Learning to Robustly Aggregate Labeling Functions for Semi-supervised Data Programming

Ayush Maheshwari 1, Krishnateja Killamsetty 2, Ganesh Ramakrishnan 1, Rishabh Iyer 2, Marina Danilevsky 3 and Lucian Popa 3

1 Indian Institute of Technology Bombay, India
2 The University of Texas at Dallas
3 IBM Research – Almaden

{ayusham, ganesh}@cse.iitb.ac.in
{krishnateja.killamsetty, rishabh.iyer}@utdallas.edu
{mdanile, lpopa}@us.ibm.com

Abstract

A critical bottleneck in supervised machine learning is the need for large amounts of labeled data which is expensive and time-consuming to obtain. Although a small amount of labeled data cannot be used to train a model, it can be used effectively for the generation of human-interpretable labeling functions (LFs). These LFs, in turn, have been used to generate a large amount of additional noisy labeled data in a paradigm that is now commonly referred to as data programming. Previous methods of generating LFs do not attempt to use the given labeled data further to train a model, thus missing opportunities for improving performance. Additionally, since the LFs are generated automatically, they are likely to be noisy, and naively aggregating these LFs can lead to suboptimal results. In this work, we propose an LF-based bi-level optimization framework WISDOM to solve these two critical limitations. WISDOM learns a joint model on the (same) labeled dataset used for LF induction along with any unlabeled data in a semi-supervised manner, and more critically, reweighs each LF according to its goodness, influencing its contribution to the semi-supervised loss using a robust bi-level optimization algorithm. We show that WISDOM significantly outperforms prior approaches on several text classification datasets. The source code can be found at https://github.com/ayushbits/robust-aggregate-lfs.

1 Introduction

Supervised machine learning approaches require large amounts of labeled data to train robust machine learning models. Human-annotated gold labels have become increasingly important to modern machine learning systems for tasks such as spam detection, (movie) genre classification, sequence labeling, etc. The creation of labeled data is, however, a time-consuming and costly process that requires large amounts of human labor. Together with the heavy reliance on labeled data for training models, this serves as a deterrent to achieving comparable performance on new tasks. As a result, various methods such as semi-supervision, distant supervision, and crowdsourcing have been proposed to reduce reliance on human annotation.

In particular, several recent data programming approaches (Bach et al., 2019; Maheshwari et al., 2021; Chatterjee et al., 2020; Awasthi et al., 2020) have proposed the use of human-crafted labeling functions to weakly associate labels with the training data. Typically, users encode supervision as rules/guides/heuristics in the form of labeling functions (LFs) that assign noisy labels to the unlabeled data, thus reducing dependence on human-labeled data. The noisy labels were aggregated using Label aggregators, which often employ generative models, to assign a label to the data instance. Examples of label aggregators are SNORKEL (Ratner et al., 2016) and CAGE (Chatterjee et al., 2020). These models provide consensus on the noisy and conflicting labels assigned by the discrete LFs to help determine the correct labels probabilistically. We could use the obtained labels to train any supervised model/classifier and evaluate on a test set. Apart from the cascaded approach described above, recently proposed semi-supervised paradigm (Awasthi et al., 2020; Maheshwari et al., 2021) learns to aggregate labels using both features and a very small labeled set in addition to labeling functions. Such approaches have been shown to outperform the completely unsupervised data programming approaches described above.

Data programming (unsupervised or semisupervised) requires carefully curated LFs, generally expressed in the form of regular expressions or conditional statements. Even though creating LFs can potentially take less time than creating large amounts of supervised data, it requires domain experts to spend considerable time identifying and
determining the patterns that should be incorporated into LFs. In this paper, we circumvent the requirement of human-curated LFs by instead automatically generating human-interpretable LFs as compositions of simple propositions on the data set by leveraging SNUBA (Varma and Ré, 2018) which utilizes a small labeled-set to induce LFs automatically. However, as we will show, SNUBA suffers from two critical limitations, which keep it from outperforming even a simple supervised baseline that is trained on the same labeled-set. First, SNUBA only uses the labeled-set to generate the LFs but does not make effective use of it in the final model training. Secondly, as it naively aggregates these LFs, it is not able to distinguish between very noisy LFs and more useful ones. This work addresses both of these limitations.

In Table 1, we present a sample set of induced LFs and assigned labels for the TREC dataset (Li and Roth, 2002). The induced LFs are likely to be less precise compared with those created by humans, and they are likely to have more mutual conflicts. Since the LFs are incomplete and noisy, existing label aggregators that merely consume their outputs do not perform well when dealing with such noisy LFs (c.f. Table 1). For instance, the sentence How long does a dog sleep? will be assigned both DESCRIPTION and NUMERIC labels due to the LFs how and how long.

As a solution, how should be given less importance due to its noisy and conflicting nature, whereas how long, associated with the NUMERIC label, should be given higher importance. In this paper, we present a bi-level optimization framework for reweighting the induced LFs, which effectively reduces the weights of noisy labels while simultaneously increasing the weights of the more useful ones.

In Figure 1, we present an overview of our approach. We leverage semi-supervision in the feature space for more effective data programming using the induced (automatically generated) labeling functions. To enable this, we split the same labeled-set (which was used to generate the LFs) into a supervised set and validation set. The supervised set is used for semi-supervised data programming, and validation set is used to tune (reweight) the LFs. As a basic framework for semi-supervised data programming, we leverage SPEAR (Maheshwari et al., 2021), which has achieved state-of-the-art performance. While the semi-supervised data programming approach helps in using the labeled data more effectively, it does not solve the problem of noise associated with the LFs. To address this, we propose an LF reweighting framework, WISDOM, which learns to reweight the labeling functions, thereby helping differentiate the noisy LFs from the cleaner and more effective ones.

The reweighting is achieved by framing the problem in terms of bi-level optimization. We argue that using a small labeled-set can help improve label prediction over hitherto unseen test instances when the labeled-set is bootstrapped for (i) inducing LFs, (ii) semi-supervision, and (iii) bi-level optimization to reweight the LFs. For most of this work, the LFs are induced automatically by leveraging part of the approach described in (Varma and Ré, 2018). The LFs are induced on the entire labeled-set, whereas the semi-supervision and reweighting are performed on the supervised set and validation set respectively (which are disjoint partitions of labeled-set).

