Learning to Detect Noisy Labels Using Model-Based Features

Zhihao Wang[*14], **Zongyu Lin**[*2], **Peiqi Liu**[3], **Guidong Zheng**[3], **Junjie Wen**[3]
Xianxin Chen[1], **Yujun Chen**[1], **Zhilin Yang**[†12]

[1]Recurrent AI, [2]Tsinghua University, [3]China Merchants Bank, [4]Meta
lucaswang@meta.com, {linzongy21, zhiliny}@tsinghua.edu.cn,
{liupeiqi,zhengguidong,wenjunjieee@cmbchina.com}
{chenxianxin,chenyujun@rcrai.com}

Abstract

Label noise is ubiquitous in various machine learning scenarios such as self-labeling with model predictions and erroneous data annotation. Many existing approaches are based on heuristics such as sample losses, which might not be flexible enough to achieve optimal solutions. Meta learning based methods address this issue by learning a data selection function, but can be hard to optimize. In light of these pros and cons, we propose SENT (Selection-Enhanced Noisy label Training) that does not rely on meta learning while having the flexibility of being data-driven. SENT transfers the noise distribution to a clean set and trains a model to distinguish noisy labels from clean ones using model-based features. Empirically, on a wide range of tasks including text classification and speech recognition, SENT improves performance over strong baselines under the settings of self-training and label corruption.[1]

1 Introduction

State-of-the-art deep neural networks require large amounts of annotated training data. Though the success of large pre-training models (Devlin et al., 2018) alleviates such requirements, high-quality labeled data are still crucial to obtain the best performance on downstream tasks. However, it is extremely expensive to acquire large-scale annotated data for every new task.

To overcome the challenge of rigorous data requirements, recent works utilize weak labels for supervision, including heuristic rules (Augenstein et al., 2016; Bach et al., 2019; Awasthi et al., 2020), large-scale datasets with cheap but noisy labels (Li et al., 2017; Lee et al., 2018);, and self-training

(Park et al., 2020; Wang et al., 2020). In self-training, one trains a model on a labeled dataset and then predicts on a large amount of unlabeled data. Then the clean labeled data and pseudo-labeled data are combined to further train the model.

The aforementioned sources of supervision share two important characteristics: First, they are learning from noisy labels. Second, the noise is dependent on data features. Thus, they could be unified into the framework of noisy label learning, for which numerous approaches have been proposed to reduce the negative impact of noise. However, there are three problems in prior work on noisy label learning: First, many existing approaches are based on heuristics such as sample losses which are not flexible enough (Han et al., 2018; Yu et al., 2019; Zhou et al., 2020; Song et al., 2019); Second, many previous works require prior knowledge of the noise distribution of the dataset to adjust the hyperparameters, which is often not available in real-world applications (Song et al., 2019, 2020). Third, meta learning based methods avoid previous problems but suffer from optimization difficulties (Ren et al., 2018; Zheng et al., 2021) such as longer training time, heavy hyper-parameter tuning and an unstable convergence process. To address the above problems, we propose a simple data-driven approach which does not rely on meta learning while being flexible.

Our contributions are summarized as follows:

- We propose a simple yet effective de-noising approach which avoids the optimization difficulty of meta learning while enjoying the flexibility of being data-driven.
- We unify the settings of both self-training and label corruption into a noisy label learning framework and demonstrate the effectiveness of our approach under both settings.
- Our approach improves performance over state-of-the-art baselines on a wide range of datasets, including text classification and auto-

---

matic speech recognition (ASR). Last but not least, our approach achieves even larger gains on few-shot learning.

2 Related Work

2.1 Self-training

Self-training is a powerful learning method that enables models to learn from huge amounts of unlabeled data by generating weak labels through either the teacher model predictions or heuristic rules. Self-training has been shown to be effective in many scenarios, including image classification (Yalniz et al., 2019), text classification (Li et al., 2019), machine translation (Wu et al., 2019), etc. However, noise contained in weak labels could largely hinge the performance of self-training.

Recently, Xie et al. (2020) improved the performance of self-training on image classfication by injecting noise to the student model, which is called NoisyStudent. Park et al. (2020) customized NoisyStudent on automatic speech recognition. One problem related with self-training is error propagation (Zou et al., 2019); in other words, pseudo labelling on unlabeled data might bring noise to the training set which leads to the degradation of further training. Most previous work simply set a fixed threshold to filter samples with low confidence (Sohn et al., 2020; Xie et al., 2020). Wang et al. (2020) used meta learning for adaptive sample re-weighting to mitigate error propagation from noisy pseudo-labels. Zhang et al. (2021) used a curriculum learning approach to re-weight unlabeled data according to the model's learning status. In our work, we alleviate the error propagation from another perspective, by learning a selection model using model-based features based on a clean dataset.

2.2 Noisy Label Learning

Learning from noisy labels has long been a research area. One of the most classical works is to add a noise adaptation layer on top of the main model to learn a label transition matrix for label correction (Goldberger et al., 2017). Bootstrapping (Reed et al., 2014) introduces the notion of perceptual consistency that a model predicts correct labels for noisy samples before overfitting to noisy labels. Co-Teaching (Han et al., 2018) and Co-Teaching+ (Yu et al., 2019) train two networks while each network selects its small-loss samples as clean samples for its peer network. However, the aforementioned

approaches only deal with class dependent noise (CDN) and make a strong assumption that noise distribution is independent of each instance, which is not flexible enough for many cases.

SEAL (Chen et al., 2020) goes beyond previous work to consider instance dependent noise (IDN), which is more realistic and common than CDN on real world datasets. SELFIE (Song et al., 2019) uses a hybrid approach that selects refurbishable samples based on the entropy of model predictions and then refurbishes the labels with model predictions. RoCL (Zhou et al., 2020) utilizes curriculum learning that starts with easy and clean samples and gradually moves to data with pseudo labels produced by a time-ensemble. However, both SELFIE and RoCL require prior knowledge of the noise distribution of the dataset and manual adjustment for hyperparameters. To avoid such efforts, meta learning is introduced to learn selection and refurbishment.

Learning to Re-weight (Ren et al., 2018) is a meta learning algorithm that learns to assign weights to training examples based on their gradient directions. Meta-weight-net (Shu et al., 2019) parameterizes the reweighting function as a multilayer perceptron network. Meta Label Correction (Zheng et al., 2021) trains the target model with corrected labels generated by a label correction model trained on clean validation data which is jointly trained by solving a bi-level optimization problem. These meta learning algorithms afford a large degree of flexibility by directly optimizing a reliable objective. However, meta learning based models are known to be sensitive to hyperparameter tuning and the quality of support data (Agarwal et al., 2021) and suffer from optimization difficulties as they are trained by propagating second-order gradients (Hospedales et al., 2020).

