Prototype-Guided Pseudo Labeling for Semi-Supervised Text Classification

Weiyi Yang[1], **Richong Zhang**[1,2]*, **Junfan Chen**[1], **Lihong Wang**[3], **Jaein Kim**[1]
[1]SKLSDE, School of Computer Science and Engineering, Beihang University, Beijing, China
[2]Zhongguancun Laboratory, Beijing, China
[3]CNCERT

Abstract

The semi-supervised text classification (SSTC) task aims at training text classification models with a few labeled data and massive unlabeled data. Recent works achieve this task by pseudo-labeling methods that assign pseudo-labels to unlabeled data as additional supervision. However, these models may suffer from incorrect pseudo-labels caused by underfitting of decision boundaries and generating biased pseudo-labels on imbalanced data. We propose a prototype-guided semi-supervised model to address the above problems, which integrates a prototype-anchored contrasting strategy and a prototype-guided pseudo-labeling strategy. Particularly, the prototype-anchored constrasting constructs prototypes to cluster text representations with the same class, forcing them to be high-density distributed, thus alleviating the underfitting of decision boundaries. And the prototype-guided pseudo-labeling selects reliable pseudo-labeled data around prototypes based on data distribution, thus alleviating the bias from imbalanced data. Empirical results on 4 commonly-used datasets demonstrate that our model is effective and outperforms state-of-the-art methods.

1 Introduction

Traditional text classification has achieved profound success by leveraging numerous labeled data. However, acquiring large-scale labeled data is expensive in many real-world scenarios, making training effective classifiers using such approaches difficult. Therefore, semi-supervised text classification (SSTC), aiming at text classification requiring a few labeled data and massive unlabeled data, has become a new appealing technique.

As a promising semi-supervised paradigm, the pseudo-labeling method (Mukherjee and Awadallah, 2020; Xie et al., 2020; che; Lee et al., 2021; Li et al., 2022), which assigns pseudo-labels to unlabeled data instances as additional supervision, has drawn attention. They follow the well-known low-density separation assumption (Chapelle and Zien, 2005) to generate pseudo-labels for the unlabeled data scattered in the high-density feature space, helping to identify the boundaries between classes. Trained with the mixture of labeled and pseudo-labeled data, the model perceives more abundant supervision signals and reduces empirical errors. Existing models attempt to assign soft pseudo-labels for unlabeled texts (Xie et al., 2020) or keep pseudo-labels with lower uncertainty (Mukherjee and Awadallah, 2020) to alleviate the influence brought by the incorrect pseudo-labeled data.

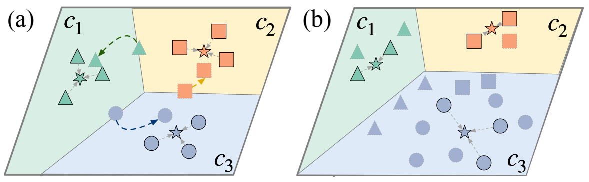


Figure 1: Illustration of two cases of pseduo-labeling problems in existing SSTC models: (a) underfitting near decision boundaries; (b) bias from imbalanced data. ★ denotes a prototype consisting of labeled data. Different symbols indicate different categories of labeled (bordered) and unlabeled data (unbordered). The areas with different colors indicate different categories.

In practice, the *underfitting of decision boundaries* is usually detrimental to the model learning (shown in Figure 1(a)). More careful considerations are expected in properly utilizing these pseudo-labeled data near decision boundaries. In addition, the pseudo-labeling process may meet with *bias from imbalanced data* and will be biased toward dominant class data (shown in Figure 1(b)). This problem may further lead the dominant classes to dominate the follow-up training and assign more incorrect labels to the unlabeled data.

