Meta Self-Refinement for Robust Learning with Weak Supervision

Dawei Zhu\textsuperscript{1}, Xiaoyu Shen\textsuperscript{1}, Michael A. Hedderich\textsuperscript{1,2}, Dietrich Klakow\textsuperscript{1}
\textsuperscript{1}Saarland University, Saarland Informatics Campus, Germany
\textsuperscript{2}Cornell University, United States
\{dzhu,xshen,mhedderich,dietrich.klakow\}@lsv.uni-saarland.de

Abstract

Training deep neural networks (DNNs) under weak supervision has attracted increasing research attention as it can significantly reduce the annotation cost. However, labels from weak supervision can be noisy, and the high capacity of DNNs enables them to easily overfit the label noise, resulting in poor generalization. Recent methods leverage self-training to build noise-resistant models, in which a teacher trained under weak supervision is used to provide highly confident labels for teaching the students. Nevertheless, the teacher derived from such frameworks may have fitted a substantial amount of noise and therefore produce incorrect pseudo-labels with high confidence, leading to severe error propagation. In this work, we propose Meta Self-Refinement (MSR), a noise-resistant learning framework, to effectively combat label noise from weak supervision. Instead of relying on a fixed teacher trained with noisy labels, we encourage the teacher to refine its pseudo-labels. At each training step, MSR performs a meta gradient descent on the current mini-batch to maximize the student performance on a clean validation set. Extensive experimentation on eight NLP benchmarks demonstrates that MSR is robust against label noise in all settings and outperforms state-of-the-art methods by up to 11.4\% in accuracy and 9.26\% in F1 score.

1 Introduction

Fine-tuning pre-trained language models has led to great success across NLP tasks. Nonetheless, it still requires a substantial amount of manual labels to achieve satisfying performance on many tasks. In reality, obtaining large amounts of high-quality labels is costly and labor-intensive (Davis et al., 2013). For certain domains, it is even infeasible due to legal issues and lack of data or domain experts. Weak supervision is a widely-used approach for reducing such cost by leveraging labels from weak sources, e.g., heuristic rules, knowledge bases or lower-quality inexpensive crowdsourcing (Ratner et al., 2017; Zhou et al., 2020; Lison et al., 2020). It has raised increasing attention in recent years, and efforts have been made to quantify the progress on weakly supervised learning, like the WRENCH benchmark (Zhang et al., 2021).

Although weak labels are inexpensive to obtain, they are often noisy and inherit biases from weak sources. Training neural networks with weak labels is challenging because of their immense capacity, which leads them to heavily overfit to the noise distribution, resulting in inferior generalization performance (Zhang et al., 2017). Various approaches have been proposed to tackle this challenge. Earlier research focused primarily on simulated noise (Bekker and Goldberger, 2016; Hendrycks et al., 2018), required prior knowledge (Ren et al., 2020; Awasthi et al., 2020) or relied on context-free aggregation rules without leveraging modern pre-trained language models (Ratner et al., 2017; Fu et al., 2020).

Recently, Yu et al. (2021) proposed a contrastive regularized self-training framework that achieved state-of-the-art (SOTA) performance in several NLP tasks from the WRENCH benchmark. It trains a teacher network on weak labels, then iteratively applies the teacher to produce pseudo-labels for training a new student model. To prevent error propagation, it filters the pseudo-labels with the model confidence scores and adds contrastive feature regularization to enforce more distinguishable representations. However, we find that this approach is effective on easy tasks but fragile on challenging ones, where the initial teacher model already have memorized a substantial amount of biases with high confidence. Consequently, confidence-based filtering is misleading and all future students will be reinforced with these initial wrong biases from the teacher.

To address this weakness, one strategy is learning to reweight the pseudo-labels with meta learning (Ren et al., 2018; Shu et al., 2019; Wang et al., 2019).
By this means, sample weights are dynamically adjusted to minimize the validation loss instead of prefixed with potentially misleading confidence scores. Nevertheless, if the initial teacher is weak and mostly produces incorrect pseudo-labels, simply reweighting the labels does not suffice to extract enough useful training signals.

In this paper, we propose Meta Self-Refinement (MSR) to go one step further. The teacher is jointly trained with a meta objective such that the student, after one gradient step, can achieve better performance on the validation set. In each training step, a copy of the current student performs one step of gradient descent based on the teacher predictions. The teacher will then update itself towards the gradient direction that minimizes the validation loss of the student. Finally, the actual student is trained by the updated teacher. In MSR, teacher’s predictions are iteratively refined, instead of only "rewighted", based on the meta objective. This will enable more efficient data utilization since the teacher still has the opportunity to refine itself to provide the proper training signal, even if its initial output label is wrong. To further stabilize the training, we enhance our framework with confidence filtering when teaching the student and apply a linearly scaled learning rate scheduler to the teacher.

In summary, the main contributions are as follows: 1) We propose a meta-learning based self-refinement framework, MSR, that allows robust learning with label noise induced by weak supervision. 2) We analyze and quantify how label noise impacts model predictions and representation learning. We find existing methods become less effective in challenging cases when the label noise can be easily fitted. In contrast, MSR is more stable and learns better representation. 3) Extensive experiments demonstrate that MSR consistently reduces the negative impact of the label noise, matching or outperforming SOTAs on six sequence classification and two sequence labeling tasks.\(^1\)

## 2 Related Work

### Learning with Noisy Labels

Learning in the presence of label noise is a long-standing problem (Angluin and Laird, 1988). Zhang et al. (2017) show that deep neural networks can memorize arbitrary noise during training, resulting in poor generalization. Noise-handling techniques - by modeling (Goldberger and Ben-Reuven, 2017; Patrini et al., 2017; Hendrycks et al., 2018) or filtering (Han et al., 2018; Li et al., 2020) the noisy instances - are proposed to conquer the label noise. While being effective, they typically assume that the noise is feature-independent which may oversimplify the noise generation process in realistic settings (Gu et al., 2021; Zhu et al., 2022). Recently, realistic and feature-dependent noise induced by weak supervision has received significant attention. To handle this type of noise, Awasthi et al. (2020) propose an implication loss that jointly denoises the noisy labels and weak sources. Ren et al. (2020) denoise the weak label by considering the reliability of different weak sources and aggregating them into one cleaned label. Zhang et al. (2021) release a benchmark, WRENCH, including various weakly supervised datasets in both text and image domains.

