Open-Set Semi-Supervised Text Classification with Latent Outlier Softening

Junfan Chen
SKLSDE
Beihang University, Beijing, China
chenjf@act.buaa.edu.cn

Richong Zhang*
SKLSDE
Beihang University, Beijing, China
zhangrc@act.buaa.edu.cn

Junchi Chen
SKLSDE
Beihang University, Beijing, China
sy2206115@buaa.edu.cn

Chunming Hu
SKLSDE
Beihang University, Beijing, China
hucm@buaa.edu.cn

Yongyi Mao
School of EECS
University of Ottawa, Ottawa, Canada
ymao@uottawa.ca

ABSTRACT

Semi-supervised text classification (STC) has been extensively researched and reduces human annotation. However, existing research assuming that unlabeled data only contains in-distribution texts is unrealistic. This paper extends STC to a more practical Open-set Semi-supervised Text Classification (OSTC) setting, which assumes that the unlabeled data contains out-of-distribution (OOD) texts. The main challenge in OSTC is the false positive inference problem caused by inadvertently including OOD texts during training. To address the problem, we first develop baseline models using outlier detectors for hard OOD-data filtering in a pipeline procedure. Furthermore, we propose a Latent Outlier Softening (LOS) framework that integrates semi-supervised training and outlier detection within probabilistic latent variable modeling. LOS softens the OOD impacts by the Expectation-Maximization (EM) algorithm and weighted entropy maximization. Experiments on 3 created datasets show that LOS significantly outperforms baselines.

CCS CONCEPTS

• **Computing methodologies** → **Semi-supervised learning settings**.

KEYWORDS

semi-supervised learning, text classification, latent variable

ACM Reference Format:
Junfan Chen, Richong Zhang, Junchi Chen, Chunming Hu, and Yongyi Mao. 2023. Open-Set Semi-Supervised Text Classification with Latent Outlier Softening. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '23), August 6–10, 2023, Long Beach, CA, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3580305.3599456

*Corresbonding author: zhangrc@act.buaa.edu.cn

KDD '23, August 6–10, 2023, Long Beach, CA, USA

ACM ISBN 979-8-4007-0103-0/23/08…$15.00
https://doi.org/10.1145/3580305.3599456

1 INTRODUCTION

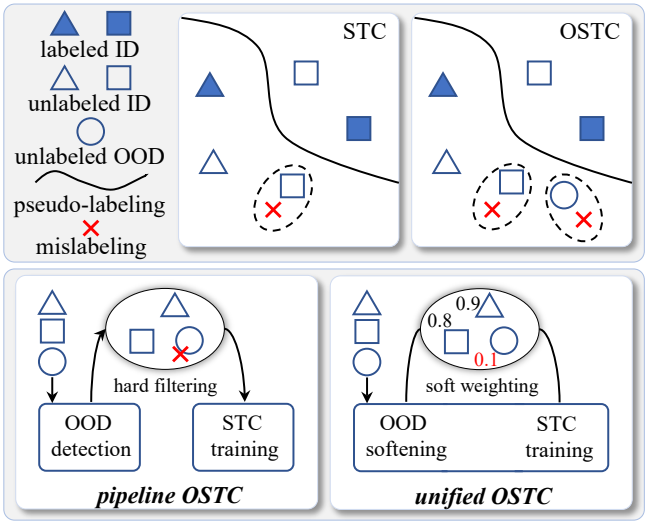


Figure 1: Illustration of the mislabelling problems in STC and OSTC and the pipeline and unified approaches for OSTC.

Training deep learning models requires annotating sufficient data. Thus, data-efficient learning has become a great demand for data-driven applications. Semi-supervised learning is a promising technique that substantially reduces human annotation. One of its attractive applications is Semi-supervised Text Classification (STC) [6, 13, 18, 19, 27, 32, 33] that utilizes a few labeled texts for training with additional unlabeled texts. Although existing STC models have achieved remarkable success, they make the impractical assumption that all unlabeled texts are in distribution (ID). To meet practical needs, we study a new task, Open-set Semi-supervised Text Classification (OSTC), which assumes the existence of OOD examples in the unlabeled-text set.

Semi-supervised learning relies on pseudo-label methods to assign labels for unlabeled texts. However, as shown in the upper of Figure 1, pseudo-labeling may produce mislabeled ID texts from the unlabeled-text set and degrade the model performance of STC. The mislabeling problem may become worse in OSTC because the OOD

Table 1: A snapshot of the evaluation results.

Model	In distribution Acc	In+out distribution Acc
UDA(ID)	71.67	71.50
UDA(ID+OOD)	72.68	28.53

examples in the unlabeled texts may also be mislabeled and worsen the training effect. A potential negative effect is *false positive inference*, a phenomenon that OOD texts in the test set are recognized as an ID class, leading to poor OSTC results. False positive inference may also make the OSTC model unsafe in real-world applications because it may assign wrong labels to OOD examples.

We first apply typical STC models (e.g., UDA and MixText) to OSTC. As the snapshot of results shown in Table 1, the STC model UDA trained on both ID and OOD examples (UDA (ID+OOD)) achieves similar in-distribution accuracy to that trained on only ID examples (UDA (ID)). However, the overall OSTC accuracy (In+out distribution Acc) of UDA (ID+OOD) significantly degrades from 71.50 to 28.53, caused by false positive inference. An intuitive understanding is: when training typical STC models on the OSTC dataset, as no specific modules or techniques are designed to recognize OOD examples in the unlabeled texts, the OOD examples therein would be treated as an ID-class example for training. As a result, the trained model is biased (or overfitting) to the ID class, making the OOD texts prone to being misclassified as an ID class during testing. Nevertheless, these STC models may serve as essential components for building OSTC models when combined with specific modules to deal with OOD examples, such as outlier detectors.

One practical solution to tackle the false positive inference problem is combining the STC model and outlier detector in a pipeline procedure, i.e., first utilizing the outlier detector to filter out OOD examples from the unlabeled-text set, then training the STC model using the remaining unlabeled texts. Open-set semi-supervised learning is a relatively new task, and just a few works explore the pipeline approach in image classification [15, 21, 25, 30, 37, 39]. To open the follow-up research on OSTC, we create baseline OSTC models in the pipeline approach. Specifically, we combine existing STC approaches (e.g., UDA [34] and MixText [6]) with outlier detection methods in text classification (e.g., MSP [14], DOC [31]).

The pipeline models are demonstrated to be effective in alleviating the false positive inference problem in OSTC compared with directly using pure STC models (see Table 5). However, as shown in the lower-left part of Figure 1, the pipeline approach relies on stepwise training and adopts hard text filtering using outlier detectors, which may encounter error-recognition when filtering OOD examples that may degrade the OSTC performance: (1) if an ID text is recognized as OOD, the model may filter out a useful unlabeled text from training; (2) otherwise, if an OOD text is recognized as ID, it may become noise data and harm training. For this reason, the pipeline approach can be improved by properly dealing with the potential error-recognization of either ID or OOD texts.

To tackle the problems in the pipeline approach, we further propose a novel unified OSTC framework, i.e., Latent Outlier Softening (LOS). As shown in the lower-right part of Figure 1, our idea is simultaneously training an STC model and softening the OOD impacts by adaptive weights within a unified training process. Specifically, we formulate the OSTC problem with latent variable modeling and derive its Expectation-Maximization solution based on a *conditional independence assumption* and an *entropy assumption*. In LOS, the estimations of posterior probabilities naturally perform as weights assigned to each example to balance their training objectives. An entropy maximization objective is specified to soften the negative impact of OOD texts. Through LOS, we can train the OSTC model in a unified procedure and prevent the negative effect caused by error-recognizing ID and OOD texts, thus further mitigating the false positive inference problem in OSTC.

In a nutshell, our work makes the following contributions. (1) We introduce a new task, Open-set Semi-supervised Text Classification (OSTC), shed light on the main challenges and contribute baseline models by combining existing STC models and outlier detection approaches. (2) We propose a novel Latent Outlier Softening (LOS) framework with latent variable modeling. To our knowledge, LOS is the first unified training framework to adopt OOD softening for open-set semi-supervised text classification. (3) We create 3 benchmarks for OSTC. Empirical results and analysis demonstrate that LOS is effective and significantly outperforms baselines.

2 RELATED WORKS

Semi-supervised text classification (STC) tries to mitigate human annotation by training a model with a few labeled texts and plenty of unlabeled texts. One branch of recent works utilizes regularization techniques or consistency training in STC [13, 20, 24, 28, 35]. For example, UDA [34] propose substituting noising operations with data augmentations and combining them with consistency training utilizing both labeled and unlabeled data. MixText proposes a new text data augmentation method based on Mixup and leverages the augmented text data to boost consistency training [6]. Other works leverage pseudo-labeling to annotate unlabeled texts as additional training data [18–20, 22, 27, 32]. For example, SALNet [19] constructs lexicons based on attention weights and leverages the lexicons to bootstrap semi-supervised training. Despite the success, the STC models may encounter degradation in the more practical OSTC setting. We reveal the challenges in OSTC and propose a better approach to combine STC models with outlier detection.

