ULF: Cross-Validation for Weak Supervision

Anonymous ACL submission

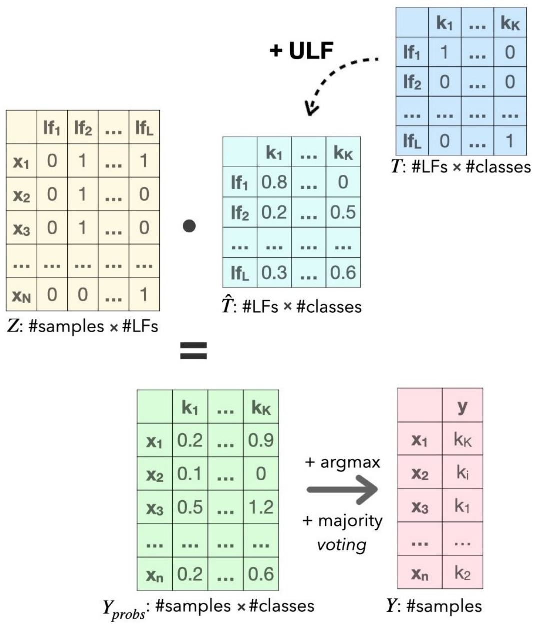
Abstract

A way to overcome expensive and time-consuming manual data labeling is *weak supervision* - automatic annotation of data samples via a predefined set of labeling functions (LFs), rule-based mechanisms that generate potentially erroneous labels. In this work, we investigate noise reduction techniques for weak supervision based on the principle of k-fold cross-validation. In particular, we extend two frameworks for detecting the erroneous samples in manually annotated data to the weakly supervised setting. Our methods profit from leveraging the information about matching LFs and detect noisy samples more accurately. We also introduce a new algorithm for denoising the weakly annotated data called *ULF*, that refines the allocation of LFs to classes by estimating the reliable LFs-to-classes joint matrix. Evaluation on several datasets shows that *ULF* successfully improves weakly supervised learning without using any manually labeled data.

1 Introduction

A large part of today's machine learning success rests upon a vast amount of annotated training data. However, a manual expert annotation turns out to be tedious and expensive work. There are different approaches to reduce this data bottleneck: fine-tuning large pre-trained models (Devlin et al., 2019), applying active (Sun and Grishman, 2012) and semi-supervised learning (Kozareva et al., 2008). However, even if in a reduced amount, these approaches still demand manually annotated data, which could be a tremendous challenge in some cases, such as tackling tasks with dynamically changing requirements or in low-resource languages.

Another strategy that does not require any manual annotation is *weak supervision* (WS), which allows getting massive amounts of training data at a low cost. In a weakly supervised setting, the data is annotated in an automized process using one or multiple weak supervision sources: for example, external knowledge bases (Lin et al., 2016; Mintz et al., 2009), manually-defined or automatically generated heuristics (Ratner et al., 2017; Varma and Ré, 2018), crowdsourcing annotations (Yang et al., 2020; Joglekar et al., 2014; Zhang et al., 2014; Hovy et al., 2013). By mapping such rules, or *labeling functions* (LFs, Ratner et al., 2017), to a large unlabeled dataset, one could quickly obtain weak training labels, which are, however, potentially error-prone and need additional denoising. Examples are provided in Figure 2.



$$\begin{array}{c|cccc} & lf_1 & lf_2 & \dots & lf_L \\ \hline x_1 & 0 & 1 & \dots & 1 \\ x_2 & 0 & 1 & \dots & 0 \\ x_3 & 0 & 1 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots \\ x_N & 0 & 0 & \dots & 1 \end{array} \quad \bullet \quad \begin{array}{c|ccc} & k_1 & \dots & k_K \\ \hline lf_1 & 0.8 & \dots & 0 \\ lf_2 & 0.2 & \dots & 0.5 \\ \dots & \dots & \dots & \dots \\ lf_L & 0.3 & \dots & 0.6 \end{array} \quad \begin{array}{c|ccc} & k_1 & \dots & k_K \\ \hline lf_1 & 1 & \dots & 0 \\ lf_2 & 0 & \dots & 0 \\ \dots & \dots & \dots & \dots \\ lf_L & 0 & \dots & 1 \end{array}$$

Z: #samples × #LFs; $\hat{T}$: #LFs × #classes; T: #LFs × #classes

$$\begin{array}{c|ccc} & k_1 & \dots & k_K \\ \hline x_1 & 0.2 & \dots & 0.9 \\ x_2 & 0.1 & \dots & 0 \\ x_3 & 0.5 & \dots & 1.2 \\ \dots & \dots & \dots & \dots \\ x_n & 0.2 & \dots & 0.6 \end{array} \quad \begin{array}{c|c} & y \\ \hline x_1 & k_K \\ x_2 & k_i \\ x_3 & k_1 \\ \dots & \dots \\ x_n & k_2 \end{array}$$

Y_{probs}: #samples × #classes; Y: #samples

Figure 1: Calculation of noisy labels Y with multiplication of a matrix with LFs matches $Z_{N \times L}$ and a matrix with LFs-to-class correspondings $\hat{T}_{L \times K}$ - an improved version of $T_{L \times K}$ matrix with ULF algorithm.

