Integrating Automatic Labelling Function Generation Methods in SPEAR

Vishal Jorwal
Dept. of Electrical Engineering
IIT Bombay
200070089@iitb.ac.in

Aziz Shameem
Dept. of Electrical Engineering
IIT Bombay
20D070020@iitb.ac.in

Kalp Vyas
Dept. of Electrical Engineering
IIT Bombay
kalp.vyas@iitb.ac.in

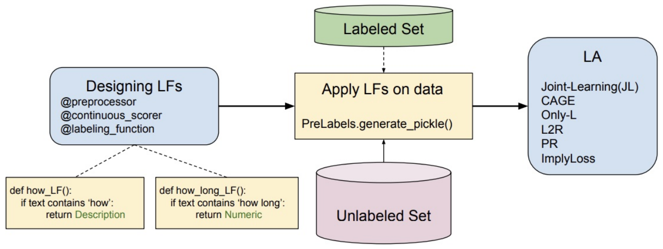
***Abstract*—This project report introduces the concept of data programming, which reduces the dependence on human-labeled data by assigning noisy labels to unlabeled data. We present the library SPEAR, an open-source Python library for data programming that provides several label aggregation approaches and allows users to define labeling functions or rules. We also describe the addition of automatic labeling function generation techniques to the SPEAR pipeline, including SNUBA, classifier weights, and WISDOM. The paper includes an example run on the IMDb and SMS spam datasets, demonstrating the effectiveness of the SPEAR library with automatic LF generation techniques. Overall, this paper provides a useful resource for researchers and practitioners looking to reduce the cost and time of creating labeled data for supervised machine learning tasks.**

Keywords — Labeling functions (LFs), optimization, SPEAR, SNUBA, weights, WISDOM

I. Introduction

Machine learning has become an increasingly popular field in recent years, with applications ranging from natural language processing to computer vision. One critical component of machine learning is the process of labelling data. In supervised learning, labelled data is used to train a model to make predictions on new, unseen data. However, the process of labelling data can be time-consuming and costly. To address this challenge, researchers have developed techniques for automating or semi-automating the labelling process. These techniques involve using labelling functions, which are functions that assign labels to data automatically.

Labelling functions play a critical role in the process of supervised learning, which is a widely used technique in machine learning for training models to make predictions on new, unseen data. In supervised learning, labelled data is used to train a model, where each data point is associated with a label that represents the ground truth. The model is trained to predict the labels of new, unseen data by learning patterns from the labelled data.



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

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

Fig. 1. Flow of the SPEAR library

II. SPEAR

[1] describes a new open-source Python library called SPEAR for data programming with semi-supervised learning. The library provides a facility to programmatically label and build training data using heuristics and association of noisy labels. SPEAR implements several label aggregation approaches that aggregate the noisy labels and then train using the noisily labeled set in a cascaded manner. The library integrates several cascade and joint data-programming approaches and allows users to define labeling functions or rules. The library is available on GitHub and has extensive documentation and video tutorials. The paper compares SPEAR with other existing packages and highlights its unique features, such as designing discrete and continuous labeling functions and integrating unsupervised and semi-supervised aggregation approaches. The package can be easily integrated with vision and speech datasets as well.

The package consists of 3 basic components: (i) Designing LFs, (ii) applying LFs, and (iii) applying a label aggregator (LA). Now, we have decided to add a few automatic LF generation techniques to the SPEAR pipeline in order to give it more useful features. We do this using 3 different methods which will be explained further in the paper.

III. Methods used

A. SNUBA

SNUBA, which is a three method for automatic LF generation. As shown in [2], firstly, SNUBA generates potential LFs (referred to as heuristics) automatically using a labeled set. Then, it filters these heuristics based on diversity and accuracy metrics, selecting only the most relevant ones. Finally, SNUBA uses the resulting set of filtered LFs and a label aggregator to calculate class probabilities for each point in the unlabeled set U. The first two steps are repeated until either the labeled set is exhausted or a predetermined number of iterations is reached. Each LF is a simple composition of propositions from the labeled set, which could be a word, a phrase, a lemma, or even a part of speech tag. The composition takes the form of a classifier, such as a decision tree or logistic regression.

B. CLASSIFIER WEIGHTS

The method involves training a linear model classifier C on the small labeled set. Suppose for N instances in our dataset, each instance x_i is denoted by its feature matrix X_i of size K. The classifier model $C(x_i) = \sigma(WX_i)$, where $W \in R^{K \times N}$ is a weight matrix, and σ represents an element-wise sigmoid function. Then, the approach find P features corresponding to the largest weights in W which is obtained by learning the classifier and creates one rule from each feature with P largest weights. If weight $w_{i,k}$ is assigned to i^{th} feature, then they create a rule associated with the i^{th} feature and the k^{th} label. Here, rule filtering is limited to choosing k rules having the largest weights in W.

C. WISDOM

WISDOM is a framework for robust aggregation of automatically generated labeling functions (LFs). The framework consists of two parts: automatic LF generation and re-weighting CAGE. The LF generation process is based on SNUBA, which involves iteratively generating candidate LFs on labeled set L and filtering them based on diversity and accuracy. The re-weighting CAGE part associates each LF with a weight parameter that acts as its reliability measure. The weight parameters are optimized on the validation set and have interactions among themselves. It uses a bi-level optimization algorithm to learn the LF weights in the outer level, and in the inner level, they learn the feature-based classifier's and labeling function aggregator's parameters jointly.

