Snorkel DryBell: A Case Study in Deploying Weak Supervision at Industrial Scale

Stephen H. Bach[†]   Daniel Rodriguez[‡]   Yintao Liu[‡]   Chong Luo[‡]

Haidong Shao[‡]   Cassandra Xia[‡]   Souvik Sen[‡]   Alex Ratner[§]

Braden Hancock[§]   Houman Alborzi[‡]   Rahul Kuchhal[‡]   Chris Ré[§]   Rob Malkin[‡]

[†]Brown University   [‡]Google   [§]Stanford University

ABSTRACT

Labeling training data is one of the most costly bottlenecks in developing machine learning-based applications. We present a first-of-its-kind study showing how existing knowledge resources from across an organization can be used as weak supervision in order to bring development time and cost down by an order of magnitude, and introduce Snorkel DryBell, a new weak supervision management system for this setting. Snorkel DryBell builds on the Snorkel framework, extending it in three critical aspects: flexible, template-based ingestion of diverse organizational knowledge, cross-feature production serving, and scalable, sampling-free execution. On three classification tasks at Google, we find that Snorkel DryBell creates classifiers of comparable quality to ones trained with tens of thousands of hand-labeled examples, converts non-servable organizational resources to servable models for an average 52% performance improvement, and executes over millions of data points in tens of minutes.

KEYWORDS

Systems for machine learning, weak supervision

1 INTRODUCTION

One of the most significant bottlenecks in developing machine learning applications is the need for hand-labeled training data sets. In industrial and other organizational deployments, the cost of labeling training sets has quickly become a significant capital expense: collecting labels at scale requires carefully developing labeling instructions that cover a wide range of edge cases; training subject matter experts to carry out those instructions; waiting sometimes months or longer for the full results; and dealing with the rapid depreciation of training sets as applications shift and evolve.

As a result, in industry and other domains there has been a major movement towards programmatic or otherwise more efficient but noisier ways of generating training labels, often referred to as *weak supervision.* Given the increasing commoditization of standard machine learning model architectures, the supervision strategy used is increasingly the key differentiator for end model performance, and recently has been a key element in state-of-the-art results [8, 20]. Many prior weak supervision approaches rely on a single source of labels, a small number of carefully chosen, manually combined sources [23, 40], or on sources that make uncorrelated errors such as independent crowd workers [9, 10]. Recent work has explored building end-to-end systems for programmatic supervision, e.g., simple heuristic rules and pattern matching scripts written from scratch by users, which may have diverse accuracies and correlations [3, 29, 30].

In this work, we present a first-of-its-kind study showing how existing organizational knowledge can be used as weak supervision to have significant impact even in some of the most heavily-engineered, high-value industrial ML applications. We introduce Snorkel DryBell, a production-scale *weak supervision management system* which extends the Snorkel framework [29] to handle three novel aspects we find to be critical: flexible template-based ingestion of *organizational knowledge*, *cross-feature* production serving, and scalable, *sampling-free* modeling. We evaluate Snorkel DryBell on three content and real-time event classification
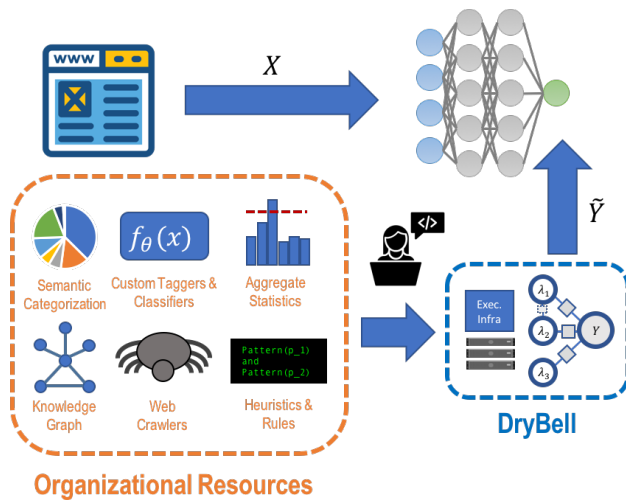
Figure 1: Rather than using hand-labeled training data, `Snorkel DryBell` uses diverse organizational resources as weak supervision to train content and event classifiers on Google's platform.

applications at Google (Figure 1), where we move beyond using simple pattern matchers over text as weak supervision for small-scale, first-cut applications (as in initial work on Snorkel [29]), and demonstrate the value of using existing organizational knowledge resources, measured against baselines with person-decades of development.

Based on our experience at Google, we outline three core principles that are central to deploying weak supervision at organizational scale, and highlight how these are implemented in `Snorkel DryBell`:

- **Flexible Ingestion of Organizational Knowledge**: In large organizations, a wide range of resources—such as models, knowledge bases, heuristics, and more—are often available; a weak supervision system should support rapid and flexible integration of as many of these resources as possible for quickly training models to the highest possible quality. We highlight the importance of this approach with three case studies involving content and event classification tasks. Engineers at Google are responsible for hundreds of separate classifiers, which often rely on hand-labeled training data. They must be responsive to everything from shifting business objectives to changes in products; updating these classifiers is often the critical blocker to core product and feature launches. We describe how a single engineer can use weak supervision to rapidly develop new classifiers, leading to average, relative quality improvements of 11.5% (measured in F1 points) over classifiers trained on small ~15K-example development sets, and reaching the quality equivalent of using 80K labels.

- **Cross-Feature Production Serving**: Organizational knowledge is often present in non-servable form factors,

i.e., too slow, expensive, or private to be used in production; instead, a weak supervision system can provide a way to use these to quickly train *servable* models suitable for deployment. For example, internal models or heuristics are often defined over features like monthly aggregate statistics, expensive internal models, etc., whereas `Snorkel DryBell` can allow users to quickly transfer this knowledge to models defined over *servable* features, e.g., inexpensive, real-time signals. We demonstrate how `Snorkel DryBell` allows users to quickly and flexibly transfer organizational knowledge from non-servable forms to new servable deployment models focused on the classification task of interest. We view this as a practical, flexible form of *transfer learning*, and show that incorporating these resources leads to 52% average gains in performance.

- **Scalable, Sampling-Free Execution**: A weak supervision system should cleanly decouple *subject matter experts (SMEs)*, who should be able to rapidly and iteratively specify weak supervision, from the details of execution and model training over industrial scale datasets. We describe how the architecture of `Snorkel DryBell` cleanly decouples the interface by which SMEs across an organization can contribute labeling strategies, and the system for executing these at massive scale while supporting rapid human-in-the-loop iteration—for example, implementing weak supervision over 6M+ data points with sub-30min. execution time—including a new TensorFlow compute graph-based generative modeling approach that avoids expensive sampling, and a MapReduce-based pipeline.

We achieve these principles in `Snorkel DryBell` by adopting the three main stages of the Snorkel pipeline: first, users write *labeling functions*, which are simply black-box functions that take in unlabeled data points and output a label or abstain, and can be used to express a wide variety of weak supervision strategies; next, a generative modeling approach is used to estimate the accuracies of the different labeling functions based on their observed agreements and disagreements; and finally, these accuracies are used to re-weight and combine the labels output by the labeling functions, producing *probabilistic* (confidence-weighted) training labels.