**Our Contributions** are as follows: While lever-

---

1Expanded as reWeIghting based Semi-supervised Data prOgraMming
2.1 Notations

Let us denote the feature space by $X$ and the label space by $Y \in \{1 \ldots K\}$ where $K$ is the number of classes. Let the automatically (or manually) generated labeling functions be denoted by $\lambda_1$ to $\lambda_m$ where $m$ is the number of labeling functions generated. Let the vector $l_i = (l_{i1}, l_{i2}, \ldots, l_{im})$ denote the firings of all the LFs on an instance $x_i$. Each $l_{ij}$ can be either 0 or 1; $l_{ij} = 1$ indicates that the LF $\lambda_j$ has fired (i.e., triggered) on the instance $x_i$ and 0 indicates it has not. Furthermore, each labeling function $\lambda_j$ is associated with some class $k_j$ and for an input $x_i$, it outputs the label $\tau_{ij} = k_j$ when triggered (i.e., $l_{ij} = 1$) and $\tau_{ij} = 0$ otherwise.

Let the labeled-set be denoted by $L = \{(x_i, y_i)\}$ where $i \in \{1 \ldots N\}$ and $N$ is the number of points in labeled-set. Similarly, we have an unlabeled dataset denoted as $U = \{x_i\}$ where $i \in \{N + 1 \ldots M\}$ and $M - N$ is the number of unlabeled points. The labeled-set is further split into two disjoint sets called supervised set and validation set. Let the supervised set be denoted by $S = \{(x_i, y_i)\}$ where $i \in \{1 \ldots N/2\}$. Let $Y = \{(x_i, y_i)\}$ denote the validation set, where $i \in \{N/2 + 1 \ldots N\}$.

2.2 SNUBA: Automatic LF Generation

Varma and Ré (2018) present SNUBA, a three step approach that (i) automatically generates candidate LFs (referred to as heuristics) using a labeled-set, (ii) filters heuristics based on diversity and accuracy metrics to select only relevant heuristics, and (iii) uses the final set of filtered LFs (heuristics) and a label aggregator to compute class probabilities for each point in the unlabeled set $U$. Steps (i) and (ii) are repeated until the labeled set is exhausted or a limit on the number of iterations is reached. Each LF is a basic composition of propositions on the labeled set. A proposition could be a word,
a phrase, or a lemma (c.f., the second column of Table 1), or an abstraction such as a part of speech tag. The composition is in the form of a classifier such as a decision stump (1-depth decision tree) or logistic regression.

Our \textsc{Wisdom} framework utilizes \textsc{Snu}ba for generating the LFs and thereafter reweigh the LFs via our reweighting framework while jointly learning the model parameters and the LF aggregation in a semi-supervised manner.

2.3 \textsc{Spear}: Joint SSL Data Programming

\textsc{Maheshwari et al. (2021)} propose a joint learning framework called \textsc{Spear} that learns the parameters of a feature-based classification model and of the label aggregation model (the LF model) in a semi-supervised manner. \textsc{Spear} has a feature-based classification model $f_\phi(x)$ that takes the features as input and predicts the class label. \textsc{Spear} employs two kinds of models: a logistic regression and a two-layer neural network model. For the LF aggregation model, \textsc{Spear} uses an LF-based graphical model inspired from \textsc{Cage} (Chatterjee et al., 2020). \textsc{Cage} aggregates the LFs by regularizing parameters such that learned joint distribution of $y$ and $\tau_j$ matches the user provided quality guides over all $y$.

$$P_\theta(i, y) = \frac{1}{Z_\theta} \prod_{j=1}^{j=m} \psi_\theta(\tau_{ij}, y)$$ \hspace{1cm} (1)

There are $K$ parameters $\theta_{j1}, \theta_{j2}..\theta_{jK}$ for each LF $\lambda_j$, where $K$ is the number of classes. The potential $\psi_\theta$ used in the \textsc{Cage} model is defined as:

$$\psi_\theta(\tau_{ij}, y) = \begin{cases} \exp(\theta_{jy}) & \text{if } \tau_{ij} \neq 0 \\ 1 & \text{otherwise} \end{cases}$$ \hspace{1cm} (2)

The loss function of \textsc{Spear} has six terms. These include the cross entropy on the labeled set, an entropy SSL term on the unlabeled dataset, a cross entropy term to ensure consistency between the feature model and the LF model, the LF graphical model terms on the labeled and unlabeled datasets, a KL divergence again for consistency between the two models, and finally a regularizer. The objective function is:

$$\sum_{i \in \mathcal{L}} \mathcal{L}_{ce}(f_\phi(x_i), y_i) + \sum_{i \in \mathcal{U}} H(f_\phi(x_i)) +$$

$$\sum_{i \in \mathcal{U}} \mathcal{L}_{ce}(f_\phi(x_i), g(l_i)) + LL_u(\theta|\mathcal{L}) + LL_u(\theta|\mathcal{U}) +$$

$$\sum_{i \in \mathcal{U} \cup \mathcal{L}} KL(P_\theta(l_i, f_\phi(x_i)) + R(\theta|\{q_j\})$$ \hspace{1cm} (3)

where $g$ is the label prediction from the LF-based graphical model. The second component $H(\cdot)$ models semi-supervision (Grandvalet and Bengio, 2005) in the form of minimization of the entropy of the predictions on the unlabeled dataset $\mathcal{U}$. It provides some semi-supervision by trying to increase the confidence of the predictions made by the model on the unlabeled dataset. (Refer Table 2 for notations used in the objective function). In the objective function above, the LF model parameters are $\theta$ while the feature model parameters are $\phi$. The learning problem in \textsc{Spear} is simply to optimize the objective jointly over $\theta$ and $\phi$. (We refer readers to \textsc{Maheshwari et al. (2021)} for details.)

