arXiv:1711.10160v1 [cs.LG] 28 Nov 2017

Snorkel: Rapid Training Data Creation with Weak Supervision

Alexander Ratner Stephen H. Bach Henry Ehrenberg
Jason Fries Sen Wu Christopher Ré
Stanford University
Stanford, CA, USA
{ajratner, bach, henryre, jfries, senwu, chrismre}@cs.stanford.edu

ABSTRACT

Labeling training data is increasingly the largest bottleneck in deploying machine learning systems. We present Snorkel, a first-of-its-kind system that enables users to train state-of-the-art models without hand labeling any training data. Instead, users write labeling functions that express arbitrary heuristics, which can have unknown accuracies and correlations. Snorkel denoises their outputs without access to ground truth by incorporating the first end-to-end implementation of our recently proposed machine learning paradigm, data programming. We present a flexible interface layer for writing labeling functions based on our experience over the past year collaborating with companies, agencies, and research labs. In a user study, subject matter experts build models 2.8× faster and increase predictive performance an average 45.5% versus seven hours of hand labeling. We study the modeling tradeoffs in this new setting and propose an optimizer for automating tradeoff decisions that gives up to 1.8× speedup per pipeline execution. In two collaborations, with the U.S. Department of Veterans Affairs and the U.S. Food and Drug Administration, and on four open-source text and image data sets representative of other deployments, Snorkel provides 132% average improvements to predictive performance over prior heuristic approaches and comes within an average 3.60% of the predictive performance of large hand-curated training sets.

PVLDB Reference Format:
A. Ratner, S. H. Bach, H. Ehrenberg, J. Fries, S. Wu, C. Ré. Snorkel: Rapid Training Data Creation with Weak Supervision. *PVLDB*, 11 (3): xxxx-yyyy, 2017.
DOI: 10.14778/3157794.3157797

1. INTRODUCTION

In the last several years, there has been an explosion of interest in machine-learning-based systems across industry, government, and academia, with an estimated spend this year of $12.5 billion [1]. A central driver has been the advent of *deep learning* techniques, which can learn task-specific representations of input data, obviating what used to be the most time-consuming development task: feature engineering. These learned representations are particularly effective for tasks like natural language processing and image analysis, which have high-dimensional, high-variance input that is impossible to fully capture with simple rules or hand-engineered features [14, 17]. However, deep learning has a major upfront cost: these methods need massive *training sets* of labeled examples to learn from—often tens of thousands to millions to reach peak predictive performance [47].

Such training sets are enormously expensive to create, especially when domain expertise is required. For example, reading scientific papers, analyzing intelligence data, and interpreting medical images all require labeling by trained *subject matter experts* (SMEs). Moreover, we observe from our engagements with collaborators like research labs and major technology companies that modeling goals such as class definitions or granularity change as projects progress, necessitating re-labeling. Some big companies are able to absorb this cost, hiring large teams to label training data [12,16,31]. However, the bulk of practitioners are increasingly turning to *weak supervision*: cheaper sources of labels that are noisier or heuristic. The most popular form is *distant supervision*, in which the records of an external knowledge base are heuristically aligned with data points to produce noisy labels [4, 7, 32]. Other forms include crowdsourced labels [37, 50], rules and heuristics for labeling data [39, 52], and others [29, 30, 30, 46, 51]. While these sources are inexpensive, they often have limited accuracy and coverage.

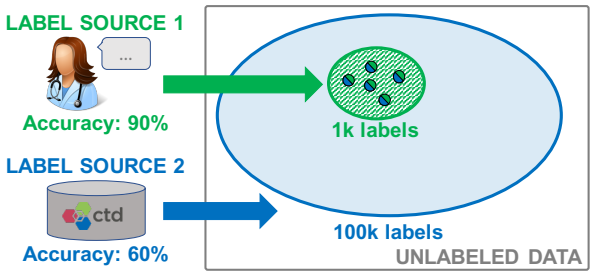

Figure 1: In Example 1.1, training data is labeled by sources of differing accuracy and coverage. Two key challenges arise in using this weak supervision effectively. First, we need a way to estimate the unknown source accuracies to resolve disagreements. Second, we need to pass on this critical lineage information to the end model being trained.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Articles from this volume were invited to present their results at The 44th International Conference on Very Large Data Bases, August 2018, Rio de Janeiro, Brazil.
Proceedings of the VLDB Endowment, Vol. 11, No. 3
Copyright 2017 VLDB Endowment 2150-8097/17/11... $ 10.00.
DOI: 10.14778/3157794.3157797

Ideally, we would combine the labels from many weak supervision sources to increase the accuracy and coverage of our training set. However, two key challenges arise in doing so effectively. First, sources will overlap and conflict, and to resolve their conflicts we need to estimate their accuracies and correlation structure, *without* access to ground truth. Second, we need to pass on critical lineage information about label quality to the end model being trained.

EXAMPLE 1.1. *In Figure 1, we obtain labels from a high accuracy, low coverage Source 1, and from a low accuracy, high coverage Source 2, which overlap and disagree (split-color points). If we take an unweighted majority vote to resolve conflicts, we end up with null (tie-vote) labels. If we could correctly estimate the source accuracies, we would resolve conflicts in the direction of Source 1.*

We would still need to pass this information on to the end model being trained. Suppose that we took labels from Source 1 where available, and otherwise took labels from Source 2. Then, the expected training set accuracy would be 60.3%*—only marginally better than the weaker source. Instead we should represent training label lineage in end model training, weighting labels generated by high-accuracy sources more.*

In recent work, we developed *data programming* as a paradigm for addressing both of these challenges by modeling multiple label sources without access to ground truth, and generating *probabilistic* training labels representing the lineage of the individual labels. We prove that, surprisingly, we can recover source accuracy and correlation structure without hand-labeled training data [5, 38]. However, there are many practical aspects of implementing and applying this abstraction that have not been previously considered.

We present *Snorkel*, the first end-to-end system for combining weak supervision sources to rapidly create training data. We built Snorkel as a prototype to study how people could use data programming, a fundamentally new approach to building machine learning applications. Through weekly hackathons and office hours held at Stanford University over the past year, we have interacted with a growing user community around Snorkel's open source implementation.[1] We have observed SMEs in industry, science, and government deploying Snorkel for knowledge base construction, image analysis, bioinformatics, fraud detection, and more. From this experience, we have distilled three principles that have shaped Snorkel's design:

1. **Bring All Sources to Bear:** The system should enable users to opportunistically use labels from all available weak supervision sources.
2. **Training Data as the Interface to ML:** The system should model label sources to produce a single, probabilistic label for each data point and train any of a wide range of classifiers to generalize beyond those sources.
3. **Supervision as Interactive Programming:** The system should provide rapid results in response to user supervision. We envision weak supervision as the REPL-like interface for machine learning.

Our work makes the following technical contributions:

A Flexible Interface for Sources: We observe that the heterogeneity of weak supervision strategies is a stumbling block for developers. Different types of weak supervision operate on different scopes of the input data. For example, distant supervision has to be mapped programmatically to specific spans of text. Crowd workers and weak classifiers often operate over entire documents or images. Heuristic rules are open ended; they can leverage information from multiple contexts simultaneously, such as combining information from a document's title, named entities in the text, and knowledge bases. This heterogeneity was cumbersome enough to completely block users of early versions of Snorkel.

To address this challenge, we built an interface layer around the abstract concept of a *labeling function (LF)*. We developed a flexible language for expressing weak supervision strategies and supporting data structures. We observed accelerated user productivity with these tools, which we validated in a user study where SMEs build models 2.8× faster and increase predictive performance an average 45.5% versus seven hours of hand labeling.

Tradeoffs in Modeling of Sources: Snorkel learns the accuracies of weak supervision sources without access to ground truth using a generative model [38]. Furthermore, it also learns correlations and other statistical dependencies among sources, correcting for dependencies in labeling functions that skew the estimated accuracies [5]. This paradigm gives rise to previously unexplored tradeoff spaces between predictive performance and speed. The natural first question is: when does modeling the accuracies of sources improve predictive performance? Further, how many dependencies, such as correlations, are worth modeling?

We study the tradeoffs between predictive performance and training time in generative models for weak supervision. While modeling source accuracies and correlations will not hurt predictive performance, we present a theoretical analysis of when a simple majority vote will work just as well. Based on our conclusions, we introduce an optimizer for deciding when to model accuracies of labeling functions, and when learning can be skipped in favor of a simple majority vote. Further, our optimizer automatically decides which correlations to model among labeling functions. This optimizer correctly predicts the advantage of generative modeling over majority vote to within 2.16 accuracy points on average on our evaluation tasks, and accelerates pipeline executions by up to 1.8×. It also enables us to gain 60%–70% of the benefit of correlation learning while saving up to 61% of training time (34 minutes per execution).

First End-to-End System for Data Programming: Snorkel is the first system to implement our recent work on data programming [5,38]. Previous ML systems that we and others developed [52] required extensive feature engineering and model specification, leading to confusion about where to inject relevant domain knowledge. While programming weak supervision seems superficially similar to feature engineering, we observe that users approach the two processes very differently. Our vision—weak supervision as the sole port of interaction for machine learning—implies radically different workflows, requiring a proof of concept.

Snorkel demonstrates that this paradigm enables users to develop high-quality models for a wide range of tasks. We report on two deployments of Snorkel, in collaboration with the U.S. Department of Veterans Affairs and Stanford Hospital and Clinics, and the U.S. Food and Drug Administration, where Snorkel improves over heuristic baselines by an average 110%. We also report results on four open-

[1] `http://snorkel.stanford.edu`

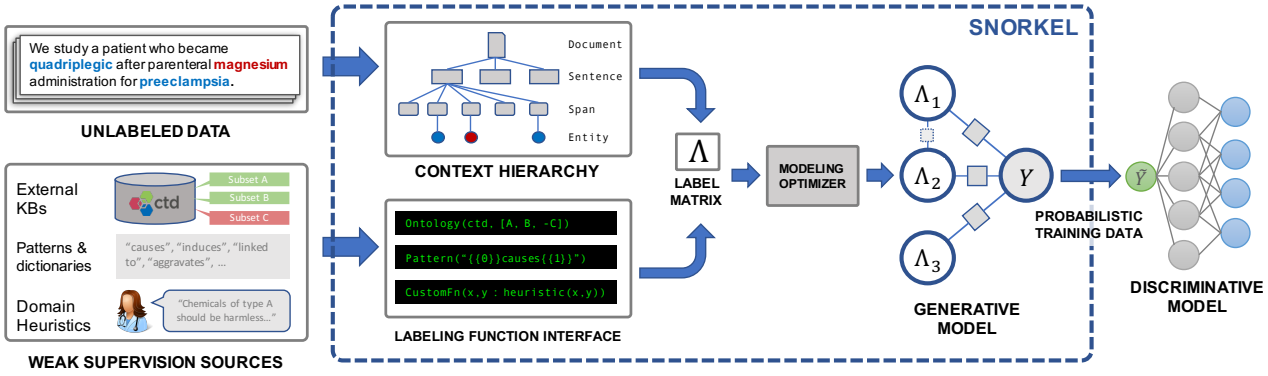

Figure 2: An overview of the Snorkel system. (1) SME users write *labeling functions (LFs)* that express weak supervision sources like distant supervision, patterns, and heuristics. (2) Snorkel applies the LFs over unlabeled data and learns a generative model to combine the LFs' outputs into probabilistic labels. (3) Snorkel uses these labels to train a discriminative classification model, such as a deep neural network.

source datasets that are representative of other Snorkel deployments, including bioinformatics, medical image analysis, and crowdsourcing; on which Snorkel beats heuristics by an average 153% and comes within an average 3.60% of the predictive performance of large hand-curated training sets.

