Validating Labeling Functions in Domain Shift

Yewon Kim
KAIST
20223137
yewon.e.kim@kaist.ac.kr

Seungjoo Lee
KAIST
20224560
juicelee@kaist.ac.kr

Abstract

Detecting domain shift and updating ML pipeline appropriately is crucial in real-world ML systems. We argue that using labeling function outputs rather than downstream model outputs is a more effective method for detecting domain shift. We propose a lightweight and simple framework that converts discrete labeling functions to continuous functions and applies density estimation method to detect out-of-distribution samples in test time. We analyze the performance of our proposed approach on three real-world datasets on binary sentiment analysis task, and show that our approach is effective at detecting domain shift.

1 Introduction

Labeling functions (LFs) are a lightweight and cost-effective way to generate labels in unlabeled data. However, in real-world applications, data shift between the train and test datasets can occur, resulting in inaccurate labeling functions and degraded downstream model performance. This data shift can occur for a variety of reasons, such as changes in data distributions, changes in domain-specific features, and temporal changes in the data. In order to ensure accurate model performance, the data scientist must be able to detect such data shifts, and take the appropriate action to update the labeling functions.

There have been various approaches to detect and solve domain shift; however, the work is limited to detecting domain shifts using the downstream model itself. This indicates that the developer must manually inspect and analyze the labeling functions in order to ensure high labeling function quality whenever the shift is detected, resulting in higher overhead to the developers.

We propose to detect domain shift from labeling function outputs. We argue that using labeling function outputs is more beneficial than using model outputs to detect domain shift. Behaviors of machine learning models, especially large language models, are complicated to interpret and debug. As such, even if the developer is informed about domain shift, it will take more time and effort to debug the model outputs and analyze data in order to update labeling functions and models. On the other hand, labeling functions are in general easier to interpret and debug than machine learning models, making it easier to why a domain shift is occurring and how to address it. Moreover, labeling functions are often faster to evaluate and update than machine learning models, which is important in the real-world deployments where data is changing largely and rapidly, and the service needs immediate actions. Overall, our approach is lightweight and efficient, and does not require additional data or manual inspection of the labeling functions.

We analyze the performance of our proposed approach on three real-world datasets on binary sentiment analysis task, and show that our approach is effective at detecting domain shift. We also compare our proposed approach to the existing method for detecting domain shift, and show that our approach is more accurate.

2 Background

2.1 Programmatic labeling

Deep learning models suffer from a label scarcity bottleneck, as collecting and annotating large training data sets is time consuming and label intensive. Programmatic labeling alleviates such bottleneck by using labeling functions for labeling (Ratner et al., 2016, 2017). Specifically, users encode weak supervision sources, such as heuristics, knowledge bases, and pre-trained models, to labeling functions (LFs). A large set of training labels is collectively generated by LFs with each labeling a subset of the data.

Some data points may have conflicting labels because the labeling functions are noisy and have

variable error rates. To deal with such cases, labeling models that aggregate the noisy output of LFs with the final training labels are developed (Ratner et al., 2017, 2019; Fu et al., 2020; Varma et al., 2019).

A number of attempts have been made to extend the scope of LFs that can be used. In CAGE (Chatterjee et al., 2020), continuous LFs are supported in conjunction with existing label models. NPLM (Yu et al., 2022) provides partial LFs that output a subset of class labels, and PLRM (Zhang et al., 2021) allows the use of indirect LFs that are limited to predicting unseen but related classes.

2.2 Domain Shift

Domain shift, or distributional shift, (Sun et al., 2016) refers to a change in the data distribution between the train and test datasets. The over-reliance on the training distribution makes the practical application of artificial intelligence challenging, as the performance of the model often degrades on deployment. Distributional shifts arise naturally in many downstream applications; any downstream NLP model can degrade significantly when exposed to new data, such as new vocabulary, expression, and writing style.

Several methods have been developed to combat such problem; one of them is out-of-distribution (OOD) detection, which is a technique for identifying when a model is being applied to data that are different from the train dataset. By detecting when a model is being used on out-of-distribution data, it is possible to flag the model predictions and either reject them or adapt the model to better suit to the new data.