We start in Section 2 with a brief description of existing work on weak supervision, and of the approach taken by Snorkel, the framework that `Snorkel DryBell` extends. In Section 3, we present three case studies of content and event classification applications at Google, where we survey the categories of weak supervision strategies that can be employed within `Snorkel DryBell`. We discuss these case studies at a high level due to the proprietary nature of the applications. In Section 4, we highlight a particularly critical *cross-feature* form of production model serving supported

in `Snorkel DryBell`, in which *non-servable* supervision resources that are expensive to run, private, or otherwise not servable in production are used to train *servable* deployment models. In Section 5, we then present the architecture of `Snorkel DryBell`, emphasizing a new sampling-free generative modeling approach, and a MapReduce-based pipeline and template library. In Section 6, we describe experimental results on benchmark data sets using Google data representative of production tasks. We show that `Snorkel DryBell` can replace hand-labeling tens of thousands of training examples. Finally, we conclude with lessons learned on how weakly supervised machine learning can be integrated into the development processes of production machine learning applications, and discuss how these lessons can be applied at many different kinds of organizations.

2 BACKGROUND

In recent years, modern machine learning models have achieved new state-of-the-art accuracies on a range of traditionally challenging tasks. However, these models generally require massive hand-labeled training sets [37]. In response, many machine learning developers have increasingly turned to *weaker* methods of supervision, in which a larger volume of cheaper, noisier labels is used [6, 9, 10, 23, 24, 29, 40].

We build on top of Snorkel, a recently proposed framework for weakly supervised machine learning [29], which allows users to generically specify multiple sources of programmatic weak supervision—such as rules and pattern matchers over text—that can vary in accuracy, coverage, and that may be arbitrarily correlated. The Snorkel pipeline follows three main stages, which we also adopt in `Snorkel DryBell`: first, users write *labeling functions*, which are simply black-box functions that take in unlabeled data points and output a label or abstain; next, a *generative model* is used to estimate the accuracies of the different labeling functions, and then to re-weight and combine their labels to produce a set of *probabilistic* training labels, effectively solving a novel data cleaning and integration problem; and finally, these labels are use to train an arbitrary end *discriminative model*, which is used as the final classifier in production.

This setup can be formalized as follows. Let $X = (X_1, \ldots, X_m)$ be a collection of unlabeled data points, $X_i \in \mathcal{X}$, with associated *unobserved* labels $Y = (Y_1, \ldots, Y_m)$. For simplicity, we focus on binary classification, $Y_i \in \{-1, 1\}$, however `Snorkel DryBell` can handle arbitrary categorical targets as well, e.g. $Y_i \in \{1, \ldots, k\}$.

In our weak supervision setting, we do not have access to these ground-truth labels Y_i, and our goal is to estimate them to use as training labels. Instead, we have access to n labeling functions $\lambda = (\lambda_1, \ldots, \lambda_n)$, where $\lambda_j : \mathcal{X} \rightarrow \{-1, 0, 1\}$, with 0 corresponding to an *abstain* vote. We use a *generative model*

wherein we model each labeling function as abstaining or not with some probability, and labeling a data point correctly with some probability. Let Λ be the matrix of labels output by the n labeling functions over the m unlabeled data points, such that $\Lambda_{i,j} = \lambda_j(X_i)$. We then estimate the parameters w of this generative labeling model $P_w(\Lambda, Y)$ by maximizing the log marginal likelihood of the observed labels Λ:

$$\hat{w} = \arg \max_w \ \log \sum_{Y \in \{-1,1\}^m} P_w(\Lambda, Y) \ .$$

Note that we are marginalizing out Y, i.e. we are not using any ground truth training labels in our learning procedure; instead, we are learning solely from the information about agreements and disagreements between the labeling functions, as contained in the observed label matrix Λ. We discuss the choice of the structure of $P_w(\Lambda, Y)$ and the unsupervised approach to estimating $\hat{w}$ further in Section 5.2.

Given the estimated generative model, we use its predicted label distributions, $\tilde{Y}_i = P_{\hat{w}}(Y_i | \Lambda)$, as *probabilistic* training labels for the end *discriminative classifier* that we aim to train. We train this discriminative classifier h_θ on our weakly labeled training set, $(X, \tilde{Y})$, by minimizing a *noise-aware* variant of a standard loss function, l, i.e. we minimize the expected loss with respect to $\tilde{Y}$:

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

A formal analysis shows that as the number of unlabeled data, i.e. m, is increased, the generalization error of the discriminative classifier should decrease at the same asymptotic rate as it would if supervised with traditional hand-labeled data [30]. More generally, we expect the discriminative classifier to provide performance gains over the generative model (i.e. the reweighted combination of the labeling function outputs) that it is trained on, both by applying to data types that the labeling functions cannot be applied to, e.g. servable versus non-servable features (see Section 4), and by learning to generalize beyond them. For example, the discriminative classifier can learn to put weight on more subtle or synonymous features that the labeling functions (and thus, the generative model) do not cover. For empirical evidence of this generalization, and further details of the actual discriminative models used, see Section 6.

In building `Snorkel DryBell`, we sought to extend Snorkel to study weak supervision in the context of an organizational-scale deployment, focusing on three key aspects. First, while Snorkel was designed to handle "de novo" weak supervision applications built with a handful of simple pattern-matching rules, written from scratch by domain experts, we design a template-based interface for ingesting existing organizational knowledge resources like internal models, taggers, legacy code, and more (Section 3). Second, we implement
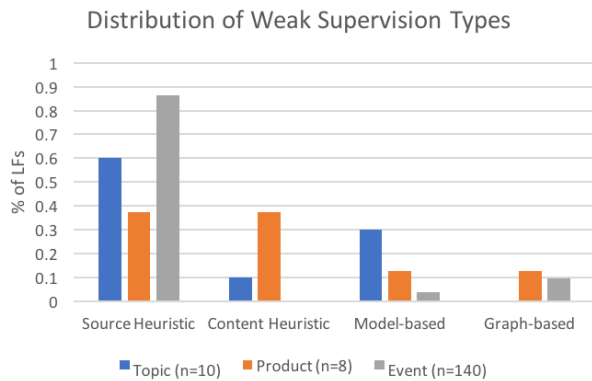
Figure 2: We plot the distribution of high-level categories of weak supervision types, counted by number of *labeling functions (LFs)*, for the three applications.

support for cross-feature production serving, where weak supervision defined over features not *servable* for application deployment—such as aggregate statistics or expensive results of model inference—can be transferred to deployable models defined over *servable* features (Section 4). Finally, our design of the Snorkel DryBell architecture focuses on handling massive scale (e.g. 6.5M data points in one application), and thus we focus on speeding up both labeling function execution and generative model training, in particular using a new sampling-free modeling approach (Section 5).

3 CASE STUDIES: WEAK SUPERVISION FOR RAPID DEVELOPMENT

We start by exploring three case studies of weak supervision applied to classification problems at Google: two on content classification related to topics and commercial product categories, and one for classifying real-time events across several serving platforms. In this section, we focus on highlighting the diversity of weak supervision signals from across the organization that developers were able to express as *labeling functions (LFs)* in Snorkel DryBell. We broadly categorize the weak supervision sources into several coarse-grained buckets, representing different types of organizational knowledge and resources (Figure 2):

- **Source Heuristics:** Labeling functions expressing heuristics (pattern or otherwise) about the *source* of the content or event, or aggregate statistics of this.
- **Content Heuristics:** Labeling functions expressing heuristics about the content or event.
- **Model-Based:** Labeling functions that use the predictions of internal models which were developed for some related or component problem. Examples include topic models and named entity recognizers applied to content.