2. SNORKEL ARCHITECTURE

Snorkel's workflow is designed around data programming [5, 38], a fundamentally new paradigm for training machine learning models using weak supervision, and proceeds in three main stages (Figure 2):

1. **Writing Labeling Functions:** Rather than hand-labeling training data, users of Snorkel write labeling functions, which allow them to express various weak supervision sources such as patterns, heuristics, external knowledge bases, and more. This was the component most informed by early interactions (and mistakes) with users over the last year of deployment, and we present a flexible interface and supporting data model.
2. **Modeling Accuracies and Correlations:** Next, Snorkel automatically learns a *generative model* over the labeling functions, which allows it to estimate their accuracies and correlations. This step uses no ground-truth data, learning instead from the agreements and disagreements of the labeling functions. We observe that this step improves end predictive performance 5.81% over Snorkel with unweighted label combination, and anecdotally that it streamlines the user development experience by providing actionable feedback about labeling function quality.
3. **Training a Discriminative Model:** The output of Snorkel is a set of *probabilistic labels* that can be used to train a wide variety of state-of-the-art machine learning models, such as popular deep learning models. While the generative model is essentially a re-weighted combination of the user-provided labeling functions—which tend to be precise but low-coverage—modern discriminative models can retain this precision while learning to generalize beyond the labeling functions, increasing coverage and robustness on unseen data.

Next we set up the problem Snorkel addresses and describe its main components and design decisions.

Setup: Our goal is to learn a parameterized classification model h_θ that, given a data point $x \in \mathcal{X}$, predicts its label $y \in \mathcal{Y}$, where the set of possible labels $\mathcal{Y}$ is discrete. For simplicity, we focus on the binary setting $\mathcal{Y} = \{-1, 1\}$, though we include a multi-class application in our experiments. For example, x might be a medical image, and y a label indicating normal versus abnormal. In the relation extraction examples we look at, we often refer to x as a *candidate*. In a traditional supervised learning setup, we would learn h_θ by fitting it to a *training set* of labeled data points. However, in our setting, we assume that we only have access to unlabeled data for training. We do assume access to a small set of labeled data used during development, called the *development set*, and a blind, held-out labeled *test set* for evaluation. These sets can be orders of magnitudes smaller than a training set, making them economical to obtain.

The user of Snorkel aims to generate training labels by providing a set of labeling functions, which are black-box functions, $\lambda : \mathcal{X} \to \mathcal{Y} \cup \{\emptyset\}$, that take in a data point and output a label where we use $\emptyset$ to denote that the labeling functions abstains. Given m unlabeled data points and n labeling functions, Snorkel applies the labeling functions over the unlabeled data to produce a matrix of labeling function outputs $\Lambda \in (\mathcal{Y} \cup \{\emptyset\})^{m \times n}$. The goal of the remaining Snorkel pipeline is to synthesize this label matrix Λ—which may contain overlapping and conflicting labels for each data point—into a single vector of *probabilistic training labels* $\tilde{Y} = (\tilde{y}_1, ..., \tilde{y}_m)$, where $\tilde{y}_i \in [0, 1]$. These training labels can then be used to train a discriminative model.

Next, we introduce the running example of a text relation extraction task as a proxy for many real-world knowledge base construction and data analysis tasks:

EXAMPLE 2.1. *Consider the task of extracting mentions of adverse chemical-disease relations from the biomedical literature (see CDR task, Section 4.1). Given documents with mentions of chemicals and diseases tagged, we refer to each co-occuring (chemical, disease) mention pair as a candidate extraction, which we view as a data point to be classified as either true or false. For example, in Figure 2, we would have two candidates with true labels y_1 = `True` and y_2 = `False`:*

```
x_1 = Causes("magnesium", "quadriplegic")
x_2 = Causes("magnesium", "preeclampsia")
```

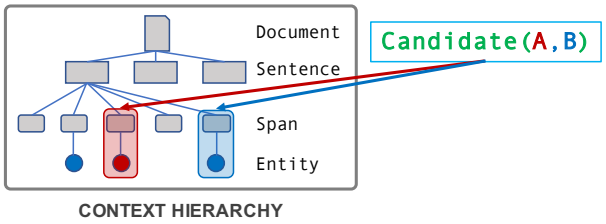

Figure 3: Labeling functions take as input a `Candidate` object, representing a data point to be classified. Each `Candidate` is a tuple of `Context` objects, which are part of a hierarchy representing the local context of the `Candidate`.

Data Model: A design challenge is managing complex, unstructured data in a way that enables SMEs to write labeling functions over it. In Snorkel, input data is stored in a *context hierarchy*. It is made up of context types connected by parent/child relationships, which are stored in a relational database and made available via an object-relational mapping (ORM) layer built with SQLAlchemy.[2] Each *context type* represents a conceptual component of data to be processed by the system or used when writing labeling functions; for example a document, an image, a paragraph, a sentence, or an embedded table. Candidates—i.e., data points x—are then defined as tuples of contexts (Figure 3).

EXAMPLE 2.2. *In our running CDR example, the input documents can be represented in Snorkel as a hierarchy consisting of `Documents`, each containing one or more `Sentences`, each containing one or more `Spans` of text. These `Spans` may also be tagged with metadata, such as `Entity` markers identifying them as chemical or disease mentions (Figure 3). A candidate is then a tuple of two `Spans`.*

2.1 A Language for Weak Supervision

Snorkel uses the core abstraction of a labeling function to allow users to specify a wide range of weak supervision sources such as patterns, heuristics, external knowledge bases, crowdsourced labels, and more. This higher-level, less precise input is more efficient to provide (see Section 4.2), and can be automatically denoised and synthesized, as described in subsequent sections.

In this section, we describe our design choices in building an interface for writing labeling functions, which we envision as a unifying programming language for weak supervision. These choices were informed to a large degree by our interactions—primarily through weekly office hours—with Snorkel users in bioinformatics, defense, industry, and other areas over the past year.[3] For example, while we initially intended to have a more complex structure for labeling functions, with manually specified types and correlation structure, we quickly found that simplicity in this respect was critical to usability (and not empirically detrimental to our ability to model their outputs). We also quickly discovered that users wanted either far more expressivity or far less of it, compared to our first library of function templates. We thus trade off expressivity and efficiency by allowing users to write labeling functions at two levels of abstraction: custom Python functions and declarative operators.

[2] `https://www.sqlalchemy.org/`

[3] `http://snorkel.stanford.edu#users`

Hand-Defined Labeling Functions: In its most general form, a labeling function is just an arbitrary snippet of code, usually written in Python, which accepts as input a `Candidate` object and either outputs a label or abstains. Often these functions are similar to extract-transform-load scripts, expressing basic patterns or heuristics, but may use supporting code or resources and be arbitrarily complex. Writing labeling functions by hand is supported by the ORM layer, which maps the context hierarchy and associated metadata to an object-oriented syntax, allowing the user to easily traverse the structure of the input data.

EXAMPLE 2.3. *In our running example, we can write a labeling function that checks if the word "causes" appears between the chemical and disease mentions. If it does, it outputs `True` if the chemical mention is first and `False` if the disease mention is first. If "causes" does not appear, it outputs `None`, indicating abstention:*

```
def LF_causes(x):
  cs, ce = x.chemical.get_word_range()
  ds, de = x.disease.get_word_range()
  if ce < ds and "causes" in x.parent.words[ce+1:ds]:
      return True
  if de < cs and "causes" in x.parent.words[de+1:cs]:
      return False
  return None
```

We could also write this with Snorkel's declarative interface:

```
LF_causes = lf_search("{{1}}.*\Wcauses\W.*{{2}}",
    reverse_args=False)
```

Declarative Labeling Functions: Snorkel includes a library of declarative operators that encode the most common weak supervision function types, based on our experience with users over the last year. These functions capture a range of common forms of weak supervision, for example:

- **Pattern-based:** Pattern-based heuristics embody the motivation of soliciting higher information density input from SMEs. For example, pattern-based heuristics encompass feature annotations [51] and pattern-bootstrapping approaches [18, 20] (Example 2.3).
- **Distant supervision:** Distant supervision generates training labels by heuristically aligning data points with an external knowledge base, and is one of the most popular forms of weak supervision [4, 22, 32].
- **Weak classifiers:** Classifiers that are insufficient for our task—e.g., limited coverage, noisy, biased, and/or trained on a different dataset—can be used as labeling functions.
- **Labeling function generators:** One higher-level abstraction that we can build on top of labeling functions in Snorkel is *labeling function generators*, which generate multiple labeling functions from a single resource, such as crowdsourced labels and distant supervision from structured knowledge bases (Example 2.4).

EXAMPLE 2.4. *A challenge in traditional distant supervision is that different subsets of knowledge bases have different levels of accuracy and coverage. In our running example, we can use the Comparative Toxicogenomics Database (CTD)[4] as distant supervision, separately modeling different subsets of it with separate labeling functions. For example,*

[4] `http://ctdbase.org/`

we might write one labeling function to label a candidate `True` *if it occurs in the "Causes" subset, and another to label it* `False` *if it occurs in the "Treats" subset. We can write this using a labeling function generator,*

```
LFs_CTD = Ontology(ctd,
    {"Causes": True, "Treats": False})
```

which creates two labeling functions. In this way, generators can be connected to large resources and create hundreds of labeling functions with a line of code.

2.2 Generative Model

The core operation of Snorkel is modeling and integrating the noisy signals provided by a set of labeling functions. Using the recently proposed approach of data programming [5, 38], we model the true class label for a data point as a latent variable in a probabilistic model. In the simplest case, we model each labeling function as a noisy "voter" which is *independent*—i.e., makes errors that are uncorrelated with the other labeling functions. This defines a generative model of the votes of the labeling functions as noisy signals about the true label.

We can also model statistical dependencies between the labeling functions to improve predictive performance. For example, if two labeling functions express similar heuristics, we can include this dependency in the model and avoid a "double counting" problem. We observe that such pairwise correlations are the most common, so we focus on them in this paper (though handling higher order dependencies is straightforward). We use our structure learning method for generative models [5] to select a set C of labeling function pairs (j, k) to model as correlated (see Section 3.2).

Now we can construct the full generative model as a factor graph. We first apply all the labeling functions to the unlabeled data points, resulting in a label matrix Λ, where $\Lambda_{i,j} = \lambda_j(x_i)$. We then encode the generative model $p_w(\Lambda, Y)$ using three factor types, representing the labeling propensity, accuracy, and pairwise correlations of labeling functions:

$$\begin{aligned}
\phi^{\text{Lab}}_{i,j}(\Lambda, Y) &= \mathbb{1}\{\Lambda_{i,j} \neq \emptyset\} \\
\phi^{\text{Acc}}_{i,j}(\Lambda, Y) &= \mathbb{1}\{\Lambda_{i,j} = y_i\} \\
\phi^{\text{Corr}}_{i,j,k}(\Lambda, Y) &= \mathbb{1}\{\Lambda_{i,j} = \Lambda_{i,k}\} \qquad (j,k) \in C
\end{aligned}$$

For a given data point x_i, we define the concatenated vector of these factors for all the labeling functions $j = 1, ..., n$ and potential correlations C as $\phi_i(\Lambda, Y)$, and the corresponding vector of parameters $w \in \mathbb{R}^{2n+|C|}$. This defines our model:

$$p_w(\Lambda, Y) = Z_w^{-1} \exp\left(\sum_{i=1}^{m} w^T \phi_i(\Lambda, y_i)\right) ,$$

where Z_w is a normalizing constant. To learn this model *without* access to the true labels Y, we minimize the negative log marginal likelihood given the observed label matrix Λ:

$$\hat{w} = \arg\min_w -\log \sum_Y p_w(\Lambda, Y) .$$

We optimize this objective by interleaving stochastic gradient descent steps with Gibbs sampling ones, similar to contrastive divergence [21]; for more details, see [5, 38]. We use the Numbskull library,[5] a Python NUMBA-based Gibbs sampler. We then use the predictions, $\tilde{Y} = p_{\hat{w}}(Y|\Lambda)$, as *probabilistic training labels*.