In this work, we explore methods for improving the quality of weakly supervised data using methods based on the principle of *k-fold cross-validation*. The intuition behind them is the following: a model, trained on the substantial part of the data samples and the corresponding weak labels,

	Matched labeling functions	Assigned Label
*(1) if your like drones, **plz subscribe** to Kamal Tayara. He takes videos with his drone that are absolutely beautiful.*	keyword("subscribe") keyword("plz")	SPAM
*(2) It looks so real and **my** daughter is a big fan and she likes a lot of your **songs**.*	keyword("my") keyword("song")	SPAM/ HAM
(3) Follow me on Twitter @mscalifornia95	no matches	—

Figure 2: An example of weakly supervised annotation from YouTube Spam Classification dataset (Alberto et al., 2015). In (1), both matched LFs correspond to the SPAM class; the sample will be assigned to this class as well. In (2), one of the matched LFs belongs to the SPAM class, while the other - to the HAM class. The easiest way to break the tie is to assign the sample to one of two classes randomly. In (3), no labeling functions matched, meaning the sample does not get any label and may be either filtered out or assigned to a random class.

can predict for the rest of the data more reliable and accurate labels than the weak ones. Such labels are often called *out-of-sample* since they are calculated basing on the other data samples.

The earlier approaches to data cleaning, which follow this idea, deal with general, non-weakly supervised data and, therefore, split the data samples into folds *randomly* (Northcutt et al., 2021; Wan et al., 2019). However, a direct application of these methods to the weakly labeled data ignores valuable knowledge stemming from the weak supervision process: for example, which LFs matched in each sample and what class each LF corresponds to. In this work, we propose extensions for these methods that leverage this additional source of knowledge via splitting the data considering the *labeling functions* matched in samples.

Apart from that, we propose **ULF** - a new method for **U**nsupervised **L**abeling **F**unction correction with k-fold cross-validation. Its primary goal is to improve the *LFs to classes allocation* in order to correct the systematically biased label assignments. ULF calculates the LFs confident matrix and estimates the joint distribution between LFs and output labels using the predicted class probabilities and the original weak labels. The improved allocation allows to calculate the labels anew and apply them in further training. Importantly, ULF also profits from the samples with no LFs matched, in contrast to others that filter them out (Ratner et al., 2017), and improves their labels as well.

Overall, our contributions are:

- We propose extensions for CrossWeigh (Wang et al., 2019b) and Cleanlab (Northcutt et al., 2021), originally created for denoising the data using k-fold cross-validation based methods, for WS setting. Our methods **WSCW** and **WSCL** profit from the WS-specific information and make the denoising of WS data more efficient.

- We propose our new method **ULF** for improving the LFs to classes allocation. ULF not only detects the erroneous LFs to classes allocations, but also improves it, what leads to more accurate labels and a better quality of the trained classifier.

- We demonstrate the effectiveness of both proposed extensions and ULF method compared to the original methods and other baselines on several weakly supervised datasets.

To the best of our knowledge, we are the first (1) to adapt the k-fold cross-validation based noise detection methods to WS domain, and (2) to refine the LFs to classes allocation in WS setting.

2 Related Work

Weak supervision has been widely applied to different tasks in various domains, such as text classification (Ren et al., 2020; Shu et al., 2020), relation extraction (Yuan et al., 2018; Hoffmann et al., 2011), named entity recognition (Lan et al., 2020, 2019; Wang et al., 2019b), video analysis (Fang et al., 2020; Kundu et al., 2019), medical domain (Fries et al., 2021; Saab et al., 2019), image classification (Li et al., 2021), and others. Weak labels are usually cheap and easy-to-obtain, but also potentially error-prone and often need additional denoising.

Some denoising algorithms build a specific model architecture or use the loss functions correction (Karamanolakis et al., 2021; Zheng et al., 2019; Hedderich and Klakow, 2018; Patrini et al., 2017; Goldberger and Ben-Reuven, 2017; Sukhbaatar et al., 2014; Mnih and Hinton, 2012). Others profit from expert annotations: for example, by adding a set of manually annotated data to the weakly labeled one (Mazzetto et al., 2021; Karamanolakis et al., 2021; Awasthi et al., 2020; Maheshwari et al., 2020; Teljstedt et al., 2015), or learning from user manual correction (Hedderich et al., 2021; Boecking et al., 2020; Saito and Imamura, 2009). All methods that we introduce in this paper, on the contrary, do not require any manual supervision and can be used with any classifier. Another popular research direction estimates the relation between noisy and clean labels (Lange et al., 2019; Northcutt et al., 2021), but often do not provide any direct label correction (in contrast to our method ULF).

There is also a group of approaches that share the idea of using k-fold cross-validation for denoising of manually labeled data (Wang et al., 2019a,b; Northcutt et al., 2021; Teljstedt et al., 2015). Wang et al., 2019b detect mistakes in crowdsourcing annotations by training the k models on $k-1$ data folds with one fold left for making the predictions by the trained model. The samples where the original noisy labels disagree with the predicted ones are downweighted for further training as unreliable. Northcutt et al. 2021 also make use of a cross-validation approach, but, instead of predicting labels in hold-out fold, they consider the confidence of the predicted class probabilities. The mislabeled samples are then identified using the ranking with respect to the self-confidence class-dependent thresholds and pruned. Both of these methods can be applied to any data; however, they miss a lot of potentially profitable information when being used *as-is* for denoising the weakly supervised data.

3 Denoising of Weakly Supervised Data using Cross-validation

In this section, we present our extension of two frameworks, CrossWeigh (Wang et al., 2019b), and Cleanlab (Northcutt et al., 2021), initially proposed for denoising the manually annotated data, for weakly supervised settings with leveraging the information about matching LFs.