#### Self-Training

Self-training (Yarowsky, 1995; Lee et al., 2013) is a simple yet effective framework that is commonly used in semi-supervised learning (SSL). It typically trains a teacher model to provide pseudo-labels for the student model. Different methods have been proposed for better generalization (Xie et al., 2020; Zoph et al., 2020; Mukherjee and Hassan Awadallah, 2020). Recently, self-training has been adopted to tackle weak supervision. Karamanolakis et al. (2021) train a teacher network that aggregates weak labels to form high-quality pseudo-labels for the student. Liang et al. (2020); Yu et al. (2021) initialize the teacher model by training a classifier directly on the weak labels, they apply early stopping to prevent this initial teacher from memorizing the label noise. The student is then trained on the highly confident pseudo-labels provided by the teacher. While the core assumption of self-training - that highly confident pseudo-labels are reliable - is generally valid in SSL, it may not be true for feature-dependent noise induced by weak supervision, especially when the noise is easy to learn. In this case, self-training inevitably suffers more from error propagation and fails to train robust models.

#### Meta-Learning

Recently, different works leveraged meta-learning techniques to develop noise-robust learning frameworks. The idea is to optimize an outer learner (e.g., sample weights) that guides the inner learner (the classifier) to generalize well. Often, a clean validation dataset is used as a proxy

---

\(^1\)Code is available on: https://github.com/uds-lsv/msr
for estimating the generalization performance. Ren et al. (2018) attempt to down weight training samples that increase the validation loss. Shu et al. (2019) employ a neural network to infer such samples. Multiple weak sources may be triggered simultaneously if a specific textual pattern is matched, after which a pre-defined label is then assigned. Otherwise, it abstains. Depending on how many weak sources are triggered, a text may obtain zero, one, or multiple weak labels.

4 Meta Self-Refinement

We propose a novel meta-learning based framework, named Meta Self-Refinement (MSR), to tackle the label noise induced by weak supervision. In contrast to conventional self-training methods, where the teacher model is fixed after being trained on weakly labeled data, MSR enables the teacher to refine itself based on student performance on the clean validation set, yielding higher-quality labels and more accurate confidence estimates. In this section, we first provide an overview of its training objective (section 4.1), then go into the training details (section 4.2). Figure 2 illustrates the full training process.

4.1 Training Objective

MSR contains a teacher network $f$ and a student network $g$, both are functions that map $X \rightarrow Y$. $f$ is initialized by fine-tuning a pre-trained language model (PLM) on the weakly labeled data $D_w$:

$$f_1 = \arg\min_f \mathbb{E}_{(x_i, y_i) \in D_w} \mathcal{L}(\hat{y}_i, f(x_i))$$

where $\mathcal{L}$ denotes the loss function. We use the cross entropy loss throughout the paper:

$$\mathcal{L}(p, q) = -\mathbb{E}_{y \sim p(y)} \log q(y)$$

$p$ and $q$ are distributions over the label space $Y$. The initial student network, $g_1$, is the PLM without fine-tuning on any data.

---

**Figure 1:** Sentiment analysis dataset annotated with rule-based weak sources. A weak source is triggered if a specific textual pattern is matched, after which a pre-defined label is then assigned. Otherwise, it abstains. Depending on how many weak sources are triggered, a text may obtain zero, one, or multiple weak labels.
Figure 2: Illustration of our proposed Meta-Self Refinement method (MSR). (a) We start by fine-tuning a PLM on weak labels with early stopping, which yields an initial teacher \( f_1 \). (b) At each training step \( t \), \( f_1 \) gets training signals by performing a “teaching experiment” on \( \tilde{g}_t \): a copy of the student network \( \tilde{g}_t \) is updated by fitting \( f_1 \) with the loss function \( \mathcal{L}_s \). \( f_1 \) is then updated to minimize the validation loss \( \mathcal{L}_v \) of \( \tilde{g}_{t+1} \). (c): \( g_t \) is updated by fitting \( f_{t+1} \) with confidence filtering under the loss \( \mathcal{L}_v \).

Algorithm 1: MSR Algorithm

Input: Initial teacher network \( f_1 \) trained according to Eq. 1. Student network \( g_t \), number of training steps \( T \), teacher’s learning rate scheduler \( R(t) \), confidence threshold \( \tau \), \( D_a \), \( D_v \).