Believing that addressing both the underfitting of decision boundaries and the bias from the im-

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

Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics
Volume 1: Long Papers, pages 16369–16382
July 9-14, 2023 ©2023 Association for Computational Linguistics

balanced data is crucial, we propose a prototype-guided pseudo labeling (PGPL) model to tackle the two problems simultaneously. PGPL improves the SSTC performance with a Prototype-Anchored Contrasting (PAC) module and a Prototype-Guided Pseudo Labeling (PGP) module. Specifically, in the PAC module, we leverage the prototypical learning (Bien and Tibshirani, 2011) to construct prototypes with representations of labeled texts and classify unseen texts according to their distance from the prototype of each class. In this way, we force the text representations to locate in high-density areas, making the unreliable pseudo-labeled data near the boundary away from decision boundaries to address the problem, as demonstrated in Figure 1(a). Furthermore, to alleviate the bias from the imbalanced data, our PGP module chooses reliable pseudo-labeled data by reducing the probability of assigning labels of dominant class for unlabeled data, and thus prevents the model from biasing on dominant classes, as demonstrated in Figure 1(b).

Finally, we conduct experiments on four commonly-used datasets, and the results demonstrate that our proposed PGPL model outperforms the state-of-the-art performance in various semi-supervised settings. The experimental analysis also suggests that our PGPL model successfully separates the different classes and assigns reliable pseudo-labels to the unlabeled instances. These results confirm the effectiveness of PGPL.

To summarize, our main contributions are in the following four folds:

- We propose a prototype-guided pseudo-labeling model for the semi-supervised text classification task, which simultaneously addresses the underfitting of decision boundaries and the bias from imbalanced data.
- We present a prototype-anchored contrasting module to generate high-density distributed representations that reduces the unreliable pseudo-labeled data chosen for training the semi-supervised models.
- We leverage a prototype-guided labeling module to choose data around prototypes that alleviates the bias from imbalanced data.
- Empirical studies indicate that our proposed model outperforms previous state-of-the-art methods on four commonly-used benchmarks and comprehensive analysis confirm the effectiveness of PGPL.

2 Related Works

2.1 Semi-supervised Text Classification

Existing methods for SSTC task can be divided into three groups: Generative methods, Graph-based methods and Pseudo-Labeling methods.

Generative methods (Croce et al., 2019, 2020; Gururangan et al., 2019) attempt to learn a generative model that establishes a mapping function based on the distribution of the labeled/unlabeled data to the classes. However, they usually require the labeled and unlabeled data strictly obey the same data distribution, which can hardly be satisfied in practical scenarios.

Graph-based methods (Linmei et al., 2019; Li et al., 2021; Cui et al., 2022) attempt to construct an graph structure underlying the data, where each node represents a labeled/unlabeled instance and each edge represents a pairwise relation in certain similarity measurement. However, they construct such graphs upon all the data, inevitably requiring huge computational resources, which can be impractical on large-scale datasets.

Pseudo-Labeling methods learn the model on the mixture of labeled data and unlabeled data (annotated with pseudo-labels), thus accessing to additional beneficial supervision. There exist two major practices: One line of the methods (Wang et al., 2021; Mukherjee and Awadallah, 2020; Lee et al., 2021; Tsai et al., 2022) is based on self-training strategy , which first trains a base model on labeled data, and then retrains the model on the unlabeled data annotated with pseudo-labels of high confidence. Another line of the methods (Najafi et al., 2019; Xie et al., 2020; Sohn et al., 2020) incorporates the idea of consistency learning, which adopts the unlabeled data to enhance model robustness with data perturbation, such as adversarial attack (Najafi et al., 2019) and data augmentation (Xie et al., 2020; Sohn et al., 2020; Chen et al., 2020). Although these methods provide potentiality and flexibility in practices, they still suffer from the underfitting near decision boundaries problem, and easily encounter learning bias from imbalanced data. In this paper, we develop the previous pseudo-labeling studies with prototype learning, which enhances low-density separation by exploiting prototypes to cluster text representations, and alleviates imbalanced class bias with prototype-guided pseudo labeling.

