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 incorpo-

[*] Equal contribution

Findings of the Association for Computational Linguistics: ACL 2022, pages 1188 - 1202
May 22-27, 2022 ©2022 Association for Computational Linguistics

Label	Generated LFs	Weighting
ENTITY	what does	↑
DESCRIPTION	what is	↓
NUMERIC	how long	↑
DESCRIPTION	how	↓
HUMAN	who	↑
DESCRIPTION	what kind	↓
LOCATION	city	↑

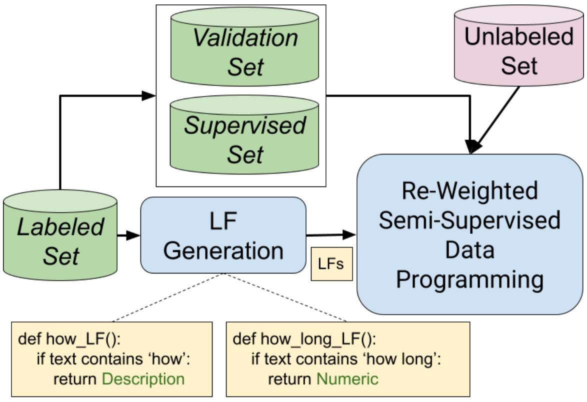
Table 1: Illustration of induced LFs, including examples of the issue of conflicting LFs, on the TREC dataset. Learning importance (weights) of LFs can be used to reduce the conflicts among LFs.

rated 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[1], which learns to reweight the labeling functions, thereby helping differentiate the noisy LFs from the cleaner and more effective ones.



```
def how_LF():
    if text contains 'how':
        return Description
```

```
def how_long_LF():
    if text contains 'how long':
        return Numeric
```

Figure 1: Pictorial depiction of our WISDOM workflow. A small *labeled-set* is used to automatically induce LFs. This labeled set is split equally into *supervised set* and *validation set* to be used by our re-weighted semi-supervised data programming algorithm along with the unlabeled set.

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 leveraging SNUBA (Varma and Ré, 2018) only for *au-*

[1] Expanded as re**W**e**I**ghting based **S**emi-supervised **D**ata pr**O**gra**M**ming

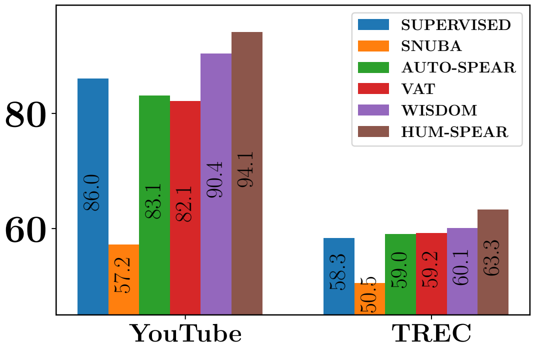


Figure 2: A summary plot contrasting the performance gains obtained using WISDOM on previous state-of-the-art approaches on YouTube and TREC (using Lemma features). WISDOM outperforms other learning approaches with auto-generated LFs.

tomatically generating LFs, we address the important limitations of SNUBA by (i) effectively using the labeled set in a semi-supervised manner using SPEAR (Maheshwari et al., 2021), and (ii) critically making the labeling function aggregation more *robust* via a reweighting framework. We do the reweighting by using our proposed bi-level optimization algorithm that weighs each LF separately, giving low importance to noisy LFs and high importance to relevant LFs. We present evaluations on six text classification datasets and show that WISDOM demonstrates better performance than current label aggregation approaches with automatically (or even human) generated labeling functions.

A summary of the results are presented in Figure 2. As mentioned, SNUBA performs worse than a simple supervised baseline that trained only on the labeled data component. Furthermore, WISDOM outperforms VAT (a state-of-the-art semi-supervised learning algorithm) and HUM-SPEAR sometimes (a state-of-the-art semi-supervised data programming algorithm with human-generated LFs), demonstrating the benefit of having both semi-supervision and robust LF reweighting with the auto-generated LFs. Finally, WISDOM gets to within 2 - 4% of HUM-SPEAR (using human crafted-LFs), without having to incur the cost of generating labeling functions manually, and which can also require significant domain knowledge.