- **Graph-Based:** Labeling functions that use a knowledge or entity graph to derive labels.

We now describe the applications, giving examples of the above weak supervision source types used in each.

3.1 Topic Classification

In the first task, an engineering team for a Google product needed to develop a new classifier to detect a topic of interest in its content. The team oversees well over 100 such classifiers, each with its own set of training data, so there is strong motivation for finding faster and more agile ways to develop or modify these models. Currently, however, the default procedure for developing a new classifier such as this one requires substantial manual data labeling effort.

In our study, we instead used Snorkel DryBell to weakly supervise 684,000 unlabeled data points, selected by a coarse-grained initial keyword-filtering step. A developer then spent a short time writing ten labeling functions that both expressed basic heuristics, and pulled on organizational resources such as existing models at Google. Specific examples of labeling functions included:

- *URL-based:* Heuristics based on the linked URL;
- *NER tagger-based:* Heuristics over entities tagged within the content, using custom named entity recognition (NER) models maintained internally at Google;
- *Topic model-based:* Heuristics based on a topic model maintained internally at Google. This topic model output semantic categorizations far too coarse-grained for the targeted task at hand, but which nonetheless could be used as effective negative labeling heuristics.

These weak supervision strategies pulled on diverse types of signal from across Google's organization, but were simple to write within the Snorkel DryBell framework. With these strategies, we matched the performance of 80K hand-labeled training labels, and get within 4.6 F1 points of a model trained on 175K hand-labeled training data points (see Section 6).

3.2 Product Classification

In a second case study with the same engineering team at Google, a strategic decision necessitated a modification of an existing classifier for detecting content references to products in a category of interest. The category of interest was expanded to include many types of accessories and parts—meaning that all previously negative class labels (i.e., "not in the category of interest") needed to be relabeled, or else discarded. In fact, our post-hoc experiments revealed that even using the previously positive labels resulted in a slight reduction in end model F1 score, highlighting the near-instantaneous depreciation of a significant labeling investment given a change in strategy.

Instead, in a similar process to the content classification scenario described above, one developer was able to write eight labeling functions, leveraging diverse weak supervision resources from across the organization. These labeling functions included:

- *Keyword-based*: Keywords in the content indicated either products and accessories in the category of interest, or other accessories not of interest;
- *Knowledge Graph-based*: In order to increase coverage across the many languages for which this classifier is used, we queried Google's Knowledge Graph for translations of keywords in ten languages;
- *Model-based*: We again used the semantic topic model to identify content obviously unrelated to the category of products of interest.

A classifier trained with these labeling functions matched the performance of 12K hand-labeled training examples, and got within 5.1 F1 points of classifier model trained on 50K hand-labeled training examples (see Section 6).

3.3 Real-Time Event Classification

Finally, we applied `Snorkel DryBell` to a real-time events classification problem over two of Google's platforms. In this setting, a common approach is to classify events based on offline (or *non-servable*) features such as aggregate statistics and relationship graphs. However, this approach induces latency between when an event occurs and when it is identified. An alternative approach is to use a machine learning model to classify events directly from real-time, *event-level* features. However, getting hand-labeled training data in this setting is challenging due to the shifting environment, as well as the cost of trained expert annotators. Instead, we used `Snorkel DryBell` to train models over the event-level features using weak supervision sources (n=140) defined over the non-servable features, spanning three broad categories:

- *Model-based*: Several smaller models that had previously been developed over various feature sets were also used as weak labelers in this setting.
- *Graph-based*: A set of models over graphs of entity and destination relationships provided higher recall but generally lower-precision signals than the heuristic classifiers.
- *Other heuristics*: A large set of existing heuristic classifiers that had previously been developed.

These sources were combined in `Snorkel DryBell` and used to train a deep neural network over real-time event-level features. Compared to the same network trained on an unweighted combination of the labeling functions, `Snorkel DryBell` identifies 58% more events of interest, with a quality improvement of 4.5% according to an internal metric.

One of the aspects that we found critical in this setting was the ability of `Snorkel DryBell` to estimate the accuracies of
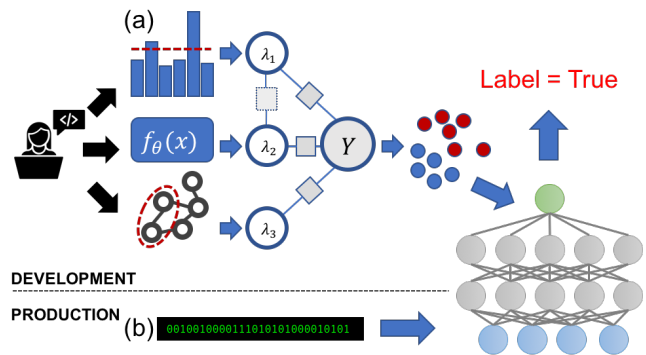

Figure 3: In `Snorkel DryBell`, developers can use *non-servable* development features for weak supervision, to train classifiers that operate over separate *servable* features in production.

different labeling functions. Given the large number of weak supervision sources in play, determining the quality or utility of each source, and tuning their combinations accordingly, would have itself been an onerous engineering task. Using `Snorkel DryBell`, these weak supervision signals could simply all be integrated as labeling functions, and the resulting estimated accuracies were found to be independently useful for identifying previously unknown low-quality sources (which were then either fixed or removed).

4 CROSS-FEATURE MODEL SERVING

One significant advantage of a weakly supervised approach, as implemented in `Snorkel DryBell`, is the ability to easily and flexibly transfer knowledge contained in *non-servable* feature sets that are too slow, expensive, or private to use in production, to *servable* feature sets such as real-time event-level signals or cheap edge-computable features, as in the real-time event application (Figure 3). Formally, this goal of transferring knowledge from a model defined over one feature set to a new model trained over another feature set can be viewed as a type of transfer learning [27], or as similar to a transductive form of model distillation [15]. However, most commonly used transfer learning techniques today apply to models with similar or identical architectures defined over the same basic feature set. Instead, with `Snorkel DryBell` we can quickly use models over one set of features—for example, aggregate statistics, results of expensive crawlers, internal models or graphs—and use these to supervise a new model over external, cheap, or otherwise *servable* features.

In the applications we survey at Google, this is an essential element. In the real-time events case study, as outlined in the preceding section, none of the weak supervision sources are directly applicable to the event-level, real-time, servable features of interest; instead, with `Snorkel DryBell` we use

them to train a new model that is defined over these servable features. For the two content applications, while some of the labeling functions could be applied at test time over servable features, others—specifically, those comprising internal models that are expensive to run, or features obtained with high-latency such as the result of web crawlers—are effectively non-servable. By incorporating the signal from these non-servable sources in `Snorkel DryBell`, we get average gains of 52% in final F1 score performance according to an ablation. We find that this ability to bridge the gap between non-servable organizational resources and servable models is one of the major advantages of a weak supervision approach like the one implemented in `Snorkel DryBell`.

