Interactive Programmatic Labeling for Weak Supervision

Benjamin Cohen-Wang
Stanford University
Stanford, CA
bencw@stanford.edu

Stephen Mussmann
Stanford University
Stanford, CA
mussmann@stanford.edu

Alex Ratner
Stanford University
Stanford, CA
ajratner@stanford.edu

Chris Ré
Stanford University
Stanford, CA
chrismre@stanford.edu

ABSTRACT

The standard supervised machine learning pipeline involves labeling individual training data points, which is often prohibitively slow and expensive. New programmatic or *weak* supervision approaches expedite this process by having users instead write *labeling functions*, simple rules or other heuristic functions that label subsets of a dataset. While these types of programmatic labeling approaches can provide significant advantages over labeling training sets by hand, there is usually little formal structure or guidance for how these labeling functions are created by users. We perform an initial exploration of processes through which users can be guided by asking them to write labeling functions over specifically-chosen subsets of the data. This can be viewed as a new form of *active learning*—a traditional technique wherein data points are intelligently chosen for labeling—applied at the labeling function level. We show in synthetic and real-world experiments how two simple labeling function acquisition strategies outperform a random baseline. In our real-world experiment we observe a 1-2% increase in accuracy after the first 100 labeling functions when using our acquisition strategies, which corresponds to a 2× reduction in the amount of data required to achieve a fixed accuracy.

KEYWORDS

data programming, labeling functions, interactive learning

ACM Reference Format:
Benjamin Cohen-Wang, Stephen Mussmann, Alex Ratner, and Chris Ré. 2019. Interactive Programmatic Labeling for Weak Supervision. In *Data Collection, Curation, and Labeling for Mining and Learning Workshop at KDD '19, August 04–08, 2019, Anchorage, AK*. ACM, New York, NY, USA, 5 pages.

1 INTRODUCTION

One of the key bottlenecks in modern machine learning is the cost of labeled data. The standard supervised pipeline involves labeling individual data points via experts or crowd-sourcing which can be expensive or completely prohibitive, for example when data requires expertise to label, cannot be shipped out of an organization, or is part of a modeling task that frequently changes. Data programming [Ratner et al. 2016] expedites this process by letting experts write labeling functions, easily computable rules or heuristic functions that can automatically label subsets of a dataset. Data programming is an example of a broader trend of programmatic or *weak supervision* approaches wherein user input is solicited in higher-level, and often noisier ways, many of which can be expressed as labeling functions.

Work around data programming has focused primarily on modeling or aggregating the labeling functions [Bach et al. 2017; Ratner et al. 2017, 2018b; Varma et al. 2017, 2019] to infer the true label. However, guiding users on how to write labeling functions, or intelligently soliciting labeling functions from them, has largely not been studied. Existing approaches [Hancock et al. 2018; Varma and Ré 2018] explore methods for auto-generating simple labeling functions, but largely do not explore how to potentially *guide* a user in writing labeling functions.

In this work, we take a step back and examine how to prompt the user with specifically-chosen data in order to guide the labeling functions that they write. In our experience working with open source users of data programming, we have observed that users often look at data samples–for example, randomly sample subsets of an unlabeled dataset, or samples based on error analysis of a model–in order to come up with ideas for labeling functions. Rather than expecting the user to come up with a labeling function based on a random sample of data, or unprompted by any data, we propose intelligently selecting subsets of unlabeled data to show to the user, thus prompting them to write a labeling function. We view this as a new functional type of *active learning* [Settles 2012], where instead of intelligently selecting individual data points to be labeled, as in traditional active learning, we intelligently select *subsets* of data to be programmatically labeled.

For our experiments we model the problem with a large number of labeling functions that are unknown to the system, but can be queried in the following way: the system provides a data point and the human expert returns a labeling function that correctly classifies the given data point. We show in synthetic and real-world experiments how two simple labeling function acquisition strategies outperform a random baseline.

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. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
Workshop at KDD '19, August 04–08, 2019, Anchorage, AK
© 2019 Copyright held by the owner/author(s). Publication rights licensed to ACM.

2 BACKGROUND

The data programming pipeline consists of the following three steps.

(1) Collect noisy labeling functions from the human-expert, which for each data point may either make a prediction or abstain.
(2) Learn an implicit generative model between labeling function outputs and the unobserved true labels that "generate" them. We use this model to reweight and combine the outputs of the labeling functions, which may overlap and disagree.
(3) Train a noise-aware discriminative end model with the aggregated soft labels from the generative model, and use it to make final predictions for each data point.