2 Background

2.1 Notations

Let us denote the feature space by $\mathcal{X}$ and the label space by $\mathcal{Y} \in \{1...K\}$ where K is the number of classes. Let the automatically (or manually) generated labeling functions be denoted by λ_1 to λ_m where m is the number of labeling functions generated. Let the vector $\mathbf{l}_i = (l_{i1}, l_{i2}, \ldots, l_{im})$ denote the firings of all the LFs on an instance $\mathbf{x}_i$. Each l_{ij} can be either 1 or 0; $l_{ij} = 1$ indicates that the LF λ_j has fired (*i.e.*, triggered) on the instance x_i and 0 indicates it has not. Furthermore, each labeling function λ_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.

Notation	Description
$\mathbf{l}_i \in \{0,1\}^m$	Firings of all the LFs, $\lambda_1..\lambda_m$ on an instance $\mathbf{x}_i$
$\tau_{ij} \in [0, K]$	class k_j associated by LF λ_j, when triggered ($l_{ij} = 1$) on x_i
f_ϕ	The feature-based model with parameters ϕ operating on feature space $\mathcal{X}$ and on label space $\mathcal{Y} \in \{1...K\}$
P_θ	The label probabilities as per the LF-based aggregation model with parameters θ
labeled-set ($\mathcal{L}$)	The entire labeled dataset: $\mathcal{L} = \{(x_i, y_i)\}$ where $i \in \{1 \cdots N\}$. This is used to induce the LFs
supervised set ($\mathcal{S}$)	Subset of $\mathcal{L}$ that is used for semi-supervision: $\mathcal{S} = \{(x_i, y_i)\}$ where $i \in \{1 \cdots N/2\}$
validation set ($\mathcal{V}$)	Subset of $\mathcal{L}$ that is used for reweighting the LFs using a bi-level optimization formulation: $\mathcal{V} = \{(x_i, y_i)\}$ where $i \in \{N/2 + 1 \cdots N\}$
*u*nlabeled-set ($\mathcal{U}$)	Unlabeled set: $\mathcal{U} = \{x_i\}$ where $i \in \{N + 1 \cdots M\}$. It is labeled using the induced LFs
$\mathcal{L}_{ce}$	Cross Entropy Loss
H	Entropy function
g	Label Prediction from the LF-based graphical model
LL_s	Supervised negative log likelihood over the parameters θ of the LF aggregation model
LL_u	Unsupervised negative log likelihood summed over labels
KL	KL Divergence between two probability models
R	Quality Guide based loss
$\mathcal{L}_{ss}(\theta, \phi, \mathbf{w})$	The semi-supervised bi-level optimization objective with additional weight parameters $\mathbf{w}$ over the LFs

Table 2: Summary of notations used in this paper.

Let the *labeled-set* be denoted by $\mathcal{L} = \{(x_i, y_i)\}$ where $i \in \{1 \cdots N\}$ and N is the number of points in *labeled-set*. Similarly, we have an unlabeled dataset denoted as $\mathcal{U} = \{x_i\}$ where $i \in \{N + 1 \cdots M\}$ 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 $\mathcal{S} = \{(x_i, y_i)\}$ where $i \in \{1 \cdots N/2\}$. Let $\mathcal{V} = \{(x_i, y_i)\}$ denote the *validation set*, where $i \in \{N/2 + 1 \cdots N\}$.

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 $\mathcal{U}$. 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 WISDOM framework utilizes SNUBA 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 SPEAR: Joint SSL Data Programming

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

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

There are K parameters $\theta_{j1}, \theta_{j2} ... \theta_{jK}$ for each LF λ_j, where K is the number of classes. The potential ψ_θ used in the CAGE model is defined as:

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

The loss function of 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:

$$\sum_{i \in \mathcal{L}} \mathcal{L}_{ce}(f_\phi(x_i), y_i) + \sum_{i \in \mathcal{U}} H(f_\phi(x_i)) +$$
$$\sum_{i \in \mathcal{U}} \mathcal{L}_{ce}(f_\phi(x_i), g(l_i)) + LL_s(\theta|\mathcal{L}) + LL_u(\theta|U) +$$
$$\sum_{i \in \mathcal{U} \cup \mathcal{L}} KL(P_\theta(l_i), f_\phi(x_i)) + R(\theta|\{q_j\}) \tag{3}$$

where g is the label prediction from the LF-based graphical model. The second component $H()$ 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 θ while the feature model parameters are ϕ. The learning problem in SPEAR is simply to optimize the objective jointly over θ and ϕ. (We refer readers to Maheshwari et al. (2021) for details.)

