Snorkel DryBell: A Case Study in Deploying Weak Supervision at Industrial Scale (SIGMOD ’19)

——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

张启凡
52194501005

Outline

- Introduction
- Background
- Case Studies
  - Topic Classification
  - Product Classification
  - Real-Time Event Classification
- System Architecture
- Experiment
- Discussion & Conclusion

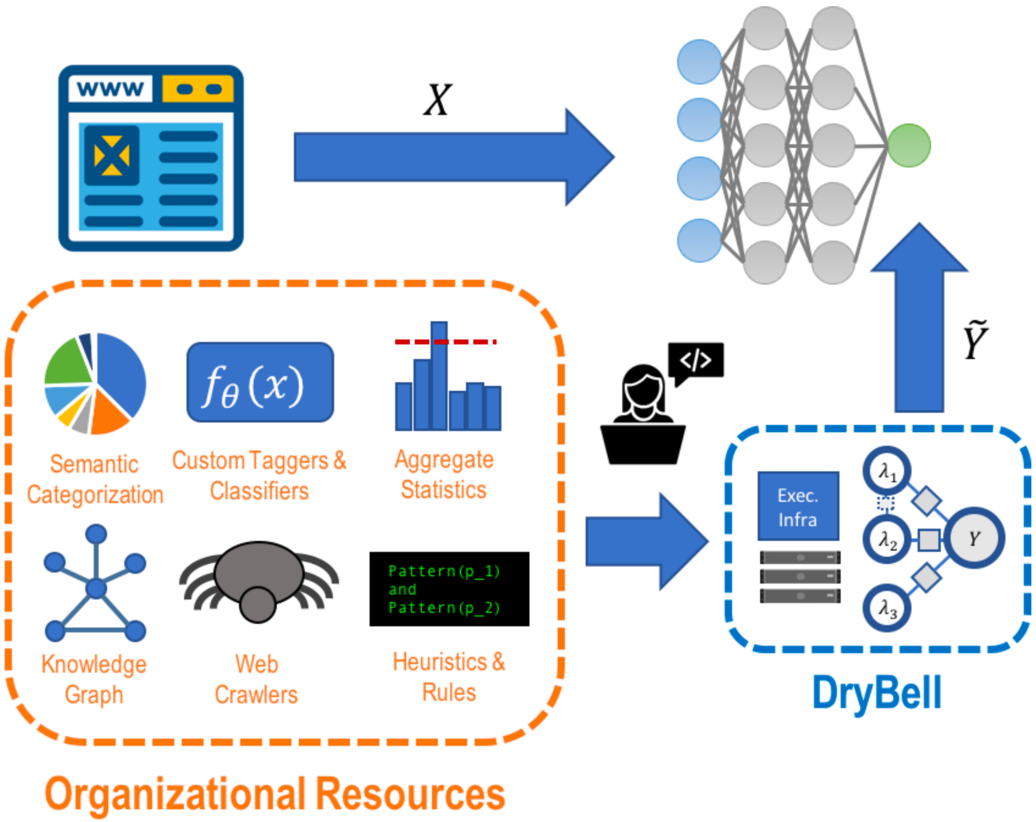
Introduction

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.

Many prior weak supervision approaches rely on a single source of labels, a small number of carefully chosen, manually combined sources, or on sources that make uncorrelated errors such as independent crowd workers.

Introduction



we present a first-of-its-kind study showing how existing organizational knowledge can be used as weak supervision to have significant impact.

1)Flexible Ingestion of Organizational Knowledge

2) Cross-Feature Production Serving

3) Scalable, Sampling-Free Execution

Background

We build on top of **Snorkel**, a recently proposed framework for weakly supervised machine learning, which allows users to generically specify multiple sources of programmatic weak supervision.

The **Snorkel** pipeline follows three main stages, we also adopt in **Snorkel DryBell:**

(a) Users write **labeling functions**;

(b) A **generative model** is used to estimate the accuracies of the different labeling functions;

(c) Final labels are use to train an arbitrary end **discriminative model.**

Background

Let $\mathbf{X} = (\mathbf{X}_1, \ldots, \mathbf{X}_m)$ be a collection of unlabeled data points, $X_i \in X$, with associated unobserved labels $\mathbf{Y} = (\mathbf{Y}_1, \ldots, \mathbf{Y}_m)$.
For simplicity, we focus on binary classification, $\mathbf{Y}_i \in \{-1, 1\}$

In our weak supervision setting, we do not have access to these **ground-truth** labels $\mathbf{Y}_i$, and our goal is to **estimate them to use as training labels**.