\textbf{CAGE loss formulation:} The learning problem proposed in \textsc{Cage} (Chatterjee et al., 2020) is a special case of \textsc{Spear} where they just use the fifth loss term $LL_u(\theta|\mathcal{U})$ along with the quality guide $R(\theta|\{q_j\})$. The specific loss formulation of \textsc{Cage} is as given below:

$$LL_u(\theta|\mathcal{U}) + R(\theta|\{q_j\})$$ \hspace{1cm} (4)

3 The \textsc{Wisdom} Workflow

In this section, we present our robust aggregation framework for automatically generated LFs. We present the LF generation approach followed by our reweighting algorithm, which solves a bi-level optimization problem. In the bi-level optimization, we learn the LF weights in the outer level, and in the inner level, we learn the feature-based classifier’s and labeling function aggregator’s parameters jointly. We describe the main components of the \textsc{Wisdom} framework below (see also Figure 1). A detailed pseudocode of \textsc{Wisdom} is provided in Algorithm 1. We describe the different components of \textsc{Wisdom} below.

\textbf{Automatic LF Generation using \textsc{Snu}ba:} Our \textsc{Wisdom} framework utilizes steps (i) and (ii) from \textsc{Snu}ba (c.f., Section 2.2) for automatically inducing LFs. That is, it initially iterates between i) candidate LF generation on labeled-set $\mathcal{L}$ and ii) filtering them based on diversity and accuracy based criteria, until a limit on the number of iterations is reached (or until the labeled set is completely covered). We refer to these steps as \textsc{Snu}balfgen.

\textbf{Re-Weighting \textsc{Cage}:} To deal with noisy labels effectively, we associate each LF $\lambda_j$ with an additional weight parameter $w_j \in [0, 1]$ that acts as its reliability measure. The $w$’s are optimized on the validation set and have interactions amongst themselves, unlike $\theta$ which is learned on the combination of unlabeled and training sets. The discrete
potential in CAGE (c.f., eq.(2)) can be modified to include weight parameters as follows:

$$
\psi_\theta(\tau_{ij}, y) = \begin{cases} 
\exp(w_j \theta_{jy}) & \text{if } \tau_{ij} \neq 0 \\
1 & \text{otherwise}
\end{cases}
$$

(5)

We observe that if the weight of the $j^{th}$ LF is zero (i.e., $w_j = 0$), the corresponding weighted potential in eq. (5) becomes one, which in turn implies that the $j^{th}$ LF is ignored while maximizing the log-likelihood during label aggregation. Similarly, if all the LFs are associated with a weight value of one (i.e., $w_j = 1$), the above weighted potential will degenerate to the discrete potential used in CAGE. The re-weighted CAGE is implicitly invoked on lines 12, 13, 17 and 18 of Algorithm 1 where $\mathcal{L}_{SS}(\theta, \phi, w)$ is invoked. We compare performance of CAGE with a bi-level variation in Table 5.

---

Algorithm 1: WISDOM

Input: $L, S,V, U, \mathcal{L}$, Learning rates: $\alpha, \beta$

Output: $\theta, \phi, w$

1. // Automatic LF generation using SNUBA ****
2. $\lambda_1, \ldots, \lambda_m = \text{SNUBA LFG}(\mathcal{L})$
3. Get LFs trigger matrix $\lambda^t$, $\lambda^n$ for sets $S, U$ using $\lambda_1, \ldots, \lambda_m$
4. Get LFs output label matrix $\tau^s, \tau^n$ for sets $S, U$ using $\lambda_1, \ldots, \lambda_m$
5. // The Reweighted Joint SSL ****
6. $t = 0$
7. Randomly initialize model parameters $\theta_0, \phi_0$ and LF weights $w_0$
8. repeat
9. Sample mini-batch $s = (x_i^s, y_i^s, \tau_i^s, \Gamma_i^s)$ of batch size $B$ from $\{S, \tau^s, \Gamma_i^s\}$
10. // Bi-level Optimization ****
11. // Inner level ****
12. $\theta_t^* = \theta_t - \alpha \nabla_{\theta} \mathcal{L}_{ss}(\theta_t, \phi_t, w_t)$
13. $\phi_t^* = \phi_t - \alpha \nabla_{\phi} \mathcal{L}_{ss}(\theta_t, \phi_t, w_t)$
14. // Outer level ****
15. $w_{t+1} = w_t - \beta \nabla_{w} \frac{1}{|V|} \sum_{i \in V} \mathcal{L}_{ss}(f_{\phi_t}(x_i), y_i)$
16. // Update net parameters $\phi, \theta$ ****
17. $\phi_{t+1} = \phi_t + \alpha \nabla_{\phi} \mathcal{L}_{ss}(\theta_{t+1}, \phi_t, w_{t+1})$
18. $\phi_{t+1} = \phi_{t+1} - \alpha \nabla_{\phi} \mathcal{L}_{ss}(\theta_t, \phi_t, w_{t+1})$
19. $t = t + 1$
20. until convergence
21. return $\theta_{t+1}, \phi_{t+1}, w_{t+1}$

---

The Reweighted Joint SSL: Since the label aggregator graphical model is now dependent on the additional LF weight parameters $w$, the joint semi-supervised learning objective function is modified as follows:

$$
\mathcal{L}_{ss}(\theta, \phi, w) = \sum_{l \in \mathcal{L}} \mathcal{L}_{ce}(f_{\phi}(x_l), y_l) + \sum_{l \in \mathcal{U}} H(f_{\phi}(x_l)) + \sum_{l \in \mathcal{U}} \mathcal{L}_{ce}(f_{\phi}(x_l), g(l, w)) + LL_u(\theta, w|\mathcal{S}) + LL_w(\theta, w|U) + \sum_{l \in \mathcal{L}} KL(P_{\theta, w}(l), f_{\phi}(x_l)) + R(\theta, w|\{y_l\})
$$

(6)

In Section 7, we present the somewhat intuitive expansions of terms that are dependent on $w$.