Open-set semi-supervised learning is a practical and understudied task. It has been explored in image classification. Most approaches adopt the pipeline procedure that first filters out OOD examples and then performs semi-supervised training. Typical solutions include designing special outlier detectors [15, 25, 30] and exploring novel optimization frameworks [21, 37, 39]. For example, The work of [30] performs a pretext task to obtain a high-level semantic understanding and proposes a cross-modal matching strategy to detect OOD samples for open-set semi-supervised learning. The work of [15] leverages an offline OOD detector based on a self-supervised vision transformer performs to tackle the open-set semi-supervised learning problem. The work of [39] proposes a joint optimization framework. To detect the OOD samples in unlabeled data, it estimates the probability of the sample belonging to OOD. Then it updates the network parameters and the OOD score alternately. We follow the pipeline approach to build baseline OSTC models and propose LOF with latent variable modeling.

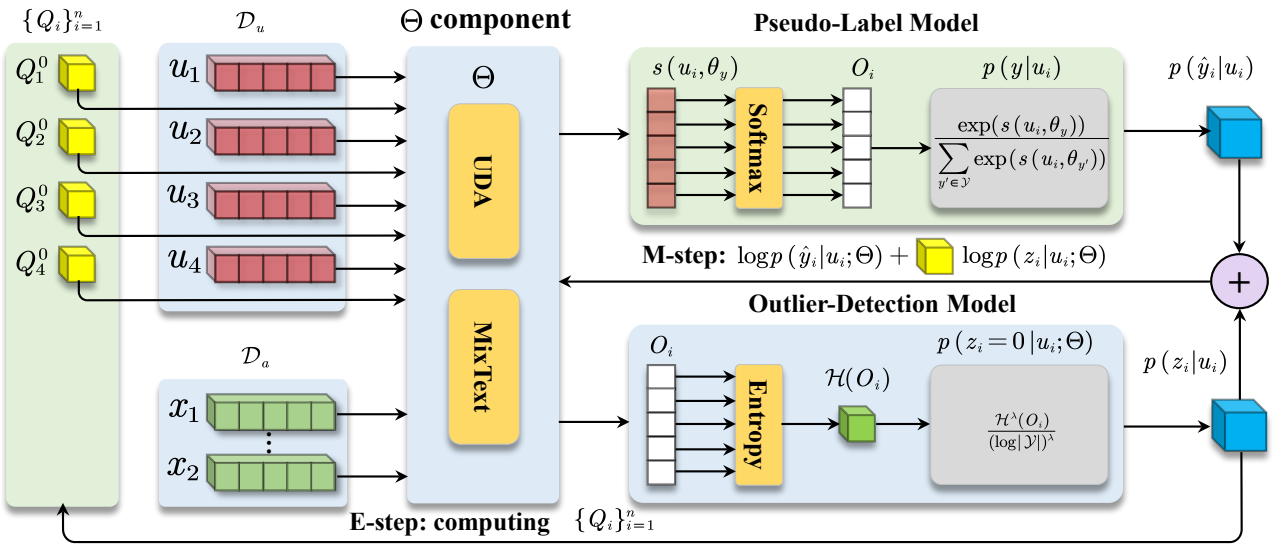


Figure 2: The structure of our LOS framework.

Latent variable models have been intensively exploited in various NLP applications. For example, in text generation and dialogues, existing models adopt the latent variable models to build hidden topic-based models to generate topic-related texts or utilize latent variable models to improve the diversity of generated sequences [2, 3, 9, 38]. In text modeling and classification [16, 17], latent variable models are combined with modeling the topics of documents or boosting the text classification model by utilizing the latent topics. In sequence matching and translation [4, 10], latent variable models are used to build the dependencies between different time steps in the sequences. The expectation-Maximization algorithm has recently been applied to train deep neural networks [1, 7, 8, 12]. In this work, we leverage latent variable modeling to build a unified OSTC framework optimized with the expectation-Maximization algorithm, which enables softening the impacts of OOD examples.

3 PRELIMINARY

3.1 Problem Statement

The open-set semi-supervised text classification (OSTC) task attempts to learn practical text classification models from a few labeled texts and plenty of unlabeled texts, which involve both in-distribution and out-of-distribution text examples. Formally, let $\mathcal{Y}$ be the set of in-distribution classes, in which each class $y \in \mathcal{Y}$ denotes a known category. The training set of OSTC consists of a labeled text set $\mathcal{D}_a$ and an unlabeled text set $\mathcal{D}_u$. Specifically, we denote the labeled-text set as $\mathcal{D}_a = \{(x_i, y_i)\}_{i=1}^{l}$, where x_i, y_i respectively represent the i^{th} text and its corresponding label. We assume that each class contains k labeled texts. The unlabeled-text set is denoted as $\mathcal{D}_u = \{u_i\}_{i=1}^{n}$, where u_i denotes the i^{th} unlabeled text. Each u_i may come from a known ID class or an unknown OOD class not belonging to $\mathcal{Y}$. During testing, the OSTC task aims to identify whether a given text is an ID or OOD example and predict the exact class type if it is an ID example.

3.2 Baseline Models for OSTC

As open-set semi-supervised learning has not been studied in text classification, we follow the pipeline approach in open-set semi-supervised image classification [15, 21, 25, 30, 37, 39] to combine the semi-supervised learning model with outlier detectors. Specifically, we train a pipeline OSTC model in two steps: (1) first, train existing outlier detectors in text classification on the labeled-text set $\mathcal{D}_a$ and leverage them to filter out OOD examples in the unlabeled-text set $\mathcal{D}_u$; (2) then, use the remaining unlabeled-text set to train the STC model. This approach reduces the negative impact of the OOD texts by excluding OOD examples by hard filtering. For outlier detection, we introduce the recent outlier detector used in text classification, including MSP [14], DOC [31], LMCL [23] and Softmax [36]. Meanwhile, we select the recent UDA [34] and MixText [6] as the STC models in the pipeline approach, which leverage BERT and consistency training with data augmentation techniques. We name these pipeline models combining the outlier detectors and STC models with a "+" symbol, e.g., UDA+MSP, UDA+DOC, etc.

4 LATENT OUTLIER SOFTENING

Although the pipeline approach for OSTC is simple and has shown to be effective, its hard OOD filtering strategy may cause error recognition of ID or OOD texts that harm the OSTC training. To tackle this problem, we propose a Latent Outlier Softening (LOS) framework shown in Figure 2 to replace hard OOD filtering with OOD softening that adaptively assigns soft weights to objectives of text examples, where the lower weights resort to weakening the negative impact of OOD texts. LOS comprises a Θ component which takes the encoder of UDA or Mixtext for text representation, a pseudo-label model that aims to assign pseudo labels for unlabeled texts, and an outlier-detection model used to identify OOD examples. LOS is naturally derived from latent variable modeling, which we will discuss in detail.

4.1 Latent Variable Model for OSTC

4.1.1 Variable Definition. We need two variables to completely describe the label of each unlabeled text u_i: (1) a variable indicating whether u_i belongs to an ID or OOD example; (2) a variable indicating which class in $\mathcal{Y}$ the text u_i belongs to. Technically, both variables are latent, making it challenging to build practical latent variable models. Fortunately, the pseudo-labeling approach assigns a pseudo-label $\hat{y}_i \in \mathcal{Y}$ for each u_i. Thus, when combining the latent variable modeling with the pseudo-labeling approach, we may use $\hat{y}_i$ as the variable indicating the ID class that u_i belongs to and treat this variable as an observed one. Then, we only need one latent variable indicating whether u_i belongs to an ID or OOD example. Here, we define this latent variable as z_i, i.e., if text u_i is an ID example, $z_i = 1$, else if text u_i is an OOD example, $z_i = 0$.

4.1.2 Probabilistic Modeling with Latent Variables. To utilize the unlabeled texts in the training set of OSTC through the pseudo-labeling approach, we need first assign a pseudo label $\hat{y}_i$ for each unlabeled text u_i based on the current model or pre-trained pseudo-label estimators. Let $\hat{\mathcal{D}}_u = \{(u_i, \hat{y}_i)\}_{i=1}^n$ denote the pseudo-labeled text set, we can then leverage the pseudo-labeled data $\hat{\mathcal{D}}_u$ to further train the OSTC model. Specifically, we can optimize the model by sending the pseudo-labeled text set $\hat{\mathcal{D}}_u$ to the text classifier and maximizing the following log-likelihood

$$\arg\max_{\Theta} \ell(\Theta) = \arg\max_{\Theta} \sum_{i=1}^{n} \log p(\hat{y}_i|u_i;\Theta) \tag{1}$$

By introducing a latent variable z_i, we can rewrite the above log-likelihood as follows

$$arg\max_{\Theta} \ell(\Theta, z) = arg\max_{\Theta} \sum_{i=1}^{n} \log \sum_{z_i \in \{0,1\}} p(\hat{y}_i, z_i|u_i;\Theta) \tag{2}$$

Here the latent variable z_i denotes whether text u_i is an ID or OOD example and assumes that both $\hat{y}_i$ and z_i depend on input text u_i. We make no assumption about the dependencies between $\hat{y}_i$ and z_i for now and will discuss them in a later section.