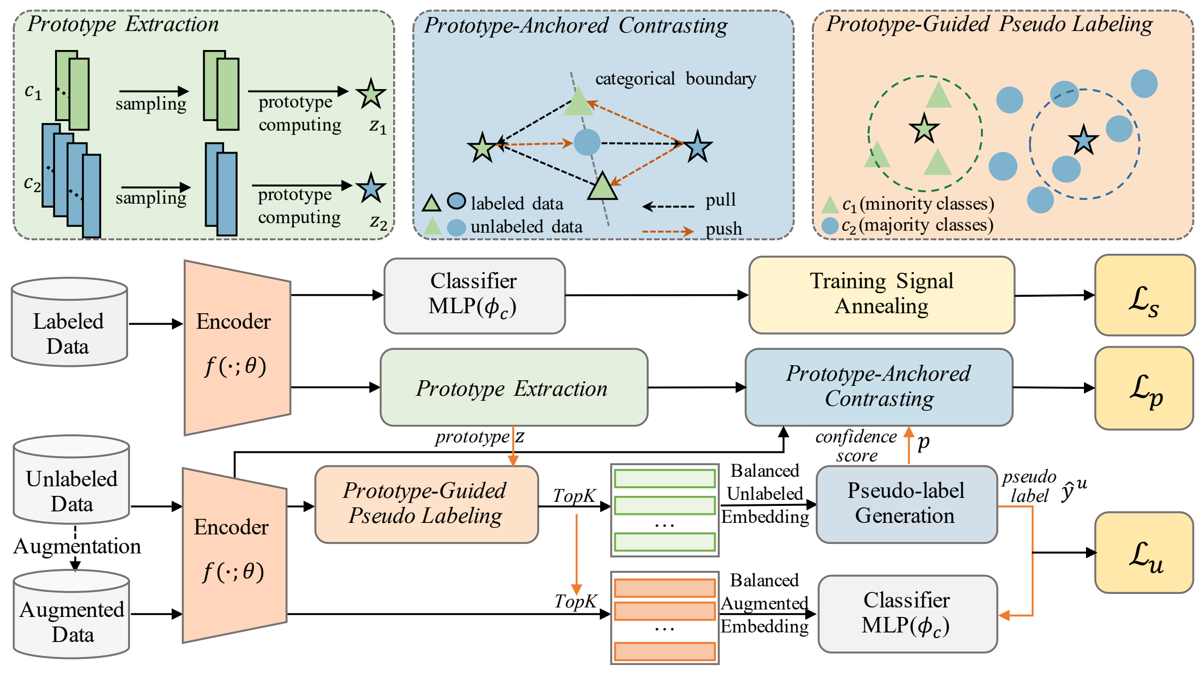


Figure 2: Overall architecture of PGPL. For labeled data, the supervised loss is computed to generate class prototypes for processing the Prototype-Anchored Contrasting module. For unlabeled data, the pseudo-labels are assigned based on Prototype-Guided Pseudo Labeling to augment the text instances for retraining the model.

2.2 Prototypical Learning

Prototypical learning (Bien and Tibshirani, 2011) aims to learn representative prototypes to summarize the instances in the same class, and is been widely used for unsupervised (Li et al., 2020; Pan et al., 2019) and semi-supervised learning (Snell et al., 2017; Arik and Pfister, 2020; Gu, 2020; Ren et al., 2018). In unsupervised scenarios, the most representative method is PCL (Li et al., 2020), which introduces a prototype as the center of mass of a cluster formed by similar samples and assigns each sample to multiple prototypes of different granularity. The goal of training is to bring each image embedding closer to its associated prototype. In semi-supervised scenarios, the prototypes can first be estimated with the labeled instances. Then, the pseudo-labels or classes of unlabeled data are obtained by calculating distances to the prototypes. Among these works, Gu (2020) attempts prototype-based pseudo-labeling for image classification but neglects the underfitting near the decision boundary and bias from the imbalanced data. We utilize prototypes to conduct low-density separation and data selection to alleviate both issues.

3 Problem Formulation

In semi-supervised text classification, we are given a small set of labeled text data and a large set of unlabeled text data. Formally, let $\mathcal{C}$ be the set of all labels of interest. $D_l = \{\ (x_1^l, y_1^l), \cdots, (x_m^l, y_m^l)\ \}$ is the set of pre-annotated m labeled text instances, where each x_i^l and $y_i^l \in \mathcal{C}$ represent a text instance and its corresponding label. $D_u = \{\ x_1^u, \cdots, x_n^u\ \}$ is the set of n unlabeled text data, where $x_i^u \in D_u$ denotes an unlabeled text instance. Semi-supervised text classification hopes to leverage the limited annotated data and expand its knowledge to the unlabeled data for pseudo-labeling and include them as additional training data.