**Bi-Level Objective:** WISDOM jointly learns the LF weights and weighted labeling aggregator and feature classifier parameters for the objective function defined in Equation (6). The LF weights are learned by WISDOM by posing a bi-level optimization problem for this objective function as defined in eq. (7) and employing alternating one-step gradient updates. As evident in eq. (7), WISDOM uses a validation set $|\mathcal{V}|$ which is a subset of labeled-set $|\mathcal{L}|$ to learn the LF weights. Furthermore, the introduced weight parameters allow filtering of LFs based on the feature model and a bilevel objective in the form of a cross-entropy loss of feature model predictions on the validation set. In essence, WISDOM tries to learn LF weights that result in minimum validation loss on the feature model that is jointly trained with weighted labeling aggregator.

$$
w^* = \arg\min_w \frac{1}{|\mathcal{V}|} \sum_{i \in \mathcal{V}} \mathcal{L}_{ce}(f_{\phi^*}(x_i), y_i)
$$

where $\phi^*, \theta^* = \arg\min_{\phi, \theta} \mathcal{L}_{ss}(\theta, \phi, w)$

(7)

However, determining the optimal solution to the above Bi-level objective function is computationally intractable. Hence, inspired by MAML (Finn et al., 2017), WISDOM adopts an iterative alternative minimizing framework, wherein we optimize the objective function at each level using a single gradient descent step. As shown in Algorithm 1, lines 12 and 13 are the inner level updates where the parameters $\theta, \phi$ are updated using the current choice of weight parameters $w$ for one gradient step, and in line 15, the weight parameter $w$ is updated using the one-step updates from lines 12 and 13. Finally, the net parameters $\phi, \theta$ are updated in lines 17 and 18. This procedure is continued till convergence (e.g., no improvement in the outer-level loss) or for a fixed number of epochs.
We use the following datasets in our experiments. We present evaluations across six datasets that we describe in the following Section 4.1. In Table 3, we present summary statistics of these datasets, including the sizes of supervised set, validation set (with labeled-set being the union of these disjoint sets) and the number of (auto-generated) LFs used in the experiments.

### 4 Experiments

We present evaluations across six datasets that we describe in the following Section 4.1. In Table 3, we present summary statistics of these datasets, including the sizes of supervised set, validation set (with labeled-set being the union of these disjoint sets) and the number of (auto-generated) LFs used in the experiments.

#### 4.1 Datasets

We use the following datasets in our experiments: (1) TREC (Li and Roth, 2002): A question classification dataset with six categories: Description, Entity, Human, Abbreviation, Numeric, Location. (2) YouTube Spam Classification (Alberto et al., 2015): A spam classification task over comments on YouTube videos. (3) IMDB Genre Classification\(^2\): A plot summary based movie genre binary classification dataset. (4) SMS Spam Classification (Almeida et al., 2011): A binary spam classification dataset to detect spam in SMS messages. (5) Twitter Sentiment (Wan and Gao, 2015): This is a 3-class sentiment classification problem extracted from Twitterfeed of popular airline handles. Each tweet is either labeled as negative, neutral, and positive labels. (6) Stanford Sentiment Treebank (SST-5) (Socher et al., 2013) is a single sentence movie review dataset, with each sentence labeled as either negative, somewhat negative, neutral, somewhat positive, or positive.

#### 4.2 Baselines

In Table 4, we compare our approach against the following baselines: SnuBa (Varma and Ré, 2018): Recall from Section 2.2 that SNUBA iteratively induces LFs from the count-based raw features of the dataset in the steps (i) and (ii). For the step (iii), as in (Varma and Ré, 2018), we employ a generative model to assign probabilistic labels to the unlabeled set. These probabilistic labels are obtained by training a 2-layered NN model.

**Supervised (SUP):** This is the model obtained by training the classifier \( P_\theta(y|x) \) only on labeled-set. This baseline does not use the unlabeled set.

**Learning to Reweight (L2R) (Ren et al., 2018):** This method trains the classifier using a meta-learning algorithm over the noisy labels in the unlabeled set obtained using the automatically generated labeling functions and aggregated using Snorkel. It uses an online algorithm that assigns importance to examples based on the gradient.

**Posterior Regularization (PR) (Hu et al., 2016):** This is a method for joint learning of a rule and feature network in a teacher-student setup. Similarly to L2R, it uses the noisy labels in the unlabeled set obtained using the automatically generated labeling functions.

**Imply Loss (IL) (Awasthi et al., 2020):** This method leverages both rules and labeled data by associating each rule with exemplars of correct firings (i.e., instantiations) of that rule. Their joint training algorithms de-noise over-generalized rules and train a classification model. This is also run on the automatically generated LFs.

**SPEAR (Maheshwari et al., 2021):** This method employs a semi-supervised framework combined with a graphical model for consensus amongst the LFs to train the model. We compare against two versions of SPEAR. The first one is SPEAR-L2R, which uses the human LFs.

**VAT:** Virtual Adversarial Training (Miyato et al., 2018) is a semi-supervised approach that uses the virtual adversarial loss on the unlabeled points, thereby ensuring robustness of the conditional label distribution on the unlabeled points.

#### 4.3 Experimental Setting

To train our model on the supervised set, we use a neural network architecture with two hidden layers (512 units) and ReLU activation function as our feature-based model \( f_\phi \). We choose our classification network to be the same as SPEAR (Maheshwari et al., 2021). We consider two types of features: a) raw words and b) lemmatizations, as an input to our supervised model (lemmatization is a technique to reduce a word, e.g., ‘walking,’ into its root form, ‘walk’). Additionally, these features are used as basic propositions over which composite LFs are built.

Each experimental run involves training Wis-
DOM for 100 epochs with early stopping based on validation set. Our model is optimized using mini-batch gradient descent with the Adam optimizer. We tuned the hyperparameters on the validation set, and the optimal configuration was found to have a dropout probability of 0.80 and a batch size of 32. Further, the optimal configuration learning rates for the classifier and LF aggregation models were 0.0003 and 0.01, respectively. Performance numbers for each experiment are obtained by averaging over five independent runs, each having a different random initialization. For evaluation on the test set, the model with the best performance on the validation set was chosen. On all datasets, macro-F1 is employed as the evaluation criterion. We implement all our models in PyTorch\(^3\) (Paszke et al., 2019). We run all our experiments on Nvidia RTX 2080 Ti GPUs with 12 GB RAM set within Intel Xeon Gold 5120 CPU having 56 cores and 256 GB RAM. Model training times range from 15 mins (YouTube) to 100 mins (TREC).