5 SYSTEM ARCHITECTURE

Deploying the Snorkel framework proposed by Ratner et. al. [29] required redesigning its implementation for an industrial, distributed computing environment, where the scale of examples (millions) is at least an order of magnitude larger than any reported data set for which Snorkel has been used. This required decoupling and redesigning the labeling function execution and generative modeling components of the pipeline around a template library and distributed compute environment, which we detail next.

5.1 Labeling Function Template Library

We implement support for user-defined labeling functions as a library of templated C++ classes. Our goal is to abstract away the repeated development of code for reading and writing to Google's distributed filesystem, and for executing MapReduce pipelines. We achieve this by implementing an `AbstractLabelingFunction` class that handles all input and output to Google's distributed filesystem. Each subclass defines a MapReduce pipeline, with class template slots for functions to be executed within the pipeline. We initially developed two labeling function pipelines.

The first pipeline is a default pipeline that does not launch any additional services; it simply executes a user-defined function written in C++ (`LabelingFunction`). This class meets the needs of many use cases, such as content heuristics, model-based heuristics for models that are executed offline as part of data collection such as semantic categorization, and graph-based heuristics that can query a knowledge graph offline (e.g., categories of products in top-ten languages).

The second pipeline integrates with Google's general-purpose natural language processing (NLP) models (`NLPLabelingFunction`). Such integration is necessary because these NLP models are too computationally expensive to run for all content submitted to Google. `Snorkel DryBell`

therefore needs to enable labeling-function writers to execute additional models in a manner that scales to the millions of examples to be labeled. To achieve this goal, `Snorkel DryBell` uses Google's MapReduce framework to launch a model server on each compute node. Other model servers besides the NLP models can be supported by creating new subclasses of `AbstractLabelingFunction`.

Engineers using this library need to write only simple main files that define the function(s) that computes the labeling function's vote for an individual example. These functions capture the engineer's knowledge about how to use existing resources at Google as heuristics for weak supervision. As an example that is analogous to a labeling function in our content classification application, suppose our goal is to identify content related to celebrities. A developer can implement a heuristic that uses a named-entity recognition model for this task as an instance of `NLPLabelingFunction`. The labeling function labels any content that *does not* contain a person as *not* related to celebrities. The first template argument is a pointer to a function that takes an example object as input and selects the text to be provided to the NLP model server. The second template argument is a pointer to a function that takes the same example object and the output of the NLP models as its inputs, and checks whether the named-entity recognition model found any proper names of people. We illustrate this example in code:

```cpp
string GetText(const Example& x) {
    return StrCat(x.title, " ", x.body);
}

LFVote GetValue(const Example& x,
                const NLPResult& nlp) {
    if (nlp.entities.people.size() == 0) {
        return NEGATIVE;
    }
    else { return ABSTAIN; }
}

int main(int argc, char *argv[]) {
    Init(argc, argv);
    NLPLabelingFunction<&GetText, &GetValue> lf;
    lf.Run();
}
```

This short bit of code captures a logical relationship between an existing model and the target task, speeding development.

5.2 Sampling-Free Generative Model

The critical task in `Snorkel DryBell` is to combine the noisy votes of the various labeling functions into estimates of the true labels for training. In Snorkel DryBell, we use a new sampling-free modeling approach which is far less CPU intensive and far simpler to distribute across compute nodes. We focus on a conditionally independent generative
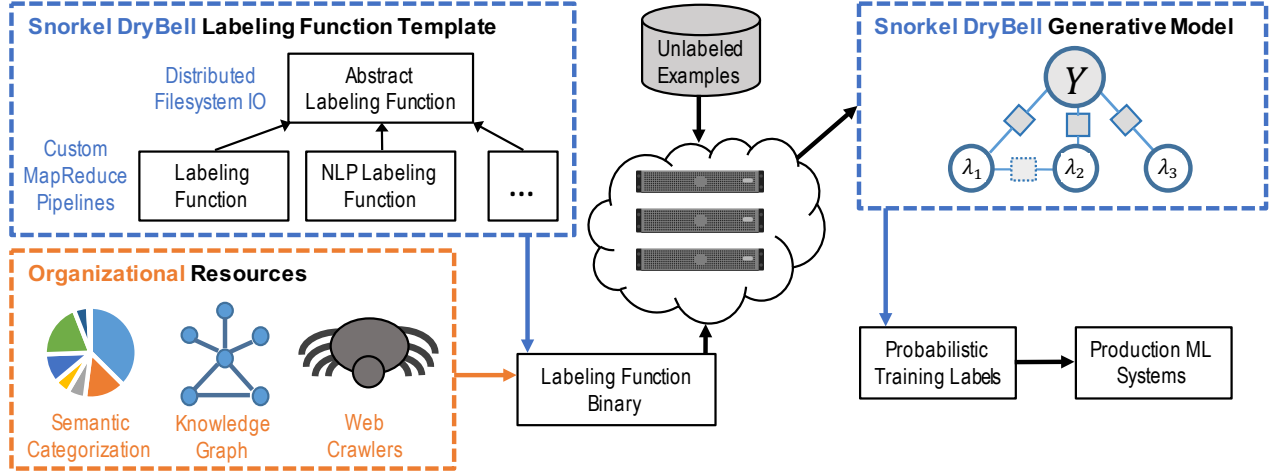
Figure 4: An overview of the Snorkel DryBell system. (1) Snorkel DryBell provides a library of templated C++ classes, each of which defines a MapReduce pipeline for executing a labeling function with the necessary services, such as natural language processing (NLP). (2) Engineers write methods for the MapReduce pipeline to determine a vote for each example's label, using Google resources. (3) Snorkel DryBell executes the labeling function binary on Google's distributed compute environment. (4) Snorkel DryBell loads the labeling functions' output into its generative model, which combines them into probabilistic training labels for use by production systems.

model, which we write as:

$$P_w(\Lambda, Y) = \prod_{i=1}^{m} P_w(Y_i) \prod_{j=1}^{n} P_w(\lambda_j(X_i)|Y_i),$$

Following prior work [29], we assume each labeling function has an accuracy given that it did not abstain, and a propensity to not abstain, i.e., we share parameters across the conditional distributions. For simplicity, here we assume that $P_w(Y_i)$ is uniform, but we can also learn this distribution.