As shown in [3], the loss function considers six different types of losses. First we have a standard cross-entropy loss on the labelled dataset L for the model f. Next, we have the semi supervised loss on the unlabelled data U, which is obtained as the entropy of the predictions on the unlabelled dataset. The third loss term is the cross entropy loss of the model using the labels from CAGE, which gets the predicted label using the LF-based graphical model. The next loss term is the supervised negative log likelihood loss on the labelled set L. The fifth loss term is the negative log likelihood loss on the unlabelled set U. Next, we have the KL (Kullback-Leibler) divergence between the predictions of the feature based model and the LF based graphical model. This tries to make the models agree in their predictions over the union of the labelled and unlabelled datasets. The last term is a regulariser term to stabilise the unsupervised likelihood training while using LFs. This gives the total loss term as:

$$L_{ss}(\theta, \phi, \mathbf{w}) = \sum_{i \in S} L_{ce}(\mathbf{f}_\phi(x_i), y_i) + \sum_{i \in U} H(\mathbf{f}_\phi(x_i)) + \sum_{i \in U} L_{ce}(\mathbf{f}_\phi(x_i), g(l_i, \mathbf{w})) + LL_s(\theta, \mathbf{w}|S) + LL_u(\theta, \mathbf{w}|U) + \sum_{i \in US} KL(P_{\theta,\mathbf{w}}(l_i), \mathbf{f}_\phi(x_i)) + R(\theta, \mathbf{w}|\{q_j\})$$

Bi-Level objective: WISDOM jointly learns the LF weights and weighted labeling aggregator and feature classifier parameters for the objective function defined by the above losses. The LF weights are learned by WISDOM by posing a bi-level optimization problem for this objective function.

$$w^* = argmin_w \frac{1}{|V|} \sum_{i \in V} L_{ce}(f_{\phi^*}(x_i), y_i) \quad , where \phi^*, \theta^* = argmin_{\theta,\phi} L_{ss}(\theta, \phi, w)$$

However, determining the optimal solution to the above Bi-level objective function is computationally intractable. To solve this, WISDOM uses an iterative algorithm where we optimize the objective function at each level using gradient descent. The algorithm used is as follows:

WISDOM employs a validation set V, which is a smaller part of the labeled-set U, to learn the weights of Labeling Functions (LFs). The weight parameters introduced in WISDOM can filter LFs based on the

Algorithm 1 WISDOM

```
Require: $L, S, V, U$, Learning rates: $\alpha, \beta$
Ensure: $\theta, \phi, w$
1: **** Automatic LF generation using SNUBA ****
2: $\lambda_1, \ldots, \lambda_m = SNUBALFGEN(L)$
3: Get LFs trigger matrix $l_s$, $l_u$ for sets $S$, $U$ using $\lambda_1, \ldots, \lambda_m$
4: Get LFs output label matrix $\tau_s$, $\tau_u$ 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 $s = (x_s^{(i)}, y_s^{(i)}, \tau_s^{(i)}, l_s^{(i)})$, $u = (x_u^{(i)}, \tau_u^{(i)}, l_u^{(i)})$ of batch size $B$ from $\{S, \tau_s, l_s\}$, $\{U, \tau_u, l_u\}$
10:    **** Bi-level Optimization ****
11:    **** Inner level ****
12:    $\theta_t^* = \theta_t - \alpha \nabla_\theta L_{ss}(\theta_t, \phi_t, w_t)$
13:    $\phi_t^* = \phi_t - \alpha \nabla_\phi L_{ss}(\theta_t, \phi_t, w_t)$
14:    **** Outer level ****
15:    $w_{t+1} = w_t - \beta \nabla_w \frac{1}{|V|} \sum_{i \in V} L_{ce}(f_{\phi_t^*}(x_i), y_i)$
16:    **** Update net parameters $\phi$, $\theta$ ****
17:    $\theta_{t+1} = \theta_t - \alpha \nabla_\theta L_{ss}(\theta_t, \phi_t, w_{t+1})$
18:    $\phi_{t+1} = \phi_t - \alpha \nabla_\phi L_{ss}(\theta_t, \phi_t, w_{t+1})$
19:    $t = t + 1$
20: until convergence
21: return $\theta_{t+1}$, $\phi_{t+1}$, $w_{t+1}$
```

feature model. Moreover, a cross-entropy loss function of feature model predictions on the validation set is used as a bilevel objective. Essentially, the main objective of WISDOM is to learn the LFs' weights that lead to the minimum validation loss on the feature model that is simultaneously trained with the weighted labeling aggregator.

IV. Example Run

All the integrated code has been tested on the data available. and example ipython notebooks have been made available, depicting the use of the implemented functions.
The relevant files have been included in the submission.

V. Conclusion

We have successfully managed to integrate automatic LF generation techniques into the SPEAR library which makes it more powerful than before. Robust frameworks such as WISDOM as well as simple models such as classifier weights were added into the pipeline and they can be used with their various parameters now making the task of automatic LF generation a new feature in SPEAR.

VI. Acknowledgement

We would like to thank Prof. Ganesh RamaKrishnan for giving us this oppurtunity to work on this project and his various useful inputs which helped us in our work. We would also like to thank Ayush Maheshwari for the help provided.

References

[1] Guttu Sai Abhishek, Harshad Ingole, Parth Laturia, Vineeth Dorna, Ayush Maheshwari, Rishabh Iyer, and Ganesh Ramakrishnan. Spear : Semi-supervised data programming in python, 2022.

[2] Ayush Maheshwari, Krishnateja Killamsetty, Ganesh Ramakrishnan, Rishabh Iyer, Marina Danilevsky, and Lucian Popa. Learning to robustly aggregate labeling functions for semi-supervised data programming, 2022.

[3] Ayush Maheshwari, Oishik Chatterjee, KrishnaTeja Killamsetty, Ganesh Ramakrishnan, and Rishabh Iyer. Semi-supervised data programming with subset selection, 2021.