3.1 Preliminaries

Let us introduce some formal notation used in the current and in the following sections. Given a dataset X with N data samples, $X = \{x_1, x_2, ..., x_N\}$; each sample is a set of words (i.e., one or several sentences). This set is used for training a classifier with K output classes, $K = \{k_1, k_2, ..., k_K\}$. In the weakly supervised setting, there are no known-to-be-correct *true* labels for training samples. Instead, we are provided with a set of LFs L, $L = \{l_1, l_2, ..., l_L\}$ (LFs). We say that a LF l_j *matches* a sample x_i if it maps this sample to a label. For example, in case of keyword-based LFs, some keyword is found in data sample and, thus, a corresponding label is assigned to this data sample. A set of LFs matched in a sample x_i is denoted as L_{x_i}. In each x_i there can be either one LF ($|L_{x_i}| = 1$), or several ($|L_{x_i}| > 1$) or none of them ($|L_{x_i}| = \emptyset$) matched. This information can be saved in a binary matrix $Z_{N \times L}$, where $Z_{ij} = 1$ means that lf_j matches in sample x_i. Each label function l_j corresponds to some class k_i. The information about this correspondence is stored in a binary matrix $T_{L \times K}$, where $T_{ij} = 1$ means LF l_i corresponds to class k_j. By multiplying Z and T, applying majority voting, and breaking the ties, we could obtain the potentially noisy labels $Y = \{y_1, y_2, ..., y_n\}$, $y_j \in K$, which can be used for training a classifier with parameters θ. An illustration of Z, T, and Y matrices is presented in Figure 1 (we are not considering the denoised $\hat{T}$ matrix in this section yet).

3.2 Weakly Supervised CrossWeigh (WSCW)

CrossWeigh (CW, Wang et al. 2019b) was proposed for tracing inconsistent labels in the crowdsourced annotations for the NER task. As in the straightforward k-fold cross-validation, the data is randomly split into k folds used to build k training and hold-out sets. Importantly, the training samples that include the entities annotated in hold-out samples are additionally filtered out, meaning each of the k trained models makes predictions only for the entities unseen during the training. Thus, if an entity was constantly mislabeled by crowdworkers, the model trained without it would be rid of this confusion.

This approach seems to be quite promising for detecting the unreliable LFs (which can be considered as samples' *annotators*) in the weakly supervised data as well. Indeed, if a potentially erroneous LF systematically annotates the samples wrongly, a reliable model trained on data without it will not make this mistake in its prediction, and, thus, the error will be traced and reduced. Thus, we propose a new **Weakly Supervised CrossWeigh** method (WSCW) with splitting the data considering the LFs: the LFs matched in a test fold are eliminated from the training folds. Thus, we refrain the model from making easy predictions and let her deduce the labels basing on the unseen LFs only.

More formally, we, firstly, randomly split LFs L

into k folds $\{f_1, ..., f_k\}$. Then, we iteratively take LFs from each fold f_i as test LFs and the others as the training LFs. So, all samples where no LFs from hold-out fold match become training samples, while the rest are used for testing.

$$X_{train_i} = \{x_j | L_{x_j} \cap f_i = \emptyset\}$$
$$X_{test_i} = X \setminus X_{train_i}$$

After that we train the k separate models on X_{train_i} and evaluate them on X_{test_i}. The same way as in the CrossWeigh algorithm, the labels predicted by the trained model for the samples in the hold-out set $\hat{y}$ are compared to the initial noisy labels y. All samples X_j where $\hat{y}_j \neq y_j$ are claimed to be potentially mislabeled; their influence is reduced in further training. The whole procedure of errors detection is performed t times with different partitions to refine the results. The sample weights w_{x_N} are then calculated as $w_{x_j} = \epsilon^{c_j}$, where c_j is the number of times a sample x_j was classified as mislabeled, $0 \leq c_j \leq t$, and ϵ is a weight reducing coefficient.

3.3 Weakly Supervised Cleanlab (WSCL)

The second method we introduce is **Weakly Supervised Cleanlab** (WSCL) - an adaptation of Cleanlab framework (Northcutt et al., 2021) for weak supervision. The Cleanlab framework allows to find erroneous labels by estimating the joint distribution between the noisy labels and out-of-sample labels calculated by k-fold cross-validation. In WSCL, we follow a similar approach, but adapt it to the weak supervision the same way as in WSCW: the cross-validation sets X_{train_i} and X_{test_i} sets are built considering the matched LFs. These sets are used to train k models, which predict the probability vector of class distribution $\hat{p}(\hat{y} = j; x_i, \theta), j \in K$ for each sample x_i in the X_{test_i}. The exact labels $\hat{y}$ are calculated later on with respect to the class expected self-confidence value t_j (see Northcutt et al. 2021 for more details):

$$t_j := \frac{\sum_{X_j} \hat{p}(y = j; x_i, \theta)}{|X_j|}, \tag{1}$$

where $X_j = \{x_i \in X_{y=j}\}, 1 < j < c$

A sample x_i is considered to confidently belong to class $j \in K$ if the probability of class j is greater than expected self-confidence for this class t_j or the maximal one in case of several classes are probable:

$$\hat{y}_i = \underset{\substack{j \in [K]: \\ \hat{p}(y=j; x_i, \theta) \geq t_j}}{\text{argmax}} \hat{p}(y = j; x_i, \theta) \tag{2}$$

The samples with no probability exceeding the thresholds have no decisive label and do not participate in the further denoising.

After that, a class-to-class confident joint matrix $C_{K \times K}$ is calculated, where:

$$C[j][k] = |\{x_i \in X | y_i = j, \hat{y}_i = j\}|$$

Notably, C contains only the information about correspondence between noisy and out-of-sample predicted labels (the same way as in Northcutt et al. 2021). So, it only counts the samples with presumably incorrect noisy labels y, but does not provide us with any insights about what LFs assigned these noisy labels and, thus, can be claimed erroneous (in contrast to the ULF approach, see Section 4).

The confident matrix C is then calibrated and normalized in order to obtain an estimate matrix of the joint distribution between noisy and predicted labels $\hat{Q}_{K \times K}$, which determines the number of samples to be pruned. We perform the pruning by noise rate following the Cleanlab default setting: $n \cdot \hat{Q}, i \neq j$ samples with $max(\hat{p}(y = j) - \hat{p}(y = i))$ are eliminated in further training.