Result: \( f_T \), \( g_T \)

1. for \( t \leftarrow 1 \ldots T \) do
2. \( \{x_i\} \leftarrow \text{SampleMiniBatch}(D_a) \)
3. \( \{x_i', y_i'\} \leftarrow \text{SampleMiniBatch}(D_v) \)
4. \( \tilde{g}_t \leftarrow \text{Copy}(g_t) \)
5. \( \tilde{g}_{t+1} \leftarrow \tilde{g}_t - \lambda_t \mathbb{E}_{x_i} \nabla_{\tilde{g}_t} \mathcal{L}(f_1(x_i), \tilde{g}_t(x_i)) \)
6. \( f_{t+1} \leftarrow f_1 - R(t) \mathbb{E}_{x_i, y_i'} (\nabla f_1(x_i', y_i') \mathcal{L}(g_t', \tilde{g}_{t+1}(x_i') \)) \)
7. \( w(f_{t+1}(x_i)) \leftarrow 1 - \frac{H(f_{t+1}(x_i))}{\log(t)} \geq \tau \)
8. \( g_{t+1} \leftarrow g_t - \lambda_v \mathbb{E}_{x_i} \nabla_{g_t} \mathcal{L}(f_{t+1}(x_i), g_t(x_i)) \)
9. end

to merge Equation 3 and 4 into an iterative training pipeline. At each training step \( t \), the teacher \( f_t \) is first updated based on the meta-objective of “learning to teach”, the student \( g_t \) is then trained by the updated teacher.

Teacher Update. To update the teacher in an efficient way, we approximate the inner loop in Equation 3 with a single-step gradient descent of the student network. Namely, the objective of the teacher is changed so that the current student, after one single gradient descent step of fitting the teacher, can perform best on the validation set. To do so, the teacher will first conduct a “teaching experiment” on a copy of the current student, denoted as \( \tilde{g}_t \). \( \tilde{g}_t \) is updated for one gradient descent step to

\[
\begin{align*}
    f^* &= \arg \min_{f} \mathbb{E}_{(x_i', y_i') \in D_a} \mathcal{L}(y_i', g_f'(x_i')) \\
    g_f' &= \arg \min_{g} \mathbb{E}_{x_i \in D_v} \mathcal{L}(f(x_i), g(x_i))
\end{align*}
\]

4.2 Training Details
Finding the exact \( f^* \) in Equation 3 involves solving two nested loops of optimization, and each loop can be computationally expensive given the large size of \( D_a \). We resort to an online approximation

\[
\begin{align*}
    g^* &= \arg \min_{g} \mathbb{E}_{x_i \in D_a} \mathcal{L}(f^*(x_i), g(x_i))
\end{align*}
\]

In conventional self-training, \( f_1 \) is used to provide pseudo-labels to train the student. By selecting higher-quality pseudo-labels via confidence filtering (Yu et al., 2021) or uncertainty estimation (Mukherjee and Hassan Awadallah, 2020), the student can often outperform its teacher. However, as the teacher is trained solely on the weak labels, it can easily inherit unexpected biases and provide misleading signals to the student. In MSR, instead of using a fixed teacher to provide pseudo-labels, we use student performance on the clean validation set as a feedback signal to dynamically refine the teacher. Specifically, the objective for the teacher \( f \), formulated as in Equation 3, is that the student network, after fitting the teacher’s output labels on \( D_a \), can perform best on the validation set \( D_v \):

\[
\begin{align*}
    f^* &= \arg \min_{f} \mathbb{E}_{(x_i', y_i') \in D_a} \mathcal{L}(y_i', f'(x_i')) \\
    g_f' &= \arg \min_{g} \mathbb{E}_{x_i \in D_v} \mathcal{L}(f(x_i), g(x_i))
\end{align*}
\]

where \( g' \) is the student network after fitting output labels from \( f \) on \( D_a \). Intuitively, MSR aims to find the best teacher to help the student achieve the lowest validation loss. After finding the optimal teacher \( f^* \) in Equation 3, the student can then be obtained by learning from the output labels of \( f^* \):

\[
\begin{align*}
    g^* &= \arg \min_{g} \mathbb{E}_{x_i \in D_a} \mathcal{L}(f^*(x_i), g(x_i))
\end{align*}
\]
fit the teacher's pseudo labels\textsuperscript{3}:
\[ \tilde{g}_{t+1} = \tilde{g}_t - \lambda_s \mathbb{E}_{x_i \sim D_a} \nabla_{\tilde{g}_t} \mathcal{L}(f_t(x_i), \tilde{g}_t(x_i)) \]
where \( \lambda_s \) is the learning rate of the student network. Afterwards, we update the teacher network to minimize the validation loss of \( \tilde{g}_{t+1} \):
\[ f_{t+1} = f_t - \lambda_t \mathbb{E}_{(x'_i, y'_i) \sim D_a} \nabla_{f_t} \mathcal{L}(y'_i, \tilde{g}_{t+1}(x'_i)) \]
where \( \lambda_t \) is the learning rate of the teacher network. It requires calculating second derivatives over \( f_t \). We always use soft labels from the teacher for \( \mathcal{L}(f_t(x_i), \tilde{g}_t(x_i)) \), so the whole process is fully differentiable. Note that \( \tilde{g}_t \) is only used in the "teaching experiment" to help update the teacher. It will be discarded after the teacher is updated.

\textbf{Student Update.} After obtaining \( f_{t+1} \), the real student network is updated with the same objective as in Equation 4, except that we use the updated teacher \( f_{t+1} \) instead of \( f^* \). As the teacher has performed the "teaching experiment", it will provide more useful signals to guide the student.\textsuperscript{4}

\textbf{Teacher Learning Rate Scheduler.} We find the teacher is rather sensitive to its learning rate in practice. If the learning rate is large from the start, the teacher may over-adjust itself due to the large performance gap between the teacher and the student. If the learning rate is small, the teacher will adjust itself too slowly so that more noisy pseudo-labels are passed to the student network. Therefore, we apply a linear learning rate scheduler \( R(t) = \frac{\lambda_t}{T} \) to the teacher network where \( t \) denotes the current iteration and \( \lambda_t \) is the targeted learning rate for the teacher. By this means, the teacher’s learning rate will gradually increase as it gets better at teaching.