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

$$LL_u(\theta|U) + R(\theta|\{q_j\}) \tag{4}$$

3 The 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 WISDOM workflow below (see also Figure 1). A detailed pseudocode of WISDOM is provided in Algorithm 1. We describe the different components of WISDOM below.

Automatic LF Generation using SNUBA: Our WISDOM framework utilizes steps (i) and (ii) from SNUBA (*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 SNUBALFGEN.

Re-Weighting CAGE: To deal with noisy labels effectively, we associate each LF λ_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 θ 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:

$$\psi_\theta(\tau_{ij}, y) = \begin{cases} \exp(w_j \theta_{jy}) & \text{if } \tau_{ij} \neq 0 \\ 1 & \text{otherwise} \end{cases} \quad (5)$$

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 $\mathcal{L}_{SS}(\theta, \phi, \mathbf{w})$ is invoked. We compare performance of CAGE with a bi-level variation in Table 5.

Algorithm 1: WISDOM

```
   Input: $\mathcal{L}, \mathcal{S}, \mathcal{V}, \mathcal{U}$, Learning rates: $\alpha, \beta$
   Output: $\theta, \phi, \mathbf{w}$
1  **** Automatic LF generation using SNUBA ****
2  $\lambda_1, \cdots, \lambda_m$ = SNUBALFGEN($\mathcal{L}$)
3  Get LFs trigger matrix $\mathbf{l}^s, \mathbf{l}^u$ for sets $\mathcal{S}, \mathcal{U}$ using
     $\lambda_1, \cdots, \lambda_m$
4  Get LFs output label matrix $\tau^s, \tau^u$ for sets $\mathcal{S}, \mathcal{U}$ using
     $\lambda_1, \cdots, \lambda_m$
5  **** The Reweighted Joint SSL ****
6  $t = 0$;
7  Randomly initialize model parameters $\theta_0$, $\phi_0$ and LF
     weights $\mathbf{w}_0$;
8  repeat
9      Sample mini-batch $s = (x_i^s, y_i^s, \tau_i^s, \mathbf{l}_i^s)$,
         $u = (x_i^u, \tau_i^u, \mathbf{l}_i^u)$ of batch size $B$ from
         $\{\mathcal{S}, \tau^s, \mathbf{l}^s\}, \{\mathcal{U}, \tau^u, \mathbf{l}^u\}$
10     **** Bi-level Optimization ****
11         **** Inner level ****
12         $\theta_t^* = \theta_t - \alpha \nabla_\theta \mathcal{L}_{ss}(\theta_t, \phi_t, \mathbf{w}_t)$
13         $\phi_t^* = \phi_t - \alpha \nabla_\phi \mathcal{L}_{ss}(\theta_t, \phi_t, \mathbf{w}_t)$
14         **** Outer level ****
15         $\mathbf{w}_{t+1} = \mathbf{w}_t - \beta \nabla_{\mathbf{w}} \frac{1}{|\mathcal{V}|} \sum_{i \in \mathcal{V}} \mathcal{L}_{ce}(f_{\phi_t^*}(x_i), y_i)$
16     **** Update net parameters $\phi, \theta$ ****
17         $\theta_{t+1} = \theta_{t+1} - \alpha \nabla_\theta \mathcal{L}_{ss}(\theta_t, \phi_t, \mathbf{w}_{t+1})$
18         $\phi_{t+1} = \phi_{t+1} - \alpha \nabla_\phi \mathcal{L}_{ss}(\theta_t, \phi_t, \mathbf{w}_{t+1})$
19         $t = t + 1$
20 until convergence
21 return $\theta_{t+1}, \phi_{t+1}, \mathbf{w}_{t+1}$
```

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

$$\begin{aligned} \mathcal{L}_{ss}(\theta, \phi, \mathbf{w}) &= \sum_{i \in \mathcal{S}} \mathcal{L}_{ce}(f_\phi(x_i), y_i) + \sum_{i \in \mathcal{U}} H(f_\phi(x_i)) \\ &+ \sum_{i \in \mathcal{U}} \mathcal{L}_{ce}(f_\phi(x_i), g(l_i, \mathbf{w})) + LL_s(\theta, \mathbf{w}|\mathcal{S}) \\ &+ LL_u(\theta, \mathbf{w}|\mathcal{U}) + \sum_{i \in \mathcal{U} \cup \mathcal{S}} KL(P_{\theta, \mathbf{w}}(l_i), f_\phi(x_i)) \\ &+ R(\theta, \mathbf{w}|\{q_j\}) \end{aligned} \quad (6)$$