4 ULF: Unsupervised Labeling Function Correction

In this section, we present a novel approach ULF - **U**nsupervised **L**abeling **F**unctions Correction algorithm. As both CrossWeigh (Wang et al., 2019b) and Cleanlab (Northcutt et al., 2021), it exploits the idea of k-fold cross-validation training. However, while those algorithms only detect the unreliable annotations, ULF correct them by refining the *LFs to classes allocation*.

4.1 Motivation

A substantial amount of noise in weakly supervised annotation is produced by LFs to class allocation being not accurate enough. For example, in Figure 2, among the LFs used to annotate the YouTube Dataset, there is a LF *"my"* that corresponds to the SPAM class. The reason for that are the often encountered spam messages like *"subscribe to **my** channel"* or *"follow **my** page"*. However, such correspondence is by no means so straightforward and sometimes may be potentially misguiding. Thus, a label for Sample 2, where this LF matched alongside another one from class HAM, cannot be defined clearly by LFs, as both classes gain 50% probability, which leads to random label assignment. However, if the information about *"my"* being related to the HAM class as well were considered, the overall HAM probability would dominate, which would make this sample being classified to the correct HAM class.

Algorithm 1 ULF: Unsupervised LFs Correction

```
Input: unsupervised training data $X$; LFs to classes matrix $T_{L\times K}$, samples to LFs matches matrix $Z_{N\times L}$
Output: a trained classifier $\theta$
1  for $t$ = 1, 2, ... $T$ do
2      Calculate noisy labels $Y_n \leftarrow Z * T$
3      Randomly split $\bigcup L_{x_i}$ into $k$ folds $f_1, ..., f_k$
4      for fold = 1, 2, ..., $k$ do
5          Build $X_{train_i}$ and $X_{test_i}$ sets through Eq. 3
6          Train model $\theta_i$ using $X_{train_i}$ and $Y_n$
7          Calculate predictions $\hat{p}(\hat{y} = j; x_i, \theta_i)$
8      Get labels $\hat{y}_i$ (Eq. 2) using thresholds $t_j$ (Eq. 1)
9      Calculate LFs-to-class confident matrix $C_{l,\hat{y}}$
10     Calculate a joint matrix $\hat{Q}_{l,\hat{y}}$ through Eq. 5
11     Recalculate $T$ matrix through Eq. 6
12 Calculate noisy labels $Y_n$ using updates $T$ matrix
13 Train model $\theta$ with $X_{train_i}$ and $Y_n$
```

4.2 Overview

Following the notation defined in Section 3.1, the main goal of the ULF algorithm is to improve the matrix of LFs to classes assignments T. It is done by, firstly, calculating the reliable probabilities of training samples belonging to each class with k-fold cross-validation and, secondly, building an LFs-to-classes confidence matrix $C_{L\times K}$. The further combining of this confidence matrix, which reflects the out-of-sample predicted LFs to classes assignment, and the original matrix T results in an adjusted $\hat{T}$ matrix, which could be used to calculate new, more accurate training labels Y. The graphical illustration is provided in Figure 1 and the algorithm is summarized in Algorithm 1.

4.3 Label Probabilities with Cross-Validation

Firstly, the class probabilities for each training sample are predicted by the k-fold cross-validation. For that, we use the unlabeled training set X and weak labels Y obtained by multiplying Z and T matrices.

There are three possible ways of data splitting:

- **randomly (ULF*rndm*)**: assigning the samples to folds same way as it would be done in standard k-fold cross-validation not considering which LFs are matching;
- **by labeling functions (ULF*lfs*)**: the same way it is done in WSCW (refer to Section 3.2 for more details)
- **by signatures (ULF*sgn*)**: for each training sample x_i we take the set of matching LFs L_{x_i} as its signature. Now the k folds $\{f_1, ..., f_k\}$ contain the signatures and not LFs as in WSCW. After that, the signatures are split into k folds, each of which becomes a test fold in turn, while others build training folds.

$$\begin{aligned} X_{train_i} &= \{x_j | L_{x_j} \notin f_i\} \\ X_{test_i} &= \{x_j | L_{x_j} \in f_i\} \end{aligned} \tag{3}$$

After training k models separately on X_{train_i}, $i \in [1, k]$, and making the predictions on the hold-out folds X_{test_i}, we obtain a matrix with out-of-sample predicted probabilities $\hat{P}_{N\times K}$. From these probabilities the reliable labels $\hat{y}$ are derived in the same way as in WSCL through Eq. 2 with reference to the expected average thresholds t_{k_j} (Eq. 1).

4.4 Re-estimate Labeling Functions

In contrast to Northcutt et al., 2021, the central idea of our approach is not to estimate the joint distribution between out-of-sample predicted labels and noisy ones (i.e., build a class-to-class confident matrix $C_{K\times K}$), but between *matched LFs and predicted labels*, what, in its turn, define the classes each LF confidently corresponds to.