4.4 Results

In Table 4, we compare the performance of WISDOM against different baselines (all using auto-generated labeling functions except VAT), for both raw and lemmatized count features (c.f. Section 2.2) across multiple datasets. We observe that SNUBA performs worse than the Supervised baseline on all datasets, exhibiting high variance over different runs (surprisingly, Varma and Ré (2018) did not compare the performance of SNUBA against the supervised baseline). Learning to Reweight (L2R) performs worse than Supervised on all datasets except YouTube. Posterior regularization, imply loss and SPEAR show gains over Supervised on a few datasets, but not consistently across all datasets and settings. Finally, VAT obtains competitive results in some settings (e.g., TREC dataset) but performs much worse on others (e.g., IMDB and SST-5). In contrast, WISDOM achieves consistent gains over Supervised and the other baselines in almost all datasets (except TREC with raw features where VAT does slightly better than WISDOM). Additionally, WISDOM yields smaller variance over different runs compared to other semi-supervised approaches. Recall that the main difference between WISDOM and Auto-SPEAR is that the former reweights the LFs in both the label aggregator as well as in the semi-supervised loss, as against Auto-SPEAR which does not reweight the LFs at all. Consequently, the aforementioned empirical gains illustrate the robustness of the bi-level optimisation algorithm. Note that these numbers are all reported using only 10% labeled data, and hence, results for some datasets (starting with Supervised) might appear lower than those reported in the literature. Note that, we compare WISDOM (using automatically induced LFs) against the HUM-SPEAR which uses the human crafted LFs in conjunction with the state-of-the-art SPEAR approach (Maheshwari et al., 2021). Although WISDOM uses auto-generated LFs, it sometimes performs better than HUM-SPEAR, which utilizes human-curated LFs. On careful analysis (presented in Section 8 of the supplementary), we observe that the human curated LFs tend to be more generic abstractions of possible patterns without assessing how precise they are for the end task. Consequently, these abstract human-LFs tend to have not only higher collective coverage but also high mutual conflicts and lower average individual precision values than the automatically induced LFs. Given the individual strengths of both human-LFs and auto-LFs, it might be interesting to consider using them in conjunction with each other in order to improve performance as future work. An ablation test in Figure 3 reveals that WISDOM performs well even for small-sized labeled-set unlike other baselines, demonstrating its robustness in scenarios with only few labeled examples.

4.5 Importance of the Bi-Level formulation

A label aggregation approach, such as CAGE, SNORKEL, may improve the consensus labeling across LFs, but not necessarily their agreement with the ground truth. Further, when LFs are noisy (or induced automatically), the performance of the CAGE model can suffer. However, the bi-level framework of CAGE can alleviate these problems since it implicitly reduces the noise in LFs. In order to demonstrate effectiveness of the bi-level formulation, we compare CAGE(Eq (4)) with two variants (i) CAGE\(^{val}\)\(^4\) that considers validation set feedback in the loss formulation for promoting LF agreement with ground-truth label and (ii) CAGEBi-level with the proposed bi-level formulation that tries to do the same\(^5\). We present our results in Table 5. The performance of our CAGEBi-level is clearly superior to the original CAGE model, as well as to the CAGE\(^{val}\) model. Thus, the bi-level formulation more effectively incorporates validation set feedback than other formulations as demonstrated by application of bi-level on both SPEAR as well

\(^3\)https://pytorch.org/

\(^4\)CAGE\(^{val}\) - equivalent to using only \(LL_u(\theta|\mathcal{L}) + LL_v(\theta, w|\mathcal{L}) + R(\theta, w|\mathcal{L})\) in Eq (3)

\(^5\)In other words, CAGEBi-level is equivalent to using only \(LL_u(\theta, w|\mathcal{L}) + R(\theta, w|\mathcal{L})\) in Eq (6)
5 Related Work

In this section, we describe some related work that was not covered in Section 2.

Automatic Rule Generation: The programming by examples paradigm produces a program from a given set of input-output pairs (Gulwani, 2012; Singh and Gulwani, 2012). It synthesises those programs that satisfy all input–output pairs. RuleNN (Sen et al., 2020) learns interpretable first-order logic rules as composition of semantic role attributes. Many of these approaches, however, learn more involved rules (using e.g., a neural network) which may not work in the realistic setting of very small labeled data. In contrast, SNUBA and WISDOM use more interpretable models (Rudin, 2019) like logistic regression and decision trees for rule induction.

Semi-supervised Learning (SSL): The goal of SSL is to effectively use unlabeled data while training. Early SSL algorithms used regularization-based approaches like margin regularization, and laplacian regularization (Chapelle et al., 2010). Most recent SSL approaches like Mean Teacher (Tarvainen and Valpola, 2018), VAT (Miyato et al., 2018),UDA (Xie et al., 2020), MixMatch (Berthelot et al., 2019) and FixMatch (Sohn et al., 2020) introduced various kinds of perturbations and augmentations that can be used along with consistency loss. Even though the current SSL approaches perform well even with minimal labels, they are computationally intensive and cannot be easily implemented in low-resource scenarios. Furthermore, it is tough to explain the discriminative behavior of the semi-supervised models.

Bi-level Optimization: The concept of bi-level optimization has been discussed in (von Stackelberg et al., 1952; Bracken and McGill, 1973; Bard, 2006). Since then, the framework of bi-level opti-
mization has been used in various machine learning applications like hyperparameter tuning (Mackay et al., 2019; Franceschi et al., 2018; Sinha et al., 2020), robust learning (Ren et al., 2018; Guo et al., 2020), meta-learning (Finn et al., 2017), efficient learning (Killamsetty et al., 2021) and continual learning (Borsos et al., 2020). Previous applications of the bi-level optimization framework for robust learning have been limited to supervised and semi-supervised learning settings. To the best of our knowledge, WISDOM is the first framework that uses a bi-level optimization approach for robust aggregation of labeling functions.

6 Conclusion

While induction of labeling functions (LFs) for data-programming has been attempted in the past by Varma and Ré (2018), we observe in our experiments that the resulting model in itself does not perform well on text classification tasks, and turns out to be even worse than the supervised baseline. A more recent semi-supervised data programming approach called SPEAR (Maheshwari et al., 2021), when used in conjunction with the induced LFs, performs better, though it fails to consistently outperform the supervised baseline. In this paper, we introduce WISDOM, a bi-level optimization formulation for reweighting the LFs, which injects robustness into the semi-supervised data programming approach, thus allowing it to perform well in the presence of noisy LFs. On a reasonably wide variety of text classification datasets, we show that WISDOM consistently outperforms all other approaches, while also coming close to the skyline of SPEAR using human-generated LFs.