[5] `https://github.com/HazyResearch/numbskull`

2.3 Discriminative Model

The end goal in Snorkel is to train a model that generalizes beyond the information expressed in the labeling functions. We train a discriminative model h_θ on our probabilistic labels $\tilde{Y}$ by minimizing a *noise-aware* variant of the loss $l(h_\theta(x_i), y)$, i.e., the expected loss with respect to $\tilde{Y}$:

$$\hat{\theta} = \arg\min_\theta \sum_{i=1}^{m} \mathbb{E}_{y\sim\tilde{Y}}\left[l(h_\theta(x_i), y)\right] .$$

A formal analysis shows that as we increase the amount of *unlabeled data*, the generalization error of discriminative models trained with Snorkel will decrease at the same asymptotic rate as traditional supervised learning models do with additional hand-labeled data [38], allowing us to increase predictive performance by adding more unlabeled data. Intuitively, this property holds because as more data is provided, the discriminative model sees more features that co-occur with the heuristics encoded in the labeling functions.

EXAMPLE 2.5. *The CDR data contains the sentence, "Myasthenia gravis presenting as weakness after magnesium administration." None of the 33 labeling functions we developed vote on the corresponding* `Causes(magnesium, myasthenia gravis)` *candidate, i.e., they all abstain. However, a deep neural network trained on probabilistic training labels from Snorkel correctly identifies it as a true mention.*

Snorkel provides connectors for popular machine learning libraries such as TensorFlow [2], allowing users to exploit commodity models like deep neural networks that do not require hand-engineering of features and have robust predictive performance across a wide range of tasks.

3. WEAK SUPERVISION TRADEOFFS

We study the fundamental question of when—and at what level of complexity—we should expect Snorkel's generative model to yield the greatest predictive performance gains. Understanding these performance regimes can help guide users, and introduces a tradeoff space between predictive performance and speed. We characterize this space in two parts: first, by analyzing when the generative model can be approximated by an unweighted majority vote, and second, by automatically selecting the complexity of the correlation structure to model. We then introduce a two-stage, rule-based optimizer to support fast development cycles.

3.1 Modeling Accuracies

The natural first question when studying systems for weak supervision is, "When does modeling the accuracies of sources improve end-to-end predictive performance?" We study that question in this subsection and propose a heuristic to identify settings in which this modeling step is most beneficial.

3.1.1 Tradeoff Space

We start by considering the *label density* d_Λ of the label matrix Λ, defined as the mean number of non-abstention labels per data point. In the low-density setting, sparsity of labels will mean that there is limited room for even an optimal weighting of the labeling functions to diverge much from the majority vote. Conversely, as the label density

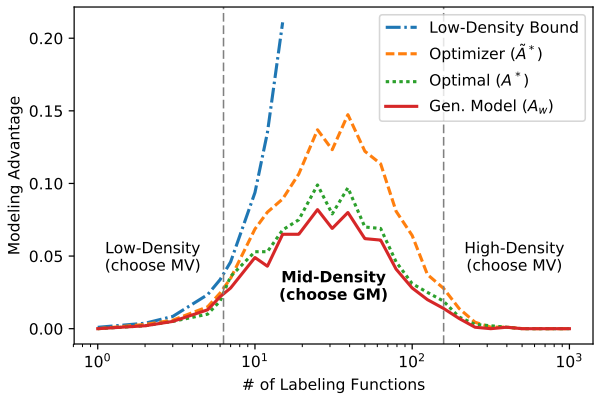

Figure 4: A plot of the *modeling advantage*, i.e., the improvement in label accuracy from the generative model, as a function of the number of labeling functions (equivalently, the label density) on a synthetic dataset.[7] We plot the advantage obtained by a learned generative model (GM), A_w; by an optimal model A^*; the upper bound $\tilde{A}^*$ used in our optimizer; and the low-density bound (Proposition 1).

grows, known theory confirms that the majority vote will eventually be optimal [27]. It is the middle-density regime where we expect to most benefit from applying the generative model. We start by defining a measure of the benefit of weighting the labeling functions by their true accuracies—in other words, the predictions of a perfectly estimated generative model—versus an unweighted majority vote:

DEFINITION 1. ***(Modeling Advantage)*** *Let the weighted majority vote of n labeling functions on data point x_i be denoted as $f_w(\Lambda_i) = \sum_{j=1}^{n} w_j \Lambda_{i,j}$, and the unweighted majority vote (MV) as $f_1(\Lambda_i) = \sum_{j=1}^{n} \Lambda_{i,j}$, where we consider the binary classification setting and represent an abstaining vote as 0. We define the* ***modeling advantage*** *A_w as the improvement in accuracy of f_w over f_1 for a dataset:*

$$A_w(\Lambda, y) = \frac{1}{m} \sum_{i=1}^{m} \left(\mathbb{1}\left\{ y_i f_w(\Lambda_i) > 0 \wedge y_i f_1(\Lambda_i) \leq 0 \right\} - \mathbb{1}\left\{ y_i f_w(\Lambda_i) \leq 0 \wedge y_i f_1(\Lambda_i) > 0 \right\} \right)$$

In other words, A_w is the number of times f_w correctly disagrees with f_1 on a label, minus the number of times it incorrectly disagrees. Let the ***optimal advantage*** *$A^* = A_{w^*}$ be the advantage using the optimal weights w^* (WMV*).*

To build intuition, we start by analyzing the optimal advantage for three regimes of label density (see Figure 6):

Low Label Density: In this sparse setting, very few data points have more than one non-abstaining label; only a small number have multiple conflicting labels. We have observed this occurring, for example, in the early stages of application development. We see that with non-adversarial labeling functions ($w^* > 0$), even an optimal generative model (WMV*) can only disagree with MV when there are disagreeing labels, which will occur infrequently. We see that the expected optimal advantage will have an upper bound that falls quadratically with label density:

[7] We generate a class-balanced dataset of $m = 1000$ data points with binary labels, and n independent labeling functions with average accuracy 75% and a fixed 10% probability of voting.

Table 1: Modeling advantage A_w attained using a generative model for several applications in Snorkel (Section 4.1), the upper bound $\tilde{A}^*$ used by our optimizer, the modeling strategy selected by the optimizer—either majority vote (MV) or generative model (GM)—and the empirical label density d_Λ.

Dataset	A_w (%)	$\tilde{A}^*$ (%)	Modeling Strategy	d_Λ
Radiology	7.0	12.4	**GM**	2.3
CDR	4.9	7.9	**GM**	1.8
Spouses	4.4	4.6	**GM**	1.4
Chem	0.1	0.3	**MV**	1.2
EHR	2.8	4.8	**GM**	1.2

PROPOSITION 1. ***(Low-Density Upper Bound)*** *Assume that $P(\Lambda_{i,j} \neq 0) = p_l \ \forall i, j$, and $w_j^* > 0 \ \forall j$. Then, the expected label density is $\bar{d} = np_l$, and*

$$\mathbb{E}_{\Lambda, y, w^*}\left[A^*\right] = O\left(\bar{d}^2\right) \tag{1}$$

Proof Sketch: We bound the advantage above by computing the expected number of pairwise disagreements.

High Label Density: In this setting, the majority of the data points have a large number of labels. For example, we might be working in an extremely high-volume crowdsourcing setting, or an application with many high-coverage knowledge bases as distant supervision. Under modest assumptions—namely, that the average labeling function accuracy $\overline{\alpha}^*$ is greater than 50%—it is known that the majority vote converges exponentially to an optimal solution as the average label density $\bar{d}$ increases, which serves as an upper bound for the expected optimal advantage as well:

THEOREM 1. *(High-Density Upper Bound [27]) Assume that $P(\Lambda_{i,j} \neq 0) = p_l \ \forall i, j$, and that $\overline{\alpha}^* = \frac{1}{n}\sum_{j=1}^{n} \alpha_j^* = \frac{1}{n}\sum_{j=1}^{n} 1/(1+\exp(w_j^*)) > \frac{1}{2}$. Then:*

$$\mathbb{E}_{\Lambda, y, w^*}\left[A^*\right] \leq e^{-2p_l\left(\overline{\alpha}^* - \frac{1}{2}\right)^2 \bar{d}} \tag{2}$$

Proof: This follows from the result in [27] for the symmetric Dawid-Skene model under constant probability sampling.

Medium Label Density: In this middle regime, we expect that modeling the accuracies of the labeling functions will deliver the greatest gains in predictive performance because we will have many data points with a small number of disagreeing labeling functions. For such points, the estimated labeling function accuracies can heavily affect the predicted labels. We indeed see gains in the empirical results using an independent generative model that only includes accuracy factors $\phi_{i,j}^{\text{Acc}}$ (Table 1). Furthermore, the guarantees in [38] establish that we can learn the optimal weights, and thus approach the optimal advantage.

3.1.2 Automatically Choosing a Modeling Strategy

The bounds in the previous subsection imply that there are settings in which we should be able to safely skip modeling the labeling function accuracies, simply taking the unweighted majority vote instead. However, in practice, the

overall label density d_Λ is insufficiently precise to determine the transition points of interest, given a user time-cost tradeoff preference (characterized by the *advantage tolerance* parameter γ in Algorithm 1). We show this in Table 1 using our application data sets from Section 4.1. For example, we see that the Chem and EHR label matrices have equivalent label densities; however, modeling the labeling function accuracies has a much greater effect for EHR than for Chem.

Instead of simply considering the average label density d_Λ, we instead develop a best-case heuristic based on looking at the ratio of positive to negative labels for each data point. This heuristic serves as an upper bound to the true expected advantage, and thus we can use it to determine when we can safely skip training the generative model (see Algorithm 1). Let $c_y(\Lambda_i) = \sum_{j=1}^{n} \mathbb{1}\{\Lambda_{i,j} = y\}$ be the counts of labels of class y for x_i, and assume that the true labeling function weights lie within a fixed range, $w_j \in [w_{min}, w_{max}]$ and have a mean $\bar{w}$.[8] Then, define:

$$\Phi(\Lambda_i, y) = \mathbb{1}\{c_y(\Lambda_i) w_{max} > c_{-y}(\Lambda_i) w_{min}\}$$

$$\tilde{A}^*(\Lambda) = \frac{1}{m} \sum_{i=1}^{m} \sum_{y \in \pm 1} \mathbb{1}\{y f_1(\Lambda_i) \leq 0\} \Phi(\Lambda_i, y) \sigma(2 f_{\bar{w}}(\Lambda_i) y)$$

where $\sigma(\cdot)$ is the sigmoid function, $f_{\bar{w}}$ is majority vote with all weights set to the mean $\bar{w}$, and $\tilde{A}^*(\Lambda)$ is the predicted modeling advantage used by our optimizer. Essentially, we are taking the expected counts of instances in which a weighted majority vote could possibly flip the incorrect predictions of unweighted majority vote under best case conditions, which is an upper bound for the expected advantage:

PROPOSITION 2. ***(Optimizer Upper Bound)*** *Assume that the labeling functions have accuracy parameters (log-odds weights) $w_j \in [w_{min}, w_{max}]$, and have $\mathbb{E}[w] = \bar{w}$. Then:*

$$\mathbb{E}_{y, w^*}\left[A^* \mid \Lambda\right] \leq \tilde{A}^*(\Lambda) \tag{3}$$

Proof Sketch: We upper-bound the modeling advantage by the expected number of instances in which WMV* is correct and MV is incorrect. We then upper-bound this by using the best-case probability of the weighted majority vote being correct given (w_{min}, w_{max}).

We apply $\tilde{A}^*$ to a synthetic dataset and plot in Figure 6. Next, we compute $\tilde{A}^*$ for the labeling matrices from experiments in Section 4.1, and compare with the empirical advantage of the trained generative models (Table 1). We see that our approximate quantity $\tilde{A}^*$ serves as a correct guide in all cases for determining which modeling strategy to select, which for the mature applications reported on is indeed most often the generative model. However, we see that while EHR and Chem have equivalent label densities, our optimizer correctly predicts that Chem can be modeled with majority vote, speeding up each pipeline execution by 1.8×. We find in our applications that the optimizer can save execution time especially during the initial stages of iterative development (see full version).