For each LF l_i, we calculate the number of samples with this LF matched and confidently assigned to each class k_j. This information is saved as a *LFs-confident matrix* $C_{L\times K}$:

$$C_{l_i,\hat{y}_j} = |\{x_i \in X : \hat{y}_i = y_j, l_i \in L_{x_i}\}| \tag{4}$$

Next, the confident matrix is calibrated and normalized to $\hat{Q}_{L\times K}$ to correspond with the values in the Z matrix: the confident matrix should sum up to the overall number of training samples and the sum of counts for each LF should be the same as in the original Z matrix:

	YouTube	Spouse	TREC	SMS
Training Data	1586	22254	4965	4502
Validation Data	150	2711	500	500
Test Data	250	2701	500	500
#Classes (C)	2	2	6	2
#LFs (R)	10	9	68	73
Average LF Hits	1.62	33.7	1.73	0.509
LF Accuracy (Majority Voting)	$82\% \pm 0.8$	$44\% \pm 0.6$	$61\% \pm 0.4$	$54\% \pm 1.8$
LF Coverage	87%	25%	85%	40%

Table 1: Statistics of all the datasets. The LF accuracy metrics are reported across 5 runs with standard deviation in order to reduce the instability caused by randomly broken ties.

$$\hat{Q}_{l_i,\hat{y}_j} = \frac{C_{l_i,\hat{y}_j} \cdot \sum_{m=1}^{L} Z_{l_m,\hat{y}_j}}{\sum_{m=1}^{L} C_{l_m,\hat{y}_j}}, \tag{5}$$

where $\sum_{\substack{i \in L,\\ j \in K}} C_{l_i,\hat{y}_j} = n$, $\sum_{j=1}^{K} Z_{l_m,\hat{y}_j} = \sum_{j=1}^{K} \hat{Q}_{l_m,\hat{y}_j}$

The joint matrix $\hat{Q}_{l,\hat{y}}$ can now be used for tuning the LFs-to-class matrix T that contains the initial LFs to class allocations. T and $\hat{Q}_{l,\hat{y}}$ are summed with multiplying coefficients p and $1-p$, where $p \in [0,1]$. The value of p determines how much information from the estimated assignment matrix $\hat{Q}_{l,\hat{y}}$ should be preserved in the refined matrix $\hat{T}$.

$$\hat{T} = p * \hat{Q}_{l,\hat{y}} + (1-p) * T \tag{6}$$

With the multiplication of Z and the newly estimated $\hat{T}$ matrices, the new set of labels Y_{upd} is calculated. Now it can be used either for rerunning the estimation process or for training the final classifier. After all iterations are done, the final classifier is trained on the training set X annotated by improved labels Y. Note that, in contrast to Northcutt et al., 2021, we do not eliminate any training samples that are considered to be unreliable but use them all with corrected labels.

Unlabeled instances. One of the challenges in weak supervision is the data samples where no LF matched. In some approaches, such samples are filtered out, while in others they are kept as belonging to a random class or to the *other* class. In ULF, such samples are initially assigned with random labels and may be partly involved in cross-validation training. The proportion of randomly labeled data included in each k-training fold is defined with hyper-parameter λ: $|\{x_i | L_{x_i} \neq \emptyset\}| = \lambda \cdot |\{x_j | L_{x_j} = \emptyset\}|$. After each $\hat{T}$ matrix recalculation, their new labels are calculated directly from the out-of-samples predicted probabilities: $\hat{y}_i = \operatorname{argmax} \hat{p}(y = j; x_i, \theta)$, where $\hat{p}(y = j; x_i, \theta) \geq t_j$.

5 Experiments

Datasets. We evaluate our methods on four well-known weakly supervised English datasets (the amount of covered tasks and language limitation of datasets certainly leaves room for future work): (1) YouTube Spam Classification dataset (Alberto et al., 2015), also used in (Ratner et al., 2017); (2) SMS Spam Classification dataset (Almeida et al., 2011); (3) Question Classification dataset from TREC-6 (Awasthi et al., 2020); (4) Spouse Relation Classification dataset based on the Signal Media One-Million News Articles Dataset (Corney et al., 2016), also used in (Ratner et al., 2017). The same LFs were used in the same way as in the previous works in order to provide a fair comparison. The data statistics is provided in Table 1.

All datasets used in experiments have their own peculiarities. For example, in the Spouse dataset, there are 75% of samples not covered with any LFs, while in the rest 25% there is a high ratio of LF overlappings. Besides, though it has a moderate overlapping and non-coverage score, the TREC dataset has quite unreliable LFs, which results in 61% LFs accuracy.

Baselines. We compare our algorithm against two baselines. (1) *Majority + Training*: the classifier is trained on the data and noisy labels acquired with simple majority voting and randomly broken ties. (2) *Snorkel-DP*: a classifier is trained using both generative and discriminative Snorkel (Ratner et al., 2017) steps.

	YouTube (Acc)	**Spouse (F1)**	**TREC (Acc)**	**SMS (F1)**
Majority + Training	86.8 ± 0.5	41.2 ± 0.2	55.9 ± 0.0	80.0 ± 0.2
Snorkel-DP (Ratner et al., 2017)	88.0 ± 0.0	41.5 ± 0.7	42.2 ± 2.2	82.8 ± 0.1
CrossWeigh (Wang et al., 2019b)	90.1 ± 0.6	41.6 ± 0.5	40.0 ± 0.1	79.6 ± 1.4
WSCW	90.8 ± 0.7	42.0 ± 0.0	46.0 ± 0.4	84.0 ± 0.9
Cleanlab (Northcutt et al., 2021)	78.0 ± 0.2	$\underline{45.1 \pm 0.1}$	$\underline{58.5 \pm 0.1}$	79.5 ± 0.2
Cleanlab-PyTorch	86.9 ± 0.7	44.4 ± 1.2	55.8 ± 0.3	86.0 ± 0.6
WSCL $_{lfs}$	87.2 ± 0.4	44.4 ± 0.7	$\mathbf{58.9 \pm 0.3}$	$\underline{86.6 \pm 0.3}$
WSCL $_{sgn}$	88.3 ± 0.5	44.6 ± 0.9	56.4 ± 0.3	85.1 ± 0.3
ULF $_{rndm}$	$\underline{92.8 \pm 0.1}$	44.8 ± 0.5	58.0 ± 0.2	85.7 ± 0.5
ULF $_{lfs}$	90.8 ± 0.9	44.0 ± 0.9	55.5 ± 0.4	70.0 ± 0.4
ULF $_{sgn}$	$\mathbf{94.6 \pm 0.2}$	$\mathbf{49.8 \pm 1.0}$	58.2 ± 0.2	$\mathbf{88.6 \pm 0.3}$

Table 2: Results of cross-validation methods across multiple datasets, averaged over ten trials and reported with standard error of the mean. The best results for each dataset are marked **bold**, the second best - underlined.