2.3 Types of OOD detection

Methods for OOD detection could be classified largely into three categories: methods that use (i) predicted class probabilities, (ii) curated training algorithms, and (iii) density estimation.

The most common OOD detection method is to use the predicted class probabilities of the model as a confidence score (Hendrycks and Gimpel, 2016). Although the raw model output could be used, this could be further improved by applying temperature scaling to the logits (Liang et al., 2017) or utilizing the intermediate features of the input (Shama Sastry and Oore, 2019).

Some researchers devised training strategies to detect OOD; one approach is to expose the model to the outlier data during the training phase (Hendrycks et al., 2018). A different approach proposed training classifier networks to become less confident for OOD samples (Lee et al., 2017), and a recent approach proposed an energy-based method (EBM) interpreting softmax probabilities as energy scores (Hsu et al., 2020).

OOD detection has also been achieved using density estimation. In a recent work (Lee et al., 2018), Mahalanobis models the class conditional probability density functions of features at intermediate layers of a DNN with Gaussian densities for in-distribution (ID) samples. Each class conditional Gaussian parameter is estimated using the empirical mean and covariance of ID training samples belonging to the class. At test-time, OOD detection is done by computing confidence score from the model output, where the score if expected to be lower for OOD samples.

While existing approaches have shown success, they are intended for downstream models, and methods for detecting OODs have not been explored for labeling functions. To our best knowledge, we are the first to address the domain shift detection problem in labeling functions.

3 Method

3.1 Overview

Figure 1 shows an overview of our system. In train phase, we develop discrete LFs using unlabeled training data, and convert discrete LFs to continuous LFs (Section 3.3). From the developed discrete LFs, we label training data and use it to train the downstream model. We train kernel density estimator using outputs from continuous LFs (Section 3.4). In the test phase, we feed the outputs from the continuous LFs to the trained kernel density estimator to detect the OOD data batch. Details of each part will be explained in the following sections.

3.2 Using LF output for OOD detection

Previous methods observe model outputs to determine if the input data instance is OOD. Specifically, the most common method is to define a score function $s(x)$ given an input x and classify the sample as OOD if $s(x) < \delta$ where δ is a predefined threshold (Hendrycks and Gimpel, 2016). Instead, in this work, we observe the outputs of LFs to detect OOD. The intuition behind the idea is that outputs of LFs itself could work as optimal feature representations of the input, as LFs are designed to identify important aspects of the data. Detecting changes in

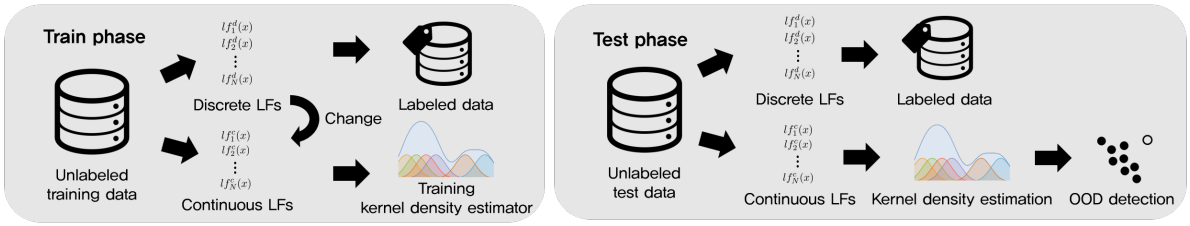


Figure 1: Overview of our system

feature values could possibly indicate shift in data; as an illustrative example, checking if the word "kid" and "like" exists will successfully label the toy reviews, e.g., "my kid likes it" as positive, but it would not be able to properly label the review of a book, e.g., "a true love story."