[8]We fix these at defaults of $(w_{min}, \bar{w}, w_{max}) = (0.5, 1.0, 1.5)$, which corresponds to assuming labeling functions have accuracies between 62% and 82%, and an average accuracy of 73%.

3.2 Modeling Structure

In this subsection, we consider modeling additional statistical structure beyond the independent model. We study the tradeoff between predictive performance and computational cost, and describe how to automatically select a good point in this tradeoff space.

Structure Learning. We observe many Snorkel users writing labeling functions that are statistically dependent. Examples we have observed include:

- Functions that are variations of each other, such as checking for matches against similar regular expressions.
- Functions that operate on correlated inputs, such as raw tokens of text and their lemmatizations.
- Functions that use correlated sources of knowledge, such as distant supervision from overlapping knowledge bases.

Modeling such dependencies is important because they affect our estimates of the true labels. Consider the extreme case in which not accounting for dependencies is catastrophic:

EXAMPLE 3.1. *Consider a set of 10 labeling functions, where 5 are perfectly correlated, i.e., they vote the same way on every data point, and 5 are conditionally independent given the true label. If the correlated labeling functions have accuracy $\alpha = 50\%$ and the uncorrelated ones have accuracy $\beta = 99\%$, then the maximum likelihood estimate of their accuracies according to the independent model is $\hat{\alpha} = 100\%$ and $\hat{\beta} = 50\%$.*

Specifying a generative model to account for such dependencies by hand is impractical for three reasons. First, it is difficult for non-expert users to specify these dependencies. Second, as users iterate on their labeling functions, their dependency structure can change rapidly, like when a user relaxes a labeling function to label many more candidates. Third, the dependency structure can be dataset specific, making it impossible to specify a priori, such as when a corpus contains many strings that match multiple regular expressions used in different labeling functions. We observed users of earlier versions of Snorkel struggling for these reasons to construct accurate and efficient generative models with dependencies. We therefore seek a method that can quickly identify an appropriate dependency structure from the labeling function outputs Λ alone.

Naively, we could include all dependencies of interest, such as all pairwise correlations, in the generative model and perform parameter estimation. However, this approach is impractical. For 100 labeling functions and 10,000 data points, estimating parameters with all possible correlations takes roughly 45 minutes. When multiplied over repeated runs of hyperparameter searching and development cycles, this cost greatly inhibits labeling function development. We therefore turn to our method for automatically selecting which dependencies to model without access to ground truth [5]. It uses a pseudolikelihood estimator, which does not require any sampling or other approximations to compute the objective gradient exactly. It is much faster than maximum likelihood estimation, taking 15 seconds to select pairwise correlations to be modeled among 100 labeling functions with 10,000 data points. However, this approach relies on a selection threshold hyperparameter ϵ which induces a tradeoff space between predictive performance and computational cost.

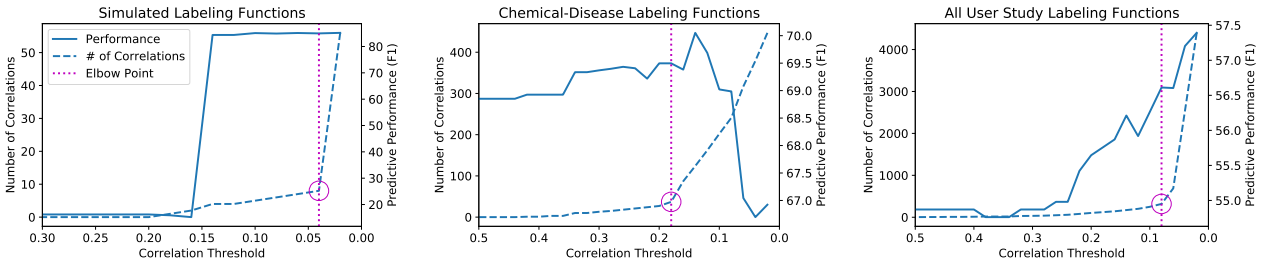

Figure 5: Predictive performance of the generative model and number of learned correlations versus the correlation threshold ϵ. The selected elbow point achieves a good tradeoff between predictive performance and computational cost (linear in the number of correlations). Left: simulation of structure learning correcting the generative model. Middle: the CDR task. Right: all user study labeling functions for the Spouses task.

3.2.1 *Tradeoff Space*

Such structure learning methods, whether pseudolikelihood or likelihood-based, crucially depend on a selection threshold ϵ for deciding which dependencies to add to the generative model. Fundamentally, the choice of ϵ determines the complexity of the generative model.[9] We study the tradeoff between predictive performance and computational cost that this induces. We find that generally there is an "elbow point" beyond which the number of correlations selected—and thus the computational cost—explodes, and that this point is a safe tradeoff point between predictive performance and computation time.

Predictive Performance: At one extreme, a very large value of ϵ will not include any correlations in the generative model, making it identical to the independent model. As ϵ is decreased, correlations will be added. At first, when ϵ is still high, only the strongest correlations will be included. As these correlations are added, we observe that the generative model's predictive performance tends to improve. Figure 5, left, shows the result of varying ϵ in a simulation where more than half the labeling functions are correlated. After adding a few key dependencies, the generative model resolves the discrepancies among the labeling functions. Figure 5, middle, shows the effect of varying ϵ for the CDR task. Predictive performance improves as ϵ decreases until the model overfits. Finally, we consider a large number of labeling functions that are likely to be correlated. In our user study (described in Section 4.2), participants wrote labeling functions for the Spouses task. We combined all 125 of their functions and studied the effect of varying ϵ. Here, we expect there to be many correlations since it is likely that users wrote redundant functions. We see in Figure 5, right, that structure learning surpasses the best performing individual's generative model (50.0 F1).

Computational Cost: Computational cost is correlated with model complexity. Since learning in Snorkel is done with a Gibbs sampler, the overhead of modeling additional correlations is linear in the number of correlations. The dashed lines in Figure 5 show the number of correlations included in each model versus ϵ. For example, on the Spouses task, fitting the parameters of the generative model at $\epsilon = 0.5$ takes 4 minutes, and fitting its parameters with $\epsilon = 0.02$ takes 57 minutes. Further, parameter estimation is often run repeatedly during development for two reasons: (i) fitting generative model hyperparameters using a development set requires repeated runs, and (ii) as users iterate on their labeling functions, they must re-estimate the generative model to evaluate them.

3.2.2 *Automatically Choosing a Model*

Based on our observations, we seek to automatically choose a value of ϵ that trades off between predictive performance and computational cost using the labeling functions' outputs Λ alone. Including ϵ as a hyperparameter in a grid search over a development set is generally not feasible because of its large effect on running time. We therefore want to choose ϵ before other hyperparameters, without performing any parameter estimation. We propose using the number of correlations selected at each value of ϵ as an inexpensive indicator. The dashed lines in Figure 5 show that as ϵ decreases, the number of selected correlations follows a pattern. Generally, the number of correlations grows slowly at first, then hits an "elbow point" beyond which the number explodes, which fits the assumption that the correlation structure is sparse. In all three cases, setting ϵ to this elbow point is a safe tradeoff between predictive performance and computational cost. In cases where performance grows consistently (left and right), the elbow point achieves most of the predictive performance gains at a small fraction of the computational cost. For example, on Spouses (right), choosing $\epsilon = 0.08$ achieves a score of 56.6 F1—within one point of the best score—but only takes 8 minutes for parameter estimation. In cases where predictive performance eventually degrades (middle), the elbow point also selects a relatively small number of correlations, giving an 0.7 F1 point improvement and avoiding overfitting.

Performing structure learning for many settings of ϵ is inexpensive, especially since the search needs to be performed only once before tuning the other hyperparameters. On the large number of labeling functions in the Spouses task, structure learning for 25 values of ϵ takes 14 minutes. On CDR, with a smaller number of labeling functions, it takes 30 seconds. Further, if the search is started at a low value of ϵ and increased, it can often be terminated early, when the number of selected correlations reaches a low value. Selecting the elbow point itself is straightforward. We use the point with greatest absolute difference from its neighbors, but more sophisticated schemes can also be applied [43]. Our full optimization algorithm for choosing a modeling strategy and (if necessary) correlations is shown in Algorithm 1.

[9]Specifically, ϵ is both the coefficient of the ℓ_1 regularization term used to induce sparsity, and the minimum absolute weight in log scale that a dependency must have to be selected.

Algorithm 1 Modeling Strategy Optimizer

Input: Label matrix $\Lambda \in (\mathcal{Y} \cup \{\emptyset\})^{m \times n}$,
advantage tolerance γ, structure search resolution η
Output: Modeling strategy

```
if Ã*(Λ) < γ then
  return MV
Structures ← [ ]
for i from 1 to 1/(2η) do
  ε ← i · η
  C ← LearnStructure(Λ, ε)
  Structures.append(|C|, ε)
ε ← SelectElbowPoint(Structures)
return GM_ε
```

4. EVALUATION

We evaluate Snorkel by drawing on deployments developed in collaboration with users. We report on two real-world deployments and four tasks on open-source data sets representative of other deployments. Our evaluation is designed to support the following three main claims:

- **Snorkel outperforms distant supervision baselines.** In *distant supervision* [32], one of the most popular forms of weak supervision used in practice, an external knowledge base is heuristically aligned with input data to serve as noisy training labels. By allowing users to easily incorporate a broader, more heterogeneous set of weak supervision sources, Snorkel exceeds models trained via distant supervision by an average of 132%.
- **Snorkel approaches hand supervision.** We see that by writing tens of labeling functions, we were able to approach or match results using hand-labeled training data which took weeks or months to assemble, coming within 2.11% of the F1 score of hand supervision on relation extraction tasks and an average 5.08% accuracy or AUC on cross-modal tasks, for an average 3.60% across all tasks.
- **Snorkel enables a new interaction paradigm.** We measure Snorkel's efficiency and ease-of-use by reporting on a user study of biomedical researchers from across the U.S. These participants learned to write labeling functions to extract relations from news articles as part of a two-day workshop on learning to use Snorkel, and matched or outperformed models trained on hand-labeled training data, showing the efficiency of Snorkel's process even for first-time users.

We now describe our results in detail. First, we describe the six applications that validate our claims. We then show that Snorkel's generative modeling stage helps to improve the predictive performance of the discriminative model, demonstrating that it is 5.81% more accurate when trained on Snorkel's probabilistic labels versus labels produced by an unweighted average of labeling functions. We also validate that the ability to incorporate many different types of weak supervision incrementally improves results with an ablation study. Finally, we describe the protocol and results of our user study.

4.1 Applications

To evaluate the effectiveness of Snorkel, we consider several real-world deployments and tasks on open-source datasets that are representative of other deployments in information extraction, medical image classification, and crowdsourced sentiment analysis. Summary statistics of the tasks are provided in Table 2.

Table 2: Number of labeling functions, fraction of positive labels (for binary classification tasks), number of training documents, and number of training candidates for each task.

Task	# LFs	% Pos.	# Docs	# Candidates
Chem	16	4.1	1,753	65,398
EHR	24	36.8	47,827	225,607
CDR	33	24.6	900	8,272
Spouses	11	8.3	2,073	22,195
Radiology	18	36.0	3,851	3,851
Crowd	102	-	505	505

Discriminative Models: One of the key bets in Snorkel's design is that the trend of increasingly powerful, open-source machine learning tools (e.g., models, pre-trained word embeddings and initial layers, automatic tuners, etc.) will only continue to accelerate. To best take advantage of this, Snorkel creates probabilistic training labels for any discriminative model with a standard loss function.

In the following experiments, we control for end model selection by using currently popular, standard choices across all settings. For text modalities, we choose a bidirectional long short term memory (LSTM) sequence model [17], and for the medical image classification task we use a 50-layer ResNet [19] pre-trained on the ImageNet object classification dataset [14]. Both models are implemented in Tensorflow [2] and trained using the Adam optimizer [24], with hyperparameters selected via random grid search using a small labeled development set. Final scores are reported on a held-out labeled test set. See full version for details.