\textbf{Confidence-Based Label Filtering.} Despite having the opportunity to refine itself, the teacher inevitably produces some wrong pseudo labels during training, especially at early iterations of self-refinement. To further reduce error propagation, we only select labels with high confidence to guide the student model. The student is updated as follows:
\[ g_{t+1} = g_t - \lambda_s \mathbb{E}_{x_i \sim D_a} \nabla_{g_t} \mathcal{L}(f_{t+1}(x_i), g_t(x_i)) \times \mathbb{1}(1 - \frac{H(f_{t+1}(x_i))}{\log(k)} \geq \tau) \]
where \( \mathbb{1} \) is the indicator function, \( H(f_{t+1}(x_i)) \) is the entropy of the distribution \( f_{t+1}(x_i) \), \( k \) is the number of classes in \( \mathcal{Y} \) and \( \tau \) is a pre-defined confidence threshold. \( \log(k) \) is the upper bound of the entropy for \( k \)-classification tasks. By this means, only low-entropy (high-confidence) predictions from the teacher are learned. Note that the filtering strategy is only applied to the actual student update step, not during the teaching experiment. Otherwise, the teacher will ignore low-confident samples as they do not contribute to teacher update.

Putting all together, Algorithm 1 summarizes the self-refinement process.

5 Experimental Settings

\textbf{Datasets.} WRENCH (Zhang et al., 2021) is a well-established benchmark for weak supervision and offers weak labels for various datasets. We compare different baselines on six NLP datasets from WRENCH including both sequence classification and Named-Entity Recognition (NER) tasks. For sequence classification, we include AGNews (Zhang et al., 2015), IMDB (Maas et al., 2011), Yelp (Zhang et al., 2015), and TREC (Li and Roth, 2002). For NER tasks, CoNLL-03 (Sang and De Meulder, 2003) and OntoNotes 5.0 (Pradhan et al., 2013) are used. In addition, we further include two sequence classification datasets in low-resource languages, Yorùbá and Hausa (Hedderich et al., 2020), to involve evaluation cases in diverse languages. Table 1 summarizes the basic statistics of the datasets. Majority voting over weak sources is used to determine a single label for each sample.

\begin{table}[h]
\begin{tabular}{|l|l|c|c|c|c|}
\hline
Dataset & Task & \# Class & \# Train & \# Val & \# Test \\
\hline
AGNews & Topic & 4 & 96,000 & 12,000 & 12,000 \\
IMDB & Sentiment & 2 & 20,000 & 2,500 & 2,500 \\
Yelp & Sentiment & 2 & 30,400 & 3,800 & 3,800 \\
TREC & Question & 6 & 4,965 & 500 & 500 \\
Yorùbá & Topic & 7 & 1,340 & 189 & 379 \\
Hausa & Topic & 5 & 2,045 & 290 & 582 \\
CoNLL03 & NER & 4 & 14,041 & 3,250 & 3,453 \\
OntoNotes5.0 & NER & 18 & 115,812 & 5,000 & 22,897 \\
\hline
\end{tabular}
\end{table}

Table 1: Dataset statistics. Refer to Appendix A for more details on datasets.

\textbf{Implementation.} RoBERTa-base (Liu et al., 2019) is used as the PLM for English datasets and

\textsuperscript{3}We use SGD for illustration purposes. The AdamW (Loshchilov and Hutter, 2019) optimizer is used in our experiments.

\textsuperscript{4}In theory, if the teacher network is strong enough to generalize among different batches, we can directly update the real student in the "teaching experiment", in the hope that the teacher from the last step can also work in the current batch. However, in practice, we find this mismatch leads to poor performance.
multilingual BERT-base (Devlin et al., 2019) for non-English ones. We utilize the higher library to perform second-order optimization. Refer to Appendix B for detailed hyper-parameter configurations.

**Baselines.** We compare our method with prior work on learning with noisy labels. 1) **Majority** applies majority vote on the weak labels. Ties are broken by randomly selecting a weak label. 2) **Snorkel** (Rattrap et al., 2017) trains a labeling model that aggregates weak labels from different weak sources. 3) **FT-WL** fine-tunes PLMs on the weak labels. 4) **FT-WLST** further applies classic self-training (Lee et al., 2013) on the model obtained by FT-WL. 5) **L2R** (Ren et al., 2018) uses a meta-learning framework to reweight weakly labeled samples. 6) **Meta-Weight-Net** (Shu et al., 2019) also applies meta-learning based sample reweighting. However, the weights are computed through an external reweighting network. 7) **Denoise** (Ren et al., 2020) iteratively corrects wrong annotations in the training set, and the classifier learns with the corrected labels. 8) **UST** (Mukherjee and Hassan Awadallah, 2020) is a self-training based method that assigns higher weights to samples that the teacher is certain about. The uncertainties are measured via MC-dropout on the predictions (Gal and Ghahramani, 2016). 9) **COSINE** (Yu et al., 2021) trains its student network with pseudo-labels which the teacher is highly confident about. In addition, contrastive regularization is introduced to further alleviate error propagation.

For our proposed framework, we report the performances of both Teacher-Init \((f_1)\): the initial teacher trained directly on weak labels, and MSR: the final student model \((g_T)\). \(f_1\) is obtained by running FT-WL five times and selecting the best one among them according to the validation performance. For a fair comparison, the same \(f_1\) is used as the initial teacher for all self-training based models. Finally, we also include the results of fine-tuning PLMs on the clean versions of each dataset, denoted by FT-CL, to represent the upper bound performance.