The above computation requires summation over exponentially many configurations of z_i, making the optimization problem in Equation (2) intractable. Fortunately, this optimization problem can be elegantly solved by maximizing a lower bound. Specifically, we define a new distribution $Q_i(z_i)$, which satisfies

$$\sum_{z_i \in \{0,1\}} Q_i(z_i) = 1, 0 \leqslant Q_i(z_i) \leqslant 1 \tag{3}$$

By introducing $Q_i(z_i)$ and following Jensen's Inequality, we can derive a lower bound of the log-likelihood in Equation (2) as follows

$$\begin{aligned} \ell(\Theta, z) &= \sum_{i=1}^{n} \log \sum_{z_i \in \{0,1\}} p(\hat{y}_i, z_i|u_i;\Theta) \\ &= \sum_{i=1}^{n} \log \sum_{z_i \in \{0,1\}} Q_i(z_i) \frac{p(\hat{y}_i, z_i|u_i;\Theta)}{Q_i(z_i)} \\ &\geqslant \sum_{i=1}^{n} \sum_{z_i \in \{0,1\}} Q_i(z_i) \log \frac{p(\hat{y}_i, z_i|u_i;\Theta)}{Q_i(z_i)} = \mathcal{L}(\Theta, \{Q_i\}_{i=1}^n) \end{aligned} \tag{4}$$

where we introduce a Q_i for each $i = 1, 2, \cdots, n$, representing the values of $Q_i(z_i)$ for $z_i = 0$ and $z_i = 1$.

4.1.3 Optimization with Expectation-Maximization Algorithm. The derivation in Equation (4) given a lower bound $\mathcal{L}(\Theta, \{Q_i\}_{i=1}^n)$ of $\ell(\Theta, z)$. Maximizing the lower bound will simultaneously maximize the original log-likelihood, which allows us to replace optimizing the intractable log-likelihood $\ell(\Theta, z)$ to optimizing the easier lower bound $\mathcal{L}(\Theta, \{Q_i\}_{i=1}^n)$. Optimizing the lower bound requires simultaneously finding optimal parameters Θ and $\{Q_i(z_i)\}_{i=1}^n$

$$(\Theta^*, \{Q_i^*\}_{i=1}^n) = \arg\max_{\Theta, \{Q_i\}_{i=1}^n} \mathcal{L}(\Theta, \{Q_i\}_{i=1}^n) \tag{5}$$

An effective training strategy for optimizing this lower bound is the Expectation-Maximization (EM) algorithm, which allows us to iteratively optimize Θ and $\{Q_i(z_i)\}_{i=1}^n$. The EM algorithm solves the optimization problem in Equation (4) adopting the coordinate ascent algorithm on the objective function $\mathcal{L}(\Theta, \{Q_i\}_{i=1}^n)$, where we iterate over two steps, i.e., the expectation step (E-step) and the maximization step (M-step). In the E-step, we maximizes $\mathcal{L}(\Theta, \{Q_i\}_{i=1}^n)$ over $\{Q_i\}_{i=1}^n$ for the current setting of parameter Θ and in the M-step, we maximizes $\mathcal{L}(\Theta, \{Q_i\}_{i=1}^n)$ for the current $\{Q_i\}_{i=1}^n$. We next describe the two steps in detail, where we will use superscript $'$ to denote the current settings of the parameters.
E-Step: In this step, we keep the parameters Θ fixed and compute $\{Q_i\}_{i=1}^n$ to maximize the lower bound $\mathcal{L}(\Theta, \{Q_i\}_{i=1}^n)$ according to the current parameters Θ' as follows:

$$Q_i(z_i) := p(z_i|u_i, \hat{y}_i; \Theta^{'}) \tag{6}$$

For simplicity, we may denote the values of Q_i corresponding to $z_i = 0$ and $z_i = 1$ as Q_i^0 and Q_i^1, respectively.
M-Step: In this step, we hold $\{Q_i\}_{i=1}^n$ and update Θ to maximize $\mathcal{L}(\Theta, \{Q_i\}_{i=1}^n)$ according to the current $\{Q_i'\}_{i=1}^n$ as follows:

$$\begin{aligned} \Theta &:= \arg\max_{\Theta} \sum_{i=1}^{n} \sum_{z_i \in \{0,1\}} Q_i(z_i) \log p(\hat{y}_i, z_i|u_i;\Theta) - Q_i(z_i) \log Q_i(z_i) \\ &= \arg\max_{\Theta} \sum_{i=1}^{n} \sum_{z_i \in \{0,1\}} Q_i(z_i) \log p(\hat{y}_i, z_i|u_i;\Theta) \end{aligned} \tag{7}$$

where the subtrahend term is omitted because it serves as a constant. During training, the EM algorithm starts with randomly initialized parameters or some guesses of $(\Theta, \{Q_i\}_{i=1}^n)$, and then iterates over the E-step and M-step until convergence.

4.2 Latent Outlier Softening Framework

4.2.1 Conditional Independence Assumption for OSTC. To implement the above latent variable model with the EM algorithm, we need to specify the posterior distribution $p(z_i|u_i, \hat{y}_i; \Theta)$ required for computing $\{Q_i\}_{i=1}^n$ and the joint distribution $p(\hat{y}_i, z_i|u_i; \Theta)$ for M-step updating. The solutions to implement the required distributions depend on the assumption made about the dependencies between $\hat{y}_i$ and z_i, which determines how we deal with $p(z_i|u_i, \hat{y}_i; \Theta)$ and $p(\hat{y}_i, z_i|u_i; \Theta)$. We make the following assumption:

Assumption 1. **Conditional Independence Assumption.** *Variable z_i is conditionally independent with variable y_i, given input u_i and the model parameters Θ. Namely, the posterior distribution $p(z_i|u_i, \hat{y}_i; \Theta)$ has a simplified form*

$$p(z_i|u_i, \hat{y}_i; \Theta) = p(z_i|u_i; \Theta) \tag{8}$$

This assumption is practical in OSTC because the fact that whether an input text u_i is an ID or an OOD example depends on the text itself and has nothing to do with its pseudo label $\hat{y}_i$.

Based on the conditional independence in Equation (8), we have

$$p(\hat{y}_i|u_i, z_i; \Theta) = \frac{p(\hat{y}_i|u_i;\Theta)p(z_i|u_i,\hat{y}_i;\Theta)}{p(z_i|u_i;\Theta)} = p(\hat{y}_i|u_i;\Theta) \quad (9)$$

According to the above model simplification, we rewrite the E-step in Equation (6) as follows

$$Q_i(z_i) := p(z_i|u_i, \hat{y}_i; \Theta^{'}) = p(z_i|u_i;\Theta^{'}) \quad (10)$$

And the M-step in Equation (7) is rewritten as

$$\begin{aligned}
\Theta &:= \arg\max_{\Theta} \mathcal{L}(\Theta, \{Q_i'\}_{i=1}^n) \\
&= \arg\max_{\Theta} \sum_{i=1}^{n} \sum_{z_i\in\{0,1\}} Q_i(z_i) \log[p(\hat{y}_i, z_i|u_i;\Theta)p(z_i|u_i;\Theta)] \\
&= \arg\max_{\Theta} \sum_{i=1}^{n} \sum_{z_i\in\{0,1\}} Q_i(z_i) [\log p(\hat{y}_i|u_i;\Theta) + \log p(z_i|u_i;\Theta)] \\
&= \arg\max_{\Theta} \sum_{i=1}^{n} \log p(\hat{y}_i|u_i;\Theta) + \sum_{i=1}^{n} \sum_{z_i\in\{0,1\}} Q_i(z_i) \log p(z_i|u_i;\Theta)
\end{aligned} \quad (11)$$

The conditional independence assumption between variable $\hat{y}_i$ and z_i allows us to simplify the posterior probability distribution $p(z_i|u_i, \hat{y}_i; \Theta)$ and specify the joint probability distribution $p(\hat{y}_i, z_i|u_i; \Theta)$. Hereafter, the implementation of the E-step and M-step only requires to model distribution $p(\hat{y}_i|u_i; \Theta)$ and $p(z_i|u_i; \Theta)$. We name the former and latter distributions *pseudo-label model* and *outlier-detection model*, as they respectively formulate the probabilities of pseudo label $\hat{y}_i$ and OOD indicator z_i, given the input u_i. We next discuss the modeling of the two distributions in detail.