Acknowledgements and Disclosure of Funding

We thank anonymous reviewers for providing constructive feedback. Ayush Maheshwari is supported by a Fellowship from Ekal Foundation (www.ekal.org). We are also grateful to IBM Research, India (specifically the IBM AI Horizon initiative) for their support and sponsorship. Rishabh Iyer and Krishnateja Killamsetty were funded by the National Science Foundation(NSF) under Grant Number 2106937, a startup grant from UT Dallas, and a Google and Adobe research award. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation, Google or Adobe.

References

Túlio C Alberto, Johannes V Lochter, and Tiago A Almeida. 2015. Tubespam: Comment spam filtering on youtube. In 2015 IEEE 14th international conference on machine learning and applications (ICMLA), pages 138–143. IEEE.

Tiago A Almeida, José Maria G Hidalgo, and Akebo Yamakami. 2011. Contributions to the study of sms spam filtering: new collection and results. In Proceedings of the 11th ACM symposium on Document engineering, pages 259–262.

Abhijeet Awasthi, Sabyasachi Ghosh, Rasna Goyal, and Sunita Sarawagi. 2020. Learning from rules generalizing labeled exemplars. In 8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020. OpenReview.net.

Stephen H Bach, Daniel Rodriguez, Yintao Liu, Chong Luo, Haidong Shao, Cassandra Xia, Souvik Sen, Alex Ratner, Braden Hancock, Houman Alborzi, et al. 2019. Snorkel drybell: A case study in deploying weak supervision at industrial scale. In Proceedings of the 2019 International Conference on Management of Data, pages 362–375.

Jonathan F. Bard. 2006. Practical Bilevel Optimization: Algorithms and Applications (Nonconvex Optimization and Its Applications). Springer-Verlag, Berlin, Heidelberg.

David Berthelot, Nicholas Carlini, Ian Goodfellow, Nicolas Papernot, Avital Oliver, and Colin Raffel. 2019. Mixmatch: A holistic approach to semi-supervised learning.

Zalán Borsos, Mojmír Mutný, and Andreas Krause. 2020. Coresets via bilevel optimization for continual learning and streaming.

Jerome Bracken and James T. McGill. 1973. Mathematical programs with optimization problems in the constraints. Operations Research, 21(1):37–44.

Olivier Chapelle, Bernhard Schlkopf, and Alexander Zien. 2010. Semi-Supervised Learning, 1st edition. The MIT Press.

Oishik Chatterjee, Ganesh Ramakrishnan, and Sunita Sarawagi. 2020. Robust data programming with precision-guided labeling functions. In AAAI.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. 2017. Model-agnostic meta-learning for fast adaptation of deep networks.

Luca Franceschi, Paolo Frasconi, Saverio Salzo, Riccardo Grazzi, and Massimilano Pontil. 2018. Bilevel programming for hyperparameter optimization and meta-learning.

Yves Grandvalet and Yoshua Bengio. 2005. Semi-supervised learning by entropy minimization. In Advances in neural information processing systems, pages 529–536.
Sumit Gulwani. 2012. Synthesis from examples: Interaction models and algorithms. In 2012 14th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing, pages 8–14. IEEE.

Lan-Zhe Guo, Zhen-Yu Zhang, Yuan Jiang, Yu-Feng Li, and Zhi-Hua Zhou. 2020. Safe deep semi-supervised learning for unseen-class unlabeled data. In Proceedings of the 37th International Conference on Machine Learning, volume 119 of Proceedings of Machine Learning Research, pages 3897–3906. PMLR.

Zhiting Hu, Xueze Ma, Zhengzhong Liu, Eduard H. Hovy, and Eric P. Xing. 2016. Harnessing deep neural networks with logic rules. CoRR, abs/1603.06318.

Krishnateja Killamsetty, Durga Sivasubramanian, Ganesh Ramakrishnan, and Rishabh Iyer. 2021. Glister: Generalization based data subset selection for efficient and robust learning. In AAAI 2021.

Xin Li and Dan Roth. 2002. Learning question classifiers. In COLING 2002: The 19th International Conference on Computational Linguistics.

Matthew Mackay, Paul Vicol, Jonathan Lorraine, David Duvenaud, and Roger Grosse. 2019. Self-tuning networks: Bilevel optimization of hyperparameters using structured best-response functions. In International Conference on Learning Representations.

Ayush Maheshwari, Oishik Chatterjee, KrishnaTeja Killamsetty, Rishabh K. Iyer, and Ganesh Ramakrishnan. 2021. Data programming using semi-supervision and subset selection. In Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics.

Takeru Miyato, Shin-ichi Maeda, Masanori Koyama, and Shin Ishii. 2018. Virtual adversarial training: a regularization method for supervised and semi-supervised learning. IEEE transactions on pattern analysis and machine intelligence, 41(8):1979–1993.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. PyTorch: An imperative style, high-performance deep learning library.

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, pages 3567–3575.

Mengye Ren, Wenyuan Zeng, Bin Yang, and Raquel Urtasun. 2018. Learning to reweight examples for robust deep learning. In International Conference on Machine Learning, pages 4334–4343. PMLR.

Cynthia Rudin. 2019. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. Nature Machine Intelligence, 1:206–215.

Prithviraj Sen, Marina Danilevsky, Yunyao Li, Siddhartha Brahma, Matthias Boehm, Laura Chiticariu, and Rajasekar Krishnamurthy. 2020. Learning explainable linguistic expressions with neural inductive logic programming for sentence classification. In Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 4211–4221.

Rishabh Singh and Sumit Gulwani. 2012. Synthesizing number transformations from input-output examples. In International Conference on Computer Aided Verification, pages 634–651. Springer.

Ankur Sinha, Tanmay Khandait, and Raja Mohanty. 2020. A gradient-based bilevel optimization approach for tuning hyperparameters in machine learning.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y 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.

Kihyuk Sohn, David Berthelot, Chun-Liang Li, Zizhao Zhang, Nicholas Carlini, Ekin D. Cubuk, Alex Kurakin, Han Zhang, and Colin Raffel. 2020. Fixmatch: Simplifying semi-supervised learning with consistency and confidence.