6 Results

**Comparison with Baselines.** Table 2 shows a comparison among different methods. MSR matches or outperforms SOTAs on all eight datasets. FT-WL outperforms majority voting over the weak labels in all cases except Hausa, which leads to a minor drop. This confirms that PLMs encode useful knowledge in their parameters, enabling them to generalize better than weak rules they are trained on. This phenomenon is particularly noticeable on AGNews, IMDB, and Yelp: direct fine-tuning on the noisy labels (FT-WL) can already achieve decent performance (accuracy above 83%). We consider them easy tasks since label noise has only a minor impact on performance of PLMs and decent generalization can be attained even without specific noise-handling. Applying self-training to such simple tasks lead to further performance improvement. COSINE, a SOTA self-training based model, can even perform comparably to the fully supervised model on these three datasets. On the other five datasets, however, FT-WL performs poorly and conventional self-training methods provide little performance boost (even a disservice on OntoNotes). This implies that self-training relies on a well-performed initial teacher to work effectively. On challenging datasets where the initial teacher is weak, it struggles to achieve further performance gain. Meta-learning based methods such as L2R performs better than COSINE on these challenging datasets. MSR can further boost the performance on all the challenging datasets by up to 11.4% in accuracy or 9.26% in F1 score while maintaining comparable results on simpler datasets.

**Error Decomposition.** Let \(y^\prime, \hat{y}, y\) denote the model prediction, the noisy weak label, and the clean label, respectively. To investigate how the label noise influences the model predictions, we decompose model prediction errors into three types: (1) **Type-A** error: \(y^\prime = \hat{y}; \hat{y} \neq y\) (2) **Type-B** error: \(y^\prime \neq \hat{y} \neq y\) and (3) **Type-C** error: \(y^\prime \neq y; y = \hat{y}\). Type-A/B errors correspond to situations in which a model complies with an incorrect weak label \(\hat{y}\), or predicts another incorrect class label. If, on the other hand, the weak label \(\hat{y}\) is correct, a Type-C error arises if the model predicts a label different than \(\hat{y}\). A higher Type-A error rate indicates that a model memorizes more label noise from the weak sources, while a model that underfits fails to learn useful knowledge from the weak sources can have a higher Type-C error rate.

Figure 3 visualizes the three types of errors on three challenging datasets: TREC, Hausa and CoNLL-03. The blue bars represent model robust-
Table 2: Accuracy and F1 score (in %) on eight NLP tasks. The mean and standard deviation over five trials are reported. Teacher-Init is the best model checkpoint selected from the five trials of FT-WL (according to the validation performance). For a fair comparison, all self-training-based models use the same Teacher-Init checkpoint. MSR matches or outperforms SOTAs on all tasks. † self-training based method, ‡ meta-learning based method.

Figure 3: Prediction error decomposition of various weak supervision baselines, evaluated on the test sets. A model is considered robust against label noise if it manages to predict the correct labels despite the wrong weak labels (the robustness is represented by the blue bars). Otherwise, it conforms to the weak label (Type-A error) or predict another incorrect label (Type-B error), which has a negative effect on generalization. The Type-C error rate quantifies the proportion of incorrect model predictions when weak labels are correct. MSR consistently reduces the Type-A error rate and attains a high level of noise robustness.

Figure 4: Accuracy vs. confidence thresholds. Accuracy vs Confidence. As confidence-based filtering is a key component in both COSINE and MSR, we show the accuracy of model predictions with different confidence thresholds in Figure 4. As can be seen, even using a high confidence threshold for COSINE, the accuracy is still low, which is why it struggles to improve on challenging datasets. MSR, on the contrary, consistently attains higher accuracy with higher confidence thresholds, and thereby confidence-based filtering on top of MSR
help lead to better performance.

**Impact of Label Noise on Feature Space.** We also analyze how the label noise influences representation learning. Figure 5 illustrates the projected feature space of different models on TREC. For a clear visualization, we present only training samples predicted as class 1 by the models in the form of circles. In figs. 5a to 5c, we further visualize the feature space of validation samples (represented by crosses). As can be seen, initially the feature space of class 1 overlaps with that of other classes from the validation set. As the training proceeds, when the teacher keeps refining itself, the MSR student gradually reduces such overlap and learns a well-split representation space. In figs. 5d to 5f, we compare the feature space between different models. The training samples are colored according to their ground truth classes to highlight the misclassification ratio (the more colorful the clusters, the higher the misclassification ratio). We observe that Teacher-Init makes many wrong predictions with high degree of confidence. In this case, utilizing the confidence score for denoising is fragile. This may explain why COSINE, despite offering a more compact cluster, still has a considerable amount of misclassification. Finally, MSR has a considerably cleaner cluster and is less affected by error propagation than COSINE.

**Effects of Validation Data Size.** The model performance reported in Table 2 is based on the original data splits from the WRENCH benchmark. The size of the validation sets is mostly less than 15% of the training sets. Typically, they are used to perform early stopping and model selection. For meta-learning based methods, they additionally rely on the validation sets for meta-update and might be more sensitive to validation size. Hence, we study how the validation size affects different models. In particular, we randomly sample a subset from the original validation set \( D_v \), and repeat the same training process. Figure 6 presents the results. We find...
that the validation size indeed has a greater impact on meta-learning approaches. However, MSR still retains its high generalization performance even with as few as 100 validation samples, suggesting that MSR is very data efficient in performing the self-refinement.