The major differences between our approach and previous methods are as follows. Compared with meta-learning based models, our approach do not suffer from optimization difficulties. Compared with models with CDN assumptions, our approach can handle the IDN settings. Compared with other state-of-the-art IDN methods such as SELFIE and RoCL, the selection strategy in our approach is learnable with model-based features. And our approach does not require the prior knowledge of noise distribution. Last but not least, we unify the settings of self-training and label corruption in the framework of noisy label learning and conduct ex-

tensive experiments on both settings.

3 Method

Now we present our approach, SENT (Selection Enhanced Noisy label Training), that learns to select a subset from a noisy dataset and only uses the selected subset for training to reduce label noise. The core idea is to transfer the noise distribution so that both clean and noisy labels are available on a data subset. A selection model is trained to distinguish clean labels from noisy ones and then applied to selecting a clean subset from a noisy dataset.

Formally, given a noisy training dataset $D_{train} = \{(x_i, y_i) | 1 \leq i \leq N\}$ that is corrupted following some unknown noise distribution P, our approach is to learn a selection strategy f to select a clean subset $D'_{train} = \{x_i | y_i = y_i^*, i = 1, 2, ..., N\}$ for training. Here (x_i, y_i) is a training sample, y_i is the noisy label, y_i^* is the corresponding unknown true label, and N is the size of training set. Let model M be our main model which is trained on the noisy dataset D_{train} and performs certain tasks such as text classification, speech recognition and others. We will also have a selection model S trained on a small clean development dataset D_{select}. The task of S is to learn the selection strategy f. There are two main stages in our approach: noise transfer and selection learning.

3.1 Noise Transfer

Now we describe the noise transfer stage. Given the corrupted training set D_{train}, we learn the unknown noise distribution P and transfer the noise to D_{select}. We train a model M on D_{train} till full convergence, which predicts a (noisy) label given a text input. M is now assumed to have learned to capture the unknown distribution P by its parameters. Then we will use the model M to get noisy labels on D_{select} by making predictions. Here we argue that the noise on D_{select} is approximately following the noise distribution P. Formally, now the development set is $D_{select} = \{(x_i^{select}, y_i^{select}, y_i^{select*})\}$, where $(x_i^{select}, y_i^{select*})$ is the original clean development sample, and y_i^{select} is the noisy label predicted by M.

3.2 Selection Learning

We model the selection learning stage as a binary classification task. On D_{select}, a model S is trained to classify whether a label is clean or noisy. In our approach, the model S is constructed as a multi-layer perceptron with one hidden layer, which uses a pre-calculated 5-dimensional feature vector as the input and outputs a binary classification probability. Given a sample from the development set $(x_i^{select}, y_i^{select}, y_i^{select*})$, the selection training sample is defined as (sg_i, sr_i), where sg_i is a 5-dimensional feature vector for the i-th sample. We will discuss how to compute the 5-dimensional feature in later sections. Meanwhile, sr_i is the corresponding true selection result, defined as $sr_i = \mathbb{I}_{[y_i^{select} = y_i^{select*}]}$. In other words, if the i-th sample has a clean label, $sr_i = 1$, and otherwise zero. Let $P_{select}(sr_i | sg_i)$ denote the probability given by the selection model S. The loss function for each sample can be written as:

$$L_i^{select} = -\log P_{select}(sr_i | sg_i) \qquad (1)$$

3.3 Model-Based Features

Now we discuss how to compute the 5-dimensional feature sg_i for each selection sample x_i^{select}.

The first feature is called the instant loss. Given a sample (x_i, y_i), let $\hat{P}(y_i | x_i)$ be the predicted probability from the main model M. The instant loss (IL) for the sample is defined as:

$$\text{IL}_i = -log\hat{P}(y_i | x_i) \qquad (2)$$

Intuitively, a larger instant loss indicates a possibly noisier sample, because the model M has low confidence in predicting the label. However, as training proceeds, the model will overfit some of the noisy labels. As a result, the instant loss will decrease for noisy samples as well. To address this issue, following Zhou et al. (2020), we additionally use an exponential moving average (EMA) loss to better differentiate noisy and clean samples. In the t-th training epoch, the EMA loss (EMAL) for the i-th sample is defined as follows:

$$\text{EMAL}_i^t = \begin{cases} \gamma\text{EMAL}_i^{t-1} + (1-\gamma)\text{IL}_i, & t \geq 1 \\ \text{IL}_i, & t = 0 \end{cases} \qquad (3)$$

where $\gamma \in [0, 1]$ is a discounting factor. Intuitively, a larger EMAL represents a possibly noisier sample as the model has lower confidence in the training history.

