examples:** We will show several examples of labeling functions which leverage popular weak supervision strategies, such as distant supervision, crowdsourced labels, and weak classifiers.

4. **Modeling and debugging:** We will demonstrate the modeling phase, and how to (a) view the resulting model over the labeling functions, and (b) use this to help debug and develop new labeling functions. Since the entire demonstration will be a sequence of turnkey notebooks, it will be very easy for users to test the effects of their work in real-time.

Our demonstration will give attendees a better "engineer's intuition" for data programming by allowing them to experiment with modifications to the workflow, such as modeling dependencies among the labeling functions, learning without modeling the uncertainty in the training data, and experimenting with different end models.

We will also be able to demonstrate the end model training phase for interested attendees, as well as discuss other larger-scale Snorkel applications from pilot deployments at technology and consulting companies, government agencies, and in research projects involving electronic health records, post-action combat reports, and the scientific literature.

5. CONCLUSION

Snorkel offers users a new way to develop information extraction systems, avoiding the most troublesome bottlenecks of hand-labeling training data and feature engineering. The demonstration will focus on the most novel aspects of the data programming experience, and show attendees how it fundamentally changes the development of machine-learning-based data systems.

6. REFERENCES

[1] S. H. Bach, B. He, A. Ratner, and C. Ré. Learning the structure of generative models without labeled data. *arXiv preprint arXiv:1703.00854*, 2017.

[2] A. P. Davis et al. A CTD–Pfizer collaboration: Manual curation of 88,000 scientific articles text mined for drug–disease and drug–phenotype interactions. *Database*, 2013.

[3] H. R. Ehrenberg, J. Shin, A. J. Ratner, J. A. Fries, and C. Ré. Data programming with DDLite: Putting humans in a different part of the loop. In *HILDA@ SIGMOD*, 2016.

[4] A. Ratner, C. De Sa, S. Wu, D. Selsam, and C. Ré. Data programming: Creating large training sets, quickly. In *Neural Information Processing Systems (NIPS)*, 2016.

[5] http://hazyresearch.github.io/snorkel/blog/dp_with_tf_blog_post.html