4 Methodology

We propose a semi-supervised text classification model PGPL to tackle problems of underfitting near the decision boundary and bias from the imbalanced data. In this section, we first illustrate the overall model structure and introduce the fundamental SSTC structure with supervised and unsupervised losses, then describe the prototype-anchored contrasting module and finally explain the prototype-guided pseudo labeling module.

4.1 Model Structure

The model structure of PGPL is illustrated in Figure 2. The labeled and unlabeled text instances are first input to the Encoder. Then, the labeled text embeddings are used to generate pseudo labels in Prototype Extraction module and encourage each instance closer to its categorical prototype and away from the other categorical prototypes via prototype-anchored contrasting. Finally, we perform prototype-guided pseudo labeling on the unlabeled data. The pseudo-labeled data is used to augment the training set and retrain the model.

4.2 Fundamental SSTC Framework

4.2.1 Backbone Text Classifier

To utilize the power of pre-trained language models in SSTC, we follow the previous practice (Chen et al., 2020) that uses BERT (Devlin et al., 2019) as the text encoder. Specifically, let x represent either a labeled instance $x^l \in D_l$ or an unlabeled instance $x^u \in D_u$. The function $f(x;\theta)$ denotes the BERT encoding process with mean pooling operation over the token representations. Afterward, a two-layer MLP with $\tanh(\cdot)$ activation is adopted to derive the relevance score of the input text x corresponding to each class:

$$g(x, \phi_y, \theta) = \mathrm{MLP}(f(x;\theta); \phi_c), \tag{1}$$

where ϕ_c is the MLP parameter corresponding to class $c \in \mathcal{C}$. Note that this backbone text classifier can be trained by two objectives: 1) a supervised loss function $\mathcal{L}_s$ (in Eq. (3)) over the labeled instances; 2) an unsupervised loss function $\mathcal{L}_u$ (in Eq. (5)) over the unlabeled instances.

4.2.2 Annealing Supervised Loss

Since the number of labeled instances is usually limited and costly in SSTC, we adopt the annealing supervised loss to fully exploit the labeled data. There is an existing work that introduces the training signal annealing (TSA) strategy (Xie et al., 2020), which releases labeled instances gradually during the training phase instead of observing them at once, to prevent the model from early over-fitting on the labeled data. Intuitively, in the initial training phase, we expect to learn about the data without overconfidence, in order not for the model to overvalue the incidental features of the data. Therefore, we employ a threshold η_t, and at training iteration t, we select the instances with the highest score of class confidence lower than the threshold η_t for loss calculation. The threshold η_t is derived as:

$$\eta_t = \frac{t}{T}(1-\tau) + \tau, \tag{2}$$

where T is the total iterations of the training process, and τ is the initial threshold that is set to a scalar larger than $\frac{1}{|\mathcal{Y}|}$ to all the classes. This strategy encourages the model to observe and learn the data with relatively lower confidence, and gradually adapt to perceive all training instances.

Based on the threshold, we define the supervised loss at t iteration with cross-entropy:

$$\mathcal{L}_s = -\frac{1}{m}\sum_{i=1}^{m} \mathbb{I}(p_{y_i^l} < \eta_t) \log \frac{\exp(g(x_i^l, \phi_{y_i^l}, \theta))}{\sum_{c\in\mathcal{C}} \exp(g(x_i^l, \phi_c, \theta))}, \tag{3}$$

where $p_{y_i^l}$ is the predicted probability of y_i^l corresponding to instance x_i^l, and $\mathbb{I}(\cdot)$ is a binary indicator which equals 1 if $p_{y_i^l} < \eta_t$ or 0 otherwise.

4.2.3 Selective Unsupervised Loss

To leverage the unlabeled text data, the selective unsupervised loss module assigns pseudo-labels for each unlabeled text instance $x_i^u \in D_u$ to augment the data. Following recent works (Xie et al., 2020; Mukherjee and Awadallah, 2020; Sohn et al., 2020) in semi-supervised learning, the original unlabeled data is augmented with the pseudo-labeled data and used to optimize the text classifier.