4.2.2 BERT-Based Pseudo-Label Model. The pseudo-label model $p(\hat{y}_i|u_i; \Theta)$ serves as a fundamental component for OSTC, which requires to equip with an encoder for text representation and a classifier to predict labels for input texts u_i. It is typically trained by feeding the labeled or pseudo-labeled texts and leveraged to pseudo-label the unlabeled texts. In addition, the pseudo-label model is also used to predict the ID-class label for a given text during testing.

Recent STC approaches, e.g., UDA [34] and MixText [6], utilize the pre-trained language model BERT [11] to build their pseudo-label models and are demonstrated to be effective. Following these works, we also adopt BERT to build the pseudo-label model expressed by $p(\hat{y}_i|u_i; \Theta)$. Formally, for an input text u_i, let $s(u_i, \theta_y)$ denote the output score from the BERT encoder corresponding to class $y \in \mathcal{Y}$, where θ_y denotes the parameters in BERT corresponding to class y. We introduce a vector $\mathbf{O}_i \in \mathbb{R}^{|\mathcal{Y}|}$ to denote the output normalized by the softmax function. Then, the y-th element of $\mathbf{O}_i$, denoted as $\mathbf{O}_{iy}$, which represents the the probability of u_i belonging to each class y, is computed by

$$\mathbf{O}_{iy} = p(y|u_i) = \frac{\exp(s(u_i, \theta_y))}{\sum_{y'\in\mathcal{Y}} \exp(s(u_i, \theta_{y'}))} \quad (12)$$

We denote the set of parameters $\{\theta_y\}$ for all $y \in \mathcal{Y}$ as Θ. The above normalized output $\mathbf{O}_i$ essentially express distribution $p(\hat{y}_i|u_i; \Theta)$. For later use, we call the BERT followed by the softmax function the Θ component, which will be reused in $p(z_i|u_i; \Theta)$ model.

4.2.3 Entropy-Driven Outlier-Detection Model. We build a shared component with parameters Θ for the two distributions $p(\hat{y}_i|u_i; \Theta)$ and $p(z_i|u_i; \Theta)$, in order to be consistent with the derivation of our latent variable model. Our investigation suggests: when u_i is an OOD example, the Θ component appears uncertain about which $y \in \mathcal{Y}$ the text u_i belongs to, i.e., the component tends to produce similar values of $\mathbf{O}_{iy}$ for all $y \in \mathcal{Y}$, resulting in evenly distributed $\mathbf{O}_i$; otherwise, when u_i is an ID example, the Θ component becomes more confident about which y the text belongs to, i.e., the Θ component tends to output much higher values of $\mathbf{O}_{iy}$ for some $y \in \mathcal{Y}$ than the others, resulting in sharply distributed $\mathbf{O}_i$. In summary, the evenness of distribution $\mathbf{O}_i$ output from the Θ component implies the probability of u_i belonging to an OOD text.

The above analysis inspires us to approximate model $p(z_i|u_i; \Theta)$ according to the evenness of $\mathbf{O}_i$. To achieve this goal, we need a measurement for the evenness of distribution. Fortunately, entropy is an elegant tool that can be used to measure the evenness of a distribution. A higher entropy indicates a more even distribution. Given a output $\mathbf{O}_i$, we compute its entropy as follows

$$\mathcal{H}(\mathbf{O}_i) = -\sum_{y\in\mathcal{Y}} \mathbf{O}_{iy} \log \mathbf{O}_{iy} \quad (13)$$

The entropy function in Equation (13) has a lower bound and an upper bound, i.e., $0 \leqslant \mathcal{H}(\mathbf{O}_i) \leqslant \log|\mathcal{Y}|$. To properly modeling $p(z_i|u_i; \Theta)$, we make the following assumption

Assumption 2. **Entropy Assumption.** *For two different input texts u_i and u_i', suppose their corresponding output from the Θ component are respectively $\mathbf{O}_i$ and $\mathbf{O}_i'$. Then, we make the following assumption: (1) If $\mathcal{H}(\mathbf{O}_i) > \mathcal{H}(\mathbf{O}_i')$, then $p(z_i = 1|u_i; \Theta) < p(z_i = 1|u_i'; \Theta)$ and $p(z_i = 0|u_i; \Theta) > p(z_i = 0|u_i'; \Theta)$. (2) When $\mathcal{H}(\mathbf{O}_i) \to 0$, then $p(z_i = 1|u_i; \Theta) \to 1$ and $p(z_i = 0|u_i; \Theta) \to 0$. (3) When $\mathcal{H}(\mathbf{O}_i) \to \log|\mathcal{Y}|$, then $p(z_i = 1|u_i; \Theta) \to 0$ and $p(z_i = 0|u_i; \Theta) \to 1$.*

This assumption indicates that an input u_i holding higher entropy $\mathcal{H}(\mathbf{O}_i)$ is more likely an OOD than an ID example, and vice versa. Based on Assumption 2 and the fact that the entropy $\mathcal{H}(\mathbf{O}_i)$ can be easily normalized by the min-max normalization approach, we can build the outlier-detection model $p(z_i|u_i; \Theta)$ with the *normalized entropy* function, which is defined as follows

$$\begin{aligned}
p(z_i = 0|u_i; \Theta) &= \left(\frac{\mathcal{H}(\mathbf{O}_i)}{\log|\mathcal{Y}|}\right)^{\lambda} = \frac{\mathcal{H}^{\lambda}(\mathbf{O}_i)}{(\log|\mathcal{Y}|)^{\lambda}} \\
p(z_i = 1|u_i; \Theta) &= 1 - \frac{\mathcal{H}^{\lambda}(\mathbf{O}_i)}{(\log|\mathcal{Y}|)^{\lambda}}
\end{aligned} \quad (14)$$

where $\lambda > 0$ serves as a hyper-parameter to control the *rate of change* of the normalized entropy function, i.e., how much the probability $p(z_i = 0|u_i; \Theta)$ changes with the changing of $\mathcal{H}(\mathbf{O}_i)$. To understand how λ affects the values of $p(z_i = 0|u_i; \Theta)$, we make an analysis on its following differentiation function

$$\frac{\mathrm{d}}{\mathrm{d}\mathcal{H}} \left(\frac{\mathcal{H}^{\lambda}(\mathbf{O}_i)}{(\log|\mathcal{Y}|)^{\lambda}}\right) = \left(\frac{\lambda}{(\log|\mathcal{Y}|)^{\lambda}}\right) \mathcal{H}^{\lambda-1}(\mathbf{O}_i) \quad (15)$$

Based on the differentiation (15), we have the following conclusion:

Conclusion 1. *(1) If $\lambda = 1$, then $p(z_i = 0|u_i; \Theta)$ increases linearly as $\mathcal{H}(\mathbf{O}_i)$ increases, with a rate $1/\log|\mathcal{Y}|$. (2) If $\lambda < 1$, as*

$\mathcal{H}^{\lambda-1}(\mathbf{O}_i)$ monotonically decease in $[0, \log|\mathcal{Y}|]$, then as $\mathcal{H}(\mathbf{O}_i)$ increases, $p(z_i = 0|u_i;\Theta)$ increases with decreased difference. (3) If $\lambda > 1$, as $\mathcal{H}^{\lambda-1}(\mathbf{O}_i)$ monotonically increases in $[0, \log|\mathcal{Y}|]$, then as $\mathcal{H}(\mathbf{O}_i)$ increases, $p(z_i = 0|u_i;\Theta)$ increases with increased difference.

Conclusion 1 motivates us: if we require the value of $p(z_i = 0|u_i;\Theta)$ increasing slowly with $\mathcal{H}(\mathbf{O}_i)$ increasing, or we expect the $p(z_i|u_i;\Theta)$ model to be more confidence in predicting ID, we may set a lower $\lambda < 1$; otherwise, if we require $p(z_i = 0|u_i;\Theta)$ increasing quickly, or we want the $p(z_i|u_i;\Theta)$ model to be more confidence on OOD, we can set a higher $\lambda > 1$. The hyper-parameter λ is essential in our model. It guarantees our model's performance by the availability to find the optimal λ, when applying our model to datasets with different class sizes $\mathcal{Y}$ or of different domains. Now we finish discussing the modeling of $p(\hat{y}_i|u_i;\Theta)$ and $p(z_i|u_i;\Theta)$ required to implement the EM training of the latent variable model.

4.3 Optimization Process and Interpretation

4.3.1 The Optimization of LOS. Training EM algorithm with neural networks is challenging because it relies on two requirements: (1) the proper initialization of $\{Q_i\}_{i=1}^n$ and parameters Θ; (2) the balance between E-step and M-step training. We design the following training process that allows us to efficiently optimize our LOS framework: at a start, we assume that an unlabeled text has equal probabilities of being an ID and OOD example, i.e., we initialize each $Q_i \in \{Q_i\}_{i=1}^n$ to $(0.5, 0.5)$ and start training at M-step then E-step, which avoids us to struggle to find a pretty good initialization of Θ. We execute one E-step computing after m SGD updates at M-step to guarantee that Θ is sufficiently updated at each M-step. The detailed training process of LOS is shown in Algorithm 1.