Song et al. (2019) has shown that the entropy of model prediction is a strong indicator to differentiate noisy and clean samples as noisy samples tend to have a larger entropy. Here we adopt two entropy signals as additional features: Instant Entropy (IE) and History Entropy (HE). The calculation of IE follows Song et al. (2019). Let $\hat{y}_i^t$ be
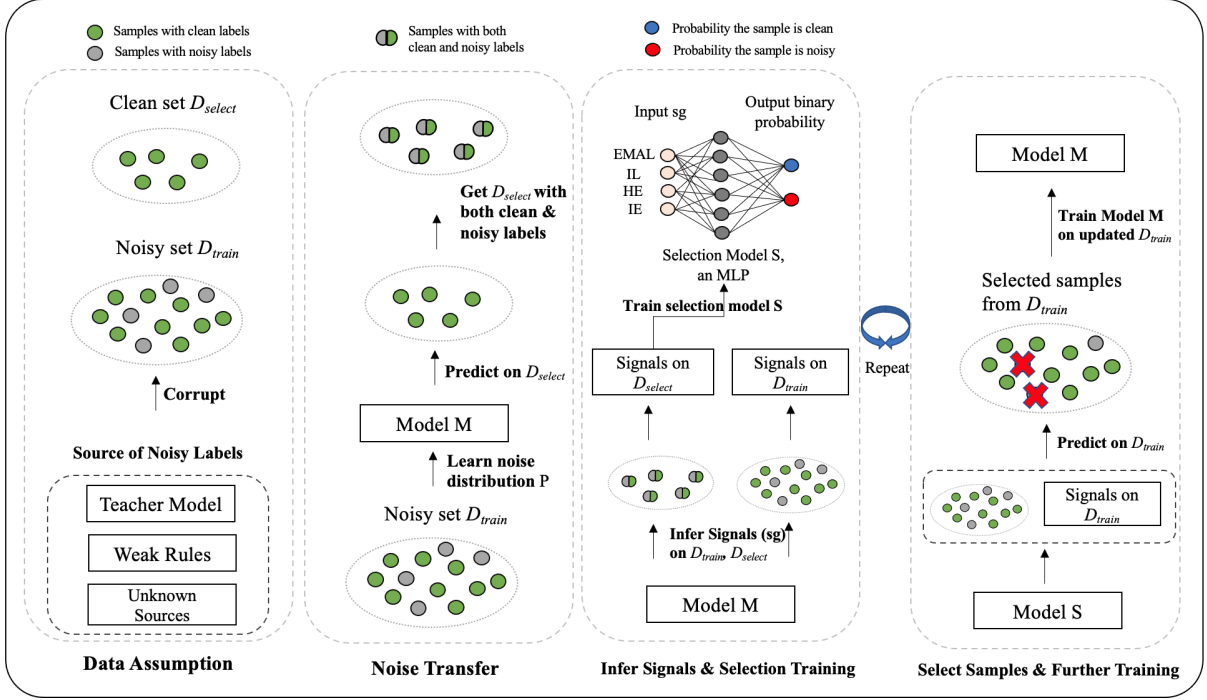
Figure 1: The pipeline of our general framework. Firstly, we are given a noisy training set and a small clean set for selection (also as development set). Secondly, we transfer the noise from the training set to the selection set. Thirdly, we compute predefined signals on both sets and train our selection model **S** using the selection set. Fourthly, We apply the trained selection model on the training set to distinguish clean samples for training model **M**. Finally, we repeat step 3 and 4.

the predicted label of the i-th sample at epoch t and let $H_i(t) = \{\hat{y}_i^0, \hat{y}_i^1, ..., \hat{y}_i^t\}$ be the prediction history of first t epochs. Then we formulate an empirical distribution

$$\tilde{P}(y|x_i, t) = \frac{1}{t} \sum_{y' \in H_i(t)} \mathbb{I}_{[y=y']}$$

which equals the ratio of prediction y in the first t epochs. The IH feature at the t-th epoch is computed as

$$\mathrm{IE}_i = (1/\tau) \times -\tilde{P}(\hat{y}_i^t|x_i, t) \log \tilde{P}(\hat{y}_i^t|x_i, t) \quad (4)$$

where $\tau = -\log(1/k)$ is a normalizing factor with k being the number of labels. The HE feature at the t-th epoch is computed as

$$\mathrm{HE}_i = (1/\tau) \times \sum_y -\tilde{P}(y|x_i, t) \log \tilde{P}(y|x_i, t) \quad (5)$$

We explore another informative feature inspired by (Han et al., 2019) who discovered that distances between low level features and high level features in a convolution model are larger on noisy samples than on clean samples. We find this holds for transformer models. Thus, we adopt the cosine similarity between the hidden states of the first layer and the last layer as another feature. Formally, FLS (First Last Similarity) is defined as:

$$\mathrm{FLS}_i = \mathrm{normalize}\left(\sum_j \cos(\mathbf{h}_{ij}^F, \mathbf{h}_{ij}^L)\right) \quad (6)$$

where $\mathbf{h}_{ij}^F$ and $\mathbf{h}_{ij}^L$ represent the hidden states of the j-th token in the first and last layers respectively, and $\cos$ refers to cosine similarity. We normalize the FLS feature into the range of $[0, 1]$.

Finally, we concatenate the above all features to be the input for the selection model S as follows,

$$sg_i = [\mathrm{EMAL}_i, \mathrm{IL}_i, \mathrm{HE}_i, \mathrm{IE}_i, \mathrm{FLS}_i] \quad (7)$$

where $[,]$ denotes concatenation.

3.4 Overall Training Procedure

Now we show the training procedure with pseudo code in Algorithm 1 and illustrations in Figure 1. In the first box of Figure 1, we clarify our data setting, where a clean set D_{select} and a corrupted set D_{train} are the input to our method. In the second box, we learn the noise distribution P of

Algorithm 1 SENT

1: Initialization: M;S;D_{train};D_{select}; total_epochs; pre-train_epochs
2: Train M on D_{train}. {Learn noise distribution P}
3: Infer on D_{select} using M. {Noise Transfer}
4: epoch = 0; Reinitialize M;
5: Pretrain M for pretrain_epochs;
6: **while** epoch < total_epochs **do**
7:     Infer signals [IL, EMAL, HE, IE, FLS] on D_{train} and D_{select} using M;
8:     **while** not early stopping S **do**
9:         Train S on D_{select};
10:     **end while**
11:     Select a clean subset D'_{train} out of D_{train} using S;
12:     Train M on D'_{train};
13: **end while**
Output: M

the training set by fitting model M on D_{train} and then transfer P to D_{select}. After this step, we will have both noisy and clean labels on D_{select}. This box corresponds to lines 2 to 4 in Algorithm 1.

Next, we will move on to the repeated training stage for model S and model M, which is shown in box 3 and 4 of the figure. First, as illustrated in box 3, we use model M to infer the aforementioned signals on D_{train} and D_{select}. Then we will use the signals on D_{select} as the input to train the selection model S. Since D_{select} has both clean and noisy labels, we can easily get the training targets sr as mentioned in Section 3.2 for the selection model S. Once we are done with selection learning, given the inferred signals on D_{train}, we can use S to predict and select clean samples from D_{train} to get D'_{train}. Then we will train M on the clean subset D'_{train} and repeat above steps. D'_{train} is not guaranteed to be entirely clean but is expected to be cleaner than D_{train}. This repeated training phase corresponds to the code from lines 6 to lines 13 in Algorithm 1.