A key takeaway of the following results is that the discriminative model generalizes beyond the heuristics encoded in the labeling functions (as in Example 2.5). In Section 4.1.1, we see that on relation extraction applications the discriminative model improves performance over the generative model primarily by increasing recall by 43.15% on average. In Section 4.1.2, the discriminative model classifies entirely new modalities of data to which the labeling functions cannot be applied.

4.1.1 Relation Extraction from Text

We first focus on four relation extraction tasks on text data, as it is a challenging and common class of problems that are well studied and for which distant supervision is often considered. Predictive performance is summarized in Table 3. We briefly describe each task.

Scientific Articles (Chem): With modern online repositories of scientific literature, such as PubMed[10] for biomedical articles, research results are more accessible than ever before. However, actually extracting fine-grained pieces of information in a structured format and using this data to answer specific questions at scale remains a significant open challenge for researchers. To address this challenge in the

[10] https://www.ncbi.nlm.nih.gov/pubmed/

Table 3: Evaluation of Snorkel on relation extraction tasks from text. Snorkel's generative and discriminative models consistently improve over distant supervision, measured in F1, the harmonic mean of precision (P) and recall (R). We compare with hand-labeled data when available, coming within an average of 1 F1 point.

	Distant Supervision			Snorkel (Gen.)				Snorkel (Disc.)				Hand Supervision		
Task	P	R	**F1**	P	R	**F1**	**Lift**	P	R	**F1**	**Lift**	P	R	**F1**
Chem	11.2	41.2	17.6	78.6	21.6	33.8	+16.2	87.0	39.2	54.1	+36.5	-	-	-
EHR	81.4	64.8	72.2	77.1	72.9	74.9	+2.7	80.2	82.6	81.4	+9.2	-	-	-
CDR	25.5	34.8	29.4	52.3	30.4	38.5	+9.1	38.8	54.3	45.3	+15.9	39.9	58.1	47.3
Spouses	9.9	34.8	15.4	53.5	62.1	57.4	+42.0	48.4	61.6	54.2	+38.8	47.8	62.5	54.2

context of drug safety research, Stanford and U.S. Food and Drug Administration (FDA) collaborators used Snorkel to develop a system for extracting chemical reagent and reaction product relations from PubMed abstracts. The goal was to build a database of chemical reactions that researchers at the FDA can use to predict unknown drug interactions. We used the chemical reactions described in the Metacyc database [8] for distant supervision.

Electronic Health Records (EHR): As patients' clinical records increasingly become digitized, researchers hope to inform clinical decision making by retrospectively analyzing large patient cohorts, rather than conducting expensive randomized controlled studies. However, much of the valuable information in electronic health records (EHRs)—such as fine-grained clinical details, practitioner notes, etc.—is not contained in standardized medical coding systems, and is thus locked away in the unstructured text notes sections. In collaboration with researchers and clinicians at the U.S. Department of Veterans Affairs, Stanford Hospital and Clinics (SHC), and the Stanford Center for Biomedical Informatics Research, we used Snorkel to develop a system to extract structured data from unstructured EHR notes. Specifically, the system's task was to extract mentions of pain levels at precise anatomical locations from clinician notes, with the goal of using these features to automatically assess patient well-being and detect complications after medical interventions like surgery. To this end, our collaborators created a cohort of 5,800 patients from SHC EHR data, with visit dates between 1995 and 2015, resulting in 500K unstructured clinical documents. Since distant supervision from a knowledge base is not applicable, we compared against regular-expression-based labeling previously developed for this task.

Chemical-Disease Relations (CDR): We used the 2015 BioCreative chemical-disease relation dataset [49], where the task is to identify mentions of causal links between chemicals and diseases in PubMed abstracts. We used all pairs of chemical and disease mentions co-occuring in a sentence as our candidate set. We used the Comparative Toxicogenomics Database (CTD) [33] for distant supervision, and additionally wrote labeling functions capturing language patterns and information from the context hierarchy. To evaluate Snorkel's ability to discover previously unknown information, we randomly removed half of the relations in CTD and evaluated on candidates not contained in the remaining half.

Spouses: Our fourth task is to identify mentions of spouse relationships in a set of news articles from the Signal Media dataset [10]. We used all pairs of person mentions (tagged with SpaCy's NER module[11]) co-occuring in the same sentence as our candidate set. To obtain hand-labeled data for evaluation, we crowdsourced labels for the candidates via Amazon Mechanical Turk, soliciting labels from three workers for each example and assigning the majority vote. We then wrote labeling functions that encoded language patterns and distant supervision from DBpedia [26].

Table 4: Evaluation on cross-modal experiments. Labeling functions that operate on or represent one modality (text, crowd workers) produce training labels for models that operate on another modality (images, text), and approach the predictive performance of large hand-labeled training datasets.

Task	Snorkel (Disc.)	Hand Supervision
Radiology (AUC)	72.0	76.2
Crowd (Acc)	65.6	68.8

4.1.2 *Cross-Modal: Images & Crowdsourcing*

In the cross-modal setting, we write labeling functions over one data modality (e.g., a text report, or the votes of crowdworkers) and use the resulting labels to train a classifier defined over a second, totally separate modality (e.g., an image or the text of a tweet). This demonstrates the flexibility of Snorkel, in that the labeling functions (and by extension, the generative model) do not need to operate over the same domain as the discriminative model being trained. Predictive performance is summarized in Table 4.

Abnormality Detection in Lung Radiographs (Rad): In many real-world radiology settings, there are large repositories of image data with corresponding narrative text reports, but limited or no labels that could be used for training an image classification model. In this application, in collaboration with radiologists, we wrote labeling functions over the text radiology reports, and used the resulting labels to train an image classifier to detect abnormalities in lung X-ray images. We used a publicly available dataset from the OpenI biomedical image repository[12] consisting of 3,851 distinct radiology reports—composed of unstructured text and Medical Subject Headings (MeSH)[13] codes—and accompanying X-ray images.

[11] `https://spacy.io/`

[12] `http://openi.nlm.nih.gov/`

[13] `https://www.nlm.nih.gov/mesh/meshhome.html`

Table 5: Comparison between training the discriminative model on the labels estimated by the generative model, versus training on the unweighted average of the LF outputs. Predictive performance gains show that modeling LF noise helps.

Task	Disc. Model on Unweighted LFs	Disc. Model	Lift
Chem	48.6	54.1	+5.5
EHR	80.9	81.4	+0.5
CDR	42.0	45.3	+3.3
Spouses	52.8	54.2	+1.4
Crowd (Acc)	62.5	65.6	+3.1
Rad. (AUC)	67.0	72.0	+5.0

Crowdsourcing (Crowd): We trained a model to perform sentiment analysis using crowdsourced annotations from the weather sentiment task from Crowdflower.[14] In this task, contributors were asked to grade the sentiment of often-ambiguous tweets relating to the weather, choosing between five categories of sentiment. Twenty contributors graded each tweet, but due to the difficulty of the task and lack of crowdworker filtering, there were many conflicts in worker labels. We represented each crowdworker as a labeling function—showing Snorkel's ability to subsume existing crowdsourcing modeling approaches—and then used the resulting labels to train a text model over the tweets, for making predictions independent of the crowd workers.

4.1.3 Effect of Generative Modeling

An important question is the significance of modeling the accuracies and correlations of the labeling functions on the end predictive performance of the discriminative model (versus in Section 3, where we only considered the effect on the accuracy of the generative model). We compare Snorkel with a simpler pipeline that skips the generative modeling stage and trains the discriminative model on an unweighted average of the labeling functions' outputs. Table 5 shows that the discriminative model trained on Snorkel's probabilistic labels consistently predicts better, improving 5.81% on average. These results demonstrate that the discriminative model effectively learns from the additional signal contained in Snorkel's probabilistic training labels over simpler modeling strategies.

4.1.4 Labeling Function Type Ablation

We also examine the impact of different types of labeling functions on end predictive performance, using the CDR application as a representative example of three common categories of labeling functions:

- *Text Patterns:* Basic word, phrase, and regular expression labeling functions.
- *Distant Supervision:* External knowledge bases mapped to candidates, either directly or filtered by a heuristic.
- *Structure-Based:* Labeling functions expressing heuristics over the context hierarchy, e.g., reasoning about position in the document or relative to other candidates.

We show an ablation in Table 6, sorting by stand-alone score. We see that distant supervision adds recall at the cost of some precision, as we would expect, but ultimately improves F1 score by 2 points; and that structure-based labeling functions, enabled by Snorkel's context hierarchy data representation, add an additional F1 point.

[14] `https://www.crowdflower.com/data/weather-sentiment/`

Table 6: Labeling function ablation study on CDR. Adding different types of labeling functions improves predictive performance.

LF Type	P	R	F1	Lift
Text Patterns	42.3	42.4	42.3	
+ Distant Supervision	37.5	54.1	44.3	+2.0
+ Structure-based	38.8	54.3	45.3	+1.0

4.2 User Study

We conducted a formal study of Snorkel to (i) evaluate how quickly SME users could learn to write labeling functions, and (ii) empirically validate the core hypothesis that writing labeling functions is more time-efficient than hand-labeling data. Users were given instruction on Snorkel, and then asked to write labeling functions for the Spouses task described in the previous subsection.

Participants: In collaboration with the Mobilize Center [25], an NIH-funded Big Data to Knowledge (BD2K) center, we distributed a national call for applications to attend a two-day workshop on using Snorkel for biomedical knowledge base construction. Selection criteria included a strong biomedical project proposal and little-to-no prior experience using Snorkel. In total, 15 researchers[15] were invited to attend out of 33 team applications submitted, with varying backgrounds in bioinformatics, clinical informatics, and data mining from universities, companies, and organizations around the United States. The education demographics included 6 bachelors, 4 masters, and 5 Ph.D. degrees. All participants could program in Python, with 80% rating their skill as intermediate or better; 40% of participants had little-to-no prior exposure to machine learning; and 53-60% had no prior experience with text mining or information extraction applications. See full version for details.

Protocol: The first day focused entirely on labeling functions, ranging from theoretical motivations to details of the Snorkel API. Over the course of 7 hours, participants were instructed in a classroom setting on how to use and evaluate models developed using Snorkel. Users were presented with 4 tutorial Jupyter notebooks providing skeleton code for evaluating labeling functions, along with a small labeled development candidate set, and were given 2.5 hours of dedicated development time in aggregate to write their labeling functions. All workshop materials are available online.[16]

Baseline: To compare our users' performance against models trained on hand-labeled data, we collected a large hand-labeled dataset via Amazon Mechanical Turk (the same set used in the previous subsection). We then split this into 15 datasets representing 7 hours worth of hand-labeling time

[15] One participant declined to write labeling functions, so their score is not included in our analysis.

[16] `https://github.com/HazyResearch/snorkel/tree/master/tutorials/workshop`

each—based on the crowd-worker average of 10 seconds per label—simulating the alternative scenario where users skipped both instruction and labeling function development sessions and instead spent the full day hand-labeling data.

Results: Our key finding is that labeling functions written in Snorkel, even by SME users, can match or exceed a traditional hand-labeling approach. The majority (8) of subjects matched or outperformed these hand-labeled data models. The average Snorkel user's score was 30.4 F1, and the average hand-supervision score was 20.9 F1. The best performing user model scored 48.7 F1, 19.2 points higher than the best supervised model using hand-labeled data. The worst participant scored 12.0 F1, 0.3 points higher that the lowest hand-labeled model. The full distribution of scores by participant, and broken down by participant background, compared against the baseline models trained with hand-labeled data is available in the full version.

5. RELATED WORK

This section is an overview of techniques for managing weak supervision, many of which are subsumed in Snorkel. We also contrast it with related forms of supervision.