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

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.

$$\begin{aligned} \mathbf{w}^* &= \underset{\mathbf{w}}{\text{argmin}} \frac{1}{|\mathcal{V}|} \sum_{i \in \mathcal{V}} \mathcal{L}_{ce}(f_{\phi^*}(x_i), y_i) \\ \text{where } \boldsymbol{\phi}^*, \boldsymbol{\theta}^* &= \underset{\boldsymbol{\phi}, \boldsymbol{\theta}}{\text{argmin}} \, \mathcal{L}_{ss}(\theta, \phi, \mathbf{w}) \end{aligned} \quad (7)$$

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 θ, ϕ are updated using the current choice of weight parameters $\mathbf{w}$ for one gradient step, and in line 15, the weight parameter $\mathbf{w}$ is updated using the one-step updates from lines 12 and 13. Finally, the net parameters ϕ, θ 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.

Dataset	$\lvert\mathcal{S}\rvert$	$\lvert\mathcal{V}\rvert$	$\lvert\mathcal{U}\rvert$	#LFs	#Class
IMDB	71	71	1278	18	2
YouTube	55	55	977	11	2
SMS	463	463	8335	21	2
TREC	273	273	4918	13	6
Twitter	707	707	12019	25	3
SST-5	568	568	9651	25	5

Table 3: Summary statistics of the datasets and the automatically generated LFs using SNUBA. The test set contains 500 instances for each dataset.

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 Twitter feed 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 that (just like L2R, PR, IL, and VAT) uses auto-generated LFs (which we call AUTO-SPEAR), and the second, *viz.*, HUM-SPEAR, 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_ϕ. 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 WISDOM for 100 epochs with early stopping based on *validation set*. Our model is optimized using mini-

[2] `www.imdb.com/datasets`

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 reweighs the LFs in both the label aggregator as well as in the semi-supervised loss, as against Auto-Spear which does not reweigh 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 *h*uman-lfs and *a*uto-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$_{\text{val}}$[4] that considers validation set feedback in the loss formulation for promoting LF agreement with ground-truth label and (ii) Cage$_{\text{Bi-level}}$ with the proposed bi-level formulation that tries to do the same[5]. We present our results in Table 5. The performance of our Cage$_{\text{Bi-level}}$ is clearly superior to the original Cage model, as well as to the Cage$_{\text{val}}$ 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 as on Cage. In Table 1, we had presented some illustrative examples (from the TREC dataset) of

[3] https://pytorch.org/

[4] Cage$_{\text{val}}$ - equivalent to using only $LL_s(\theta|\mathcal{L}) + LL_u(\theta, \mathbf{w}|\mathcal{U}) + R(\theta, \mathbf{w}|\{q_j\})$ in Eq (3)

[5] In other words, Cage$_{\text{Bi-level}}$ is equivalent to using only $LL_u(\theta, \mathbf{w}|\mathcal{U}) + R(\theta, \mathbf{w}|\{q_j\})$ in Eq (6)