In this work we seek to optimize the first step of this pipeline, and use data programming engine from Snorkel MeTaL [Ratner et al. 2018a] for the latter two steps.

3 METHOD

We propose two basic strategies for selecting data points to show to the human expert: (1) select data points where the current labeling functions abstain most and (2) select data points where the current labeling functions disagree most. Our baseline is choosing random points which is standard in open source frameworks for data programming for collecting labeling functions [Ratner et al. 2017].

In the abstain-based strategy, we simply query the data point for which the number of current labeling functions that abstain is maximized. In the frequent case where multiple data points tie for the most abstentions, one of these data points is selected randomly. Intuitively this strategy promotes obtaining more information about data points which we currently do not know much about. This strategy often selects data points with no predictions, as coverage (the fraction of data points with at least one prediction) rarely reaches one.

For the disagreement-based strategy, we first define the agreement of our labeling functions on some data point x as $\left|\sum_j \lambda_j(x)\right|$ where λ_j represents the j'th labeling function and outputs 1 for positive labels, -1 for negative labels, and 0 for abstains (for simplicity of exposition, we consider binary classification problems, but our approach could be easily extended to multiclass settings). At each step we select the data point to show the human expert from the data points where agreement is equal to the minimum agreement across the dataset, again breaking ties by choosing one point randomly. In most cases, this means that the data point selected currently has no non-abstain predictions or has equally many positive and negative votes. As points are selected uniformly in case of ties in disagreement, this strategy tends to select more points with no predictions early on, and more points with tied positive and negative votes as coverage increases.

4 EXPERIMENTS

4.1 Synthetic Data Experiments

Our synthetic experiments are intended to measure the performance of our two strategies compared to random on differently structured datasets. The synthetic datasets we generate are binary classification problems with 10,000 data points and perfectly balanced classes. To simulate the human expert for these experiments, we define a large pool of possible labeling functions that each tend to label certain areas or *partitions* of the data, based on our our observations of common patterns by which human experts write labeling functions [Ratner et al. 2017]. We divide the data into p equally-sized partitions, and assign L/p labeling functions to each partition (where $L = 4000$ is the total number of labeling functions) such that each labeling function primarily fires on data points within its assigned partition. For each query, the simulated human expert randomly select one labeling function from the pool which correctly predicts on the given data point.

We parameterize the labeling function behavior of our synthetic dataset as follows, to simulate the noisy labeling performance of user-written labeling functions [Ratner et al. 2017]. We assign to each labeling function an inside fire rate (IF), an outside fire rate (OF), a false fire rate (FF), and a class. The inside and outside fire rates are the fraction of the data points within the labeling function's partition and outside of the labeling function's partition, respectively, on which the labeling function fires, allowing us to control the degree to which labeling functions label in tightly correlated clusters (i.e., in their partitions). We randomly select data points according to these rates and refer to these data points as the region of the labeling function. The labeling function fires on every data point of its class in its region, and (incorrectly) fires on data points of the opposite class in its region at its false fire rate. Labeling functions are randomly assigned their class such that exactly half of the labeling functions assigned to each partition have positive class, and the other half have negative class. Letting λ_j denote the j'th labeling function, $\mathcal{P}_j$ denote the partition λ_j is assigned to and c_j denote the class of λ_j, the probability that point x belongs to the region $\mathcal{R}_j$ of λ_j is

$$\Pr[x \in \mathcal{R}_j] = \begin{cases} \text{IF if } x \in \mathcal{P}_j \\ \text{OF otherwise} \end{cases}$$

and the probability that λ_j fires on x is

$$\Pr[\lambda_j(x) = c_j] = \Pr[x \in \mathcal{R}_j] \cdot \begin{cases} 1 \text{ if } c(x) = c_j \\ \text{FF otherwise} \end{cases}$$

To measure the effects of differently structured data on our methods' performances we vary our parameters in three ways. First, we vary labeling function accuracy by changing the false fire rates while keeping the inside fire rates fixed at 1.0, the outside fire rates at 0.0, and the number of partitions fixed at $p = 20$ (Figure 1). Second, we vary the extent to which labeling functions fire within their assigned partition, which intuitively reflects the degree of clustering among labeling function predictions, by changing the ratio between the outside fire rate and inside fire rate while keeping the overall fire rate (the total fraction of data points each labeling function fires on) fixed, false fire rates fixed at 0.0, and again the number of partitions fixed at $p = 20$ (Figure 2). Finally, we vary the number of partitions p while fixing false fire rates at 0.0, inside fire rates at 1.0, and false fire rates at 0.0 (Figure 3).