Combining Weak Supervision Sources: The main challenge of weak supervision is how to combine multiple sources. For example, if a user provides two knowledge bases for distant supervision, how should a data point that matches only one knowledge base be labeled? Some researchers have used multi-instance learning to reduce the noise in weak supervision sources [22, 41], essentially modeling the different weak supervision sources as soft constraints on the true label, but this approach is limited because it requires using a specific end model that supports multi-instance learning.

Researchers have therefore considered how to estimate the accuracy of label sources without a gold standard with which to compare—a classic problem [13]—and combine these estimates into labels that can be used to train an arbitrary end model. Much of this work has focused on crowdsourcing, in which workers have unknown accuracy [11, 23, 53]. Such methods use generative probabilistic models to estimate a latent variable—the true class label—based on noisy observations. Other methods use generative models with hand-specified dependency structures to label data for specific modalities, such as topic models for text [4] or denoising distant supervision sources [42, 48]. Other techniques for estimating latent class labels given noisy observations include spectral methods [35]. Snorkel is distinguished from these approaches because its generative model supports a wide range of weak supervision sources, and it learns the accuracies and correlation structure among weak supervision sources without ground truth data.

Other Forms of Supervision: Work on *semi-supervised learning* considers settings with some labeled data and a much larger set of unlabeled data, and then leverages various domain- and task-agnostic assumptions about smoothness, low-dimensional structure, or distance metrics to heuristically label the unlabeled data [9]. Work on *active learning* aims to automatically estimate which data points are optimal to label, thereby hopefully reducing the total number of examples that need to be manually annotated [45]. *Transfer learning* considers the strategy of repurposing models trained on different datasets or tasks where labeled training data is more abundant [34]. Another type of supervision is self-training [3, 44] and co-training [6], which involves training a model or pair of models on data that they labeled themselves. Weak supervision is distinct in that the goal is to solicit input directly from SMEs, however at a higher level of abstraction and/or in an inherently noisier form. Snorkel is focused on managing weak supervision sources, but combing its methods with these other types of supervision is straightforward.

Related Data Management Problems: Researchers have considered related problems in data management, such as data fusion [15, 40] and truth discovery [28]. In these settings, the task is to estimate the reliability of data sources that provide assertions of facts and determine which facts are likely true. Many approaches to these problems use probablistic graphical models that are related to Snorkel's generative model in that they represent the unobserved truth as a latent variable, e.g., the latent truth model [54]. Our setting differs in that labeling functions assign labels to user-provided data, and they may provide any label or abstain, which we must model. Work on data fusion has also explored how to model user-specified correlations among data sources [36]. Snorkel automatically identifies which correlations among labeling functions to model.

6. CONCLUSION

Snorkel provides a new paradigm for soliciting and managing weak supervision to create training data sets. In Snorkel, users provide higher-level supervision in the form of labeling functions that capture domain knowledge and resources, without having to carefully manage the noise and conflicts inherent in combining weak supervision sources. Our evaluations demonstrate that Snorkel significantly reduces the cost and difficulty of training powerful machine learning models while exceeding prior weak supervision methods and approaching the quality of large, hand-labeled training sets. Snorkel's deployments in industry, research labs, and government agencies show that it has real-world impact, offering developers an improved way to build models.

Acknowledgments. Alison Callahan and Nigam Shah of Stanford, and Nicholas Giori of the US Dept. of Veterans Affairs developed the electronic health records application. Emily Mallory, Ambika Acharya, and Russ Altman of Stanford, and Roselie Bright and Elaine Johanson of the US Food and Drug Administration developed the scientific articles application. Joy Ku of the Mobilize Center organized the user study. Nishith Khandwala developed the radiograph application. We thank the contributors to Snorkel including Bryan He, Theodoros Rekatsinas, and Braden Hancock. We gratefully acknowledge the support of DARPA under No. N66001-15-C-4043 (SIMPLEX), No. FA8750-17-2-0095 (D3M), No. FA8750-12-2-0335, and No. FA8750-13-2-0039, DOE 108845, NIH U54EB020405, ONR under No. N000141210041 and No. N000141310129, the Moore Foundation, the Okawa Research Grant, American Family Insurance, Accenture, Toshiba, the Stanford Interdisciplinary Graduate and Bio-X fellowships, and members of the Stanford DAWN project: Intel, Microsoft, Teradata, and VMware. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views, policies, or endorsements, either expressed or implied, of DARPA, DOE, NIH, ONR, or the U.S. Government.

7. REFERENCES

[1] Worldwide semiannual cognitive/artificial intelligence systems spending guide. Technical report, International Data Corporation, 2017.

[2] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, et al. TensorFlow: A system for large-scale machine learning. In *USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, 2016.

[3] A. K. Agrawala. Learning with a probabilistic teacher. *IEEE Transactions on Infomation Theory*, 16:373–379, 1970.

[4] E. Alfonseca, K. Filippova, J.-Y. Delort, and G. Garrido. Pattern learning for relation extraction with a hierarchical topic model. In *Meeting of the Association for Computational Linguistics (ACL)*, 2012.

[5] S. H. Bach, B. He, A. Ratner, and C. Ré. Learning the structure of generative models without labeled data. In *International Conference on Machine Learning (ICML)*, 2017.

[6] A. Blum and T. Mitchell. Combining labeled and unlabeled data with co-training. In *Workshop on Computational Learning Theory (COLT)*, 1998.

[7] R. C. Bunescu and R. J. Mooney. Learning to extract relations from the Web using minimal supervision. In *Meeting of the Association for Computational Linguistics (ACL)*, 2007.

[8] R. Caspi, R. Billington, L. Ferrer, H. Foerster, C. A. Fulcher, I. M. Keseler, A. Kothari, M. Krummenacker, M. Latendresse, L. A. Mueller, Q. Ong, S. Paley, P. Subhraveti, D. S. Weaver, and P. D. Karp. The MetaCyc database of metabolic pathways and enzymes and the BioCyc collection of pathway/genome databases. *Nucleic Acids Research*, 44(D1):D471–D480, 2016.

[9] O. Chapelle, B. Schölkopf, and A. Zien, editors. *Semi-Supervised Learning*. Adaptive Computation and Machine Learning. MIT Press, 2009.

[10] D. Corney, D. Albakour, M. Martinez, and S. Moussa. What do a million news articles look like? In *Workshop on Recent Trends in News Information Retrieval*, 2016.

[11] N. Dalvi, A. Dasgupta, R. Kumar, and V. Rastogi. Aggregating crowdsourced binary ratings. In *International World Wide Web Conference (WWW)*, 2013.

[12] 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.

[13] A. P. Dawid and A. M. Skene. Maximum likelihood estimation of observer error-rates using the EM algorithm. *Journal of the Royal Statistical Society C*, 28(1):20–28, 1979.

[14] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, IEEE Conference on (CVPR)*, 2009.

[15] X. L. Dong and D. Srivastava. *Big Data Integration*. Synthesis Lectures on Data Management. Morgan & Claypool Publishers, 2015.

[16] L. Eadicicco. Baidu's Andrew Ng on the future of artificial intelligence, 2017. Time [Online; posted 11-January-2017].

[17] A. Graves and J. Schmidhuber. Framewise phoneme classification with bidirectional LSTM and other neural network architectures. *Neural Networks*, 18(5):602–610, 2005.

[18] S. Gupta and C. D. Manning. Improved pattern learning for bootstrapped entity extraction. In *CoNLL*, 2014.

[19] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

[20] M. A. Hearst. Automatic acquisition of hyponyms from large text corpora. In *Meeting of the Association for Computational Linguistics (ACL)*, 1992.

[21] G. E. Hinton. Training products of experts by minimizing contrastive divergence. *Neural computation*, 14(8):1771–1800, 2002.

[22] R. Hoffmann, C. Zhang, X. Ling, L. Zettlemoyer, and D. S. Weld. Knowledge-based weak supervision for information extraction of overlapping relations. In *Meeting of the Association for Computational Linguistics (ACL)*, 2011.

[23] M. Joglekar, H. Garcia-Molina, and A. Parameswaran. Comprehensive and reliable crowd assessment algorithms. In *International Conference on Data Engineering (ICDE)*, 2015.

[24] D. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[25] J. P. Ku, J. L. Hicks, T. Hastie, J. Leskovec, C. Ré, and S. L. Delp. The Mobilize center: an NIH big data to knowledge center to advance human movement research and improve mobility. *Journal of the American Medical Informatics Association*, 22(6):1120–1125, 2015.

[26] J. Lehmann, R. Isele, M. Jakob, A. Jentzsch, D. Kontokostas, P. Mendes, S. Hellmann, M. Morsey, P. van Kleef, S. Auer, and C. Bizer. DBpedia - A large-scale, multilingual knowledge base extracted from Wikipedia. *Semantic Web Journal*, 2014.

[27] H. Li, B. Yu, and D. Zhou. Error rate analysis of labeling by crowdsourcing. In *ICML Workshop: Machine Learning Meets Crowdsourcing. Atalanta, Georgia, USA*, 2013.

[28] Y. Li, J. Gao, C. Meng, Q. Li, L. Su, B. Zhao, W. Fan, and J. Han. A survey on truth discovery. *SIGKDD Explor. Newsl.*, 17(2), 2015.

[29] P. Liang, M. I. Jordan, and D. Klein. Learning from measurements in exponential families. In *International Conference on Machine Learning (ICML)*, 2009.

[30] G. S. Mann and A. McCallum. Generalized expectation criteria for semi-supervised learning with weakly labeled data. *Journal of Machine Learning Research*, 11:955–984, 2010.

[31] C. Metz. Google's hand-fed AI now gives answers, not just search results, 2016. Wired [Online; posted 29-November-2016].

[32] M. Mintz, S. Bills, R. Snow, and D. Jurafsky. Distant supervision for relation extraction without labeled data. In *Meeting of the Association for Computational Linguistics (ACL)*, 2009.

[33] D. A. P., C. J. Grondin, R. J. Johnson, D. Sciaky, B. L. King, R. McMorran, J. Wiegers, T. Wiegers, and C. J. Mattingly. The comparative toxicogenomics database: update 2017. *Nucleic Acids Research*, 2016.

[34] S. J. Pan and Q. Yang. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359, 2010.

[35] F. Parisi, F. Strino, B. Nadler, and Y. Kluger. Ranking and combining multiple predictors without labeled data. *Proceedings of the National Academy of Sciences of the USA*, 111(4):1253–1258, 2014.

[36] R. Pochampally, A. Das Sarma, X. L. Dong, A. Meliou, and D. Srivastava. Fusing data with correlations. In *ACM SIGMOD International Conference on Management of Data (SIGMOD)*, 2014.

[37] A. J. Quinn and B. B. Bederson. Human computation: A survey and taxonomy of a growing field. In *ACM SIGCHI Conference on Human Factors in Computing Systems (CHI)*, 2011.

[38] 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.

[39] T. Rekatsinas, X. Chu, I. F. Ilyas, and C. Ré. HoloClean: Holistic data repairs with probabilistic inference. *PVLDB*, 10(11):1190–1201, 2017.

[40] T. Rekatsinas, M. Joglekar, H. Garcia-Molina, A. Parameswaran, and C. Ré. SLiMFast: Guaranteed results for data fusion and source reliability. In *ACM SIGMOD International Conference on Management of Data (SIGMOD)*, 2017.

[41] S. Riedel, L. Yao, and A. McCallum. Modeling relations and their mentions without labeled text. In *European Conference on Machine Learning and Knowledge Discovery in Databases (ECML PKDD)*, 2010.

[42] B. Roth and D. Klakow. Combining generative and discriminative model scores for distant supervision. In *Conference on Empirical Methods on Natural Language Processing (EMNLP)*, 2013.

[43] V. Satopaa, J. Albrecht, D. Irwin, and B. Raghavan. Finding a "kneedle" in a haystack: Detecting knee points in system behavior. In *International Conference on Distributed Computing Systems Workshops*, 2011.

[44] H. J. Scudder. Probability of error of some adaptive pattern-recognition machines. *IEEE Transactions on Infomation Theory*, 11:363–371, 1965.