In order to evaluate the performance of our weakly supervised adaptations of CrossWeigh and Cleanlab, we also introduce the original frameworks as baselines: (3) *CrossWeigh* (Wang et al., 2019b), (4a) *Cleanlab* - the original implementation of Northcutt et al. 2021 with Scikit-learn library, (4b) *Cleanlab-PyTorch* - our PyTorch-based reimplementation of the Cleanlab algorithm in order to provide a fair comparison with our methods implemented with PyTorch library (Paszke et al., 2017). Following (Northcutt et al., 2021), we use a logistic regression model in our experiments.

Implementation Details. The training data were encoded with TF-IDF vectors. For training, we set the number of epochs to 20 and applied the early stopping with patience = 5. We ran each experiment 10 times; the models which showed the best scores on the development set were evaluated on the test sets. Development set is also used to estimate the number of iterations t: initially, it is set to $t = 20$, but if training labels do not change after three iterations, the algorithm stops, and the last saved model is used for final testing. The actual number of iterations in training of the best performing ULF models (ULF$_{sng}$ and ULF$_{rndm}$), alongside other hyperparameter values found using grid-search, are provided in Appendix A.

For our implementations we used the setting of the weak supervision framework *Knodle* (Sedova et al., 2021), which by providing an access to all WS components allowed us to implement and benchmark all algorithms described above.[1] All experiments were performed on a machine with CPU frequency of 2.2GHz with 40 cores; the full set up took on average 20 hours for each dataset.

5.1 Experimental Results

In Table 2, there are the average results across all datasets reported with the standard error of the mean. Due to the skewness of the Spouse and SMS datasets, we report the F1 score for them; for other datasets, an accuracy score is provided.

Overall, our WSCW and WSCL extensions of CrossWeigh and Cleanlab frameworks outperform the corresponding base methods in most cases and lead to consistent improvements compared to the Majority and Snorkel-DP baselines. It proves our hypothesis of LFs' importance in applying cross-validation techniques for weak supervision. At the same time, the ULF algorithm shows the best result overall on most of the datasets. ULF$_{sgn}$ outperforms the classifier trained on the data with label chosen with majority voting without any additional denoising by 6.8% on average and Snorkel-DP by 9.2%, ULF$_{rndm}$ - by 4.4% and 6.7% respectively. Interestingly, the ULF$_{lfs}$ demonstrates a worse result compared to ULF$_{sign}$ and even ULF$_{rndm}$, which could be explained by multiple LF overlappings that makes using them for splitting the data in scope of ULF a complicated task. The analysis of hyperparameter values in Tables 4 and 5 showed that in most cases the signature-based data splitting

[1]The code will be publicly released on acceptance.

Sample		LFs matched	Noisy Label	Corrected Label	True Label
(1) Sample:	*sub my channel for no reason -_-*	*short_comment*	HAM	SPAM	SPAM
(2) Sample:	*:):):) Like This Comment :):):)*	*short_comment*	HAM	SPAM	SPAM
(3) Sample:	*OMG I LOVE YOU KATY PARRY YOUR & SONGS ROCK!!!!!!!!!!!!!!!!! THATS A TOTAL SUBSCRIBE*	*keyword_subscribe*	SPAM	HAM	HAM
(4) Sample:	*s...from the training manual it show there is no tech process:) its all about password reset and troubleshooting:)*	*keyword_password*	SPAM	HAM	HAM
(5) Sample:	*thanks for your message. i really appreciate your sacrifice. i'm not sure of the process of direct pay but will find out on my way back from the test tomorrow. i'm in class now. do have a wonderful day.*	*keyword_thanks* *keyword_direct*	SPAM	HAM	HAM

Figure 3: Examples of changed labels in YouTube and SMS datasets after denoising with ULF$_{lfs}$.

requires smaller number of folds k and iterations t compared to the random data splitting. In the meantime, the values of multiplying coefficient p and non-labeled data rate λ, being rather data- and not algorithm-specific parameters, remain the same. Note that our method does not involve any manually annotated data and, therefore, cannot be directly compared to the results reported for settings that include it (Karamanolakis et al., 2021; Awasthi et al., 2020).

5.2 Case Study

Table 3 demonstrates several samples from YouTube and SMS datasets and their erroneous labels that were detected and changed by ULF$_{lfs}$. While Sample 1 is definitely a SPAM comment, no SPAM LF matched in it (since the potentially spam keyword *"subscribe"* is here reduced to *"sub"*). However, a LF *short_comment* corresponding to class HAM matched there, which results in assigning a wrong label HAM to it. Sample 2 is a SPAM message not covered by any keywords-based LFs: while they are primarily concerned with the cases where a user asks to subscribe to the channel (using the key words *"subscribe"*, *"my"*), here there is a request to like the comment. Thus, a lack of LFs involves misclassification. However, in Sample 3, a word *"subscribe"* is mentioned indeed, but in a related to the video context, what makes this message not-SPAM in contrast to the assigned label HAM in effect. In the same way, keyword *"password"*, which is defined as a LF corresponding to SPAM class in SMS dataset (in order to detect spam messages like *Send me your id and password*), is matched in a HAM message and annotated it with SPAM label. A different one is Sample 5: here, two LFs from different classes match; as a result, the label SPAM was defined not by LFs but randomly. All of these various misclassifications were corrected by ULF$_{lfs}$, and the corresponding samples were reassigned to the correct classes.