Specifically, we incorporate the augmented data $\tilde{x}_i^u$ of x_i^u with its pseudo label $\hat{y}_i^u$ into the new training dataset to train the classifier with cross-entropy loss. Formally, for a unlabeled instance x_i^u, its pseudo label is determined by:

$$\hat{y}_i^u = \arg\max_{c\in\mathcal{C}} \exp(g(x_i^u, \phi_c, \theta)), \tag{4}$$

Among the unlabeled instances with pseudo-labels, we further select high-quality pseudo-labeled instances that are used in the next training phase by the following formula:

$$\mathcal{L}_u = -\frac{1}{n}\sum_{i=1}^{n} \mathbb{I}(x_i^u) \log \frac{\exp(g(\tilde{x_i^u}, \phi_{y_i^u}, \theta))}{\sum_{c\in\mathcal{C}} \exp(g(\tilde{x_i^u}, \phi_c, \theta))}, \tag{5}$$

where the binary indicator $\mathbb{I}(\cdot) \in \{0,1\}$ is:

$$\mathbb{I}(x_i^u) = \mathbb{I}(\hat{y}_i^u = c) \wedge \mathbb{I}(\mathrm{d}(f(x_j^u), z_c) \leq d_c), \tag{6}$$

where $\wedge$ represents the logic operation *AND*. $\mathrm{d}(\cdot,\cdot)$ denotes the distance function that measures the distance between the text representation $f(x_j^u)$ and prototype z_c of class c. d_c serves as the threshold to determine the value in the binary indicator. The selected instances will be used in the next training phase as additional training data.

Dataset	Classification Type	Class	Train	Unlabeled	Dev	Test
AG News	News Topic	4	200	5000	2000	1900
DBpedia	Wikipedia Topic	14	200	5000	2000	5000
Yahoo! Answer	QA Topic	10	200	5000	2000	6000
IMDB	Movie Review Sentiment	2	200	5000	2000	12500

Table 1: The dataset statistics for the per-class number of unlabeled, dev and test data.

4.3 Prototype-Anchored Contrasting Module

To alleviate the underfitting near decision boundaries problem, inspired by the low-density separation hypothesis, the prototype-anchored cluster contrasting module (PAC) forces each text representation to be closer to its categorical prototype and far away from unrelated prototypes. As such, the number of text instances lie near the decision boundaries can be reduced.

4.3.1 Categorical Prototypes Extraction

The prototype of a class is defined by the mean vector of text embeddings within the class. Concretely, we first obtain the text embedding $f(x_i^l)$ for each $x_i^l \in D_l$, and generate prototypes according to their classes. For each $c \in \mathcal{C}$, the prototype z_c is computed by:

$$z_c = \frac{1}{n_c} \sum_{y_i^l = c} f(x_i^l), \tag{7}$$

$$n_c = \sum_{y_i^l \in D_l} \mathbb{I}(y_i^l = c), \tag{8}$$

where n_c is the instance number of class c.

4.3.2 Prototype-Anchored Contrasting

After we obtain the prototype of each class, for a subset of labeled instances $x_i^l \in D_l$ and unlabeled instances $x_j^u \in D_u$, we expect the embedding $f(x_i^l)$ (resp. $f(x_j^u)$) to be closer to its prototype $z_{y_i^l}$ (resp. $z_{\hat{y}_i^u}$) and away from the others. For the unlabeled text instances, we migrate them within the representation space to make each cluster more distinct based on the prediction confidence. More specifically, a confidence score for each instance is calculated and is used as a weight value for determining how much it should be migrated. In other words, as the instances obtained low confidence are more likely to be assigned with incorrect pseudo-labels, how far the model allows the instances to migrate is proportional to confidence score.

Thus, the loss function of this module can be formulated as:

$$\begin{aligned}\mathcal{L}_p = &-\frac{1}{m}\sum_i^m \sum_c \mathbb{I}(y_i^l = c) \log \frac{\exp\left(-\mathrm{d}(f(\tilde{x_i^l}), z_c)\right)}{\sum_{k \in \mathcal{C}} \exp(-\mathrm{d}(f(\tilde{x_i^l}), z_k))} \\ &- \frac{\lambda}{n}\sum_j^n \sum_c p_{\hat{y}_j^u} \mathbb{I}(\hat{y}_j^u = c) \log \frac{\exp\left(-\mathrm{d}(f(\tilde{x_j^u}), z_c)\right)}{\sum_{k \in \mathcal{C}} \exp(-\mathrm{d}(f(\tilde{x_j^u}), z_k))},\end{aligned} \tag{9}$$

where m and n are the instance number of labeled and unlabeled data, respectively. The function $\mathrm{d}(\cdot, \cdot)$ is a distance metric function and λ is a weight factor. We use Euclidean distance for the distance metric between each instance and the prototype embedding, and use softmax normalization for the instances in the representation space. Here we also consider the pseudo-label confidence $p_{\hat{y}_i^u}$ (see Eq. (4)) to balance the training.