3.5 Adaptation to Self-Training

Our above framework can be directly applied to learning scenarios with noisy labels. In this section, we will further discuss how to adapt this method to self-training, a classic semi-supervised learning paradigm. Generally, it trains the teacher model on labeled data to infer pseudo labels on unlabeled data and add them back to the original training set. Then a student model is trained on the combined data. After that, the student becomes a new teacher model and the above process is repeated. Obviously, the process of pseudo labelling will bring noise since it cannot ensure 100% accuracy on the predicted labels. Therefore, it fits the nature of our proposed framework. Specifically, the noise dis-

tribution P in self-training is known because the source of noise is the teacher model. Thus, it is natural to directly leverage the teacher model to infer noisy labels on D_{select}.
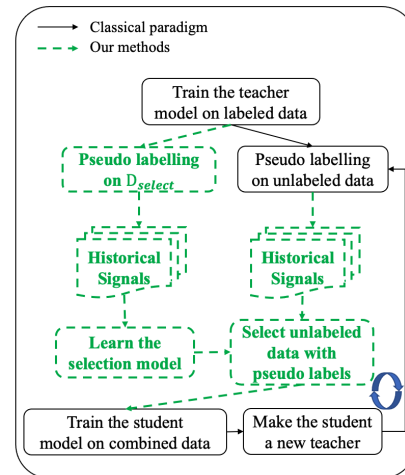

Figure 2: The pipeline of our general system applied to the classical self-training paradigm.

The whole pipeline of adapting our framework to classic self-training is displayed in Figure 2, and the corresponding algorithm is shown in Algorithm 2 in Appendix. After training the teacher model on labeled data L, we use the teacher model to infer on the unlabeled data U and the dev set D_{select}. This is followed by training the selection model based on the signals of D_{select}. Then, we utilize the selection model to predict on the unlabeled data U to judge whether to choose each pseudo-labeled sample or not. Then, we train the student model on the combination of original labeled data and selected samples from unlabeled data. The above procedure is repeated as in classical self-training.

4 Experiments

4.1 Experimental Setup

4.1.1 Overview

To verify the effectiveness of SENT, we conduct extensive experiments on text classification and automatic speech recognition (ASR) benchmarks.

We use text classification tasks to evaluate the self-training setting. We perform finetuning based on the BERT (Devlin et al., 2018) model. We use ASR tasks to evaluate the label corruption setting. Our approach and other baselines are built on top of an encoder-decoder transformer network. Details of model configuration can be found in Ap-

	IMDB	SMS	SMS*	TREC	TREC*	YOUTUBE	AGNEWS
Train	250	69	61	68	60	77	60
Test	25000	500	392	500	500	392	7600
Unlabeled	24500	4502	4948	4965	5409	1409	11876
Dev_eval	126	250	31	251	30	40	32
Dev_select	124	250	31	249	32	38	32

Table 1: Statistics of text classifcation datasets.

Method	TREC	TREC*	SMS	SMS*	AGNEWS	IMDB	YOUTUBE
Supervised	83.4	84.2	97.8	97.5	78.1	85.1	92.9
Co-teaching+	81.8	79.0	98.2	98.4	82.8	86.8	92.1
L2R	80.8	86.0	98.6	97.2	84.2	84.7	92.9
SELF	81.0	81.2	98.4	98.6	82.4	83.5	92.9
Self-train	84.0	84.8	97.9	98.4	82.4	87.0	93.6
Self-train (thres)	84.2	87.8	97.9	98.4	83.3	85.6	93.1
Noisy Student	**85.0**	88.6	98.4	99.0	85.0	87.7	93.9
Ours	**85.0**	86.6	**99.0**	**99.2**	83.4	88.3	94.4
Ours + noisy	**85.0**	**89.2**	**99.0**	**99.2**	**86.0**	**89.0**	**95.2**

Table 2: Results for text classification in the self-training setting. We compare our approach with baselines under the BERT-based models.

pendix A.2.

4.1.2 Datasets

For text classification in the self-training setting, we evaluate our framework on the following five benchmark datasets: question classification TREC-6 (Li and Roth, 2002), spam classification of SMS messages (Almeida et al., 2011), spam classification of YouTube comments (Alberto et al., 2015), AG's news topic classification dataset (Zhang et al., 2015), and sentiment classification on IMDB movie reviews (Maas et al., 2011). Our data splits follow previous work (Karamanolakis et al., 2021). Related details are shown in Table 1. For the SMS and TREC datasets, we consider two separate versions. The datasets with ∗ have smaller development sets D_{eval} and D_{select} while the ones without ∗ have larger developments. Smaller development sets are more challenging for noisy label learning because selection learning has to perform on a smaller clean set. We use these two separate versions to test the robustness of our approach.

For ASR in the label corruption learning setting, we use AISHELL-1 (Bu et al., 2017) as the benchmark. Following prior work, we model IDN using DNNs' prediction error (Du and Cai, 2015;

Menon et al., 2018). Specifically, we train three small transformer models to corrupt the training set to different corruption levels: hard, medium, easy. The higher the error rate, the harder the corrupted dataset. In the following experiments of SENT, the prior noise information (i.e. how the training set is corrupted) is assumed to be unknown. Related statistics are shown in Table 8 in Appendix.

4.1.3 Evaluation

For text classification, we report micro F1 for SMS and accuracy for the rest of the datasets. For ASR, we follow Bu et al. (2017) to use the character error rate (CER) for evaluation.

Since our approach relies on an additional clean set D_{select}, we split the normal development set into two halves. We use one half as D_{select} for selection learning and the other half D_{eval} for standard model tuning, so as to set up fair comparison with the baselines.

4.1.4 Baselines

For text classification, we compare with the following baselines: (a) "Supervised" refers to supervised learning using only labeled data; (b) "self-train" is standard self training that utilizes both labeled and unlabeled data for iterative train-

Model	Hard	Medium	Easy
Vanilla	32.63	21.82	16.10
Co-Teaching+	32.08	21.67	15.11
L2R	30.43	20.07	15.15
RoCL	27.16	17.87	14.95
SELFIE	27.31	19.10	13.98
Ours	**26.91**	**17.67**	**13.47**

Table 3: Comparison with baselines on AISHELL-1 test set in the label corruption setting. We use CER as the metric of performance. A lower CER indicates a better model.