6 Conclusion

In our work, we explored the k-fold cross-validation based methods for denoising the weakly annotated data. Firstly, we introduced our extensions of two frameworks, initially proposed for denoising the manually annotated data: CrossWeigh (Wang et al., 2019b) and Cleanlab (Northcutt et al., 2021). We adapted them for weakly supervised setting by leveraging the information about labeling functions matching in the data samples and proved the efficiency of our adaptations. In the second part of our work, we propose the ULF algorithm for unsupervised labeling functions denoising. Based on counting the confident labeling functions-to-classes matrix and estimating their joint distribution, ULF helps to refine the labeling functions to classes allocations without any manually annotated data, allowing for the correction (and not just removal) of the weak labels. Thus, the entire training set can then be used for training a reliable classifier without any data reduction.

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.

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*, DocEng '11, page 259–262, New York, NY, USA. Association for Computing Machinery.

Abhijeet Awasthi, Sabyasachi Ghosh, Rasna Goyal, and Sunita Sarawagi. 2020. Learning from rules generalizing labeled exemplars. *CoRR*, abs/2004.06025.

Benedikt Boecking, Willie Neiswanger, Eric Po Xing, and Artur Dubrawski. 2020. Interactive weak supervision: Learning useful heuristics for data labeling. *CoRR*, abs/2012.06046.

D. Corney, M-Dyaa Albakour, Miguel Martinez-Alvarez, and Samir Moussa. 2016. What do a million news articles look like? In *NewsIR@ECIR*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Zhiyuan Fang, Shu Kong, Zhe Wang, Charless C. Fowlkes, and Yezhou Yang. 2020. Weak supervision and referring attention for temporal-textual association learning. *CoRR*, abs/2006.11747.

Jason A. Fries, Ethan Steinberg, Saelig Khattar, Scott L. Fleming, Jose Posada, Alison Callahan, and Nigam H. Shah. 2021. Ontology-driven weak supervision for clinical entity classification in electronic health records. *Nature Communications*, 12(1):2017.

Jacob Goldberger and Ehud Ben-Reuven. 2017. Training deep neural-networks using a noise adaptation layer. In *ICLR*.

Michael A. Hedderich and Dietrich Klakow. 2018. Training a neural network in a low-resource setting on automatically annotated noisy data. In *Proceedings of the Workshop on Deep Learning Approaches for Low-Resource NLP*, pages 12–18, Melbourne. Association for Computational Linguistics.

Michael A. Hedderich, Lukas Lange, and Dietrich Klakow. 2021. ANEA: distant supervision for low-resource named entity recognition. *CoRR*, abs/2102.13129.

Raphael Hoffmann, Congle Zhang, Xiao Ling, Luke Zettlemoyer, and Daniel S. Weld. 2011. Knowledge-based weak supervision for information extraction of overlapping relations. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 541–550, Portland, Oregon, USA. Association for Computational Linguistics.

Dirk Hovy, Taylor Berg-Kirkpatrick, Ashish Vaswani, and Eduard Hovy. 2013. Learning whom to trust with MACE. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1120–1130, Atlanta, Georgia. Association for Computational Linguistics.

Manas Joglekar, Hector Garcia-Molina, and Aditya G. Parameswaran. 2014. Comprehensive and reliable crowd assessment algorithms. *CoRR*, abs/1411.3377.

Giannis Karamanolakis, Subhabrata Mukherjee, Guoqing Zheng, and Ahmed Hassan Awadallah. 2021. Self-training with weak supervision. *CoRR*, abs/2104.05514.

Zornitsa Kozareva, Ellen Riloff, and Eduard Hovy. 2008. Semantic class learning from the web with hyponym pattern linkage graphs. In *Proceedings of ACL-08: HLT*, pages 1048–1056, Columbus, Ohio. Association for Computational Linguistics.

Gourab Kundu, Prahal Arora, Ferdi Adeputra, Polina Kuznetsova, Daniel McKinnon, Michelle Cheung, Larry K. Anazia, and Geoffrey Zweig. 2019. Multi-modal content localization in videos using weak supervision.

Ouyu Lan, Xiao Huang, Bill Yuchen Lin, He Jiang, Liyuan Liu, and Xiang Ren. 2019. Learning to contextually aggregate multi-source supervision for sequence labeling. *CoRR*, abs/1910.04289.

Ouyu Lan, Xiao Huang, Bill Yuchen Lin, He Jiang, Liyuan Liu, and Xiang Ren. 2020. Learning to contextually aggregate multi-source supervision for sequence labeling. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2134–2146, Online. Association for Computational Linguistics.

Lukas Lange, Michael A. Hedderich, and Dietrich Klakow. 2019. Feature-dependent confusion matrices for low-resource NER labeling with noisy labels. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3554–3559, Hong Kong, China. Association for Computational Linguistics.

Jiahui Li, Wen Chen, Xiaodi Huang, Zhiqiang Hu, Qi Duan, Hongsheng Li, Dimitris N. Metaxas, and Shaoting Zhang. 2021. Mixed supervision learning for whole slide image classification. *CoRR*, abs/2107.00934.

Yankai Lin, Shiqi Shen, Zhiyuan Liu, Huanbo Luan, and Maosong Sun. 2016. Neural relation extraction with selective attention over instances. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2124–2133, Berlin, Germany. Association for Computational Linguistics.

Ayush Maheshwari, Oishik Chatterjee, KrishnaTeja Killamsetty, Rishabh K. Iyer, and Ganesh Ramakrishnan. 2020. Data programming using semi-supervision and subset selection. *CoRR*, abs/2008.09887.

Alessio Mazzetto, Dylan Sam, Andrew Park, Eli Upfal, and Stephen Bach. 2021. Semi-supervised aggregation of dependent weak supervision sources with performance guarantees. In *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*, volume 130 of *Proceedings of Machine Learning Research*, pages 3196–3204. PMLR.