<table>
<thead>
<tr>
<th>Configuration</th>
<th>Seq. Classification (Acc)</th>
<th>NER (F1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Teacher-Init</td>
<td>73.75</td>
<td>68.99</td>
</tr>
<tr>
<td>Student</td>
<td>83.43 (↓ 1.05%)</td>
<td>81.50</td>
</tr>
<tr>
<td>w/o Teacher Scheduler</td>
<td>82.38 (↓ 1.63%)</td>
<td>80.26 (↓ 1.24%)</td>
</tr>
<tr>
<td>w/o Confidence Filtering</td>
<td>82.32 (↓ 1.11%)</td>
<td>81.09 (↓ 0.41%)</td>
</tr>
<tr>
<td>w/o Both</td>
<td>81.63 (↓ 1.80%)</td>
<td>79.95 (↓ 1.55%)</td>
</tr>
</tbody>
</table>

Table 3: Summary of ablation experiments aggregated across multiple datasets. See Appendix D for results in each dataset.

Ablation Study. Table 3 summarizes the impact of different components of our method. In general, our student model performs slightly better than the teacher. This is as expected because a) the teacher’s goal is to guide the student to generalize better, the training loss does not explicitly encourage the teacher to improve its accuracy, and b) the confidence filtering helps the student avoid fitting some wrong pseudo-labels from the teacher. This is also confirmed by the decreased performance when the filter is removed. In addition, applying a learning rate scheduler is better than using a fixed learning rate throughout training.

7 Conclusion

We present MSR, a meta-learning based self-refinement framework that enables robust learning with weak labels. Unlike conventional self-training which relies on a fixed teacher, MSR dynamically refines the teacher based on the student’s performance on the validation set. To further suppress error propagation, we introduce a learning rate scheduler to the teacher and add confidence filtering to the student. We demonstrate that our framework performs on par with or better than current SOTAs on both sequence classification and labeling tasks.

Limitations

In this work, Our primary focus is to propose a strong weak supervision method that works reliably under various weak supervision settings. We employ meta-learning techniques to address the issue of unreliable confidence scores under challenging settings (Figure 4). Despite the effectiveness, the main limitation of our method, just like other meta-learning based frameworks, is the computational overhead. The teacher update step (Algorithm 1, Line 4-6) requires computing both the first and second-order derivatives, which incurs additional computation time and higher memory consumption. Consequently, our method requires longer training.6 Implementation-wise and computation-wise, MSR is as complex as other existing meta-learning based methods, like L2R (Ren et al., 2018) and MW-Net (Shu et al., 2019), but performs substantially better than them in all weak supervision scenarios we evaluated. It is worth noting that MSR has no overhead at inference time. In weak supervision, the data annotation cost is considered the most significant bottleneck. A stronger model is often obtained by trading some more computation with the cost and effort of obtaining more human-generated, manual annotations. Hence, the one-off investment of training MSR can be worthwhile for real-world weak supervision applications.

Acknowledgments

This work has been partially funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – Project-ID 232722074 – SFB 1102 and the EU Horizon 2020 projects ROXANNE under grant number 833635.

References


Allan Peter Davis, Thomas C. Wiegers, Phoebe M. Roberts, Benjamin L. King, Jean M. Lay, Kelley Lennon-Hopkins, Daniela Sciaky, Robin J. Johnson, Heather Keating, Nigel Greene, Robert Hernandez, 6Detailed training time on each dataset can be found in Appendix E The most costly training of MSR takes roughly 3 hours.


Chen Liang, Yue Yu, Haoming Jiang, Siawpeng Er, Ruijia Wang, Tuo Zhao, and Chao Zhang. 2020. Bond: Bert-assisted open-domain named entity recognition with distant supervision. In ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.


A Dataset Details

We experiment with eight NLP datasets, including six English datasets and two datasets in low-resource languages. All datasets come with their ground truth annotations and as well as the weak labels.

A.1 Datasets Selection Criteria

The WRENCH (Zhang et al., 2021) benchmark contains 23 NLP datasets. We choose representative datasets (like previous research in weak supervision) that a) overlap with previous works to enable direct comparisons. b) are diverse in terms of weak label quality, languages and tasks to approve the applicability of different baselines.

A.2 English Datasets

We experiment with four popular sequence classification datasets: AGNews, IMDB, Yelp and TREC.

1. **AGNews** (Zhang et al., 2015): originates from AG, which is a large collection of news articles. The news are categorized in four classes: “World”, “Sports”, “Business” and “Sci/Tech”.

2. **IMDB** (Maas et al., 2011): consists of movie reviews with binary labels. It is a commonly used benchmark dataset for sentiment analysis.

3. **Yelp** (Zhang et al., 2015): obtained from the Yelp Dataset Challenge in 2015. Similar to IMDB, it is a sentiment analysis dataset.


and with the two sequence labeling datasets: CoNLL-03 and OntoNotes 5.0.

1. **CoNLL-03** (Sang and De Meulder, 2003): NER dataset with four named-entity categories.

2. **OntoNotes 5.0** (Pradhan et al., 2013): NER dataset with 18 named-entity categories.

All weak labels are obtained from the WRENCH benchmark7 (Zhang et al., 2021).

A.3 Datasets in Low-Resource Languages

Most datasets in the current WRENCH benchmarks are in English. Although weak supervision is desired in low-resource languages, it is understudied as finding annotators for them is more difficult. Hence, we include two low-resource languages, Yorùbá and Hausa, to cover this scenario. Often, learning with weak labels in low-resource languages is more challenging. First, the training set is often much smaller than English datasets. For example, Hausa has only about 2k training samples while AGNews have 96k. Second, the weak labels in low-resource languages can have lower quality as experts for weak source development are harder to find. A set of simple rules is often used for labeling (which is the case in Yorùbá and Hausa). Hence, weak supervision with low-resource languages is a combination of two challenges: training with small datasets which have low-quality labels.