Data	Method	Pse-Acc.	Sel-Pre.	Sel-Rec.	#Selected
IMDB	Self-train	85.7	85.7	**100.0**	24500
	Self-train (thres)	85.2	94.3	58.1	12870
	Ours	**88.1**	**97.9**	47.5	7742
YOUTUBE	Self-train	95.0	95.0	**100.0**	1409
	Self-train (thres)	**95.7**	96.1	98.7	1386
	Ours	94.6	**96.9**	94.1	1294
SMS*	Self-train	98.1	98.1	100.0	4948
	Self-train (thres)	96.0	96.0	98.6	4880
	Ours	**98.3**	**98.3**	100.0	4897
TREC*	Self-train	77.4	77.4	**100.0**	4965
	Self-train (thres)	77.2	77.2	99.8	4944
	Ours	**78.3**	**81.2**	67.1	4897
AGNEWS	Self-train	**84.4**	84.4	**100.0**	11876
	Self-train (thres)	83.4	83.6	99.7	11826
	Ours	84.3	**88.0**	85.6	9728

Table 4: The key metrics of pseudo labeling and sample selection during the self-training. We report the accuracy of pseudo labeling, the precision and recall of sample selection, and the number of selected samples.

ing; (c) "self-train (thres)" means self-training that uses the development set to select a threshold of confidence score for filtering pseudo-labeled data; (d) "noisy student" (Xie et al., 2020) adds dropout noise to the student model in self training; (e) "co-teaching+" (Yu et al., 2019) uses two neural networks to select small-loss samples for each other and applies a disagreement strategy; (f) "L2R" (Ren et al., 2018) learns to re-weight noisy labels via meta-learning (g) "SELF" (Nguyen et al., 2019) utilizes self-ensemble predictions to progressively remove noisy labels. We also evaluate the performance when combining noisy student and our method, denoted as "ours + noisy".

In the experiments of ASR, we include the following baselines: Vanilla (a naive encoder-decoder transformer network), Co-Teaching+ (Yu et al., 2019), L2R (Ren et al., 2018), RoCL (Zhou et al., 2020) and SELFIE (Song et al., 2019). The details of these baselines can be found in Appendix A.1.1.

4.2 Experimental Results

4.2.1 Results in Text Classification.

As shown in Table 2, the self-training baseline improves text classification performance. In comparison, our selection approach can stably lift the performance, which shows that our selection model has learned an informative selection strategy. Last, although using SENT alone outperforms self-train and noisy student, our approach can be combined with the noisy student approach to achieve even better performance. Overall, this combined approach achieves the best performance among all the datasets we consider.

4.2.2 Results in ASR

Table 3 shows that Co-Teaching+ is not able to handle our setting as it only achieves similar result to the vanilla model. L2R is effective on our problem setting with improved performance. However, meta learning based L2R underperforms RoCL and SELFIE. Compared with above baselines, our approach consistently excels on all error levels, which demonstrates the effectiveness of our approach.

4.3 Empirical Analysis

Case Study: Key Metrics During Self-Training
In order to gain a deeper understanding of how our method improves over the traditional self-training methods, we investigate some of the key metrics regarding pseudo labelling and selection performance. We consider self-train, self-train (thres), and our model. Specifically, we display the accuracy of the pseudo labelling on unlabeled data (Pse-Acc.), the precision of sample selection (Sel-Pre.), the recall of sample selection (Sel-Rec.) and the number of selected samples in the best round (i.e., the training round that achieves the best performance in the repeated self-training process). As can be seen in Table 4, our approach achieves a better pseudo labeling accuracy. This is because our approach obtains a more balanced tradeoff between selection precision and selection recall compared to self training. Because of a more rigorous selection model, our approach tends to only select samples with a higher probability of having clean labels; i.e., increasing the selection precision. This is also reflected in small decrease in the number of selected samples. We believe this is crucial for mitigating error propagation (Xie et al., 2020) and thus for better performance.

Method	TREC	SMS	YOUTUBE
Supervised	66.5	93.3	91.0
Co-teaching+	66.8	98.3	93.3
L2R	66.4	98.0	93.3
SELF	66.3	98.1	92.3
Self-train	**71.1**	95.1	92.5
Self-train (thres)	70.1	98.1	92.1
Noisy Student	68.9	98.1	92.1
Ours	70.3	98.2	93.3
Ours+Noisy	70.0	**98.4**	**93.4**

Table 5: Comparison with baselines under the MLP models.

Signal	Hard	Medium	Easy
All	**26.91**	**17.67**	**13.47**
-EMAL	27.31	17.95	13.85
-IL	28.56	19.32	14.94
-HE	29.32	19.88	15.01

Table 6: Ablation experiments for features on AISHELL-1. 'All' is all features. Each line removes one feature based on the last line.

Ablation Study: Substituting with Simpler Models To further test the robustness of our approach, we substitute the base model from BERT to simple multi-layer percetrons (MLPs) without pretraining. As show in Table 5, the performance will decrease after applying simpler MLPs. However, our approach remains effective compared to the other baselines. The relative gain and absolute improvement from "Supervised" to our approach is still significant.

Ablation Study: Features We study the effects of the model-based features we introduced in Section 3.3 with an ablation. Note that we did not use FLS for ASR because the the first layer and the last layer have different lengths. As shown in Table 6 and Table 7, all of the features contribute to the final performance. Among the features, adding the instant loss (IL) feature results in the most relative gain for performance.

Also, we do similar experiments on text classification tasks. As seen in Table 7, we can do basic search on feature engineering to improve the final performance. Among all signals, IL leads to the greatest relative gain of the performance. But across all the datasets, each signal has its own role and contributes to our final performance.

Signal	IMDB	YT	TREC	TREC*	SMS	SMS*	AG
All	85.9	**95.2**	**85.0**	**89.2**	**99.0**	**99.2**	85.1
-FLS	88.2	92.9	83.6	82.0	99.0	99.0	84.1
-EMAL	**89.0**	92.1	84.2	86.8	99.0	99.0	**86.0**
-IL	87.8	92.6	83.4	85.8	99.0	98.8	82.5
-HE	88.0	93.4	82.6	79.8	98.4	98.0	79.6

Table 7: Ablation experiments for features on text classification. 'All' is all signals. Each line removes one feature from the previous line. YT represents YouTube dataset, and AG represents AG News dataset.
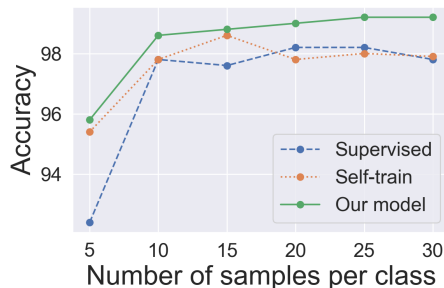

Figure 3: Evaluation on SMS under the few-shot setting.

Performance Under Few-shot Setting LST (Li et al., 2019) has shown that self-training paradigm can be customized on few-shot classification. Here, we also investigate the effectiveness of our method when applying to the few-shot setting. Specifically, we evaluate on selected text classification datasets (i.e., YOUTUBE, SMS and IMDB). Figure 5 showcases that generally self-train performs better than "Supervised" (using only labeled data), while our model achieves the best performance in most cases, indicating the robustness of our method. More results are displayed in Appendix A.3.2.