Algorithm 1 The Training Process of the LOS Framework

Input: the unlabeled-text set $\mathcal{D}_u$ and the Θ component
Output: the optimized values of $\{Q_i\}_{i=1}^n$ and parameters Θ

initialize each Q_i with $(0.5, 0.5)$, randomly initialize Θ;
repeat
  load values of $\{Q'_i\}_{i=1}^n$, relax Θ and update Θ by M-step:
  for $j \leftarrow 1$ **to** m **do**
    sample a batch of unlabeled texts $\mathcal{B} = \{u\}$ from $\mathcal{D}_u$;
    obtain $\hat{\mathcal{B}} = \{(u, \hat{y})\}$ by pseudo-labeling with Eq. (12);
    update Θ by $\max \mathcal{L}(\Theta, \{Q'_i\}_{i=1}^n)$ with $\{Q'_i\}_{i=1}^n$ and $\hat{\mathcal{B}}$;
  end for
  fix Θ as Θ' and compute $\{Q_i\}_{i=1}^n$ by E-step:
  for $i \leftarrow 1$ **to** n **do**
    obtain $\mathbf{O}'_i$ of u_i from the Θ component with Θ';
    compute Q_i by the function in Eq. (14) with $\mathbf{O}'_i$;
  end for
  store the values $\{Q_i\}_{i=1}^n$ as $\{Q'_i\}_{i=1}^n$;
until convergence

4.3.2 The Interpretation of LOS. Our LOS framework is probabilistically interpreted. By introducing the *conditional independence* (Assumption 1) and *entropy* assumption (Assumption 2), we derive the EM solution in Equation (5), whose M-step in Equation (11) elegantly integrates the pseudo-label model and outlier-detection model, eliminating the reliance on extra outlier detectors in the traditional pipeline framework. According to Equations (10) and (11), at the M-step, the values of Q_i in the second term of Equation (11) serve as soft weights attached to the objectives of the outlier detection model. Particularly, when $Q_i(z_i = 0)$ is larger than $Q_i(z_i = 1)$, the M-step training tends to maximize the entropy of the output of Θ component. This maximization softens the negative impact of the OOD examples and relieves the false positive inference problem. Furthermore, at the E-steps, LOS iteratively refines the weights $\{Q_i\}_{i=1}^n$ for all unlabeled texts. In addition, based on Equation (11), the objective of our LOS framework could be viewed as maximizing the log-likelihood of the pseudo-label model with an entropy-based regularizer. Therefore, our LOS framework can be combined with any pseudo-label models. In this work, we combine LOS with UDA and MixText and name the new OSTC models **UDA+LOS** and **MixText+LOS**, respectively.

5 EXPERIMENT

5.1 Datasets and Experiment Setting

5.1.1 Datasets. We create 3 benchmarks from existing text classification datasets to evaluate OSTC: AGNews, DBPedia and Yahoo.
AGNews [29] is extrated from the AG news corpus and is processed by compiling the titles and descriptions of articles from the 4 categories, including *World*, *Sports*, *Business* and *Sci/Tech*.
DBPedia [26] is extracted from from Wikipedia and is commonly used in query understanding. This dataset contains 14 classes for text classification, including *Company*, *Educational Institution*, etc.
Yahoo [5] is a widely used question classification dataset. The question/answer pairs of Yahoo are extracted from the Yahoo! Answers website. This dataset contains 10 top-level categories, which includes *Society & Culture*,*Health*,*Education & Reference*, etc.

Table 2: The statistics of the datasets. *# lab*, *#unl.*, *#val* and *#test* respectively denote the labelled, unlabeled, validation and test texts for each class. And *#ID ($|\mathcal{Y}|$)*, *#OOD* denote the number of ID classes and OOD classes, respectively.

Dataset	#lab.	#unl.	#val.	#test	#ID ($\lvert\mathcal{Y}\rvert$)	#OOD
AGNews	10/50/100	5000	2000	1900	2	2
Yahoo	10/50/100	5000	2000	6000	6	4
DBPedia	10/50/100	5000	2000	5000	8	6

5.1.2 Dataset Construction and Experimental Settings. To guarantee the fairness of OSTC dataset construction, we randomly split the classes into an ID and an OOD class set and split the training sets to $k = 10$, 50 and 100 labeled texts and 5000 unlabelled texts for each ID class and each OOD class. We split 2000 texts for each ID and OOD class to construct the validation set and keep the original test set for evaluation. The dataset statistics are shown in Table 2. We introduce 4 metrics to evaluate the OSTC models. Specifically, we use *Acc* and *F1* to denote the overall accuracy and F1, which evaluates a model's performance in both ID classification and OOD detection. To easily analyze the results, we also introduce metrics *In* and *Out* to denote the ID accuracy on only ID examples and outlier-detection accuracy on all examples, respectively.

Table 3: The open-set semi-supervised text classification results on AGNews, Yahoo and DBPedia.

Method	AGNews						Yahoo						DBPedia					
	$k=10$		$k=50$		$k=100$		$k=10$		$k=50$		$k=100$		$k=10$		$k=50$		$k=100$	
	Acc	F1	Acc	F1	Acc	F1	Acc	F1	Acc	F1	Acc	F1	Acc	F1	Acc	F1	Acc	F1
UDA+MSP	41.10	37.56	48.53	35.98	59.92	57.50	49.42	11.03	59.91	40.01	60.53	52.52	51.09	10.65	71.57	77.01	76.77	80.87
	(0.19)	(0.84)	(0.45)	(2.04)	(4.71)	(5.12)	(0.08)	(1.32)	(0.03)	(0.91)	(0.06)	(0.05)	(0.64)	(1.68)	(0.49)	(0.62)	(0.39)	(0.07)
UDA+LSoftmax	38.77	35.24	39.13	39.12	51.96	41.79	32.12	37.75	38.87	45.42	38.10	47.45	48.71	65.08	53.13	64.88	57.33	64.74
	(2.97)	(1.61)	(0.52)	(0.56)	(0.02)	(0.89)	(1.02)	(0.20)	(1.17)	(0.64)	(0.17)	(0.67)	(0.09)	(1.29)	(0.36)	(0.36)	(0.20)	(0.39)
UDA+DOC	35.22	31.94	49.36	37.20	57.94	55.48	49.79	12.03	50.87	12.68	57.25	31.83	49.89	8.42	56.27	28.29	78.63	82.23
	(0.42)	(0.93)	(0.73)	(2.41)	(3.55)	(1.76)	(0.22)	(2.43)	(0.29)	(1.08)	(0.29)	(1.65)	(0.02)	(0.55)	(1.98)	(5.26)	(0.80)	(0.48)
UDA+LMCL	43.44	28.45	45.10	43.93	54.65	35.22	49.11	12.14	53.41	35.68	57.14	41.05	50.29	64.75	65.67	61.62	67.40	63.01
	(7.55)	(0.28)	(0.64)	(0.74)	(0.12)	(0.58)	(0.35)	(0.06)	(1.54)	(6.69)	(0.20)	(0.39)	(0.74)	(0.61)	(0.15)	(2.27)	(0.89)	(0.55)
MixText+MSP	37.68	32.59	50.99	46.67	59.29	57.83	50.74	13.44	57.57	55.07	58.67	48.81	56.64	40.09	71.39	77.54	75.04	80.04
	(6.75)	(1.89)	(1.04)	(9.26)	(4.39)	(2.44)	(0.26)	(1.31)	(0.26)	(1.44)	(1.89)	(4.51)	(0.83)	(1.90)	(3.44)	(1.73)	(2.90)	(1.20)
MixText+LSoftmax	37.19	37.84	41.01	41.02	46.42	42.41	38.36	43.06	38.47	47.10	39.02	46.46	47.64	61.28	48.84	61.95	51.25	63.23
	(0.96)	(0.79)	(0.38)	(1.10)	(2.08)	(3.37)	(0.22)	(1.01)	(0.55)	(0.69)	(0.14)	(0.43)	(0.08)	(0.45)	(1.26)	(1.17)	(0.83)	(1.10)
MixText+DOC	37.95	38.32	49.67	51.67	51.88	53.42	51.02	11.47	58.39	43.99	59.78	53.59	49.75	32.14	72.40	77.95	77.48	82.57
	(0.91)	(0.99)	(1.40)	(1.71)	(1.03)	(0.76)	(1.43)	(2.70)	(1.04)	(0.63)	(0.69)	(0.28)	(0.15)	(4.79)	(7.13)	(3.66)	(0.35)	(0.02)
MixText+LMCL	35.13	31.27	42.99	39.56	43.44	40.29	49.49	11.66	51.50	42.53	56.03	40.82	48.21	64.36	65.46	62.97	67.14	62.20
	(0.39)	(0.45)	(0.25)	(1.11)	(0.23)	(0.63)	(0.81)	(2.95)	(2.06)	(0.10)	(0.48)	(6.51)	(0.10)	(0.13)	(0.03)	(0.29)	(0.79)	(0.92)
UDA+LOS	**64.30**	**51.03**	68.11	**64.11**	**83.16**	**79.20**	53.15	47.77	66.16	**63.96**	68.56	64.49	**78.84**	**80.08**	**84.38**	**86.51**	79.44	81.84
	(0.84)	(6.24)	(3.48)	(3.10)	(1.72)	(1.37)	(1.00)	(3.57)	(1.65)	(1.01)	(2.38)	(1.67)	(3.99)	(3.55)	(1.57)	(0.65)	(3.24)	(3.41)
MixText+LOS	58.48	29.91	**71.43**	58.56	79.16	73.29	**65.93**	**51.51**	**67.82**	62.34	**68.57**	**66.24**	74.65	75.88	80.76	83.77	**79.59**	**82.54**
	(3.45)	(1.11)	(3.30)	(6.35)	(2.82)	(4.31)	(1.76)	(4.65)	(0.91)	(2.56)	(0.75)	(0.45)	(2.71)	(3.62)	(6.13)	(3.00)	(5.63)	(1.55)