Figures 1, 2, and 3 show the accuracies of the majority vote of the collective labeling functions after each query for the abstain-based, disagreement-based, and random data point selection strategies on datasets with different parameter values as described above. These results are averaged over 50 trials. Notice that in all three plots, the blue line represents the "ideal" case where inside fire rates are 1.0, outside fire rates are 0.0, false fire rates are 0.0 and $p = 20$.

4.2 Real Data Experiments

For our real-world dataset experiments we used the Amazon review data from [He and McAuley 2016], which consists of text reviews of Amazon products and integer ratings between one and five inclusive associated with each review. The data includes reviews about different categories of products such as "Home and Kitchen", "Electronics", "Books" and "Apps for Android". To construct our binary classification dataset, we choose 500 random reviews with rating one and 500 random reviews with rating five from every category with at least 100,000 reviews, which amounts to 18 categories total. The task we define is simply predicting whether the review has rating one or five (i.e. is positive or negative).

To simulate the human expert for this task, we use the opinion lexicon from [Hu and Liu 2004] which provides the sentiment (positive or negative) of 6789 common words and alternative spellings. When presented with a review, the simulated human expert picks one random word in the review whose sentiment (according to the opinion lexicon) is the same as the sentiment of the review and returns a labeling function of the class of the review which fires if that word is present, a reasonable proxy for labeling functions written by real human experts [Ratner et al. 2017]. For instance, for the review "This was a terrible and disappointing book. The happy ending doesn't make sense." the simulated human expert would either return a labeling function which labels every review with the word "terrible" with the negative class and abstains everywhere else, or the labeling function which behaves the same way for the word "disappointing". In the case where no words with sentiment matching the label of the review are present, the query yields no labeling function.

To evaluate the labeling functions produced, we run the complete data programming pipeline from [Ratner et al. 2016] with logistic regression on the bag-of-words representation of the reviews as our end model. We query 250 labeling functions and evaluate the collective labeling functions every 10 queries. We record coverages and the accuracies of the majority vote, label model, and end model predictions, and average these results over 50 trials.

As a baseline for the data programming pipeline on this task, we also train a logistic regression model on the bad-of-words representation of the reviews on 50 to 250 labeled data points, once again averaging results over 50 trials. This baseline seems reasonable as manually labeling some number of reviews should take a human expert roughly as much time as providing one indicator word for each review. This means that we can reasonably compare the logistic regression baseline trained on q labeled data points to the data programming pipeline with q human expert queries. We observe that the end model for the disagreement-based strategy outperforms this baseline by roughly 4% at 50 labeled data points/queries and continues to outperform the baseline until about 250 labeled data points/queries. This baseline verifies that data programming is relevant in this setting, and thus that improving upon the data programming pipeline is valuable.

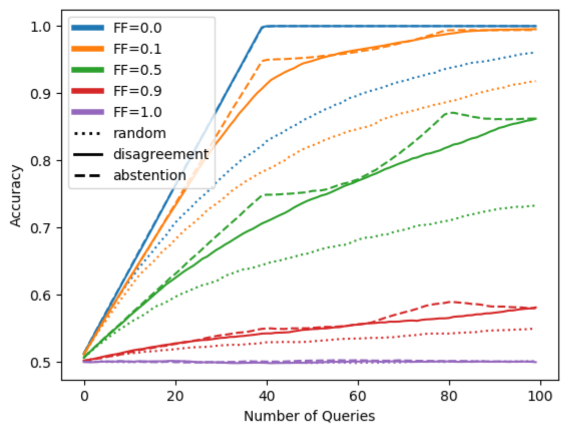

Figure 1: Varying false fire rates: The two smart strategies outperform random for both high and low accuracy labeling functions.