Instead, we have access to **n** labeling functions $\boldsymbol{\lambda} = (\boldsymbol{\lambda}_1,\ldots,\boldsymbol{\lambda}_n)$, where $\boldsymbol{\lambda}_j$: $\mathbf{X} \rightarrow \{-1,0,1\}$,with 0 corresponding to an abstain vote.

Background

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 $\mathbf{\Lambda}$ be the matrix of labels output by the **n** labeling functions over the **m** unlabeled data points, such that $\mathbf{\Lambda_{i,j} = \lambda_j (X_i)}$.

We then estimate the parameters **w** of this generative labeling model $\mathbf{P_w (\Lambda, Y)}$ by **maximizing** the log marginal likelihood of the observed labels $\mathbf{\Lambda}$:

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

Note that we are marginalizing out **Y**, i.e. **we are not using any ground truth training labels in our learning procedure.**

Background

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** $\mathbf{h}_{\boldsymbol{\theta}}$ 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 $\tilde{Y}$:

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

Case Studies - 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.

We 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 **(URL-based|NER tagger-based|Topic model-based)**

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.

Case Studies - 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**.

A developer was able to write **eight** labeling functions, leveraging diverse weak supervision resources from across the organization**. (Keyword-based|Knowledge Graph-based|Model-based)**

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.

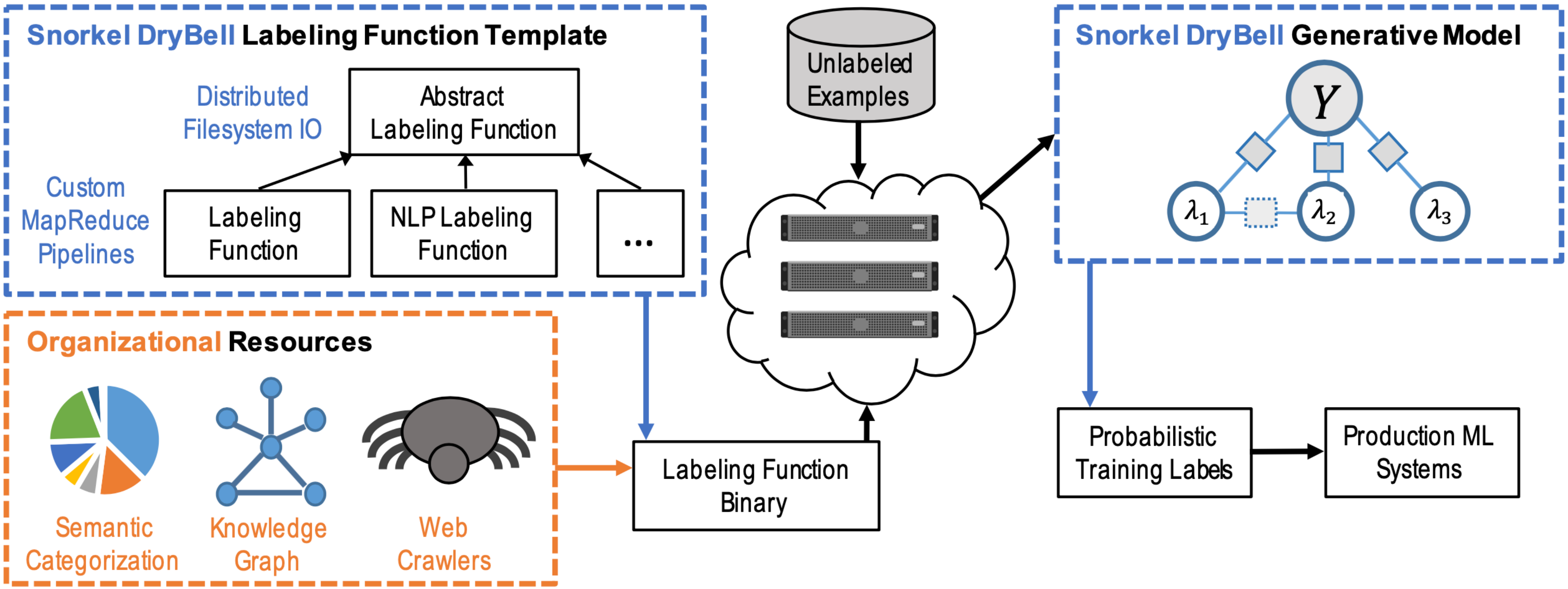
Case Studies - Real-Time Event Classification

Finally, we applied Snorkel DryBell to a real-time events classification problem over two of Google's platforms.