5.2 Implementation and Baseline Models

5.2.1 Implementation. Our LOS framework is implemented with PyTorch and released for reproduction[1]. We leverage the BERT encoder used in UDA and MixText to build the Θ component. When optimizing LOS with the EM algorithm, we first initialize Q_i for each unlabeled text u_i as $(0.5, 0.5)$ and start the training by the M-step, then iteratively execute the E-step and M-step until convergence. At the M-step, we add a weight 0.0001 to the loss term of the outlier detection model to balance training. We execute $m = 100$ SGD updates at each M-step, then execute one E-step to compute $\{Q_i\}_{i=1}^{n}$. All hyper-parameters are selected by grid search on the validation set. The training batch size is set to 8, and the learning rate is set to $1e-5$. The hyper-parameter λ in the normalized entropy function is set to 1.25. For baseline STC models and outlier detection methods, we use their released code to run experiments on the OSTC setting. We run each experiment setting 3 times and report the average performance and standard deviation. We use the code released in the original paper to run experiments in the OSTC setting for all STC models and outlier detection methods. We run all experiments on an NVIDIA Tesla P100 GPU with 16GB memory.

5.2.2 Baseline Models. We evaluate the models on the three benchmarks and compare them with the following baselines:
BERT: A baseline supervised text classification method utilizing the BERT-based-uncased model without using the unlabeled texts [11].
UDA: A baseline STC model that substitutes noising operations with data augmentations combined with consistency training [34].
MixText: A baseline STC model that proposes a new data augmentation method based on Mixup to boost consistency training [6].
UDA+MSP, MixText+MSP: Pipeline OSTC models combine UDA, MixText with OOD detection by maximum softmax probability [14].
UDA+LSoftmax, MixText+LSoftmax: Pipeline OSTC models combine UDA, MixText that learn discriminative features using softmax and adopt OOD detection using Local Outlier Factor [36].
UDA+DOC, MixText+DOC: Pipeline OSTC models combine UDA, MixText with m 1-vs-rest sigmoid classifiers for OOD detection [31].
UDA+LMCL, MixText+LMCL: Pipeline OSTC models combine UDA, MixText with large margin cosine loss for OOD detection [23].

[1] code: https://github.com/BDBC-KG-NLP/SIGKDD2023_Latent_Outlier_Softening

5.3 OSTC Evaluation Results

Table 3 shows the open-set semi-supervised Text Classification Results. LOS combined with UDA and MixText significantly outperforms pipeline models that combine UDA and MixText with outlier detection methods. In the $k = 100$ setting of DBPedia, MSP and DOC combined with UDA and MixText perform slightly worse than our LOS framework. However, in other settings, LOS outperforms the baselines near 10%-20% Acc and F1. The significant improvements demonstrate the effectiveness of our model. Another observation is that the pipeline models perform worse when the labeled texts become fewer. This result can be explained as follows: when feeding fewer labeled texts, the outlier detector may perform worse and can not effectively filter out the OOD texts, leading to worse OSTC results. Nevertheless, benefiting from the OOD softening strategy, LOS is less sensitive to insufficient labeled texts and generally keeps the performance when labeled texts become fewer.

Table 4: The evaluation results of training pure STC models with ID and all (ID+OOD) unlabeled texts .

Method	AGNews						Yahoo						DBPedia					
	$k=10$		$k=50$		$k=100$		$k=10$		$k=50$		$k=100$		$k=10$		$k=50$		$k=100$	
	In	Acc	In	Acc	In	Acc	In	Acc	In	Acc	In	Acc	In	Acc	In	Acc	In	Acc
BERT (ID)	57.70	-	57.26	-	83.40	-	65.38	-	77.01	-	78.56	-	96.15	-	98.37	-	98.42	-
UDA (ID)	71.67	**71.50**	85.54	85.77	86.67	87.29	69.77	69.42	77.10	76.93	78.40	77.95	96.16	96.40	98.36	97.25	98.24	**97.82**
MixText (ID)	68.04	68.52	**86.36**	**86.98**	**87.11**	**87.65**	**74.54**	**73.78**	**78.53**	**78.28**	78.97	**78.92**	96.98	**97.13**	97.98	**97.58**	97.46	97.49
UDA (ID+OOD)	**72.68**	28.53	84.81	33.72	86.87	34.49	64.51	31.60	77.01	38.25	79.17	39.28	**97.84**	48.76	**98.47**	49.32	**98.46**	49.48
MixText (ID+OOD)	70.31	27.73	85.68	34.09	86.45	34.37	70.68	35.30	78.40	38.99	**80.00**	39.58	96.90	48.63	97.92	49.80	98.21	49.86

Table 5: The evaluation results of accuracy on ID examples (In) and OOD detection accuracy on all examples (Out).

Method	AGNews						Yahoo						DBPedia					
	$k=10$		$k=50$		$k=100$		$k=10$		$k=50$		$k=100$		$k=10$		$k=50$		$k=100$	
	In	Out	In	Out	In	Out	In	Out	In	Out	In	Out	In	Out	In	Out	In	Out
UDA+MSP	72.38	46.77	84.47	50.60	86.98	65.80	60.41	50.24	75.47	61.62	**79.53**	63.83	94.95	61.62	98.38	72.49	98.33	77.19
UDA+LSoftmax	69.52	45.17	84.96	42.88	87.01	52.63	63.30	49.85	76.90	48.84	78.98	47.14	**97.19**	50.01	98.39	53.34	98.27	57.81
UDA+DOC	70.41	50.02	85.50	50.70	86.54	60.26	67.87	49.95	75.98	50.98	78.02	57.72	92.23	49.97	98.13	56.27	98.45	79.50
UDA+LMCL	69.50	49.46	86.01	51.85	86.17	54.89	58.16	49.67	76.97	56.13	78.00	58.98	96.49	50.80	**98.41**	65.74	**98.46**	67.47
MixText+MSP	61.68	49.31	**86.13**	54.22	**87.15**	60.10	67.04	50.95	**78.38**	61.85	79.25	60.69	96.67	57.62	98.08	71.95	98.11	73.96
MixText+LSoftmax	73.23	44.37	83.84	46.30	86.64	49.19	**71.15**	49.91	77.02	48.59	78.59	48.97	96.75	48.76	97.99	49.58	97.98	51.78
MixText+DOC	**73.28**	47.54	86.08	54.51	86.89	56.76	64.67	51.82	77.28	60.30	79.30	62.88	96.20	49.91	98.20	73.50	98.18	77.63
MixText+LMCL	70.66	49.94	85.57	47.63	86.37	49.56	67.32	49.80	77.62	54.60	79.41	56.94	95.77	50.03	98.21	65.67	98.30	67.50
UDA+LOS	69.15	**67.68**	83.33	70.48	86.67	**85.63**	62.06	58.30	76.33	71.78	78.18	**73.51**	92.18	**80.71**	96.17	**95.39**	94.77	**81.06**
MixText+LOS	59.35	58.80	83.84	**72.22**	86.39	80.61	68.81	**69.63**	77.24	**72.99**	79.20	72.49	93.93	75.26	96.44	81.31	97.07	79.88

5.4 Analysis on False Positive Inference

5.4.1 The Impact of False Positive Inference on Pure STC Models. We make an ablation study to analyze how much the false positive inference problem affects the STC models. We report the accuracy on ID examples (In) and the overall accuracy (Acc) (incorporating Local Outlier Factor (LOF) for outlier detection) in Table 4. When training with only ID examples, i.e., traditional semi-supervised learning, both UDA and MixText improve upon BERT without leveraging the unlabeled text, demonstrating the effectiveness of the STC models. However, when UDA and MixText are trained on unlabeled texts with additional OOD texts, their Acc suffers a significant drop (about 50%, 40% and 50% drops on AGNews, Yahoo and DBPedia, respectively). As the In accuracy only has little change, we may conclude that the drop in Acc substantially cause by incorrectly predicting OOD examples as ID examples, i.e., UDA and MixText are heavily affected by false positive inference.