The learning objective of the generative model is to minimize the negative marginal log-likelihood of the observed labeling function outputs $-\log P(\Lambda)$. The open-source Snorkel implementation[1] uses a Gibbs sampler to compute the gradient of this likelihood, but sampling is relatively CPU intensive and complicated to distribute across compute nodes. Instead, we design a new TensorFlow-based [1] implementation for sampling-free optimization. For numeric stability, we represent the model parameters in log space. Let α_j be the unnormalized log probability that labeling function j is correct given that it did not abstain, and let β_j be the unnormalized log probability that it did not abstain. Then, to define a static compute graph, as required by TensorFlow, we use 0-1 indicator functions for each possible label value

---

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

and multiply by the corresponding log likelihood:

$$-\log P(\Lambda) = -\sum_{i=1}^{m} \log \left(P(\Lambda_i, Y_i = 1) + P(\Lambda_i, Y_i = -1) \right),$$

where

$$\log P(\Lambda_i, Y_i = 1) = \sum_{j=1}^{n} (\mathbf{1}[\lambda_j(X_i) = 1](\alpha_j + \beta_j - Z_j) \\ + \mathbf{1}[\lambda_j(X_i) = -1](-\alpha_j + \beta_j - Z_j) \\ - \mathbf{1}[\lambda_j(X_i) = 0]Z_j),$$

$$\log P(\Lambda_i, Y_i = -1) = \sum_{j=1}^{n} (\mathbf{1}[\lambda_j(X_i) = 1](-\alpha_j + \beta_j - Z_j) \\ + \mathbf{1}[\lambda_j(X_i) = -1](\alpha_j + \beta_j - Z_j) \\ - \mathbf{1}[\lambda_j(X_i) = 0]Z_j),$$

$$Z_j = \log(\exp(\alpha_j + \beta_j) + \exp(-\alpha_j + \beta_j) + 1).$$

The result is a fast implementation that can take hundreds of gradient steps per second on a single compute node. For example, in our product classification application, in which there are ten labeling functions, the optimizer takes an average > 100 steps per second with a batch size of 64. With ten labeling functions and a batch size of 64, a Gibbs sampler averages < 50 examples per second, so Snorkel DryBell provides a 2× speedup. Implementing the generative model

as a static compute graph in TensorFlow has another advantage over a Gibbs sampler. It is easy to parallelize across multiple compute nodes using TensorFlow's API. (Here we report timing using a single process for a fair comparison.)

It is also possible to relax the conditional independence assumption by defining model functions in TensorFlow that capture specific, low-tree-width graphical model structures, which we leave for future work. It is also possible to directly plug-in matrix factorization models of the kind recently used for denoising labeling functions [31] as TensorFlow model functions.

5.3 Discriminative Model Serving

To create discriminative models that are servable in production, we integrated `Snorkel DryBell` with TFX [4], Google's platform for end-to-end production-scale machine learning. The probabilistic training labels estimated by `Snorkel DryBell` are passed to TFX, where users can configure a model to train with a noise-aware loss function. Once trained, we use TFX to automatically stage it for serving.

As described in Section 4, the discriminative model acts on a more compact feature representation than the labeling functions, enabling a cross-feature transfer of knowledge from non-servable resources used in labeling functions to a servable model. TFX supports both logistic regression and deep neural network models, which can operate on user-specified features that are available in production, or on the "raw" content, e.g., an LSTM [16] that embeds each token of text in a vector space.

5.4 Comparison with Snorkel Architecture

There are several other key differences between `Snorkel DryBell` and Snorkel's existing open-source implementation beyond the changes detailed above. Snorkel is designed to run on a single, shared-memory compute node. In contrast, Google, like many large organizations, uses a distributed job scheduling and accounting system for large-scale computing. It therefore was necessary to integrate `Snorkel DryBell` with Google's MapReduce framework.

Further, Snorkel is designed to be accessible to novice programmers with limited machine learning experience. It uses a Jupyter notebook interface and enforces a strict *context hierarchy* data model for representing training data. This rigid approach is not appropriate for the wide range of tasks that arise in a large organization. Snorkel also uses a relational database backend for storing data, which does not easily integrate with Google's existing data-storage systems. We therefore developed the more loosely coupled system described above, in which labeling functions are independent executables that use a distributed filesystem to share data.

Table 1: Number of unlabeled examples used during training n, number of labeled examples in the development set n_{Dev} and test set n_{Test}, fraction of positive labels in n_{Test}, and number of labeling functions used for each task, for the content classification applications.

Task	n	n_{Dev}	n_{Test}	% Pos.	# LFs
Topic Classification	684K	11K	11K	0.86	10
Product Classification	6.5M	14K	13K	1.48	8

6 EXPERIMENTS

To evaluate the performance of `Snorkel DryBell`, we created benchmark data sets using Google data representative of the production tasks described in Section 3. We first show results on the content classification applications, and use them to illustrate trade-offs between weak supervision and collecting hand-labeled data, as well as the benefits of using non-servable features for weak supervision. We then show results on the real-time events application. Due to the sensitive nature of these applications, we report relative improvement to our baselines for the content classification applications. We are unable to describe the details of internal metrics used to evaluate real-time events, but include a high-level description as corroborating evidence that `Snorkel DryBell` is widely applicable.

6.1 Topic and Product Classification

To evaluate on the topic and product classification applications, we used the probabilistic training labels estimated by `Snorkel DryBell` to train logistic regression discriminative classifiers with servable features similar to those used in production. We have access to hundreds of thousands to millions of unlabeled examples for these tasks. We also create a small, hand-labeled *development set* (n_{Dev} in Table 1) which is used by the developer while formulating labeling functions, for hyperparameter tuning of the end discriminative classifier, and as a supervised learning baseline in our experiments.

We use a logistic regression model in TFX. We train using the FTLR optimization algorithm [22], a variant of stochastic gradient descent that tunes per-coordinate learning rates, with an initial step size of 0.2. We train for 10K iterations for the topic classification task and 100K iterations for the product classification task, in order to have a similar training time to production classifiers. (The topic classification task has an order-of-magnitude more features than the product classification task.) All experiments use a batch size of 64.

Table 2 shows the results of applying the `Snorkel DryBell` system on the product and topic classification tasks. We report all results relative to the baseline approach of training

Table 2: Evaluation of `Snorkel DryBell` on content classification tasks, optimizing for F1 score. We report numbers relative to the baseline of training directly on the hand-labeled development set. Reported scores are normalized relative to the precision, recall, and F1 scores of these baselines, using a true/false threshold of 0.5 for prediction. Lift is reported relative to the baseline F1. We compare the generative model of `Snorkel DryBell`, i.e., a weighted combination of the labeling functions, and the discriminative logistic regression classifier trained with `Snorkel DryBell`. Note that the generative model is not servable, i.e., it cannot be used to make predictions in production.

Task	Relative:	Generative Model Only				Snorkel DryBell			
		P	R	**F1**	**Lift**	P	R	**F1**	**Lift**
Topic Classification		84.4%	101.7%	93.9%	*-6.1%*	100.6%	132.1%	117.5%	*+17.5%*
Product Classification		103.8%	102.0%	102.7%	*+2.7%*	99.2%	110.1%	105.2%	*+5.2%*

the discriminative classifier directly on the hand-labeled development set.

We also report the predictive accuracy of `Snorkel DryBell`'s generative model, i.e., using the weighted combination of labeling functions directly to make predictions. We do so to demonstrate that the discriminative classifier learns to generalize beyond the information contained in the labeling functions. Note that the generative model is not actually viable for production tasks, because labeling functions often depend on non-servable features of the data.

The results show that on both tasks, the discriminative classifiers trained on `Snorkel DryBell`-produced training data has higher predictive accuracy in F1 score on the test sets than classifiers trained directly on the development set. The weakly supervised classifiers also have higher predictive accuracy than the corresponding generative models. This result demonstrates that `Snorkel DryBell` effectively transfers the knowledge contained in the non-servable resources to classifiers that only depend on servable features.