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 | Graph-based | Other heuristics**
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.

System Architecture



System Architecture

Labeling Function Template Library

- templated **C++** classes
- Each subclass defines a **MapReduce** pipeline

```
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();
}
```

System Architecture

Sampling-Free Generative Model

■ combine the noisy votes of the various labeling functions into estimates of the true labels for training.

■ sampling-free modeling approach: far less **CPU intensive** and far simpler to **distribute** across compute nodes

We focus on a **conditionally independent** **generative model**, which we write as:

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

System Architecture

Sampling-Free Generative Model

The learning objective of the generative model is to minimize the negative marginal log-likelihood of the observed labeling function outputs **$-\log P(\Lambda)$.**

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

System Architecture

Sampling-Free Generative Model

Let $\boldsymbol{\alpha}_j$ be the unnormalized log probability that labeling function j is correct given that it did not abstain, and let $\boldsymbol{\beta}_j$ be the unnormalized log probability that it did not abstain.

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

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

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

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

System Architecture

Discriminative Model Serving

■ integrated Snorkel DryBell with **TFX**, Google's platform for end-to-end production-scale machine learning.

■ 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.

System Architecture

Comparison with Snorkel Architecture

■ Snorkel is designed to run on a **single, shared-memory** compute node.

■ 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.

Experiment - Topic and Product Classification

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 also create a small, hand-labeled **development set** (**nDev** in Table) 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.

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

Experiment - Topic and Product Classification

Table below 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 the **discriminative classifier** directly on the **hand-labeled development set**.

On both tasks, the **discriminative classifiers** has **higher** predictive accuracy in F1 score on the test sets than classifiers trained directly on the development set.

Tips: 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.

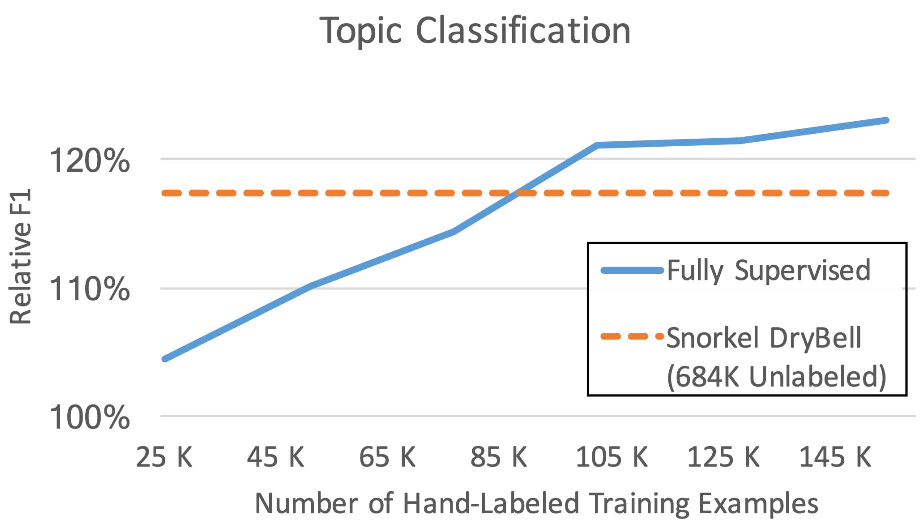
.

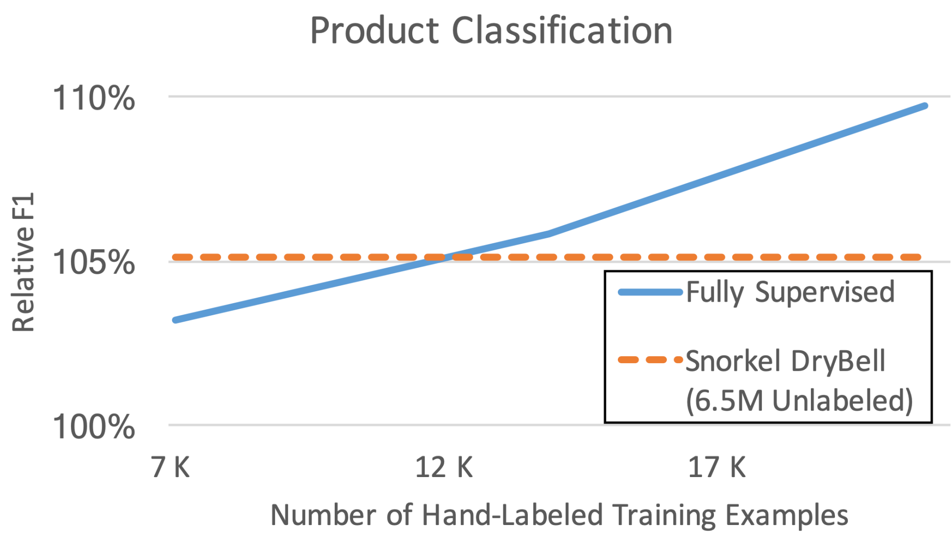
		Generative Model Only				Snorkel DryBell			
Task	Relative:	P	R	**F1**	**Lift**	P	R	**F1**	**Lift**
Topic Classification		84.4%	101.7%	93.9%	*-6.1%*	100.6%	132.1%	117.5%	*+17.5%*
Product Classification		103.8%	102.0%	102.7%	*+2.7%*	99.2%	110.1%	105.2%	*+5.2%*