5 Conclusions

In this paper, we propose SENT to address the problem of label noises. Compared with meta learning based models, our selection model is trained with full supervision using cross entropy loss which facilitates the convergence process. In the meantime, we model IDN noise without the prior knowledge of noise distribution. We also unify the setting of self-training and label corruption in the framework of noisy label learning and conduct extensive experiments on both settings.

6 Limitations

Although our framework has been proved to be effective under the setting of self-training and label corruption on text classification and speech recognition tasks, adapting our approach to more sequence-

level tasks such as named entity recognition (NER) and machine translation will also be interesting. Besides, the selection model in our framework is feature-based. These features are very informative but might be limited in terms of expressivity. This reserves the space for further improvement to learn more data-driven features under our framework.

References

Mayank Agarwal, Mikhail Yurochkin, and Yuekai Sun. 2021. On sensitivity of meta-learning to support data. *Advances in Neural Information Processing Systems*, 34.

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

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

Isabelle Augenstein, Tim Rocktäschel, Andreas Vlachos, and Kalina Bontcheva. 2016. Stance detection with bidirectional conditional encoding. *arXiv preprint arXiv:1606.05464*.

Abhijeet Awasthi, Sabyasachi Ghosh, Rasna Goyal, and Sunita Sarawagi. 2020. Learning from rules generalizing labeled exemplars.

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

Hui Bu, Jiayu Du, Xingyu Na, Bengu Wu, and Hao Zheng. 2017. Aishell-1: An open-source mandarin speech corpus and a speech recognition baseline. In *2017 20th Conference of the Oriental Chapter of the International Coordinating Committee on Speech Databases and Speech I/O Systems and Assessment (O-COCOSDA)*, pages 1–5. IEEE.

Pengfei Chen, Junjie Ye, Guangyong Chen, Jingwei Zhao, and Pheng-Ann Heng. 2020. Beyond class-conditional assumption: A primary attempt to combat instance-dependent label noise. *arXiv preprint arXiv:2012.05458*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Jun Du and Zhihua Cai. 2015. Modelling class noise with symmetric and asymmetric distributions. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*.

J Goldberger, E Ben-Reuven, et al. 2017. Deep neural—networks using a noise adaptation layer. In *Proc. 5th Int. Conf. Learn. Represent.(ICLR) Conf. Track*, 2014, pages 1–9.

Bo Han, Quanming Yao, Xingrui Yu, Gang Niu, Miao Xu, Weihua Hu, Ivor Tsang, and Masashi Sugiyama. 2018. Co-teaching: Robust training of deep neural networks with extremely noisy labels. *arXiv preprint arXiv:1804.06872*.

Jiangfan Han, Ping Luo, and Xiaogang Wang. 2019. Deep self-learning from noisy labels. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5138–5147.

Timothy Hospedales, Antreas Antoniou, Paul Micaelli, and Amos Storkey. 2020. Meta-learning in neural networks: A survey. *arXiv preprint arXiv:2004.05439*.

Zhiting Hu, Xuezhe Ma, Zhengzhong Liu, Eduard Hovy, and Eric Xing. 2016. Harnessing deep neural networks with logic rules. *arXiv preprint arXiv:1603.06318*.

Giannis Karamanolakis, Subhabrata Mukherjee, Guoqing Zheng, and Ahmed Hassan Awadallah. 2021. Self-training with weak supervision. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 845–863, Online. Association for Computational Linguistics.

Kuang-Huei Lee, Xiaodong He, Lei Zhang, and Linjun Yang. 2018. Cleannet: Transfer learning for scalable image classifier training with label noise. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5447–5456.

Wen Li, Limin Wang, Wei Li, Eirikur Agustsson, and Luc Van Gool. 2017. Webvision database: Visual learning and understanding from web data. *arXiv preprint arXiv:1708.02862*.

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

Xinzhe Li, Qianru Sun, Yaoyao Liu, Qin Zhou, Shibao Zheng, Tat-Seng Chua, and Bernt Schiele. 2019. Learning to self-train for semi-supervised few-shot classification. *Advances in Neural Information Processing Systems*, 32:10276–10286.

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, Portland, Oregon, USA. Association for Computational Linguistics.

Aditya Krishna Menon, Brendan Van Rooyen, and Nagarajan Natarajan. 2018. Learning from binary labels with instance-dependent noise. *Machine Learning*, 107(8):1561–1595.

Duc Tam Nguyen, Chaithanya Kumar Mummadi, Thi Phuong Nhung Ngo, Thi Hoai Phuong Nguyen, Laura Beggel, and Thomas Brox. 2019. Self: Learning to filter noisy labels with self-ensembling. *arXiv preprint arXiv:1910.01842*.

Daniel S Park, Yu Zhang, Ye Jia, Wei Han, Chung-Cheng Chiu, Bo Li, Yonghui Wu, and Quoc V Le. 2020. Improved noisy student training for automatic speech recognition. *arXiv preprint arXiv:2005.09629*.

Alexander Ratner, Stephen H Bach, Henry Ehrenberg, Jason Fries, Sen Wu, and Christopher Ré. 2017. Snorkel: Rapid training data creation with weak supervision. In *Proceedings of the VLDB Endowment. International Conference on Very Large Data Bases*, volume 11, page 269. NIH Public Access.

Scott Reed, Honglak Lee, Dragomir Anguelov, Christian Szegedy, Dumitru Erhan, and Andrew Rabinovich. 2014. Training deep neural networks on noisy labels with bootstrapping. *arXiv preprint arXiv:1412.6596*.

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

Jun Shu, Qi Xie, Lixuan Yi, Qian Zhao, Sanping Zhou, Zongben Xu, and Deyu Meng. 2019. Meta-weight-net: Learning an explicit mapping for sample weighting. *arXiv preprint arXiv:1902.07379*.

Kihyuk Sohn, David Berthelot, Chun-Liang Li, Zizhao Zhang, Nicholas Carlini, Ekin D Cubuk, Alex Kurakin, Han Zhang, and Colin Raffel. 2020. Fixmatch: Simplifying semi-supervised learning with consistency and confidence. *arXiv preprint arXiv:2001.07685*.

Hwanjun Song, Minseok Kim, and Jae-Gil Lee. 2019. Selfie: Refurbishing unclean samples for robust deep learning. In *International Conference on Machine Learning*, pages 5907–5915. PMLR.

Hwanjun Song, Minseok Kim, Dongmin Park, Yooju Shin, and Jae-Gil Lee. 2020. Learning from noisy labels with deep neural networks: A survey. *arXiv preprint arXiv:2007.08199*.