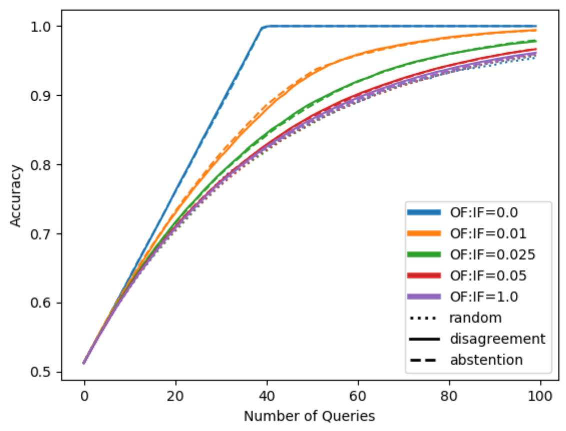

Figure 2: Varying ratio of outside to inside fire rates: The two smart strategies have higher performance gain relative to random for higher degrees of clustering, which performs the same for different degrees of clustering (random results overlap in the plot)

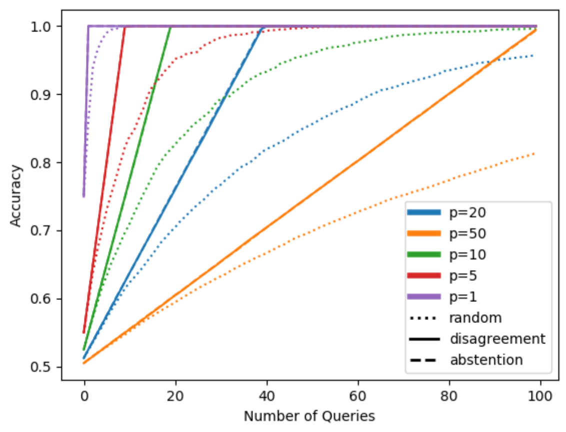

Figure 3: Varying number of partitions: All strategies exhibit similar behaviors for different numbers of partitions.

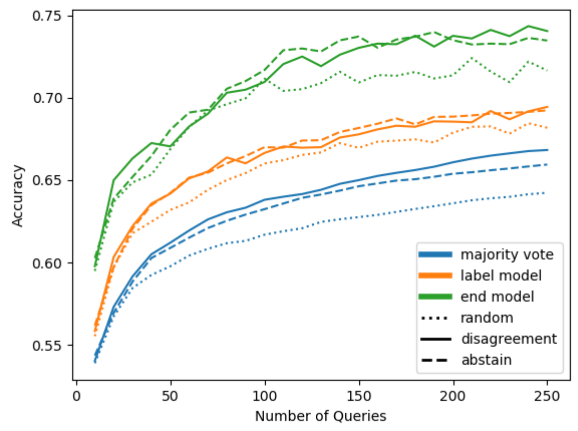

Figure 4: Accuracies on Amazon review data: Smart strategies have accuracy gain of roughly 1-2% over random, or 2× increase in data efficiency.

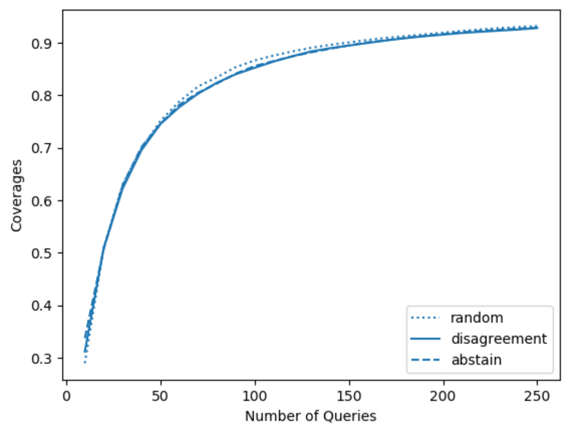

Figure 5: Coverages on Amazon review data: All strategies have similar coverages throughout.

5 DISCUSSION

In this section we summarize some high-level observations about performance gains from using our two smarter strategies to query data points over random, and discuss some potential next steps in this research direction. We begin by discussing the results of our synthetic experiments. First, we notice in Figure 1 that the abstain and disagreement-based strategies outperform random for both low and high labeling function accuracies. Intuitively, for very accurate labeling functions performing well involves maximizing coverage, and for less accurate labeling functions performing well involves maximizing coverage as well as breaking ties between existing labeling functions. The abstain-based and disagreement-based strategies promote these behaviors more so than the random strategy.

We observe in Figure 2 that the performance gain from the smart strategies grows with the degree of clustering structure between the labeling functions, or more generally with the degree of correlation among labeling functions in where they predict and abstain. This is because when labeling functions are more correlated with each other each new labeling function is more likely to be redundant, which the smarter query strategies are designed to avoid. While traditional crowdsourcing work often considers a model of uncorrelated labelers, in data programming we see that labeling functions are indeed often very correlated in where they label, corresponding to different distinct parts of the feature space [Ratner et al. 2016].