Dataset		Methods								
		Supervised	SNUBA	L2R	VAT	PR	IL	AUTO-SPEAR	WISDOM	HUM-SPEAR
IMDB	Raw	68.8 (0.2)	-5.9 (2)	-6.6 (1.6)	-12.3 (1)	+2.7 (15.6)	+2.4 (1.7)	+2.4 (1.6)	**+3.4** (0.1)	NA
	Lemma	72.4 (1.3)	-14.4 (5.7)	-3.7 (14.7)	-19.3 (0.1)	-11.7 (4.1)	-6.4 (8.2)	-2.4 (1.6)	**+3.6** (1.4)	NA
YouTube	Raw	90.8 (0.3)	-33.2 (1.8)	+0.5 (0.5)	+0.5 (0)	-4.7 (0.4)	+0.2 (0.3)	+0.8 (0.5)	**+1.4** (0.0)	+3.8 (0.2)
	Lemma	86 (0.3)	-28.7 (2.9)	-2.2 (0.7)	-3.8 (0.2)	-7.5 (0.5)	-2.6 (0.3)	-7.9 (3.7)	**+4.4** (0.2)	+6.9(0.7)
SMS	Raw	92.3 (0.5)	-16.7 (9.8)	-5.6 (0.4)	+1.1 (0.1)	+0.3 (0.1)	0 (0.3)	0.4 (0.8)	**+1.5** (0.1)	+0.1 (0.5)
	Lemma	91.4 (0.5)	-16.1 (5.3)	-5.9 (0.5)	+1.6 (0.5)	+0.6 (0.3)	+1.5 (0.3)	-1.5 (1.8)	**+2** (0.5)	0 (0.1)
TREC	Raw	58.3 (3.1)	-6.8 (4.1)	-11.8 (0.8)	**+3.7** (0.5)	-2.2 (0.6)	-0.3 (0.8)	-0.9 (0.5)	+3.4 (0.5)	+5 (0.5)
	Lemma	56.3 (0.3)	-5.8 (5.1)	-5.5 (0.6)	+3.0 (0.5)	+0.4 (0.4)	+0.8 (0.8)	+2.7 (0.1)	**+3.9** (0.5)	+4.7(0.3)
Twitter	Raw	52.61 (0.12)	-7 (4.1)	-5 (2.3)	+0.41 (3.5)	-4.49 (3.6)	-0.85 (0.6)	-4.24 (0.4)	**+1.04** (0.8)	NA
	Lemma	61.24 (0.52)	-9.28 (5.1)	-18.03 (1.5)	-10.8 (5.3)	-8.12 (2.1)	-3.79 (0.1)	+1.9 (0.1)	**+3.97** (0.7)	NA
SST-5	Raw	27.54 (0.12)	-9 (2.2)	-7.98 (0.2)	-6.12 (0.12)	-5.59 (0.2)	-2.11 (0.1)	-4.12 (0.1)	**+0.97** (0.3)	NA
	Lemma	27.52 (0.52)	-8.31 (3.1)	-8.1 (8.1)	-7.89 (1.6)	-7 (4.7)	-3.4 (0.16)	-3.13 (2.1)	**+0.79** (0.3)	NA

Table 4: Performance of different approaches on six datasets, *viz.*, IMDB, YouTube, SMS, TREC, Twitter, and SST-5. Results are shown for both 'Raw' or 'Lemmatized' features. The numbers reported are macro-F1 scores over the test set averaged over 5 runs, and for all methods after the double-line are reported as gains over the baseline (*Supervised*). L2R, PR, IL, AUTO-SPEAR, and WISDOM all use the automatically generated LFs; *Supervised* and VAT do not use LFs; and HUM-SPEAR uses the human generated LFs. 'NA' in HUM-SPEAR column is when human LFs are not available. Numbers in brackets '()' represents standard deviation of the original scores and not of the gains.

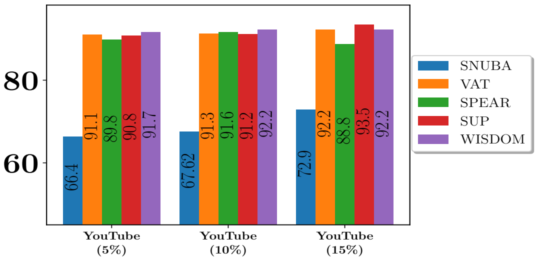


Figure 3: Ablation study with different *labeled-set* sizes on the YouTube dataset.

	Youtube	SMS	TREC
CAGE	62.45	18.1	14.1
CAGE$_{\text{val}}$	84.62	39.61	37.99
CAGE$_{\text{Bi-level}}$	87.11	43.22	39.34

Table 5: Comparison of CAGE model with two variants. CAGE$_{\text{val}}$ includes validation set feedback in the original CAGE loss function and CAGE$_{\text{Bi-level}}$ is bi-level formulation of CAGE objective using Eq 5.

automatically induced LFs whose weights are relatively higher based on the bi-level formulation along with those that are down-weighted owing to their conflicting signals. We present additional examples as well as further qualitative analysis in Section 9 of the supplementary.

5 Related Work

In this section, we describe some additional 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, 2017), 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 optimization 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 Networks - IIT Bombay 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 María 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 A Raffel. 2019. Mixmatch: A holistic approach to semi-supervised learning. *Advances in Neural Information Processing Systems*, 32.