[45] B. Settles. *Active Learning*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers, 2012.

[46] R. Stewart and S. Ermon. Label-free supervision of neural networks with physics and other domain knowledge. In *AAAI Conference on Artificial Intelligence (AAAI)*, 2017.

[47] C. Sun, A. Shrivastava, S. Singh, and A. Gupta. Revisiting unreasonable effectiveness of data in deep learning era. *arXiv preprint arXiv:1707.02968*, 2017.

[48] S. Takamatsu, I. Sato, and H. Nakagawa. Reducing wrong labels in distant supervision for relation extraction. In *Meeting of the Association for Computational Linguistics (ACL)*, 2012.

[49] C.-H. Wei, Y. Peng, R. Leaman, D. A. P., C. J. Mattingly, J. Li, T. Wiegers, and Z. Lu. Overview of the BioCreative V chemical disease relation (CDR) task. In *BioCreative Challenge Evaluation Workshop*, 2015.

[50] M.-C. Yuen, I. King, and K.-S. Leung. A survey of crowdsourcing systems. In *Privacy, Security, Risk and Trust (PASSAT) and Inernational Conference on Social Computing (SocialCom)*, 2011.

[51] O. F. Zaidan and J. Eisner. Modeling annotators: A generative approach to learning from annotator rationales. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2008.

[52] C. Zhang, C. Ré, M. Cafarella, C. De Sa, A. Ratner, J. Shin, F. Wang, and S. Wu. DeepDive: Declarative knowledge base construction. *Commun. ACM*, 60(5):93–102, 2017.

[53] Y. Zhang, X. Chen, D. Zhou, and M. I. Jordan. Spectral methods meet EM: A provably optimal algorithm for crowdsourcing. *Journal of Machine Learning Research*, 17:1–44, 2016.

[54] B. Zhao, B. I. Rubinstein, J. Gemmell, and J. Han. A Bayesian approach to discovering truth from conflicting sources for data integration. *PVLDB*, 5(6):550–561, 2012.

APPENDIX

A. ADDITIONAL MATERIAL FOR SEC. 3.1

A.1 Minor Notes

Note that for the independent generative model (i.e., $|C| = 0$), the weight corresponding to the accuracy factor, w_j, for labeling function j is just the log-odds of its accuracy:

$$\begin{aligned}\alpha_j &= P(\Lambda_{i,j} = 1 \mid Y_i = 1, \Lambda_{i,j} \neq 0) \\ &= \frac{P(\Lambda_{i,j} = 1, Y_i = 1, \Lambda_{i,j} \neq 0)}{P(Y_i = 1, \Lambda_{i,j} \neq 0)} \\ &= \frac{\exp(w_j)}{\exp(w_j) + \exp(-w_j)} \\ \implies w_j &= \frac{1}{2}\log\left(\frac{\alpha_j}{1-\alpha_j}\right)\end{aligned}$$

Also note that the accuracy we consider is conditioned on the labeling function not abstaining, i.e.,:

$$P(\Lambda_{i,j} = 1 \mid Y_i = 1) = \alpha_j * P(\Lambda_{i,j} \neq \emptyset)$$

because a separate factor $\phi_{i,j}^{\text{Lab}}$ captures how often each labeling function votes.

A.2 Proof of Proposition 1

In this proposition, our goal is to obtain a simple upper bound for the expected optimal advantage $\mathbb{E}_{\Lambda,y,w^*}[A^*]$ in the low label density regime. We consider a simple model where all the labeling functions have a fixed probability of emitting a non-zero label,

$$P(\Lambda_{i,j} \neq \emptyset) = p_l \ \forall i, j \tag{4}$$

and that the labeling functions are all non-adversarial, i.e., they all have accuracies greater than 50%, or equivalently,

$$w_j^* > 0 \ \forall j \tag{5}$$

First, we start by only counting cases where the optimal weighted majority vote (WMV*)—i.e., the predictions of the generative model with perfectly estimated weights—is correct and the majority vote (MV) is incorrect, which is an upper bound on the modeling advantage:

$$\begin{aligned}&\mathbb{E}_{\Lambda,y,w^*}[A_{w^*}(\Lambda, y)] \\ &= \frac{1}{m}\sum_{i=1}^{m}\Big(\mathbb{E}_{\Lambda_i,y_i,w^*}\left[\mathbb{1}\left\{y_i f_{w^*}(\Lambda_i) > 0 \wedge y_i f_1(\Lambda_i) \leq 0\right\}\right. \\ &- \left.\mathbb{1}\left\{y_i f_{w^*}(\Lambda_i) \leq 0 \wedge y_i f_1(\Lambda_i) > 0\right\}\right]\Big) \\ &\leq \frac{1}{m}\sum_{i=1}^{m}\mathbb{E}_{\Lambda_i,y_i,w^*}\left[\mathbb{1}\left\{y_i f_{w^*}(\Lambda_i) > 0 \wedge y_i f_1(\Lambda_i) \leq 0\right\}\right]\end{aligned}$$

Next, by (5), the only way that WMV* and MV could possibly disagree is if there is at least one disagreeing pair of labels:

$$\mathbb{E}_{\Lambda,y,w^*}[A^*(\Lambda, y)] \leq \frac{1}{m}\sum_{i=1}^{m}\mathbb{E}_{\Lambda,y}\left[\mathbb{1}\left\{c_1(\Lambda_i) > 0 \wedge c_{-1}(\Lambda_i) > 0\right\}\right]$$

where $c_y(\Lambda_i) = \sum_{j=1}^{n}\mathbb{1}\{\Lambda_{i,j} = y\}$, in other words, the counts of positive or negative labels for a given data point x_i. Then, we can bound this by the expected number of disagreeing, non-abstaining pairs of labels:

$$\begin{aligned}&\mathbb{E}_{\Lambda,y,w^*}[A^*(\Lambda, y)] \\ &\leq \frac{1}{m}\sum_{i=1}^{m}\mathbb{E}_{\Lambda,y}\left[\sum_{j=1}^{n-1}\sum_{k=j+1}^{n}\mathbb{1}\left\{\Lambda_{i,j} \neq \Lambda_{i,k} \wedge \Lambda_{i,j}, \Lambda_{i,k} \neq 0\right\}\right] \\ &= \frac{1}{m}\sum_{i=1}^{m}\sum_{j=1}^{n-1}\sum_{k=j+1}^{n}\mathbb{E}_{\Lambda,y}\left[\mathbb{1}\left\{\Lambda_{i,j} \neq \Lambda_{i,k} \wedge \Lambda_{i,j}, \Lambda_{i,k} \neq 0\right\}\right] \\ &= \frac{1}{m}\sum_{i=1}^{m}\sum_{j=1}^{n-1}\sum_{k=j+1}^{n}\sum_{y'\in\pm 1}\sum_{\lambda\in\pm 1}P(\Lambda_{i,j} = \lambda, \Lambda_{i,k} = -\lambda, y_i = y')\end{aligned}$$

Since we are considering the independent model, $\Lambda_{i,j} \perp \Lambda_{i,k\neq j} \mid y_i$, we have that:

$$\begin{aligned}&P(\Lambda_{i,j} = \lambda, \Lambda_{i,k} = -\lambda, y_i = \lambda) \\ &= P(\Lambda_{i,j} = \lambda \mid y_i = \lambda)P(\Lambda_{i,k} = -\lambda \mid y_i = \lambda)P(y_i = \lambda) \\ &= \alpha_j(1-\alpha_k)p_l^2 P(y_i = \lambda)\end{aligned}$$

Thus we have:

$$\begin{aligned}\mathbb{E}_{\Lambda,y,w^*}[A^*(\Lambda, y)] &\leq \sum_{j=1}^{n-1}\sum_{k=j+1}^{n}p_l^2\left(\alpha_j(1-\alpha_k) + (1-\alpha_j)\alpha_k\right) \\ &= \sum_{j=1}^{n}\sum_{k\neq j}p_l^2\alpha_j(1-\alpha_k) \\ &\leq \sum_{j=1}^{n}\sum_{k=1}^{n}p_l^2\alpha_j(1-\alpha_k) \\ &= n^2p_l^2\bar{\alpha}(1-\bar{\alpha}) \\ &= \bar{d}^2\bar{\alpha}(1-\bar{\alpha})\end{aligned}$$

where we have defined the average labeling function accuracy as $\bar{\alpha}$, and where the label density is defined as $\bar{d} = p_l n$. Thus we have shown that the expected advantage scales at most quadratically in the label density. □

A.3 Explanation of Theorem 1

The Dawid-Skene model [13] of crowd workers classically models each crowd worker as conditionally independent, and having some class-dependent but data point-independent probability of emitting a correct label. In our setting, considering the binary classification case (as Dawid-Skene treats), we can think of each crowd worker as a labeling function, in which case we have:

$$\begin{aligned}\alpha_j^+ &= P(\Lambda_{i,j} = 1 \mid y_i = 1, \Lambda_{i,j} \neq 0) \\ \alpha_j^- &= P(\Lambda_{i,j} = -1 \mid y_i = -1, \Lambda_{i,j} \neq 0)\end{aligned}$$

The *symmetric* Dawid-Skene setting is the one we consider, where $\alpha_j \equiv \alpha_j^+ = \alpha_j^-$. Furthermore, we can refer to the matrix of probabilities of a worker j being given data point i to label (i.e., in our syntax, the probability of not abstaining) as the *sampling probability matrix*. If the entries are all the same, this is refered to as a *constant probability sampling strategy*, and is equivalent to our assumption $P(\Lambda_{i,j} \neq \emptyset) = p_l \ \forall i, j$.

In this setting, and assuming that the mean labeling function / crowd worker accuracy $\bar{\alpha} > \frac{1}{2}$ (or equivalently, $\bar{w}^* > 0$), then Corrollary 9 in [27] provides us with the following upper bound on the mean error rate:

$$\frac{1}{m}P(f_1(\Lambda_i) \neq y_i) \leq e^{-2np_l^2(\bar{\alpha}^* - \frac{1}{2})^2} \tag{6}$$

We then use (6) to upper bound the expected advantage $\mathbb{E}_{\Lambda,y,w^*}[A^*]$, substituting in $\bar{d} = np_l$ for clarity.

A.4 Proof of Proposition 2

In this proposition, our goal is to find a tractable upper bound on the conditional modeling advantage, i.e., the modeling advantage given the observed label matrix Λ. This will be useful because, given our label matrix, we can compute this quantity and, when it is small, safely skip learning the generative model and just use an unweighted majority vote (MV) of the labeling functions. We assume in this proposition that the true weights of the labeling functions lie within a fixed range, $w_j \in [w_{min} > 0, w_{max}]$ and have a mean $\bar{w}$. For notational convenience, let

$$y' = \begin{cases}1 & f_1(\Lambda) > 0 \\ 0 & f_1(\Lambda) = 0 \\ -1 & f_1(\Lambda) < 0\end{cases}$$

We start with the expected advantage, and upper-bound by the expected number of instances in which WMV* is correct and MV

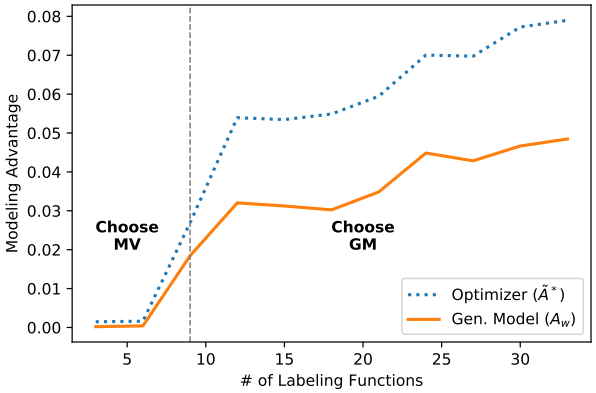

Figure 6: The advantage of using the generative labeling model (GM) over majority vote (MV) as predicted by our optimizer ($\tilde{A}^*$), and empirically (A_w), on the CDR application as the number of LFs is increased. We see that the optimizer correctly chooses MV during early development stages, and then GM in later ones.