3.3 Converting to continuous LF

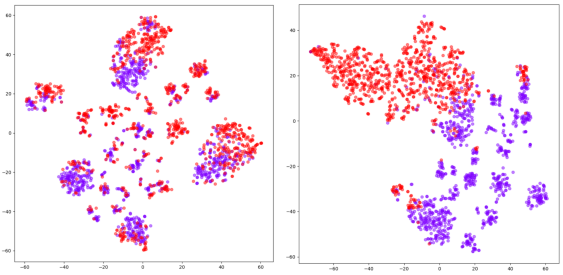
Instead of using outputs from discrete LFs, we convert disrete LFs to continuous LFs. Algorithm 1 shows an example discrete LF for text sentiment classification. It returns POSITIVE if the specified keyword is in the text. Algorithm 2 is continuous version of Algorithm 1. It first converts keyword and words in the text to GloVe embedding (Pennington et al., 2014), and calculates the cosine similarity between the keyword and words in the text. The mean of the top 10% cosine similarity values is returned.

Figure 2 shows the benefit of converting discrete LFs to continuous LFs. Figure 2a and 2b show the T-SNE visualization of LFs' outputs with IMDB (Maas et al., 2011) and Yelp (Zhang et al., 2015) dataset. The distribution of both datasets cannot be distinguished when using discrete LFs; on the other hand, it can be distinguished well when using continuous LFs. This is due to the fact that a discrete LF can only output limited values (3 in the sentiment analysis task), thus not enough information can be obtained from them.

The idea of converting discrete LFs to continuous LFs by using word embedding and cosine similarity is similar to CAGE (Chatterjee et al., 2020). The continuous LFs from CAGE, however, only return maximum cosine similarity values and can only be used for spouse relationship extraction.

3.4 Kernel Density Estimation

Given the outputs of the continuous LFs, we apply kernel density estimation (KDE) (Feinman et al., 2017) to model the marginal feature probability



(a) 8 discrete LFs (b) 8 continuous LFs

Figure 2: T-SNE visualization of LF's output

Algorithm 1 Discrete LF

```
function PosKeywordLF(text, keyword)
    if keyword in text then
        return POSITIVE
    end if
    return ABSTAIN
end function
```

density function from train data. Specifically, given an input $x_i \in D_{ID}^{Tr}$ where D_{ID}^{Tr} is in-distribution train data, let f_{x_i} be a vector of labeling function outputs from x_i. We estimate the marginal feature probability density function $p(f)$ based on Gaussian kernel:

$$p(f) \approx \hat{p}(f) = \frac{1}{|D_{ID}^{Tr}|h} \sum_{j=1}^{|D_{ID}^{Tr}|} \mathcal{K}(\frac{f - f_j}{h})$$

where h is a smoothing bandwidth and $\mathcal{K}(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}}$ is a Gaussian kernel function. Given $\hat{p}(f)$ and a predefined threshold δ, we can determine that a new feature f' is OOD if $\hat{p}(f') < \delta$.

4 Evaluation Setup

We evaluated the performance with AUROC (Bradley, 1997). AUROC means area under ROC (Receiver Operating Characterisitic) curve, where an ROC curve illustrates the diagnostic abilities of

Algorithm 2 Continuous LF

Require:

Vec(`word`) ▷ GloVe embedding of word

MeanTopK(`input`, `k`) ▷ Mean of top k input

function POSKEYWORDLF(`text`, `keyword`)

$K \leftarrow$ Vec(`keyword`)

$T \leftarrow$ Vec(`word` in `text`)

$C \leftarrow$ CosSimilarity(K, T)

$Num \leftarrow$ MAX(len(`text`) $* 0.1, 1)$

return MeanTopK(C, Num)

end function

binary classifiers as their discrimination threshold is changed. AUROC measures the performance of a binary classification model. An AUROC of 0.5 indicates that the classifier is no better than random, while an AUROC of 1.0 indicates a perfect classifier. In general, the higher the AUROC, the better the performance of the classifier.

We measured labeling accuracy, which means how accurately the label model (discrete LFs) labeled the dataset (25,000 instances). We used Snorkel label model (Ratner et al., 2017) to combine results of discrete LFs. The labeling accuracy of ID is the result of applying the LFs of the ID dataset to the ID dataset. In contrast, the labeling accuracy of OOD is the result of applying the LFs of the ID dataset to the OOD dataset.