Mike Mintz, Steven Bills, Rion Snow, and Daniel Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 1003–1011, Suntec, Singapore. Association for Computational Linguistics.

Volodymyr Mnih and Geoffrey E. Hinton. 2012. Learning to label aerial images from noisy data. In *ICML*.

Curtis G. Northcutt, Lu Jiang, and Isaac L. Chuang. 2021. Confident learning: Estimating uncertainty in dataset labels.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in pytorch.

Giorgio Patrini, Alessandro Rozza, Aditya Menon, Richard Nock, and Lizhen Qu. 2017. Making deep neural networks robust to label noise: a loss correction approach.

Alexander Ratner, Stephen H. Bach, Henry R. Ehrenberg, Jason Alan Fries, Sen Wu, and Christopher Ré. 2017. Snorkel: Rapid training data creation with weak supervision. *CoRR*, abs/1711.10160.

Wendi Ren, Yinghao Li, Hanting Su, David Kartchner, Cassie Mitchell, and Chao Zhang. 2020. Denoising multi-source weak supervision for neural text classification. *CoRR*, abs/2010.04582.

Khaled Saab, Jared Dunnmon, Roger Goldman, Alex Ratner, Hersh Sagreiya, Christopher Ré, and Daniel Rubin. 2019. Doubly weak supervision of deep learning models for head ct. In *Medical Image Computing and Computer Assisted Intervention – MICCAI 2019*, pages 811–819, Cham. Springer International Publishing.

Kuniko Saito and Kenji Imamura. 2009. Tag confidence measure for semi-automatically updating named entity recognition. In *Proceedings of the 2009 Named Entities Workshop: Shared Task on Transliteration (NEWS 2009)*, pages 168–176, Suntec, Singapore. Association for Computational Linguistics.

Anastasiia Sedova, Andreas Stephan, Marina Speranskaya, and Benjamin Roth. 2021. Knodle: Modular weakly supervised learning with pytorch. *CoRR*, abs/2104.11557.

Kai Shu, Guoqing Zheng, Yichuan Li, Subhabrata Mukherjee, Ahmed Hassan Awadallah, Scott Ruston, and Huan Liu. 2020. Leveraging multi-source weak social supervision for early detection of fake news. *CoRR*, abs/2004.01732.

Sainbayar Sukhbaatar, Joan Bruna, Manohar Paluri, Lubomir D. Bourdev, and Rob Fergus. 2014. Training convolutional networks with noisy labels. *arXiv: Computer Vision and Pattern Recognition*.

Ang Sun and Ralph Grishman. 2012. Active learning for relation type extension with local and global data views. pages 1105–1112.

Christopher Teljstedt, Magnus Rosell, and Fredrik Johansson. 2015. A semi-automatic approach for labeling large amounts of automated and non-automated social media user accounts. In *2015 Second European Network Intelligence Conference*, pages 155–159.

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

Mengting Wan, Rishabh Misra, Ndapa Nakashole, and Julian McAuley. 2019. Fine-grained spoiler detection from large-scale review corpora. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2605–2610, Florence, Italy. Association for Computational Linguistics.

Yisen Wang, Xingjun Ma, Zaiyi Chen, Yuan Luo, Jinfeng Yi, and James Bailey. 2019a. Symmetric cross entropy for robust learning with noisy labels. *CoRR*, abs/1908.06112.

Zihan Wang, Jingbo Shang, Liyuan Liu, Lihao Lu, Jiacheng Liu, and Jiawei Han. 2019b. Crossweigh: Training named entity tagger from imperfect annotations. *CoRR*, abs/1909.01441.

Jingru Yang, Ju Fan, Zhewei Wei, Guoliang Li, Tongyu Liu, and Xiaoyong Du. 2020. A game-based framework for crowdsourced data labeling. *The VLDB Journal*, 29(6):1311–1336.

Changsen Yuan, Heyan Huang, Chong Feng, Xiao Liu, and Xiaochi Wei. 2018. Distant supervision for relation extraction with linear attenuation simulation and non-iid relevance embedding. *CoRR*, abs/1812.09516.

Yuchen Zhang, Xi Chen, Dengyong Zhou, and Michael I. Jordan. 2014. Spectral methods meet em: A provably optimal algorithm for crowdsourcing.

Guoqing Zheng, Ahmed Hassan Awadallah, and Susan T. Dumais. 2019. Meta label correction for learning with weak supervision. *CoRR*, abs/1911.03809.

A Appendix

Hyperparameter	Values
Multiplying coefficient p	0.1, 0.2, 0.3, 0.4, 0.5, 0.7, 0.9
Learning rate lr	1e1, 1e2, 1e3, 1e4
Number of folds k	3, 5, 8, 10, 15, 20 (with respect to the overall number of LFs)
Number of iterations t	1, 2, 3
Non-labeled data rate λ	0, 0.5, 1, 2, 3

Table 3: Hyperparameter values tried in grid search.

	YouTube	Spouse	TREC	SMS
Multiplying coefficient p	0.5	0.2	0.3	0.1
Learning rate lr	1e-2	1e-2	1e-1	1e-1
Number of folds k	8	3	3	10
Number of iterations t	5	1	1	2
Non-labeled data rate λ	0	3	1	0.5

Table 4: ULF$_{sng}$ selected hyperparameters.

	YouTube	Spouse	TREC	SMS
Multiplying coefficient p	0.5	0.2	0.3	0.2
Learning rate lr	1e-2	1e-2	1e-1	1e-2
Number of folds k	8	5	5	5
Number of iterations	10	1	1	1
Non-labeled data rate λ	0	3	1	0.5

Table 5: ULF$_{rndm}$ selected hyperparameters.