Experiment - Trade-Off

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.





Experiment - Ablation Study

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:	P	R	**F1**	**Lift**
Topic Classification				
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%*

Experiment - Ablation Study

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.

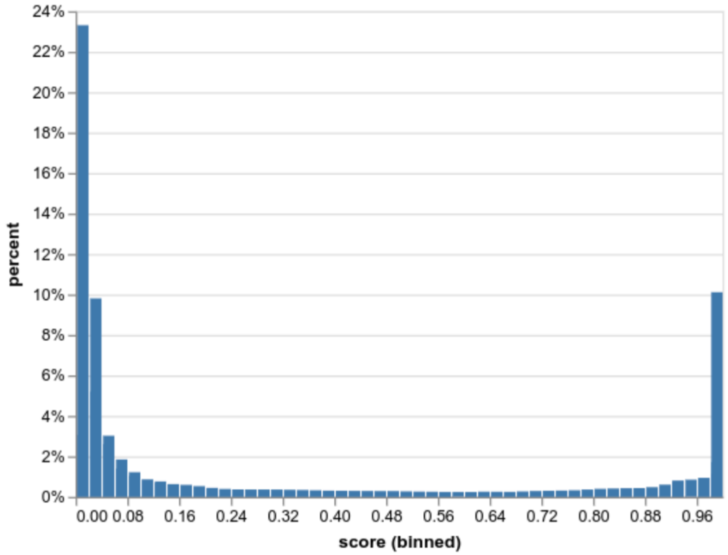
Relative:	P	R	**F1**	**Lift**
Topic Classification				
Equal Weights	54.1%	163.7%	109.0%	
+ Generative Model	100.6%	132.1%	117.5%	*+7.7%*
Product Classification				
Equal Weights	94.3%	110.9%	103.24%	
+ Generative Model	99.2%	110.1%	105.2%	*+1.9%*

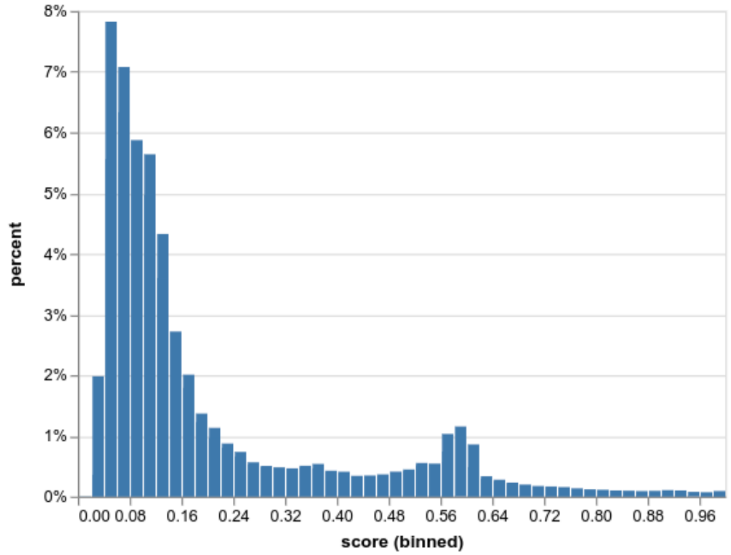
Experiment - Real-Time Events

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.





Discussion & Conclusion

1.We find that **labeling functions** are an effective abstraction for **encapsulating** all these types of heterogeneity.

2. 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**.

3. Snorkel DryBell's architecture is designed for **high throughput**, enabling rapid **human-in-the-loop** development of labeling functions.

Discussion & Conclusion

4. This **code-as-supervision paradigm** also has the potential to meet additional challenges that modern machine learning teams face.

5. We believe weakly supervised machine learning has the potential to affect organizational structures.

The End