We also measured coverage, which means how much portion of the data could be labeled by the label model. Keyword-based LFs may not label all of the data, because they can return an ABSTAIN value, which indicates that the data point has not been labeled. Note that labeling accuracy is calculated only with labeled data points.

For the evaluation, we fixed the batch size to 16, and smoothing factor of the kernel density estimation to 0.05.

4.1 Datasets

We evaluate our method on a text sentiment analysis task. We use 4 different datasets (IMDB (Maas et al., 2011), Yelp Polarity (Zhang et al., 2015), SST-2 (Socher et al., 2013), Amazon (McAuley et al., 2015)) to simulate domain shift. SST-2 and IMDB dataset contain movie review of varying lengths. Yelp polarity dataset, on the other hand, contains reviews of different businesses, which represents a shift in domain from the SST-2 and IMDB datasets. Amazon review dataset contains product reviews from Amazon, categorized by product types. We chose 5 product types (baby, electronics, jewelry, home, sports) among them.

For each dataset, we chose the first 25,000 data instances from the train split. Within the chosen 25,000 data instances, we use first 20,000 data instances for training, and last 5,000 data instances for testing. For example, if we use IMDB dataset as ID (in-distribution) and Yelp dataset as OOD (out-of-distribution), training data is first 20,000 instances of IMDB and testing data is total 10,000 from last 5,000 instances of IMDB and Yelp.

4.2 Baselines

We compare our evaluation results to the paper that detects OOD texts using large language models (Arora et al., 2021). It is important to note that we did not implement and experiment with the paper's method, but instead compared our performance with the paper's reported performance. As our baseline study did not experiment on all pairs we experimented with, we only compared the pairs tested in the baseline paper.

4.3 Discrete LF development

We created keyword-based labeling functions for IMDB and Yelp Polarity dataset separately using Argilla(Argilla). Argilla provides web UI that we can query unlabeled training data using keywords and create keyword-based LFs. We created 22 LFs for Yelp Polarity, and 31 LFs for IMDB. Developed discrete LFs are further converted to continuous LFs like Algorithm 2 for OOD detection. The list of keywords that are used to contruct LFs are shown in table 1.

5 Result

5.1 Overall Performance

Overall performance of our system is shown in Table 2. All AUROC scores are close to 1, which indicates that OOD samples are well detected without satisfying FPR (False Positive Rate). We achieved higher AUROC score compared to baseline method.

We can also observe that the labeling accuracy of the OOD data is comparable to ID data, but the coverage of OOD data is significantly lower compared to ID data. This result suggests that the LFs are capable of identify certain aspects of the data. Note that the coverage of continuous LFs

Label	IMDB (Maas et al., 2011)	Yelp (Zhang et al., 2015)
Positive	impress, adorable, enjoy, excellent, beautiful, wonderful, recommend, best, masterpiece, performance * best, performance *good	recommend, amaze, love, fresh, clean, friendly, perfect, favorite, delicious, service * great, food * great
Negative	terrible, poor, stupid, wrong, disappoint, painful, awful, boring, worse, worst, bad, cliche, killer, unnecessary, waste, least try, nothing * special, nothing * even, performance * worst, acting * bad	leave, finally, terrible, understand, worse, worst, disappoint, bad, awful, rude, wait

Table 1: List of keywords that are used for constructing LFs

ID	OOD	AUROC	Baseline AUROC	Labeling accuracy OOD (Coverage)	Labeling accuracy ID (Coverage)
IMDB	Yelp	0.93	0.78	0.78 (0.57)	0.74 (0.82)
	SST-2	1.00	0.97	0.87 (0.09)	
	Amazon-baby	0.96		0.78 (0.41)	
	Amazon-electronics	0.95		0.75 (0.42)	
	Amazon-jewelry	1.00		0.86 (0.39)	
	Amazon-home	0.98		0.80 (0.39)	
	Amazon-sports	0.99		0.79 (0.33)	