Yaqing Wang, Subhabrata Mukherjee, Haoda Chu, Yuancheng Tu, Ming Wu, Jing Gao, and Ahmed Hassan Awadallah. 2020. Adaptive self-training for few-shot neural sequence labeling. *arXiv preprint arXiv:2010.03680*.

Lijun Wu, Yiren Wang, Yingce Xia, Tao Qin, Jianhuang Lai, and Tie-Yan Liu. 2019. Exploiting monolingual data at scale for neural machine translation. 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 4207–4216.

Qizhe Xie, Minh-Thang Luong, Eduard Hovy, and Quoc V Le. 2020. Self-training with noisy student improves imagenet classification. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10687–10698.

I Zeki Yalniz, Hervé Jégou, Kan Chen, Manohar Paluri, and Dhruv Mahajan. 2019. Billion-scale semi-supervised learning for image classification. *arXiv preprint arXiv:1905.00546*.

Xingrui Yu, Bo Han, Jiangchao Yao, Gang Niu, Ivor W. Tsang, and Masashi Sugiyama. 2019. How does disagreement help generalization against label corruption?

Bowen Zhang, Yidong Wang, Wenxin Hou, Hao Wu, Jindong Wang, Manabu Okumura, and Takahiro Shinozaki. 2021. Flexmatch: Boosting semi-supervised learning with curriculum pseudo labeling. *arXiv preprint arXiv:2110.08263*.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. *Advances in neural information processing systems*, 28:649–657.

Guoqing Zheng, Ahmed Hassan Awadallah, and Susan Dumais. 2021. Meta label correction for noisy label learning. In *Proceedings of the 35th AAAI Conference on Artificial Intelligence*.

Tianyi Zhou, Shengjie Wang, and Jeff Bilmes. 2020. Robust curriculum learning: From clean label detection to noisy label self-correction. In *International Conference on Learning Representations*.

Yang Zou, Zhiding Yu, Xiaofeng Liu, BVK Kumar, and Jinsong Wang. 2019. Confidence regularized self-training. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5982–5991.

A Appendix

We provide details of our datasets (Section A.1) and experimental results (Section A.2).

A.1 Details of Implementation

Algorithm 2 Adaptation of SENT on classical self-training.

1: Initialization:Labeled data L, Unlabeled data U, total dev data D_{total}, dev_select data D_{select};
2: Initialize selected_train_set $L_{chosen} = L$;
3: Initialize the teacher model $M_{teacher}$, selection model S;
4: Train the teacher model $M_{teacher}$ on L_{chosen};
5: Infer on U, D_{select} using $M_{teacher}$;
6: Infer signals $sg = [IL, EMAL, HE, IE, FLS]$ on U, D_{select} using $M_{teacher}$ and get signals $sg_{unlabeled}, sg_{select}$;
7: Train S on D_{select};
8: $P_{select} = S(U, sg_{unlabeled})$;
9: $sr = argmax(P_{select})$;
10: $L_{chosen} = L \cup U[sr]$;
11: Train the student model $M_{student}$ on L_{chosen};
12: Make the student a teacher and go back to step 5 and repeat.

A.1.1 Baselines

In the first experiment of text classification, we also take the same baselines which consider rules and report the same results as utilized in ASTRA (Karamanolakis et al. (2021)). (a) Majority predicts the majority vote of the rules with ties resolved by predicting a random class. (b) Snorkel+Labeled (Ratner et al. (2017)) trains classifiers using weakly-labeled data with a generative model. . (c) L2R (Ren et al. (2018)) learns to re-weight noisy labels from rules by meta learning. (d) PosteriorReg (Hu et al. (2016)) uses rules as soft constraints for training by posterior regularization. (e) ImplyLoss (Awasthi et al. (2020)) learns both instance-specific and rule specific weights by minimizing an implication loss (h) ASTRA (Karamanolakis et al. (2021)) introduces an rule attention network to leverage multiple sources of weak supervision with trainable weights to compute soft weak labels for unlabeled data.

For ASR baselines, Vanilla is a naive encoder-decoder transformer network without any denoising moduels. All following baselines and our approache are build on top of the vanilla model. (b) Co-Teaching+ trains two networks with each network selecting its small-loss samples as clean samples for its peer network. (c) L2R is the same as mentioned before. (d) RoCL starts learning with easy and clean samples and gradually moves to

learn noisy-labeled data with pseudo labels produced by a time-ensemble of the model and data augmentations.(e) SELFIE selects refurbishable samples based on the entropy of model predictions and refurbs the samples with model predictions.

A.2 Details of Experiments

For ASR models, the transformer model contains 12 layers of encoder and 6 layers of decoder. For each transformer block, the number of heads in the multiheadattention module is 4. The dimension of the encoder and decoder input is 256. The dimension of the feedforward network is 2048. We use 80-dimensional filter bank coefficient as input features. The hyper parameter for training is show in Table 9. Batch_size_in_s2 means the maximum allowed length of audio in seconds in one batch. History_length represents the maximum allowed length for stored history predicted labels. These history predictions are used to calculate entropy.

It should be noted that in both text classification as ASR tasks, we split the total development set into dev_select and dev_eval, where dev_select is used to train the selection model and dev_eval is used to evaluate the selection model.

There are three details worth to noice in ASR: a) we perform an additional correction module for ASR. The correction module has the same architecture as the selection module. Correction module takes the same signal as selection module, and it outputs three weights which sum to one. The weights are assigned to the noisy labels (NL), model predicted probabilities (Pred), accumulated corrected labels respectively. The corrected label(CL) at T-th epoch is:

$$CL^T = w_1 * NL + w_2 * Pred + w_3 * CL^{T-1} \quad (8)$$

after correction, we will perform the normal selection module to select clean labels from corrected labels. b) Since ASR is sequence level problem, we can not correct and select each token independently which would ignore the word dependencies. We first align the predicted word sequence to the noisy target sequence accoding to their longest common sequence. Then we will correct and select the token that are not common in both sequence. c) As ASR is a generation problem and the length of input and output is different, we do not extract FSL as a feature for our approach.

Stats	OriginalTrain	HardTrain	MediumTrain	EasyTrain	OriginalDev	OriginalTest
CER	NA	37.11	26.19	16.14	NA	NA
MaxLen	44	82	89	44	35	37
AvgLen	14.41	14.30	14.35	14.41	14.33	14.60
Num	120098	120098	120098	120098	14326	7176

Table 8: Statistics of AISHELL-1. 'Worse','Normal','Better' are three notation we use to represent different error levels. 'NA' means not char error rate because original datasets are assumed to be clean. 'Num' represents the data volume of each set.