Yorùbá and Hausa are text classification datasets obtained from (Hedderich et al., 2020).8


2. **Hausa**: consists of news headlines from VOA Hausa which have the same seven classes as Yorùbá. However, only five classes are considered in the text classification task. “Entertainment” and “Sport” have been removed due to the lack of samples of these classes.

Hedderich et al. (2020) provided both the clean labels and weak labels on the two datasets. A gazetteer is created for each class for weak supervision. For example, a gazetteer containing names of agencies, organizations, states and cities in Nigeria is used to label the class “Nigeria”.

A.4 More Dataset Statistics

We provide more details on the datasets we used in our experiments in Table 4. In general, not all data can be covered by weak sources. No weak source is triggered for some training samples and they remain unlabeled. The coverage of the datasets ranges from 69.08% to 100%. Note that for NER tasks, the coverage is always 100% since if no weak source is triggered for a token, we assign

---

7https://github.com/JieyuZ2/wrench

8https://github.com/uds-lsv/transfer-distant-transformer-african
Some extracted text is as follows:

| Dataset   | Task       | # Class | $|D_w|$ | $|D_a|$ | Coverage | Conflict | $|D_v|$ | $|D_t|$ |
|-----------|------------|---------|-------|-------|----------|----------|-------|-------|
| AGNews    | Topic      | 4       | 66,314 | 96,000 | 69.08%   | 14.17%   | 12,000 | 12,000 |
| IMDB      | Sentiment  | 2       | 17,515 | 20,000 | 87.58%   | 11.96%   | 2,500  | 2,500  |
| Yelp      | Sentiment  | 2       | 25,165 | 30,400 | 82.78%   | 18.29%   | 3,800  | 3,800  |
| TREC      | Question   | 6       | 4,723  | 4,965  | 95.13%   | 23.76%   | 500    | 500    |
| Yorùbá    | Topic      | 7       | 1,340  | 1,340  | 100.00%  | 1.87%    | 189    | 379    |
| Hausa     | Topic      | 5       | 2,045  | 2,045  | 100.00%  | 1.90%    | 290    | 582    |
| CoNLL03   | NER        | 4       | 14,041 | 14,041 | 100.00%  | 4.05%    | 3,250  | 3,453  |
| OntoNotes5.0 | NER     | 18      | 115,812| 115,812| 100.00%  | 1.86%    | 5,000  | 22,897 |

Table 4: Dataset statistics. $|D_w|$: number of training samples with weak labels. $|D_a|$: total number of training samples (weakly labeled + unlabeled). Coverage: fraction of samples that are weakly labeled, i.e., $\frac{|D_w|}{|D_a|}$. Conflict: samples that are labeled by at least two weak sources with contradicted weak labels. $|D_v|$: number of validation samples. $|D_t|$: number of test samples.

Hyperparameter search.

<table>
<thead>
<tr>
<th>Hyperparameter</th>
<th>Search Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>Teacher Learning Rate</td>
<td>3e-6, 5e-6, 2e-5, 3e-5</td>
</tr>
<tr>
<td>Teacher Warm-Up Steps</td>
<td>500, 100, 2000, 3000</td>
</tr>
<tr>
<td>Confidence Filter Threshold</td>
<td>0.4, 0.5, 0.6, 0.7, 0.8, 0.95</td>
</tr>
</tbody>
</table>

Table 5: Hyperparameter search.

Label “O” (i.e., non-entity) to it. On the other hand, some samples can be covered by two or more weak sources with contradicted weak labels. In this case, we have a conflict. The conflict ratio ranges from 1.86% to 22.76% in the datasets we tested.

B Implementation Details

Models. All baselines in our paper, except the majority vote and the Snokerl model (Ratner et al., 2017) which work with label space only, use the official RoBERTa model9 (Liu et al., 2019) from Huggingface as the classification backbone for all English datasets, and the multilingual BERT10 for datasets in African languages. We use the base version of the two models which contain roughly 120M and 110M parameters, respectively.

Fine-Tuning on Classification Task. We fine-tune all layers using AdamW (Loshchilov and Hutter, 2019) as the optimizer. For sequence classification tasks, we pass the final layer of the [CLS] token representation ($\mathbb{R}^{768}$) to a feed-forward layer for prediction. For sequence labeling tasks, the final layers of all tokens ($\mathbb{R}^{768 \times L}$, where $L$ is the sentence length) are passed to a shared feed-forward layer to predict the class of each token in the sentence. We report the score where the model performs the best on the validation set during training.

Hyper-Parameters of MSR. We apply grid search on the warm-up steps for the teacher and the confidence threshold for the student network. Table 5 shows our hyperparameter search configuration. We choose the final configurations of the hyperparameters according to the model’s performance on the validation set. Table 6 shows the best configurations of parameters we used to produce the results in Table 2.

Evaluation Metrics. For model evaluation, we report accuracy for sequence classification tasks and F1 Score for sequence labeling tasks. In our implementation, we call the function classification_report() from the scikit-learn library11 to compute the accuracy, and use the Seqeval class from Huggingface12 to compute the F1 Score.

C Validation Performance

The average test performance of MSR is reported in Table 2. We further report the corresponding validation performance in Table 7.

---

9https://huggingface.co/roberta-base
10https://huggingface.co/bert-base-multilingual-cased
12https://github.com/huggingface/datasets/blob/master/metrics/seqeval/seqeval.py
Table 6: Selected hyperparameters. mBERT: multilingual BERT.