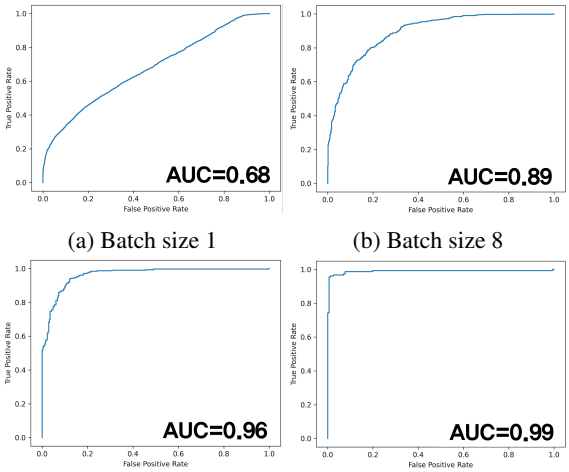
Table 2: Overall performance

that are used for OOD detection is 100%, since continuous LFs do not return ABSTAIN value.

5.2 Trade-off between batch size and AUROC

We identified a trade-off between batch size and AUROC. Figure 3 shows the AUROC score of varying batch size when IMDB dataset is used as ID and Amazon-baby dataset is used as OOD. AUC increases as batch size increases, which means the classifier is getting closer to ideal classifier. Such phenomena is observed on all other combinations of datasets.

The reason behind this is that we can observe more data points at the same time, density estimation results become more robust to noise, while sacrificing the precision of detection. In other words, smaller the batch size, it is possible to detect domain shift in a fine-grained manner but could be sensitive to noise, and vice versa.



(a) Batch size 1 (b) Batch size 8

(c) Batch size 16 (d) Batch size 32

Figure 3: ROC curve and AUROC for varying batch sizes on IMDB dataset (ID) - Amazon baby dataset(OOD)

6 Discussion and Future Work

6.1 Explainability

It is critical to provide explainability that helps to deal with domain shifts. Through our method, we are able to further identify which LFs are malfunctioning. We can train a separate OOD detector for each LFs. When the domain shift is detected on global OOD detector, we can run each LF's OOD detectors and find out which LF is vunerable to domain shift. Then engineers can adjust the LFs to cope with domain shift.

6.2 Converting discrete to continuous LFs

In our work, we only used GloVe embedding (Pennington et al., 2014) to convert discrete keyword-based LFs to continuous LFs. It is possible to convert in other ways, like using Bert (Devlin et al., 2018) or FastText (Joulin et al., 2016) embedding methods. Other than word embeddings, there may be another way to convert.

Additionally, the labeling functions used in this study were only keyword-based. We can also use other labeling functions such as subjectivity and polarity analysis from TextBlob (TextBlob).

6.3 Coverage as OOD predictor

During the experiment, we identified that the coverage of the labeling model significantly dropped when confronting OOD samples. Labeling functions identify certain aspects of data, so this is considered natural behavior. Using this coverage value as an OOD predictor may also be useful.

6.4 More experiments

We only evaluated on text sentiment analysis tasks. It would be helpful to evaluate the system on other NLP tasks like topic classification and relation classification. It would also be beneficial to evaluate it in other domains like vision.

7 Division of roles

Two authors equally contributed to the project. The work of 20223137 Yewon Kim primarily focused on developing a domain shift detection algorithm using kernel density estimation given continuous LFs, while the work of 20224560 Seungjoo Lee primarily focused on developing a method for converting discrete to continuous LFs. All the other works, including ideation, literature search, evaluation, discussion, and paper writing were equally contributed by the authors.

References

Argilla. Argilla library. Accessed: 2022-12-11.

Udit Arora, William Huang, and He He. 2021. Types of out-of-distribution texts and how to detect them. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 10687–10701, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Andrew P. Bradley. 1997. The use of the area under the roc curve in the evaluation of machine learning algorithms. *Pattern Recognition*, 30(7):1145–1159.

Oishik Chatterjee, Ganesh Ramakrishnan, and Sunita Sarawagi. 2020. Robust data programming with precision-guided labeling functions. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 3397–3404.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Reuben Feinman, Ryan R Curtin, Saurabh Shintre, and Andrew B Gardner. 2017. Detecting adversarial samples from artifacts. *arXiv preprint arXiv:1703.00410*.