Table 7: Number of candidates in the training, development, and test splits for each dataset.

Task	# Train.	# Dev.	# Test
Chem	65,398	1,292	1,232
EHR	225,607	913	604
CDR	8,272	888	4,620
Spouses	22,195	2,796	2,697
Radiology	3,851	385	385
Crowd	505	63	64

is incorrect (note that for tie votes, we simply upper bound by trivially assuming an expected advantage of one):

$$\begin{aligned}
&\mathbb{E}_{w^*,y}\left[A^*(\Lambda,y) \mid \Lambda\right] \\
&= \mathbb{E}_{w^*,y\sim P(\cdot \mid \Lambda,w^*)}\left[A_{w^*}(\Lambda,y)\right] \\
&\leq \frac{1}{m}\sum_{i=1}^{m}\mathbb{E}_{w^*,y\sim P(\cdot \mid \Lambda_i,w^*)}\left[\mathbb{1}\left\{y_i \neq y_i'\right\}\mathbb{1}\left\{y_i' f_{w^*}(\Lambda_i)\leq 0\right\}\right] \\
&= \frac{1}{m}\sum_{i=1}^{m}\mathbb{E}_{w^*}\left[\mathbb{E}_{y\sim P(\cdot \mid \Lambda_i,w^*)}\left[\mathbb{1}\left\{y_i \neq y_i'\right\}\right]\mathbb{1}\left\{y_i' f_{w^*}(\Lambda_i)\leq 0\right\}\right] \\
&= \frac{1}{m}\sum_{i=1}^{m}\mathbb{E}_{w^*}\left[P(y_i \neq y_i' \mid \Lambda_i,w^*)\mathbb{1}\left\{y_i' f_{w^*}(\Lambda_i)\leq 0\right\}\right]
\end{aligned}$$

Next, define:

$$\Phi(\Lambda_i,y'') = \mathbb{1}\left\{c_{y''}(\Lambda_i)w_{max} - c_{-y''}(\Lambda_i)w_{min}\right\}$$

i.e. this is an indicator for whether WMV* could *possibly* output y'' as a prediction under best-case circumstances. We use this in turn to upper-bound the expected modeling advantage again:

$$\begin{aligned}
&\mathbb{E}_{w^*,y\sim P(\cdot \mid \Lambda,w^*)}\left[A_{w^*}(\Lambda,y)\right] \\
&\leq \frac{1}{m}\sum_{i=1}^{m}\mathbb{E}_{w^*}\left[P(y_i \neq y_i' \mid \Lambda_i,w^*)\Phi(\Lambda_i,-y_i')\right] \\
&= \frac{1}{m}\sum_{i=1}^{m}\Phi(\Lambda_i,-y_i')\mathbb{E}_{w^*}\left[P(y_i \neq y_i' \mid \Lambda_i,w^*)\right] \\
&\leq \frac{1}{m}\sum_{i=1}^{m}\Phi(\Lambda_i,-y_i')P(y_i \neq y_i' \mid \Lambda_i,\bar{w})
\end{aligned}$$

Now, recall that, for $y' \in \pm 1$:

$$\begin{aligned}
P(y_i = y' \mid \Lambda_i, w) &= \frac{P(y_i = y', \Lambda_i \mid w)}{\sum_{y''\in\pm 1} P(y_i = y'', \Lambda_i \mid w)} \\
&= \frac{\exp\left(w^T\phi_i(\Lambda_i, y_i = y')\right)}{\sum_{y''\in\pm 1}\exp\left(w^T\phi_i(\Lambda_i, y_i = y'')\right)} \\
&= \frac{\exp\left(w^T\Lambda_i y'\right)}{\exp\left(w^T\Lambda_i\right) + \exp\left(-w^T\Lambda_i\right)} \\
&= \sigma\left(2f_w(\Lambda_i)y'\right)
\end{aligned}$$

where $\sigma(\cdot)$ is the sigmoid function. Note that we are considering a simplified independent generative model with only accuracy factors; however, in this discriminative formulation the labeling propensity factors would drop out anyway since they do not depend on y, so their omission is just for notational simplicity.

Putting this all together by removing the y_i' placeholder, simplifying notation to match the main body of the paper, we have:

$$\begin{aligned}
&\mathbb{E}_{w^*,y}\left[A^*(\Lambda,y) \mid \Lambda\right] \\
&\leq \frac{1}{m}\sum_{i=1}^{m}\sum_{y\in\pm 1}\mathbb{1}\left\{yf_1(\Lambda_i)\leq 0\right\}\Phi(\Lambda_i,y)\sigma\left(2yf_{\bar{w}}(\Lambda_i)\right) \\
&= \tilde{A}^*(\Lambda)\ \square.
\end{aligned}$$

A.5 Modeling Advantage Notes

In Figure 6, we measure the modeling advantage of the generative model versus a majority vote of the labeling functions on random subsets of the CDR labeling functions of different sizes. We see that the modeling advantage grows as the number of labeling functions increases, indicating that the optimizer can save execution time especially during the initial stages of iterative development.

Note that in Section 4, due to known negative class imbalance in relation extraction problems, we count instances in which the generative model emits no label—i.e., a 0 label—as negatives, as is common practice (essentially, we are giving the generative model the benefit of the doubt given the known class imbalance). Thus our reported F1 score metric hides instances in which the generative model learns to apply a -1 label where majority vote applied 0. In computing the empirical modeling advantage, however, we *do* count such instances as improvements over majority vote, as these instances *do* have an effect on the training of the end discriminative model.

B. ADDITIONAL EVALUATION DETAILS

B.1 Data Set Details

Additional information about the sizes of the datasets are included in Table 7. Specifically, we report the size of the (unlabeled) training set and hand-labeled development and test sets, in terms of number of candidates. Note that the development and test sets can be orders of magnitude smaller that the training sets. Labeled development and test sets were either used when already available as part of a benchmark dataset, or labeled with the help of our SME collaborators, limited to several hours of labeling time maximum.

B.2 User Study

Figures 7 and 8 show the distribution of scores by participant, and broken down by participant background, compared against the baseline models trained with hand-labeled data. Figure 8 shows descriptive statistics of user factors broken down by their end model's predictive performance.

C. IMPLEMENTATION DETAILS

Note that all code is open source and available—with tutorials, blog posts, workshop lectures, and other material—at `snorkel.stanford.edu`.

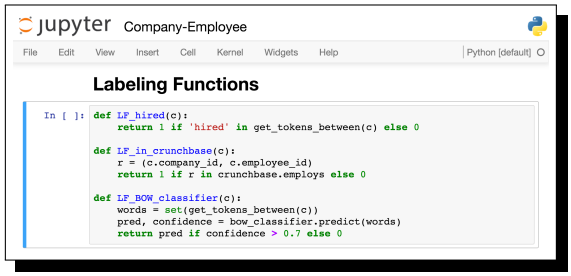

Figure 9: Labeling functions which express pattern-matching, distant supervision, and weak classifier heuristics, respectively, in Snorkel's Jupyter notebook interface.

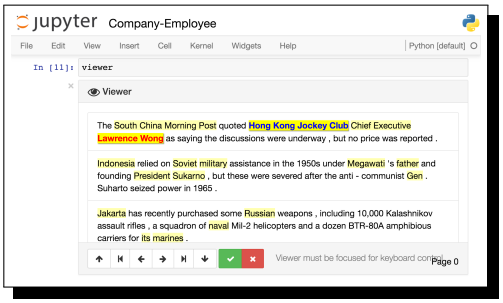

Figure 10: The Viewer utility in Snorkel, showing candidate company-employee relation mentions from the ACE benchmark, composed of candidate person and company mention pairs.

Table 8: Self-reported skill levels—beginner (Beg.), intermediate (Int.), and advanced (Adv.)—for all user study participants.

Subject	New	Beg.	Int.	Adv.
Python	0	3	8	4
Machine Learning	5	1	4	5
Info. Extraction	2	6	5	2
Text Mining	3	6	4	2

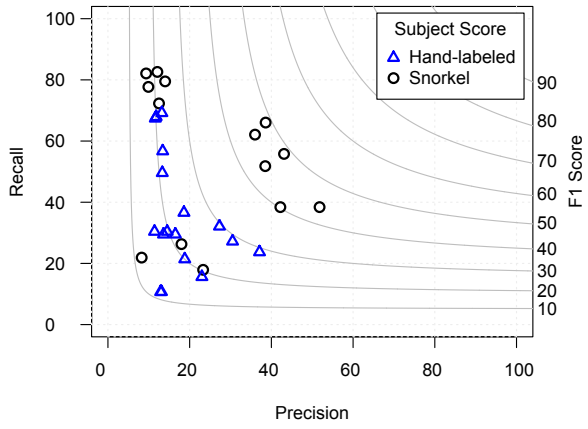

Figure 7: Predictive performance of our 14 user study participants. The majority of users matched or exceeded the performance of a model trained on 7 hours (2500 instances) of hand-labeled data.

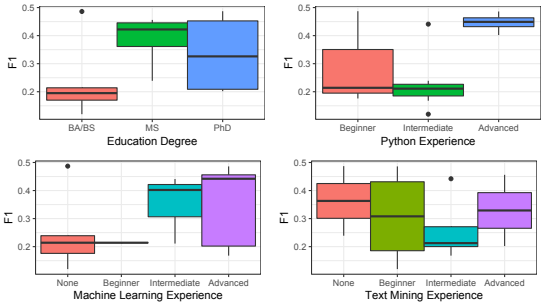

Figure 8: The profile of the best performing user by F1 score, was a MS or Ph.D. degree in any field, strong Python coding skills, and intermediate to advanced experience with machine learning. Prior experience with text mining added no benefit.

Interface Implementation. Snorkel's interface is designed to be accessible to SMEs without advanced programming skills. All components run in Jupyter (`http://jupyter.org/`) iPython notebooks, including writing labeling functions. Users can therefore write labeling functions as arbitrary Python functions for maximum flexibility (Figure 9). We also provide a library of labeling function primitives and generators to declaratively program weak supervision.

A key aspect of labeling function development is that the process is iterative. After developing an initial set of labeling functions, it is important for users to visualize the errors of the end model. Therefore, when the model is evaluated on the development data set, the candidates are separated into true positive, false positive, true negative, and false negative sets. Each of these buckets can be loaded into a viewer in a notebook (Figure 10) so that SMEs can identify common patterns that are either not covered or misclassified by their current labeling functions. The viewer also supports labeling candidates directly in order to create or expand development and test sets.

Execution Model. Since labeling functions are self-contained and operate on discrete candidates, their execution is embarrassingly parallel. If Snorkel is connected to a relational database that supports simultaneous connections, e.g., PostgreSQL, then the master process (usually the notebook kernel) distributes the primary keys of the candidates to be labeled to Python worker processes. The workers independently read from the database to materialize the candidates via the ORM layer, then execute the labeling functions over them. The labels are returned to the master process which persists them via the ORM layer. Collecting the labels at the master is more efficient than having workers write directly to the database, due to table-level locking.

Snorkel includes a Spark (`https://spark.apache.org/`) integration layer, enabling labeling functions to be run across a cluster. Once the set of candidates is cached as a Spark data frame, only the closure of the labeling functions and the resulting labels need to be communicated to and from the workers. This is particularly helpful in Snorkel's iterative workflow. Distributing a large unstructured data set across a cluster is relatively expensive, but only has to be performed once. Then, as users refine their labeling functions, they can be rerun efficiently.

This same execution model is supported for preprocessing utilities—such as natural language processing for text and candidate extraction—via a common class interface. Snorkel provides wrappers for Stanford CoreNLP (`https://stanfordnlp.github.io/CoreNLP/`) and SpaCy (`https://spacy.io/`) for text preprocessing, and supports automatically defining candidates using their named-entity recognition features.