Finally, in Figure 3 where we vary the number of partitions we observe similar relationships between our two strategies and random but roughly stretched proportionally to the number of partitions. This is consistent with what we would expect, since the abstain and disagreement-based strategies simply produce one labeling function of each class for each partition and the random strategy randomly collects labeling functions. Letting p be the number of partitions and q be the number of queries, the expected accuracies for the strategies according to this behavior are

$$E[\text{Abstain Strategy Accuracy}] = \min(1, \frac{q}{2p})$$

$$E[\text{Disagreement Strategy Accuracy}] = \min(1, \frac{q}{2p})$$

$$E[\text{Random Strategy Accuracy}] = 1 - (1 - \frac{1}{2p})^q$$

On the Amazon review data in Figure 4 we observe that the abstain and disagreement-based strategies outperform random on real data with somewhat correlated labeling functions of varying accuracies. As the coverages of the three strategies are roughly equal after each query as seen in Figure 5, we expect this improvement derives from the abstain-based strategy correcting aggregate predictions for data points with few votes, and the disagreement-based strategy producing labeling functions that break ties between existing labeling functions. While the improvement we observe is most consistent in the majority vote of labeling functions, the label models and end models trained on the labeling functions produced by the disagreement-based strategy also outperform those produced by random, particularly for higher number of queries. We expect the disagreement-based strategy to eventually outperform the abstain-based strategy, since as coverage approaches one without reaching it the data points selected by the abstain-based strategy are typically outliers.

Next Steps. As follow up work to this initial exploration, we plan to investigate more sophisticated selection strategies. For example, we have not yet explored strategies which depend on the label or end model predictions, which could provide more fine-grain information than the individual labeling function votes about points for which the current labeling functions are insufficient. We could also experiment with showing the human expert multiple data points at once, which we have so far avoided since simulating the labeling function the human expert produces after seeing multiple points would be difficult. Other important next steps are to assess performance of smart selection strategies on other real-world datasets/tasks, and to run experiments involving real human experts producing the labeling functions.

REFERENCES

Stephen H Bach, Bryan He, Alexander Ratner, and Christopher Ré. 2017. Learning the structure of generative models without labeled data. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 273–282.

Braden Hancock, Martin Bringmann, Paroma Varma, Percy Liang, Stephanie Wang, and Christopher Ré. 2018. Training classifiers with natural language explanations. In *Proceedings of the conference. Association for Computational Linguistics. Meeting*, Vol. 2018. NIH Public Access, 1884.

Ruining He and Julian McAuley. 2016. Ups and Downs: Modeling the Visual Evolution of Fashion Trends with One-Class Collaborative Filtering. *CoRR* abs/1602.01585 (2016). arXiv:1602.01585 http://arxiv.org/abs/1602.01585

Minqing Hu and Bing Liu. 2004. Mining and Summarizing Customer Reviews. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD ’04)*. ACM, New York, NY, USA, 168–177. https://doi.org/10.1145/1014052.1014073

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.

Alex Ratner, Braden Hancock, Jared Dunnmon, Roger Goldman, and Christopher Ré. 2018a. Snorkel MeTaL: Weak Supervision for Multi-Task Learning. In *Proceedings of the Second Workshop on Data Management for End-To-End Machine Learning*. ACM, 3.

Alexander Ratner, Braden Hancock, Jared Dunnmon, Frederic Sala, Shreyash Pandey, and Christopher Ré. 2018b. Training complex models with multi-task weak supervision. *arXiv preprint arXiv:1810.02840* (2018).

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.

Burr Settles. 2012. Active learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning* 6, 1 (2012), 1–114.

Paroma Varma, Bryan D He, Payal Bajaj, Nishith Khandwala, Imon Banerjee, Daniel Rubin, and Christopher Ré. 2017. Inferring generative model structure with static analysis. In *Advances in neural information processing systems*. 240–250.

Paroma Varma and Christopher Ré. 2018. Snuba: automating weak supervision to label training data. *Proceedings of the VLDB Endowment* 12, 3 (2018), 223–236.

Paroma Varma, Frederic Sala, Ann He, Alexander Ratner, and Christopher Ré. 2019. Learning Dependency Structures for Weak Supervision Models. *arXiv preprint arXiv:1903.05844* (2019).