<table>
<thead>
<tr>
<th>BERT Backbone</th>
<th>AGNews</th>
<th>IMDB</th>
<th>Yelp</th>
<th>TREC</th>
<th>Yorùbá</th>
<th>Hausa</th>
<th>CoNLL-03</th>
<th>OntoNotes 5.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Batch Size</td>
<td>32</td>
<td>16</td>
<td>16</td>
<td>32</td>
<td>32</td>
<td>32</td>
<td>32</td>
<td>32</td>
</tr>
<tr>
<td>Maximum Sequence Length</td>
<td>128</td>
<td>256</td>
<td>256</td>
<td>64</td>
<td>64</td>
<td>128</td>
<td>64</td>
<td>64</td>
</tr>
<tr>
<td>Student Learning Rate</td>
<td>2e-5</td>
<td>2e-5</td>
<td>2e-5</td>
<td>2e-5</td>
<td>2e-5</td>
<td>2e-5</td>
<td>2e-5</td>
<td>2e-5</td>
</tr>
<tr>
<td>Teacher Learning Rate</td>
<td>2e-5</td>
<td>2e-5</td>
<td>2e-5</td>
<td>2e-5</td>
<td>5e-6</td>
<td>2e-5</td>
<td>2e-5</td>
<td>2e-5</td>
</tr>
<tr>
<td>Teacher Warm-Up Steps</td>
<td>500</td>
<td>500</td>
<td>3000</td>
<td>500</td>
<td>1000</td>
<td>3000</td>
<td>2000</td>
<td>2000</td>
</tr>
<tr>
<td>Confidence Filter Threshold</td>
<td>0.7</td>
<td>0.7</td>
<td>0.5</td>
<td>0.5</td>
<td>0.7</td>
<td>0.4</td>
<td>0.8</td>
<td>0.5</td>
</tr>
</tbody>
</table>

Table 7: The average test and validation accuracy/F1 score (in %) of MSR over five trials.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Test</th>
<th>Validation</th>
</tr>
</thead>
<tbody>
<tr>
<td>AGNews</td>
<td>89.92</td>
<td>89.90</td>
</tr>
<tr>
<td>IMDB</td>
<td>89.16</td>
<td>89.21</td>
</tr>
<tr>
<td>Yelp</td>
<td>95.00</td>
<td>94.79</td>
</tr>
<tr>
<td>TREC</td>
<td>94.80</td>
<td>94.42</td>
</tr>
<tr>
<td>Yorùbá</td>
<td>72.56</td>
<td>75.13</td>
</tr>
<tr>
<td>Hausa</td>
<td>59.11</td>
<td>62.34</td>
</tr>
<tr>
<td>CoNLL-03</td>
<td>88.41</td>
<td>87.86</td>
</tr>
<tr>
<td>OntoNotes</td>
<td>74.59</td>
<td>75.20</td>
</tr>
</tbody>
</table>

D Ablation Studies

We report the detailed ablation results for each dataset in Table 8.

E Hardware and Average Runtime.

We use Nvidia Tesla V100 to accelerate training. The average runtime for each method and dataset is summarized in Table 9.
<table>
<thead>
<tr>
<th>MSR Configuration</th>
<th>AGNews (Acc)</th>
<th>IMDB (Acc)</th>
<th>Yelp (Acc)</th>
<th>TREC (Acc)</th>
<th>Yorùbá (Acc)</th>
<th>Hausa (Acc)</th>
<th>CoNLL-03 (F1)</th>
<th>OntoNotes 5.0 (F1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Student</td>
<td>89.92</td>
<td>89.16</td>
<td>95.00</td>
<td>94.80</td>
<td>72.56</td>
<td>59.11</td>
<td>88.41</td>
<td>74.59</td>
</tr>
<tr>
<td>Teacher</td>
<td>89.02</td>
<td>88.08</td>
<td>94.37</td>
<td>93.80</td>
<td>68.87</td>
<td>60.14</td>
<td>87.30</td>
<td>73.22</td>
</tr>
<tr>
<td>w/o Teacher Scheduler</td>
<td>89.68</td>
<td>87.68</td>
<td>93.78</td>
<td>93.60</td>
<td>70.71</td>
<td>55.32</td>
<td>87.82</td>
<td>72.48</td>
</tr>
<tr>
<td>w/o Confidence Filtering</td>
<td>89.87</td>
<td>89.04</td>
<td>94.76</td>
<td>93.60</td>
<td>71.50</td>
<td>55.15</td>
<td>88.07</td>
<td>74.11</td>
</tr>
<tr>
<td>w/o Both</td>
<td>89.55</td>
<td>87.68</td>
<td>93.33</td>
<td>93.40</td>
<td>70.50</td>
<td>55.32</td>
<td>87.82</td>
<td>72.08</td>
</tr>
</tbody>
</table>

Table 8: Ablation studies. The numbers represent the test accuracy and F1 Score.

<table>
<thead>
<tr>
<th></th>
<th>AGNews</th>
<th>IMDB</th>
<th>Yelp</th>
<th>TREC</th>
<th>Yorùbá</th>
<th>Hausa</th>
<th>CoNLL-03</th>
<th>OntoNotes 5.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Running time (hours)</td>
<td>2.5</td>
<td>1.6</td>
<td>0.5</td>
<td>1.2</td>
<td>0.5</td>
<td>0.7</td>
<td>1.1</td>
<td>3.0</td>
</tr>
</tbody>
</table>

Table 9: Average runtime (in hours) for training a MSR model. One single Nvidia Tesla V100 GPU is used in each experiment to accelerate the computation.