5.4.2 The Effectiveness of LOS in Mitigating False Positive Inference. To investigate the effectiveness of LOS and the pipeline OSTC models in mitigating the false positive inference problem, we report the accuracy of ID examples (In) and outlier detection accuracy (Out) in Table 5. The table shows that our LOS framework outperforms pipeline OSTC models with large margins on the outlier detection (Out) accuracy results, although slightly sacrificing the accuracy on ID examples (In). These results suggest that our LOS framework is more likely to avoid misclassifying an ID example to OOD or misclassifying an ID example to OOD than the pipeline OSTC models, manifesting the effectiveness of our LOS framework in alleviating the false positive inference problem. Another observation is that when k is larger, the performance gap between the pipeline models and LOS becomes smaller, indicating that outlier detectors may rely on abundant data, and LOS is more robust with scarce data.

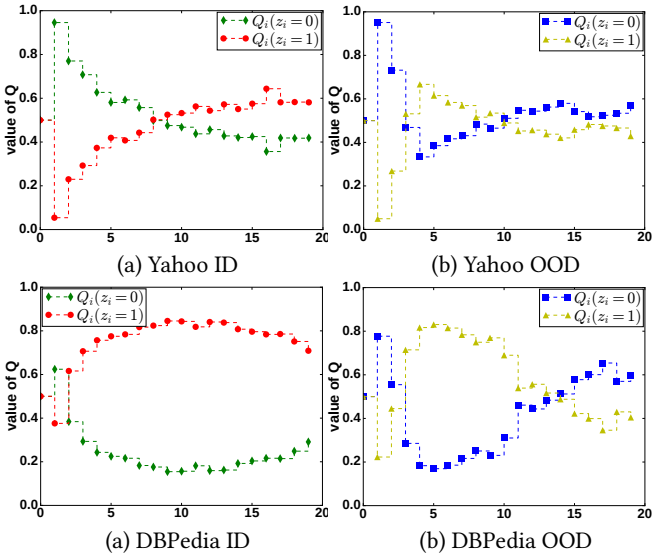
5.5 EM Training and Hyper-parameter Analysis

5.5.1 The Effectiveness of the EM training. To study the effect of EM optimization, we visualize the changes of the $\{Q_i\}_{i=1}^n$ values during each E-step of the training process on Yahoo and DBPedia datasets. Specifically, we report the $Q_i(z_i = 0)$ and $Q_i(z_i = 1)$ values averaged over ID examples or OOD examples in the unlabeled texts in Figure 3. The visualization results illustrate that $Q_i(z_i = 0)$ and $Q_i(z_i = 1)$ generally increase on ID examples and decrease on OOD examples, demonstrating that the EM training in LOS adopts a correctly updating direction of $\{Q_i\}_{i=1}^n$. At the early training stage, $Q_i(z_i = 0)$ obtains high values because the Θ component is nearly random and tends to output even distributions on ID classes, leading to high entropy that produces a high $Q_i(z_i = 0)$.

5.5.2 The Hyper-parameter Analysis for λ. The hyper-parameter λ controls the rate of change of the normalized entropy function, which is used to adapt the function to the data. Specifically, a lower

Table 6: Case study on two selected text examples from the AGNews dataset.

Id	Text	Label	MixText+MSP		MixText+LOS	
			prediction	confidence	prediction	confidence
1	'AGHDAD, Iraq, Aug. 17 A delegation of Iraqis was delayed for security reasons today but still intended to visit Najaf to try to convince a rebellious Shiite cleric and his militia to evacuate a shrine in the holy city and end ...'	\	Sci/Tech	0.6051	\	0.942
2	Unions representing workers at Turner Newall say they are 'disappointed' after talks with stricken parent firm Federal Mogul..	Business	\	0.5891	Business	0.674



(a) Yahoo ID (b) Yahoo OOD

(a) DBPedia ID (b) DBPedia OOD

Figure 3: The $Q_i(z_i = 0)$ and $Q_i(z_i = 1)$ values in EM training.

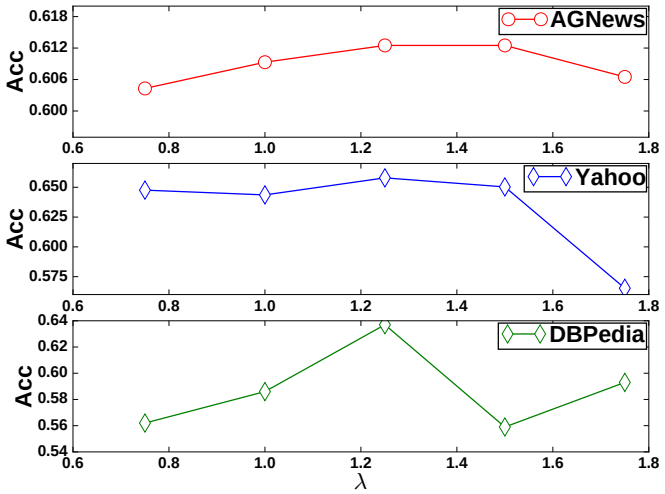


Figure 4: Evaluation with different hyper-parameter λ.

λ encourages predicting a text to an ID example and a higher λ encourages more confidence in predicting OOD. λ is an essential hyper-parameter for our LOS framework to guarantee the model's performance. To investigate how λ affects the performance of LOS, we attempt different λ values to train LOS on AGNews, Yahoo and DBPedia. The Acc results with different λ are shown in Figure 4. From the figure, we observe that using different λ may result in different Acc and the models on all datasets obtain the best results near $\lambda = 1.25$. These results suggest that we may adapt our LOS framework to the data by adjusting the hyper-parameter λ.

5.5.3 Case Study. We make case studies to compare the pipeline OSTC methods and our LOS framework. Specifically, we select two text examples from the AGNews dataset and report the predictions of MixText+MSP and MixText+LOS in Table 6. In case 1, the input text is an OOD text, which we represent its label using "\". Input this text. The MixText+MSP model predicts a wrong label *Sci/Tech* with confidence of 0.6051. Nevertheless, our MixText+LOS model correctly identifies the input text as an OOD example with a high confidence of 0.942. In the second case, the correct label of the input text is *Business*. However, the MixText+MSP model incorrectly identifies this text as an OOD example with a 0.5891 confidence. In this case, our MixText+LOS model still correctly predicts the label *Business* with a confidence 0.674. The two cases demonstrate that our LOS framework can improve the OSTC models by reducing misclassifications from OOD to ID or ID to OOD.

6 CONCLUSION AND FUTURE WORK

This work studies a new task, open-set semi-supervised text classification (OSTC). We propose a Latent Outlier Softening (LOS) framework to tackle the false positive inference problem in OSTC, which is demonstrated effective. Future work could take several forms. First, one optional realization of the outlier-detection model is employing a separate network component, incorporating its parameters in Θ and fixing corresponding parameters at the E-step and M-step during optimization. Second, our outlier-detection model is based on the static normalized entropy function with the hyper-parameter λ. It may be beneficial to investigate more dynamic functions that can adjust to the training condition. Third, it is important to gather and construct larger datasets with more classes in order to evaluate the OSTC's effectiveness in more complex circumstances.

ACKNOWLEDGMENTS

This work is supported partly by the National Key R&D Program of China under Grant 2021ZD0110700, partly by the Fundamental Research Funds for the Central Universities, and partly by the State Key Laboratory of Software Development Environment.

REFERENCES

[1] Randall Balestriero, Sebastien Paris, and Richard G. Baraniuk. 2020. Analytical Probability Distributions and Exact Expectation-Maximization for Deep Generative Networks. In *NeurIPS*.

[2] Siqi Bao, Huang He, Fan Wang, Hua Wu, and Haifeng Wang. 2020. PLATO: Pre-trained Dialogue Generation Model with Discrete Latent Variable. In *ACL*. Association for Computational Linguistics, 85–96.

[3] Yu Bao, Hao Zhou, Shujian Huang, Dongqi Wang, Lihua Qian, Xinyu Dai, Jiajun Chen, and Lei Li. 2022. latent-GLAT: Glancing at Latent Variables for Parallel Text Generation. In *ACL*. 8398–8409.

[4] Iacer Calixto, Miguel Rios, and Wilker Aziz. 2019. Latent Variable Model for Multi-modal Translation. In *ACL*. 6392–6405.

[5] Ming-Wei Chang, Lev-Arie Ratinov, Dan Roth, and Vivek Srikumar. 2008. Importance of Semantic Representation: Dataless Classification.. In *AAAI*, Vol. 2. 830–835.

[6] Jiaao Chen, Zichao Yang, and Diyi Yang. 2020. MixText: Linguistically-Informed Interpolation of Hidden Space for Semi-Supervised Text Classification. In *ACL*. 2147–2157.