Antti Tarvainen and Harri Valpola. 2018. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results.

Paroma Varma and Christopher Ré. 2018. Snuba: Automating weak supervision to label training data. Proc. VLDB Endow., 12(3):223–236.

Paroma Varma and Christopher Ré. 2018. Snuba: automating weak supervision to label training data. In Proceedings of the VLDB Endowment. International Conference on Very Large Data Bases, volume 12, page 223. NIH Public Access.

H. von Stackelberg, S.H. Von, and A.T. Peacock. 1952. The Theory of the Market Economy. Oxford University Press.

Yun Wan and Qigang Gao. 2015. An ensemble sentiment classification system of twitter data for airline services analysis. In 2015 IEEE international conference on data mining workshop (ICDMW), pages 1318–1325. IEEE.
Qizhe Xie, Zihang Dai, Eduard Hovy, Minh-Thang Luong, and Quoc V. Le. 2020. *Unsupervised data augmentation for consistency training.*
Appendix

7 Explanation of loss terms

First Component (L1): The first component (L1) of the loss $L_{CE} \left( P_{\phi}^f(y|x_i), y_i \right) = - \log \left( P_{\phi}^f(y = y_i|x_i) \right)$ is the standard cross-entropy loss on the labelled dataset $L$ for the model $P_{\phi}^f$.

Second Component (L2): The second component L2 is the semi-supervised loss on the unlabelled data $U$. In our framework, we can use any unsupervised loss function.

Third Component (L3): The third component $L_{CE} \left( P_{\phi}^f(y|x_i), g(l_i), w \right)$ is the cross entropy of the classification model using the hypothesised labels from CAGE (Chatterjee et al., 2020) on $U$. Given that $l_i$ is the output vector of all labelling functions for any $x_i \in U$, we specify the predicted label for $x_i$ using the LF-based graphical model $P_{\theta}(l_i, y)$ as: $g(l_i) = \arg\max P_{\theta,w}(l_i, y)$

Fourth Component (L4): The fourth component $LL_s(\theta|L)$ is the (supervised) negative log likelihood loss on the labelled dataset $L$: $LL_s(\theta, w|L) = - \sum_{i=1}^{N} \log P_{\theta,w}(l_i, y_i)$

Fifth Component (L5): The fifth component $LL_u(\theta, w|U)$ is the negative log likelihood loss for the unlabelled dataset $U$. Since the true label information is not available, the probabilities need to be summed over $y$: $LL_u(\theta, w|U) = - \sum_{i=N+1}^{M} \log \sum_{y \in Y} P_{\theta,w}(l_i, y)$

Sixth Component (L6): The sixth component $KL(P_{w}^j(y|x_i), P_{\theta}(y|l_i))$ is the Kullback-Leibler (KL) divergence between the predictions of both the models, viz., feature-based model $f_{\phi}$ and the LF-based graphical model $P_{\theta}$ summed over every example $x_i \in U \cup L$. Through this term, we try and make the models agree in their predictions over the union of the labelled and unlabelled datasets.

Quality Guides (QG): As a last component in our objective, we use quality guides $R(\theta, w|\{q_j\})$ on LFs which have been shown (Chatterjee et al., 2020) to stabilise the unsupervised likelihood training while using labelling functions. Let $q_j$ be the fraction of cases where $l_j$ correctly triggered. And let $q_j^t$ be the user’s belief on the fraction of examples $x_i$ where $y_i$ and $l_{ij}$ agree. If user’s beliefs weren’t available, we consider precision of LFs on validation set as the user’s beliefs. Except SMS dataset, we take precision of LFs on validations set as quality guides. If $P_{\theta,w}(y_i = k_j|l_{ij} = 1)$ is the model-based precision over the LFs, the quality guide based loss can be expressed as $R(\theta, w|\{q_j^t\}) = - \left( \sum_j q_j^t \log P_{\theta,w}(y_i = k_j|l_{ij} = 1) + (1 - q_j^t) \log(1 - P_{\theta,w}(y_i = k_j|l_{ij} = 1)) \right)$.

8 LF Analysis

We compare statistics of automatically induced LFs and human-curated LFs in Table 6. While developing LFs, humans generally tend to design LFs based on generalizibility of the pattern without worrying much about the conflicts among the patterns. Whereas the LF induction in WISDOM focuses on inducing individually precise LFs without necessarily focusing on the overall coverage. Except in the case of the SMS dataset, collective coverage of human designed LFs is much higher than that of the automatically induced LFs. We also observe in Table 6 that higher coverage leads to higher conflicts. Whereas, on an average, the precision is higher for each of the automatically induced LFs in the case of every dataset.

9 Qualitative Analysis of Automatically Induced LFs

For the six datasets used for experimentation, we automatically induce LFs using Snuba (Varma and Ré, 2018). We show the automatically induced LFs and their respective weights assigned by WISDOM for three datasets TREC, IMDB, and SMS below.

In Table 7, we present LFs produced by the Snuba for the TREC dataset sorted in descending order of weights for each class along with the weights assigned by WISDOM to each of the LFs. From analysis, we observe that WISDOM does a good job of reweighting LFs. For instance, how many was given higher weightage than how and many for class Numeric; this sounds logical as well since sentences containing the keyword how many are more likely to belong to class Numeric than sentences containing...
| Class  | LF               | Weights |
|--------|------------------|---------|
| NUM    | how many         | 1       |
| NUM    | how              | 1       |
| NUM    | many             | 0.62    |
| DESC   | what kind        | 1       |
| DESC   | what was         | 0.54    |
| LOC    | city             | 1       |
| LOC    | country          | 0.84    |
| LOC    | where            | 0.05    |
| ENTY   | what does        | 1       |
| ENTY   | def              | 1       |
| ENTY   | why              | 0.8     |
| ENTY   | what is          | 0.65    |
| HUM    | who              | 0.00012 |

Table 7: Automatically induced LFs by Snuba (Varma and Ré, 2018) for the TREC dataset sorted in descending order of weights per class assigned by WISDOM. Column 1 refers to the class associated with the induced LF. No LFs were induced for class Abbreviation.