6.2 Trade-Off Between Weak Supervision Hand-Labeled Data

We next investigate the trade-off between using weak supervision and collecting hand-labeled training examples. We train the discriminative classifier for each content classification task on increasingly large hand-labeled training sets. Figure 5 shows the predictive performance in relative F1 score of the the classifiers versus the number of hand-labeled training examples. On the topic classification task, we find that it takes roughly 80K hand-labeled examples to match the predictive accuracy of the weakly supervised classifier. On the product classification task, we find that it takes roughly 12K hand-labeled examples. This result shows that weak supervision can significantly reduce the need for hand-labeled training data in content classification applications.
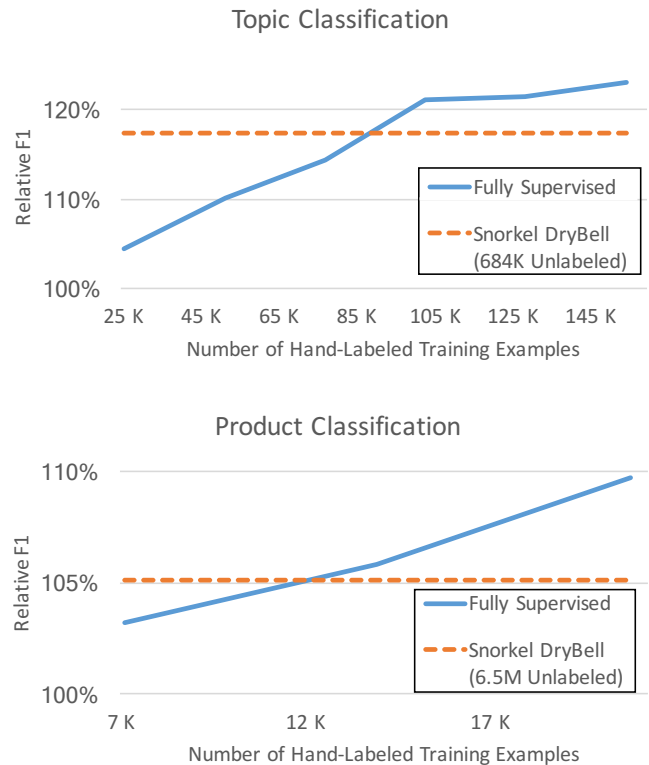

Figure 5: Relative difference in predictive accuracy measured in F1 of supervised classifiers trained on increasing numbers of hand-labeled training examples for the topic and product classification tasks. The dashed line shows the normalized F1 score of the weakly supervised classifier trained on `Snorkel DryBell`-inferred labels.

6.3 Ablation Study

We measured the importance of including non-servable organizational supervision resources by removing all labeling functions that depend on them from the topic and product

Table 3: An ablation study of `Snorkel DryBell` using only labeling functions that depend on servable features ("Servable LFs") compared with all labeling functions, including non-servable resources. All scores are normalized to the precision, recall, and F1 of the logistic regression classifier trained directly on the development set. Lift is reported relative to Servable LFs.

Relative: Topic Classification	P	R	F1	Lift
Servable LFs	50.9%	159.2%	86.1%	
+ Non-Servable LFs	100.6%	132.1%	117.5%	*+36.4%*
Product Classification				
Servable LFs	38.0%	119.2%	62.5%	
+ Non-Servable LFs	99.2%	110.1%	105.2%	*+68.2%*

Table 4: An ablation study of `Snorkel DryBell` using equal weights for all labeling functions to label training data ("Equal Weights") compared with using the weights estimated by the generative model. All scores are normalized to the precision, recall, and F1 of the logistic regression classifier trained directly on the development set. Lift is reported relative to Equal Weights.

Relative: Topic Classification	P	R	F1	Lift
Equal Weights	54.1%	163.7%	109.0%	
+ Generative Model	100.6%	132.1%	117.5%	*+7.7%*
Product Classification				
Equal Weights	94.3%	110.9%	103.24%	
+ Generative Model	99.2%	110.1%	105.2%	*+1.9%*

classification applications. The only labeling functions that remained were pattern-based rules. Table 2 shows the results. We find that incorporating non-servable Google resources in labeling functions leads to a 52% average performance improvement for the end discriminative classifier. This result shows that the non-servable resources contain valuable information that are effectively transferred.

We also measured the importance of using the generative model to estimate the weights of the labeling function votes by training an identical logistic regression classifier giving equal weight to all the labeling functions for the topic and product classification applications. In other words, the probabilistic training labels were an unweighted average of the labeling function votes. Table 4 shows the results. We
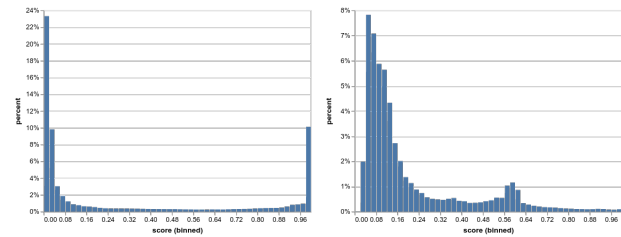

Figure 6: We compare a histogram of the predicted probabilities ("scores") of an event using a model trained with a baseline Logical-OR approach to combining weak supervision sources (left), and trained using `Snorkel DryBell`'s output (right). We see that the baseline approach results in greatly over-estimating the score of events, whereas the model trained using `Snorkel DryBell` produces a smoother distribution. This results in better performance, and offers more useful output to those monitoring the system.

find that using the generative model to weight labeling functions leads to a 4.8% average performance improvement for the end discriminative classifier. This result shows that the generative model is an effective component of the `Snorkel DryBell` pipeline.

6.4 Real-Time Events

We evaluate the application of `Snorkel DryBell` to the real-time events application as compared to a baseline weak supervision approach of training the same deep neural network architecture on a simpler combination of the same set of labeling functions. Specifically, we compare:

- *Logical-OR Weak Supervision:* Here, the weak supervision sources, defined over the non-servable features, are combined using a logical OR. The resulting labels are then used to train a deep neural network (DNN) discriminative classifier over the servable features.
- `Snorkel DryBell`: Here, we use `Snorkel DryBell` to combine the weak supervision sources, and then use the resulting probabilistic training labels to train a DNN over the servable features.

We observed that `Snorkel DryBell` identifies an additional 58% of events of interest as compared to what the baseline Logical-OR approach captures, and the quality of the events identified by `Snorkel DryBell` offer a 4.5% improvement according to an internal metric.

Finally, we note that `Snorkel DryBell` leads to an end discriminative classifier that produces a more reasonable distribution of *scores*, i.e. predicted probabilities of a certain event label, as compared to the Logical-OR weak supervision baseline (Figure 6). Whereas the DNN trained using

the latter approach ends up predicting labels with nearly absolute confidence, the distribution produced by Snorkel DryBell is far more nuanced and consistent with the expected distribution—resulting not only in better quality, but more interpretable and usable end predictions.

7 DISCUSSION

The experimental results above demonstrate the importance of Snorkel DryBell's design principles to deploying weakly supervised machine learning in an industrial setting. First, the ability to incorporate diverse organizational resources was critical to both the content and real-time event applications. In both cases, Google has a variety of tools from which we constructed weak supervision sources, from existing machine learning classifiers, to structured background knowledge, to previously developed heuristic functions. These tools are heterogeneous, not just in the information they contain, but how they are maintained and executed within Google. Some, like semantic categorization, are maintained by one team and applied generally to incoming content. Others, like the natural language processing models, are maintained by another team and must be executed as part of the weak supervision development process because they are too expensive to run on all incoming content. We find that labeling functions are an effective abstraction for encapsulating all these types of heterogeneity.