[7] Junfan Chen, Richong Zhang, Yongyi Mao, Hongyu Guo, and Jie Xu. 2019. Uncover the Ground-Truth Relations in Distant Supervision: A Neural Expectation-Maximization Framework. In *EMNLP-IJCNLP*. 326–336.

[8] Junfan Chen, Richong Zhang, Jie Xu, Chunming Hu, and Yongyi Mao. 2022. A Neural Expectation-Maximization Framework for Noisy Multi-Label Text Classification. *TKDE* 01 (2022), 1–12.

[9] Wei Chen, Yeyun Gong, Song Wang, Bolun Yao, Weizhen Qi, Zhongyu Wei, Xiaowu Hu, Bartuer Zhou, Yi Mao, Weizhu Chen, Biao Cheng, and Nan Duan. 2022. DialogVED: A Pre-trained Latent Variable Encoder-Decoder Model for Dialog Response Generation. In *ACL*. 4852–4864.

[10] Jihun Choi, Taeuk Kim, and Sang-goo Lee. 2019. A Cross-Sentence Latent Variable Model for Semi-Supervised Text Sequence Matching. In *ACL*. 4747–4761.

[11] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *NAACL-HLT*. 4171–4186.

[12] Angela F. Gao, Jorge C. Castellanos, Yisong Yue, Zachary E. Ross, and Katherine L. Bouman. 2021. DeepGEM: Generalized Expectation-Maximization for Blind Inversion. In *NeurIPS*. 11592–11603.

[13] Suchin Gururangan, Tam Dang, Dallas Card, and Noah A. Smith. 2019. Variational Pretraining for Semi-supervised Text Classification. In *ACL*. Association for Computational Linguistics, 5880–5894.

[14] Dan Hendrycks and Kevin Gimpel. 2017. A Baseline for Detecting Misclassified and Out-of-Distribution Examples in Neural Networks. In *ICLR*.

[15] Junkai Huang, Chaowei Fang, Weikai Chen, Zhenhua Chai, Xiaolin Wei, Pengxu Wei, Liang Lin, and Guanbin Li. 2021. Trash to Treasure: Harvesting OOD Data with Cross-Modal Matching for Open-Set Semi-Supervised Learning. In *ICCV*. 8290–8299.

[16] Shuning Jin, Sam Wiseman, Karl Stratos, and Karen Livescu. 2020. Discrete Latent Variable Representations for Low-Resource Text Classification. In *ACL*. 4831–4842.

[17] Canasai Kruengkrai. 2019. Better Exploiting Latent Variables in Text Modeling. In *ACL*. 5527–5532.

[18] Dong-Hyun Lee et al. 2013. Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. In *Workshop on challenges in representation learning, ICML*, Vol. 3. 896.

[19] Ju Hyoung Lee, Sang-Ki Ko, and Yo-Sub Han. 2021. SALNet: Semi-supervised Few-Shot Text Classification with Attention-based Lexicon Construction. In *AAAI*. 13189–13197.

[20] Changchun Li, Ximing Li, and Jihong Ouyang. 2021. Semi-Supervised Text Classification with Balanced Deep Representation Distributions. In *ACL*. 5044–5053.

[21] Haoran Li, Chun-Mei Feng, Tao Zhou, Yong Xu, and Xiaojun Chang. 2022. Prompt-driven efficient Open-set Semi-supervised Learning. *CoRR* abs/2209.14205 (2022).

[22] Shujie Li, Min Yang, Chengming Li, and Ruifeng Xu. 2022. Dual Pseudo Supervision for Semi-Supervised Text Classification with a Reliable Teacher. In *SIGIR*. 2513–2518.

[23] Ting-En Lin and Hua Xu. 2019. Deep Unknown Intent Detection with Margin Loss. In *ACL*. 5491–5496.

[24] Chen Liu, Mengchao Zhang, Zhibing Fu, Panpan Hou, and Yu Li. 2021. FLiText: A Faster and Lighter Semi-Supervised Text Classification with Convolution Networks. In *EMNLP*. 2481–2491.

[25] Yen-Cheng Liu, Chih-Yao Ma, Xiaoliang Dai, Junjiao Tian, Peter Vajda, Zijian He, and Zsolt Kira. 2022. Open-Set Semi-Supervised Object Detection. In *ECCV*. 143–159.

[26] Pablo Mendes, Max Jakob, and Christian Bizer. 2012. DBpedia: A Multilingual Cross-domain Knowledge Base. In *LREC*. Istanbul, Turkey, 1813–1817.

[27] Yu Meng, Jiaming Shen, Chao Zhang, and Jiawei Han. 2018. Weakly-Supervised Neural Text Classification. In *CIKM*. 983–992.

[28] Takeru Miyato, Andrew M. Dai, and Ian J. Goodfellow. 2017. Adversarial Training Methods for Semi-Supervised Text Classification. In *ICLR*.

[29] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog* 1, 8 (2019), 9.

[30] Kuniaki Saito, Donghyun Kim, and Kate Saenko. 2021. OpenMatch: Open-set Consistency Regularization for Semi-supervised Learning with Outliers. In *NeurIPS*.

[31] Lei Shu, Hu Xu, and Bing Liu. 2017. DOC: Deep Open Classification of Text Documents. In *EMNLP*. 2911–2916.

[32] Antti Tarvainen and Harri Valpola. 2017. Weight-averaged consistency targets improve semi-supervised deep learning results. *CoRR* abs/1703.01780 (2017).

[33] Austin Cheng-Yun Tsai, Sheng-Ya Lin, and Li-Chen Fu. 2022. Contrast-Enhanced Semi-supervised Text Classification with Few Labels. In *AAAI*. 11394–11402.

[34] Qizhe Xie, Zihang Dai, Eduard H. Hovy, Thang Luong, and Quoc Le. 2020. Unsupervised Data Augmentation for Consistency Training. In *NeurIPS*.

[35] Hai-Ming Xu, Lingqiao Liu, and Ehsan Abbasnejad. 2022. Progressive Class Semantic Matching for Semi-supervised Text Classification. In *NAACL*. 3003–3013.

[36] Guangfeng Yan, Lu Fan, Qimai Li, Han Liu, Xiaotong Zhang, Xiao-Ming Wu, and Albert Y. S. Lam. 2020. Unknown Intent Detection Using Gaussian Mixture Model with an Application to Zero-shot Intent Classification. In *ACL*. 1050–1060.

[37] Qing Yu, Daiki Ikami, Go Irie, and Kiyoharu Aizawa. 2020. Multi-task Curriculum Framework for Open-Set Semi-supervised Learning. In *ECCV*, Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm (Eds.), Vol. 12357. 438–454.

[38] Qingfu Zhu, Wei-Nan Zhang, Ting Liu, and William Yang Wang. 2021. Neural Stylistic Response Generation with Disentangled Latent Variables. In *ACL*. 4391–4401.

[39] Ronghang Zhu and Sheng Li. 2022. CrossMatch: Cross-Classifier Consistency Regularization for Open-Set Single Domain Generalization. In *ICLR*.

A ADDITIONAL EXPERIMENTS

A.1 Evaluation on Large-Scale Long-Tailed Datasets

To evaluate the models on datasets with noisier and more difficult OOD examples with a large set of long-tail classes, we construct new datasets with extreme text classification datasets RCV1-2K and EURLex-4K, which involve 2456 and 3993 classes, respectively. As the extreme text classification datasets are used for the multi-label classification task, they are unsuitable to be used as in-distribution classes. However, they can be used to mimic complex OOD distributions because the labels of OOD examples are not needed. Thus, we use examples from the AGNews dataset as in-distribution examples and examples from RCV1-2K or EURLex-4K as out-of-distribution examples to construct two new datasets and evaluate a baseline MixText+DOC and our model MixText+LOS, the results on RCV1-2K and EURLex-4K are shown in Table 7 and Table 8, respectively.

Table 7: The evaluation results with RCV1-2K.

Method	RCV1-2K					
	$k = 10$		$k = 50$		$k = 100$	
	Acc	F1	Acc	F1	Acc	F1
MixText+DOC	44.71	33.05	48.95	39.53	68.63	50.78
MixText+LOS	**48.00**	**36.37**	**61.54**	**49.35**	**73.85**	**54.24**

Table 8: The evaluation results with EURLex-4K.

Method	EURLex-4K					
	$k = 10$		$k = 50$		$k = 100$	
	Acc	F1	Acc	F1	Acc	F1
MixText+DOC	34.76	28.85	45.71	39.46	62.84	48.77
MixText+LOS	**44.25**	**37.18**	**58.00**	**47.87**	**69.53**	**53.32**

The metrics "Acc", "F1" and "Out" denote the overall accuracy, F1 score and outlier detection accuracy on all examples, respectively. The above results show that our model MixText+LOS works well on datasets with noisier and large sets of long-tail OOD examples and outperforms the baseline.