Zalán Borsos, Mojmir Mutny, and Andreas Krause. 2020. Coresets via bilevel optimization for continual learning and streaming. *Advances in Neural Information Processing Systems*, 33:14879–14890.

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. In *International conference on machine learning*, pages 1126–1135. PMLR.

Luca Franceschi, Paolo Frasconi, Saverio Salzo, Riccardo Grazzi, and Massimiliano Pontil. 2018. Bilevel programming for hyperparameter optimization and meta-learning. In *International Conference on Machine Learning*, pages 1568–1577. PMLR.

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, Xuezhe Ma, Zhengzhong Liu, Eduard Hovy, and Eric Xing. 2016. Harnessing deep neural networks with logic rules. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2410–2420.

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, Nicholas Carlini, Zizhao Zhang, Han Zhang, Colin A Raffel, Ekin Dogus Cubuk, Alexey Kurakin, and Chun-Liang Li. 2020. Fixmatch: Simplifying semi-supervised learning with consistency and confidence. *Advances in Neural Information Processing Systems*, 33:596–608.

Antti Tarvainen and Harri Valpola. 2017. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. *Advances in neural information processing systems*, 30.

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, Thang Luong, and Quoc Le. 2020. Unsupervised data augmentation for consistency training. *Advances in Neural Information Processing Systems*, 33:6256–6268.

Appendix

7 Explanation of loss terms

First Component (L1): The first component (L1) of the loss $L_{CE}\left(P_\phi^f(y|\mathbf{x}_i), y_i\right) = -\log\left(P_\phi^f(y = y_i|\mathbf{x}_i)\right)$ is the standard cross-entropy loss on the labelled dataset $\mathcal{L}$ for the model P_ϕ^f.
Second Component (L2): The second component L2 is the semi-supervised loss on the unlabelled data $\mathcal{U}$. In our framework, we can use any unsupervised loss function.
Third Component (L3): The third component $L_{CE}\left(P_\phi^f(y|\mathbf{x}_i), g(\mathbf{l}_i), w\right)$ is the cross entropy of the classification model using the hypothesised labels from CAGE (Chatterjee et al., 2020) on $\mathcal{U}$. Given that $\mathbf{l}_i$ is the output vector of all labelling functions for any $\mathbf{x}_i \in \mathcal{U}$, we specify the predicted label for $\mathbf{x}_i$ using the LF-based graphical model $P_\theta(\mathbf{l}_i, y)$ as: $g(\mathbf{l}_i) = \underset{y}{\text{argmax}} P_{\theta,w}(\mathbf{l}_i, y)$
Fourth Component (L4): The fourth component $LL_s(\theta|\mathcal{L})$ is the (supervised) negative log likelihood loss on the labelled dataset $\mathcal{L}$: $LL_s(\theta, w|\mathcal{L}) = -\sum\limits_{i=1}^{N} \log P_{\theta,w}(\mathbf{l}_i, y_i)$
Fifth Component (L5): The fifth component $LL_u(\theta, w|\mathcal{U})$ is the negative log likelihood loss for the unlabelled dataset $\mathcal{U}$. Since the true label information is not available, the probabilities need to be summed over y: $LL_u(\theta, w|\mathcal{U}) = -\sum\limits_{i=N+1}^{M} \log \sum\limits_{y\in\mathcal{Y}} P_{\theta,w}(\mathbf{l}_i, y)$
Sixth Component (L6): The sixth component $KL(P_{\phi,w}^f(y|\mathbf{x}_i), P_\theta(y|\mathbf{l}_i))$ is the Kullback-Leibler (KL) divergence between the predictions of both the models, *viz.*, feature-based model f_ϕ and the LF-based graphical model P_θ summed over every example $\mathbf{x}_i \in \mathcal{U} \cup \mathcal{L}$. 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 λ_j correctly triggered. And let q_j^t be the user's belief on the fraction of examples $\mathbf{x}_i$ 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 $R(\theta, w|\{q_j^t\}) = -\Big(\sum_j q_j^t \log P_{\theta,w}(y_i = k_j|l_{ij} = 1) + (1 - q_j^t)\log(1 - P_{\theta,w}(y_i = k_j|l_{ij} = 1))\Big)$.

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

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

Table 6: Comparison of automatically generated LFs with human-curated LFs. Coverage is fraction of instances in $\mathcal{U}$ 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.

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

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.