Second, we find that the mechanism of denoising labeling functions to produce training data and train new classifiers used in Snorkel DryBell effectively transfers knowledge from non-servable resources to servable models. This is crucial in an industrial environment in which products are composed of many services that are connected via latency agreements. When engineers have to ensure that classifiers make predictions within allotted times, they have to be very selective about what features to use. In contrast, writing labeling functions affords developers flexibility because they are executed as part of an offline training process.

Third, we find that the labeling function abstraction is user friendly, in the sense that developers in the organization can write new labeling functions to capture domain knowledge. Snorkel DryBell's architecture is designed for high throughput, enabling rapid human-in-the-loop development of labeling functions. For example, developing each content classification application was possible because of the ability to rapidly iterate on labeling functions. In contrast, waiting for human annotators to hand-label training data can cause lengthy delays.

We anticipate that this low-latency development of machine learning classifiers will be increasingly important as businesses and other large organizations increasingly depend on machine learning. This is because machine learning teams are now responsible for implementing business strategies. For example, if a company like Google decides to add a feature to a product that requires identifying content on a specific topic, the machine learning team currently must respond by curating training examples for this topic. If the strategy changes, then the training examples must change too. Weakly supervised machine learning systems like Snorkel DryBell enable these teams to respond by writing code, rather than pushing high-latency tasks to data annotators. When launch schedules for products that depend on machine learning are short, time spent curating training data is costly.

This code-as-supervision paradigm also has the potential to meet additional challenges that modern machine learning teams face. A single team in a large organization is often now responsible for hundreds or more different classifiers. Each one currently needs its own hand-labeled set of training examples. We have demonstrated that Snorkel DryBell enables Google to leverage existing resources—including other machine learning classifiers—to create new classifiers. We anticipate that the problem of managing these large networks of classifiers that share knowledge will be a significant area of future work in the near future.

Finally, we believe weakly supervised machine learning has the potential to affect organizational structures. Google is beginning to experiment with reorganizing machine learning development around the separation between subject matter expertise and infrastructure enabled by weak supervision. Dedicated teams could potentially focus on writing labeling functions while others stay focused on serving the resulting classifiers in production.

7.1 Lessons for Other Organizations

As the code-as-supervision paradigm grows in use, we anticipate that several lessons learned from our work are adoptable beyond Google. First, resources that can be used for weak supervision are abundant, and our findings demonstrate that labeling functions combined via a generative model are a new way to extract value from them. In our work, access to broad-purpose natural language processing models saved significant developer time, even though these models were not designed specifically for our tasks. We observe other organizations already developing broad-purpose models for domains such as language modeling [13, 14], object recognition [20], and object detection [39]. In addition, cloud-service providers are making pre-trained, broad-purpose models for language and vision available to consumers [12, 35]. Further, many open-source, broad-purpose models for tasks like named entity recognition [17], sentiment analysis [21], and

object detection [25] are freely available and could be incorporated into labeling functions for more specific tasks. While our study focused on existing resources, our results also indicate that further investment in broad-purpose models and knowledge graphs that provide background knowledge for weak supervision will be increasingly worthwhile.

Second, cross-feature transfer to servable models was critical in our applications, and represents a new perspective on model serving strategies which we believe may be of general applicability. In this approach, developers use a set of features not servable in production—for example, expensive internal models or private entity network graphs—to create training data for models that are defined over production-servable features. We learned that having multiple representations of the same data is an effective way to weakly supervise models with service-level agreements. This technique can potentially benefit the many applications where efficient model serving is needed.

Third, the design choices made in the original Snorkel implementation can be improved for many use cases. Some changes are applicable to both systems for novice users and experts. We found that implementing the generative model in an optimization framework with automatic differentiation was faster to develop, easier to distribute, and faster to execute than an MCMC sampling approach. Other lessons came from the different needs of advanced machine learning engineers versus novice users. We found that advanced users want maximum flexibility in implementing labeling functions, including being able to launch additional services and call remote procedures during execution. This approach is in contrast to Snorkel's focus on novice users, in which data to be labeled is represented with a prescribed class hierarchy. The tradeoff is that while `Snorkel DryBell` offers fewer higher-level helper functions for labeling function writers, it was easy to apply to a wider range of data, from online content to events. We anticipate that other implementations of weak supervision for advanced users will want to follow `Snorkel DryBell`'s approach.

8 RELATED WORK

Weakly supervised machine learning as implemented in `Snorkel DryBell`—using multiple noisy but inexpensive sources of labels as an alternative to hand-labeled training data—is related to other areas of research in machine learning and data systems that also seek to learn and make inferences with limited labeled data.

In machine learning, *semi-supervised learning* [7] combines labeled data with unlabeled data. It is a broad category of methods that generally seek to use the unlabeled data to discover structure in the data, such as dense clusters or low-dimensional manifolds, that enables better extrapolation

from the limited labeled examples. *Transfer learning* [26] exploits labeled data available for one or more tasks to reduce the need for labeled data for a new task. Methods in which a learner labels additional data for itself to train on include *self-training* [2, 34], *co-training* [5], and *pseudo-labeling* [18]. *Zero-shot learning* [38] attempts to learn a sufficiently general mapping between class descriptions and labeled examples that new classes can be identified at test time from just a description, without any additional training examples. *Active learning* [36] methods select data points for human annotators to label. They aim to minimize the amount of labeling needed by interleaving learning and requests for new labels.

Related problems in data systems include *data fusion* [11, 33] and *truth discovery* [19]. Here the goal is to estimate the accuracy of possibly conflicting records in different data sources and integrate them into the most likely set of correct records. A similar problem is *data cleaning* [28], which aims to identify and correct errors in data sets. Recently, Rekatsinas et. al. [32] proposed HoloClean, which uses weakly supervised machine learning to learn to correct errors. Many methods for these problems, e.g., the latent truth model [41], use generative models similar to the one in `Snorkel DryBell` in that they represent the unobserved truth as a latent variable. `Snorkel DryBell`'s generative model differs in that it models the output of labeling functions executed on input data, and these functions can provide any class label or abstain.

9 CONCLUSION

In this paper we presented the first results from deploying the `Snorkel DryBell` framework for weakly supervised machine learning in a large-scale, industrial setting. We find that weak supervision can train classifiers that would otherwise require tens of thousands of hand-labeled examples to obtain, and that `Snorkel DryBell`'s design enables developers to effectively connect a wide range of organizational resources to new machine learning problems in order to improve predictive accuracy. These results indicate that weak supervision has the potential to play a significant role in industrial development of machine learning applications in the near future.

REFERENCES

[1] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, Manjunath Kudlur, Josh Levenberg, Rajat Monga, Sherry Moore, Derek G. Murray, Benoit Steiner, Paul Tucker, Vijay Vasudevan, Pete Warden, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2016. TensorFlow: A System for Large-scale Machine Learning. In *USENIX Conference on Operating Systems Design and Implementation (OSDI)*.

[2] A. K. Agrawala. 1970. Learning with a Probabilistic Teacher. *IEEE Transactions on Infomation Theory* 16 (1970), 373–379.

[3] Stephen H. Bach, Bryan He, Alexander Ratner, and Christopher Ré. 2017. Learning the Structure of Generative Models without Labeled Data. In *International Conference on Machine Learning (ICML)*.