Daniel Fu, Mayee Chen, Frederic Sala, Sarah Hooper, Kayvon Fatahalian, and Christopher Ré. 2020. Fast and three-rious: Speeding up weak supervision with triplet methods. In *International Conference on Machine Learning*, pages 3280–3291. PMLR.

Dan Hendrycks and Kevin Gimpel. 2016. A baseline for detecting misclassified and out-of-distribution examples in neural networks. *arXiv preprint arXiv:1610.02136*.

Dan Hendrycks, Mantas Mazeika, and Thomas Dietterich. 2018. Deep anomaly detection with outlier exposure. *arXiv preprint arXiv:1812.04606*.

Yen-Chang Hsu, Yilin Shen, Hongxia Jin, and Zsolt Kira. 2020. Generalized odin: Detecting out-of-distribution image without learning from out-of-distribution data. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10951–10960.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2016. Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759*.

Kimin Lee, Honglak Lee, Kibok Lee, and Jinwoo Shin. 2017. Training confidence-calibrated classifiers for detecting out-of-distribution samples. *arXiv preprint arXiv:1711.09325*.

Kimin Lee, Kibok Lee, Honglak Lee, and Jinwoo Shin. 2018. A simple unified framework for detecting out-of-distribution samples and adversarial attacks. *Advances in neural information processing systems*, 31.

Shiyu Liang, Yixuan Li, and Rayadurgam Srikant. 2017. Enhancing the reliability of out-of-distribution image detection in neural networks. *arXiv preprint arXiv:1706.02690*.

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, Portland, Oregon, USA. Association for Computational Linguistics.

Julian McAuley, Christopher Targett, Qinfeng Shi, and Anton Van Den Hengel. 2015. Image-based recommendations on styles and substitutes. In *Proceedings of the 38th international ACM SIGIR conference on research and development in information retrieval*, pages 43–52.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

Alexander Ratner, Braden Hancock, Jared Dunnmon, Frederic Sala, Shreyash Pandey, and Christopher Ré. 2019. Training complex models with multi-task weak supervision. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 4763–4771.

Alexander J. Ratner, Stephen H. Bach, Henry R. Ehrenberg, and Chris Ré. 2017. Snorkel: Fast training set generation for information extraction. In *Proceedings of the 2017 ACM International Conference on Management of Data*, SIGMOD '17, page 1683–1686, New York, NY, USA. Association for Computing Machinery.

Alexander J Ratner, Christopher M De Sa, Sen Wu, Daniel Selsam, and Christopher Ré. 2016. Data programming: Creating large training sets, quickly. *Advances in neural information processing systems*, 29.

Chandramouli Shama Sastry and Sageev Oore. 2019. Detecting out-of-distribution examples with in-distribution examples and gram matrices. *arXiv e-prints*, pages arXiv–1912.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Seattle, Washington, USA. Association for Computational Linguistics.

Baochen Sun, Jiashi Feng, and Kate Saenko. 2016. Return of frustratingly easy domain adaptation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 30.

TextBlob. Textblob library. Accessed: 2022-12-11.

Paroma Varma, Frederic Sala, Shiori Sagawa, Jason Fries, Daniel Fu, Saelig Khattar, Ashwini Ramamoorthy, Ke Xiao, Kayvon Fatahalian, James Priest, et al. 2019. Multi-resolution weak supervision for sequential data. *Advances in Neural Information Processing Systems*, 32.

Peilin Yu, Tiffany Ding, and Stephen H Bach. 2022. Learning from multiple noisy partial labelers. In *International Conference on Artificial Intelligence and Statistics*, pages 11072–11095. PMLR.

Jieyu Zhang, Bohan Wang, Xiangchen Song, Yujing Wang, Yaming Yang, Jing Bai, and Alexander Ratner. 2021. Creating training sets via weak indirect supervision. *arXiv preprint arXiv:2110.03484*.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. *Advances in neural information processing systems*, 28.