HP	HardTrain	MediumTrain	EasyTrain
Max_LR	5e-4	3e-4	3e-4
Min_LR	5e-6	1e-6	1e-6
Warmup_step	20000	20000	20000
Max_steps	80000	50000	50000
Batch_size	300	300	300
Batch_size_in_s2	500	500	500
History_length	18	12	12

Table 9: Hyperparameter for SENT on ASR. HP means hyperparameters.

Model	Hard	Medium	Easy
Vanilla	30.06	20.21	14.42
Co-Teaching+	29.67	20.02	13.71
L2R	27.79	19.08	14.12
RoCL	24.50	16.34	13.22
Selfie	24.40	17.22	13.01
ours	**23.97**	**16.01**	**11.96**

Table 10: Comparison with other baselines on dev set of AISHELL-1

$$FX - score = \frac{(1 + X^2) * precision * recall}{X^2 * preision + recall} \quad (9)$$

Noted that this metric becomes F1-score if we set X as 1. The X measures the preference of precision to recall. If X approaches 0, it becomes precision. If X approaches infinite, it becomes recall. As shown in Figure 4, the performance gradually decreases as the X grows, which implicitly indicates that precision matters more than recall when we are going to select samples from unlabeled data.
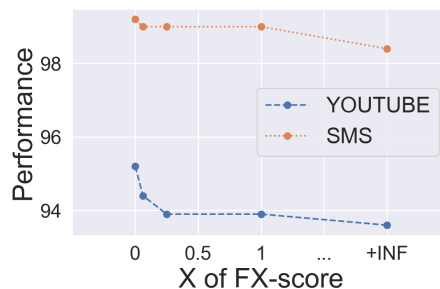

Figure 4: The influence of X of FX-score as the selection threshold on the final performance.

A.3 Details of Ablation Study

A.3.1 The Influence of Selection Threshold On The Final Performance

In practice, we find that choosing a proper threshold for selection model might have some influence on the final performance. In detail, we choose FX-score as the target to choose the threshold which yields best FX-score on the dev_eval set, and use this threshold to select the samples from the unlabeled data based on the output of the selection model. We investigate the influence on final performance by changing the X of FX-score on YOUTUBE and SMS. The computation of this metric is displayed as follow:

A.3.2 The Performance Under The Few-shot Setting

We also investigate the performance of IMDB and YOUTUBE under the few-shot learning setting. The results are shown in Figure 5.

A.3.3 Case Study: Signal Difference On Train and Development Sets.

We show the difference of signals on training and development sets in Figure 6, 7, 8. We can see that the all signals show close statistics on both sets. This indicates that our noise transfer approach holds and has a good performance. This phenomena exists in all three error levels.
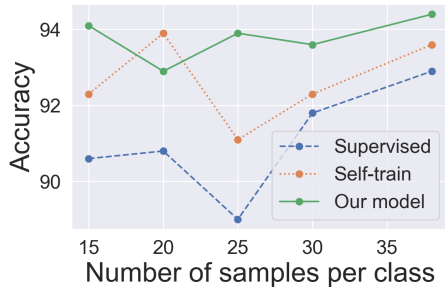
(a) YOUTUBE
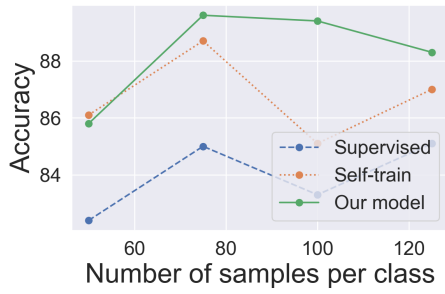
(b) IMDB

Figure 5: Evaluation on Youtube, SMS and IMDB under the few-shot setting.

A.3.4 Influence of Signals On AISHELL-1 Dev

We show the influence of different signals on AISHELL-1 development set in Table 11. We can see that the tendency is same as the tendency on the test set.

Signal	Hard	Medium	Easy
All	**23.97**	**16.01**	**11.96**
-EMAL	24.35	16.43	12.45
-IL	25.48	17.55	13.71
-HE	25.89	18.01	13.99

Table 11: Ablation experiments for signals on dev set of AISHELL-1. Each line is removing one signal from previous line.
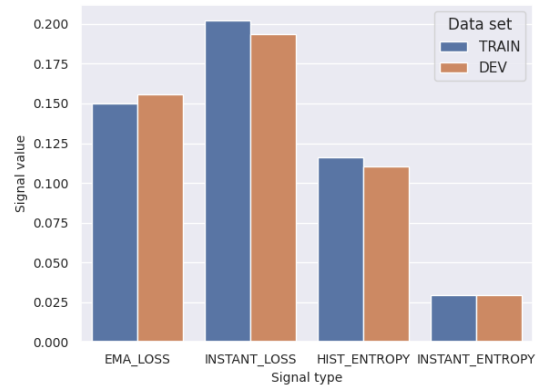

Figure 6: Mean of signals on hard level training and development set
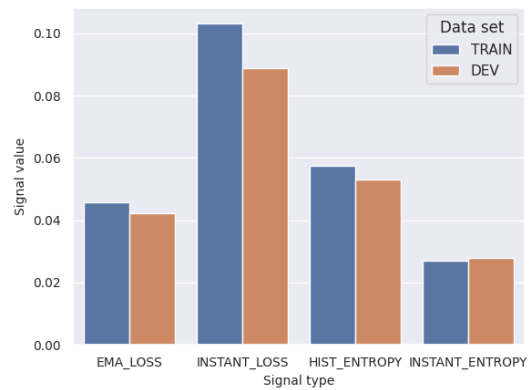

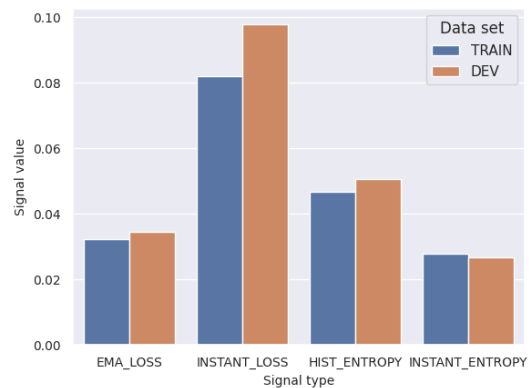Figure 7: Mean of signals on medium level training and development set

Figure 8: Mean of signals on easy level training and development set

5808