[4] Denis Baylor, Eric Breck, Heng-Tze Cheng, Noah Fiedel, Chuan Yu Foo, Zakaria Haque, Salem Haykal, Mustafa Ispir, Vihan Jain, Levent Koc, et al. 2017. TFX: A TensorFlow-based production-scale machine learning platform. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*.

[5] A. Blum and T. Mitchell. 1998. Combining Labeled and Unlabeled Data with Co-Training. In *Workshop on Computational Learning Theory (COLT)*.

[6] Jakramate Bootkrajang and Ata Kabán. 2012. Label-noise robust logistic regression and its applications. In *Joint European conference on machine learning and knowledge discovery in databases*. Springer, 143–158.

[7] O. Chapelle, B. Schölkopf, and A. Zien (Eds.). 2006. *Semi-Supervised Learning*. MIT Press.

[8] Ekin D Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V Le. 2018. AutoAugment: Learning Augmentation Policies from Data. *arXiv preprint arXiv:1805.09501* (2018).

[9] Nilesh Dalvi, Anirban Dasgupta, Ravi Kumar, and Vibhor Rastogi. 2013. Aggregating crowdsourced binary ratings. In *Proceedings of the 22nd international conference on World Wide Web*. ACM, 285–294.

[10] Alexander Philip Dawid and Allan M Skene. 1979. Maximum likelihood estimation of observer error-rates using the EM algorithm. *Applied statistics* (1979), 20–28.

[11] X. L. Dong and D. Srivastava. 2015. *Big Data Integration*. Morgan & Claypool Publishers.

[12] Google. 2019. Cloud AI. https://cloud.google.com/products/ai/.

[13] Edouard Grave, Moustapha M Cisse, and Armand Joulin. 2017. Unbounded cache model for online language modeling with open vocabulary. In *Advances in Neural Information Processing Systems (NeurIPS*.

[14] Sonal Gupta, Rushin Shah, Mrinal Mohit, Anuj Kumar, and Mike Lewis. 2018. Semantic parsing for task oriented dialog using hierarchical representations. *arXiv preprint arXiv:1810.07942* (2018).

[15] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531* (2015).

[16] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation* 9, 8 (1997), 1735–1780.

[17] Matthew Honnibal and Ines Montani. 2017. spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing. (2017).

[18] Dong-Hyun Lee. 2013. Pseudo-label: The simple and efficient semisupervised learning method for deep neural networks. In *ICML Workshop on Challenges in Representation Learning*.

[19] Yaliang Li, Jing Gao, Chuishi Meng, Qi Li, Lu Su, Bo Zhao, Wei Fan, and Jiawei Han. 2015. A Survey on Truth Discovery. *SIGKDD Explor. Newsl.* 17, 2 (2015).

[20] Dhruv Mahajan, Ross Girshick, Vignesh Ramanathan, Kaiming He, Manohar Paluri, Yixuan Li, Ashwin Bharambe, and Laurens van der Maaten. 2018. Exploring the Limits of Weakly Supervised Pretraining. In *European Conference on Computer Vision (ECCV)*.

[21] Christopher Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Annual meeting of the Association for Computational Linguistics: System Demonstrations*.

[22] H Brendan McMahan, Gary Holt, David Sculley, Michael Young, Dietmar Ebner, Julian Grady, Lan Nie, Todd Phillips, Eugene Davydov, Daniel Golovin, et al. 2013. Ad click prediction: A view from the trenches. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*.

[23] Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2*. Association for Computational Linguistics, 1003–1011.

[24] Volodymyr Mnih and Geoffrey E Hinton. 2012. Learning to label aerial images from noisy data. In *Proceedings of the 29th International conference on machine learning (ICML-12)*. 567–574.

[25] ONNX. 2017. Open Neural Network Exchange. https://github.com/onnx/onnx.

[26] S. J. Pan and Q. Yang. 2010. A Survey on Transfer Learning. *IEEE Transactions on Knowledge and Data Engineering* 22, 10 (2010), 1345–1359.

[27] Sinno Jialin Pan, Qiang Yang, et al. 2010. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering* 22, 10 (2010), 1345–1359.

[28] Erhard Rahm and Hong Hai Do. 2000. Data cleaning: Problems and current approaches. *IEEE Data Eng. Bull.* 23, 4 (2000), 3–13.

[29] Alexander Ratner, Stephen H Bach, Henry Ehrenberg, Jason Fries, Sen Wu, and Christopher Ré. 2017. Snorkel: Rapid training data creation with weak supervision. *Proceedings of the VLDB Endowment* 11, 3 (2017), 269–282.

[30] Alexander J Ratner, Christopher M De Sa, Sen Wu, Daniel Selsam, and Christopher Ré. 2016. Data programming: Creating large training sets, quickly. In *Advances in neural information processing systems*. 3567–3575.

[31] Alexander J Ratner, Braden Hancock, Jared Dunnmon, Frederic Sala, Shreyash Pandey, and Christopher Ré. 2019. Training Complex Models with Multi-Task Weak Supervision. In *AAAI*.

[32] Theodoros Rekatsinas, Xu Chu, Ihab F. Ilyas, and Christopher Ré. 2017. HoloClean: Holistic Data Repairs with Probabilistic Inference. *PVLDB* 10, 11 (2017), 1190–1201.

[33] Theodoros Rekatsinas, Manas Joglekar, Hector Garcia-Molina, Aditya Parameswaran, and Christopher Ré. 2017. SLiMFast: Guaranteed Results for Data Fusion and Source Reliability. In *ACM SIGMOD International Conference on Management of Data (SIGMOD)*.

[34] H. J. Scudder. 1965. Probability of Error of some Adaptive Pattern-Recognition Machines. *IEEE Transactions on Infomation Theory* 11 (1965), 363–371.

[35] Amazon Web Services. 2019. Amazon Comprehend. https://aws.amazon.com/comprehend/.

[36] B. Settles. 2012. *Active Learning*. Morgan & Claypool Publishers.

[37] Chen Sun, Abhinav Shrivastava, Saurabh Singh, and Abhinav Gupta. 2017. Revisiting Unreasonable Effectiveness of Data in Deep Learning Era. *CoRR* abs/1707.02968 (2017). arXiv:1707.02968 http://arxiv.org/abs/1707.02968

[38] Yongqin Xian, Christoph H Lampert, Bernt Schiele, and Zeynep Akata. 2018. Zero-shot learning - A comprehensive evaluation of the good, the bad and the ugly. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2018).

[39] Andrew Zhai, Dmitry Kislyuk, Yushi Jing, Michael Feng, Eric Tzeng, Jeff Donahue, Yue Li Du, and Trevor Darrell. 2017. Visual discovery at Pinterest. In *International Conference on the World Wide Web (WWW)*.

[40] Ce Zhang, Jaeho Shin, Christopher Ré, Michael Cafarella, and Feng Niu. 2016. Extracting databases from dark data with deepdive. In *Proceedings of the 2016 International Conference on Management of Data*. ACM, 847–859.

[41] Bo Zhao, Benjamin IP Rubinstein, Jim Gemmell, and Jiawei Han. 2012. A Bayesian approach to discovering truth from conflicting sources for data integration. *PVLDB* 5, 6 (2012), 550–561.