The keyword how or many. Another example is among LFs associated with Location class, LFs city and country were given higher weightage than where. However, WISDOM does a poor job by assigning a very small weight value to the single LF who associated with the Human class.

In Table 8, we present LFs produced by the Snuba for the IMDB dataset sorted in descending order of weights for each class along with the weights assigned by WISDOM to each of the LFs. For the IMDB dataset as well, we can see that WISDOM does a good job of reweighting LFs. For instance, among the LFs associated with the class ROMANCE, wife and love were given higher weightage than other LFs like friendship, wealthy, town; this sounds logical as well since ROMANCE is often associated with the sentences containing the keywords wife, love than sentences containing the keyword friendship, town, wealthy. One more key observation is that apart from LFs wife and love, all other LFs associated with the class ROMANCE are given weights of 0 (equivalent to ignoring them). However, assigning 0 weights is controversial for LFs like boyfriend since there is a possibility of ROMANCE associated with the sentence containing keyword boyfriend. Similarly for LFs associated with Action class, LFs government, agent, and plan were given higher weightage than race, and team.

In Table 9, we present LFs produced by the Snuba for the SMS dataset sorted in descending order of weights for each class along with the weights assigned by WISDOM to each of the LFs. For the SMS dataset, we can see that WISDOM did not do as good a job of reweighting as done on other datasets. For instance, among the LFs associated with the class SPAM, ur, video and cam were given higher weightage while completely ignoring (i.e., assigned a weight of zero) to other important LFs like free, claim, won. Whereas for LFs associated with the class NOT SPAM, WISDOM did a good job. One possible reason for the poor job of WISDOM for reweighting LFs associated with the class SPAM is that class imbalance present in the unlabeled set, where the sample count of samples of the class SPAM is

```
| Auto LFs | Human LFs |
|----------|-----------|
| #LFs     | #LFs      |
| Precision| Precision |
| Conflict | Conflict  |
| Cover (%)| Cover (%) |
| YouTube  | 11        | 10       |
| 94.3     | 79.8      |
| 8.1      | 28.7      |
| 63.4     | 88.0      |
| SMS      | 25        | 73       |
| 94.9     | 92.3      |
| 3.2      | 1.0       |
| 47.9     | 33.3      |
| TREC     | 13        | 68       |
| 70.1     | 59.9      |
| 2.3      | 22.3      |
| 62.3     | 95.1      |
```

Table 6: Comparison of automatically generated LFs with human-curated LFs. Coverage is fraction of instances in $U$ covered by at least one rule. Precision refers to micro precision of rules. Conflict denotes the fraction of instances covered by conflicting rules among all the covered instances.

In Table 8, we present LFs produced by the Snuba for the IMDB dataset sorted in descending order of weights for each class along with the weights assigned by WISDOM to each of the LFs. For the IMDB dataset as well, we can see that WISDOM does a good job of reweighting LFs. For instance, among the LFs associated with the class ROMANCE, wife and love were given higher weightage than other LFs like friendship, wealthy, town; this sounds logical as well since ROMANCE is often associated with the sentences containing the keywords wife, love than sentences containing the keyword friendship, town, wealthy. One more key observation is that apart from LFs wife and love, all other LFs associated with the class ROMANCE are given weights of 0 (equivalent to ignoring them). However, assigning 0 weights is controversial for LFs like boyfriend since there is a possibility of ROMANCE associated with the sentence containing keyword boyfriend. Similarly for LFs associated with Action class, LFs government, agent, and plan were given higher weightage than race, and team.

In Table 9, we present LFs produced by the Snuba for the SMS dataset sorted in descending order of weights for each class along with the weights assigned by WISDOM to each of the LFs. For the SMS dataset, we can see that WISDOM did not do as good a job of reweighting as done on other datasets. For instance, among the LFs associated with the class SPAM, ur, video and cam were given higher weightage while completely ignoring (i.e., assigned a weight of zero) to other important LFs like free, claim, won. Whereas for LFs associated with the class NOT SPAM, WISDOM did a good job. One possible reason for the poor job of WISDOM for reweighting LFs associated with the class SPAM is that class imbalance present in the unlabeled set, where the sample count of samples of the class SPAM is
| Class       | LF      | Weights |
|------------|---------|---------|
| ROMANCE    | wife    | 0.412   |
| ROMANCE    | love    | 0.042   |
| ROMANCE    | boyfriend | 0      |
| ROMANCE    | friendship | 0      |
| ROMANCE    | wealthy | 0       |
| ROMANCE    | story   | 0       |
| ROMANCE    | town    | 0       |
| ROMANCE    | friend  | 0       |
| ACTION     | government | 1    |
| ACTION     | plan    | 0.985   |
| ACTION     | agent   | 0.913   |
| ACTION     | team    | 0.753   |
| ACTION     | race    | 0.685   |

Table 8: Automatically induced LFs by Snuba (Varma and Ré, 2018) for the IMDB dataset sorted in descending order of weights per class assigned by WISDOM. Column 1 refers to the class associated with the induced LF.

| Class    | LF    | Weights |
|----------|-------|---------|
| SPAM     | ur    | 1       |
| SPAM     | video | 1       |
| SPAM     | com   | 1       |
| SPAM     | contact | 0.2213 |
| SPAM     | holiday | 0.1593 |
| SPAM     | free  | 0       |
| SPAM     | claim | 0       |
| SPAM     | stop  | 0       |
| SPAM     | won   | 0       |
| SPAM     | win   | 0       |
| SPAM     | uk    | 0       |
| SPAM     | text  | 0       |
| SPAM     | urgent | 0      |
| NOTSPAM  | come  | 1       |
| NOTSPAM  | ok    | 1       |
| NOTSPAM  | got   | 1       |
| NOTSPAM  | like  | 1       |
| NOTSPAM  | sorry | 0.03731254 |

Table 9: Automatically induced LFs by Snuba (Varma and Ré, 2018) for the SMS dataset sorted in descending order of weights per class assigned by WISDOM. Column 1 refers to the class associated with the induced LF.
eight times smaller than the sample count of the class SPAM. From our LF analysis results across the three datasets, we observe that WISDOM tries to up weigh LFs that are more specific and precise and downweigh LFs that are abstract and less precise.