4.4 Prototype-guided Pseudo Labeling Module

As discussed in the introduction, the unlabeled instances are easily misguided by the dominant classes, inevitably leading to assigning incorrect pseudo-labels. To alleviate the bias from imbalanced data, we propose a prototype-guided pseudo labeling module (PGP). The idea is to select different k-nearest pseudo-labeled instances corresponding to the center (i.e., the prototype) for each class according to a short training history.

To balance the training process, we expect not to select (or select fewer) pseudo-labeled instances for a class that has been over-trained in previous iterations. Thus, we record a statistic value μ_t^c during training, which returns the number of selected pseudo-labeled texts for class c at the t^{th} iteration. For simplicity, we use $\mu_{<t}^c$ to denote the summation of $\mu_{t'}^c$ (with $t' < t$) from a training history of previous iterations. Then, at each iteration t, we can find the least trained class and compute its total number of selected pseudo-labeled texts during the training history by:

$$\mu_t^c = \sum_{\tilde{x_j^u} \in B_u} \mathbb{I}(\hat{y}_j = c), \tag{10}$$

$$\gamma_t = \arg\min_{c \in \mathcal{C}} \mu_{<t}^c. \tag{11}$$

We hope this least trained class utilizes more pseudo-labeled instances in the next training iteration than other classes to keep balanced training. To this end, we determine the number of nearest text instances k_c for class $c \in \mathcal{C}$ according to:

$$k_c = \begin{cases} \mu_t^c & \text{if } \mu_{<t}^c - \gamma_t = 0 \\ \mu_t^c - (\mu_{<t}^c - \gamma_t) & \text{if } 0 \le \mu_{<t}^c - \gamma_t < \mu_t^c \\ 0 & \text{if } \mu_{<t}^c - \gamma_t \ge \mu_t^c \end{cases}, \tag{12}$$

Here we encourage to select more pseudo-labeled instances for the less-trained classes, and fewer pseudo-labeled instances for the over-trained classes, thus alleviating the imbalanced training problem. Based on k_c, we select the top-k_c pseudo-labeled instances near the categorical prototype of class c by Euclidean distance $\mathrm{d}(\cdot,\cdot)$ as follows:

$$d_c = \mathrm{TopK}(\mathrm{d}(f(\tilde{x_j^u}), z_c), k_c), \tag{13}$$

where d_c is the distance threshold used in Eq. (6) to select pseudo-labeled texts. By selecting k-nearest pseudo-labeled text instances around the corresponding categorical prototypes, the model obtains more reliable training data for SSTC.

The overall objective $\mathcal{L}$ is combined with three losses, including the cross-entropy losses on the labeled and unlabeled data, and the prototype-anchored contrastive loss:

$$\mathcal{L} = \mathcal{L}_s + \beta_1 \mathcal{L}_u + \beta_2 \mathcal{L}_p, \tag{14}$$

where β_1 and β_2 are harmonic factors to balance the weight of the different losses and β_1 is usually fixed to 1.

5 Experimental Setup

5.1 Dataset and Pre-processing

We conduct intensive experiments on four widely used text classification benchmark datasets: IMDB (Maas et al., 2011), AG News (Zhang et al., 2015), Yahoo! Answers (Chang et al., 2008), and DBpedia (Lehmann et al., 2015). We follow the original setting of test set splitting and randomly sample from the training set to form the labeled and unlabeled sets used for the training. The experimental results are averaged after five runs. The statistics and splitting information of the dataset are shown in Table 1.

5.2 Baselines

To verify the effectiveness of our method, we compare our method to the several baselines:

BERT (Devlin et al., 2019) adopts a pre-trained model for text classification, trained with only labeled data.
Delta-training (Jo and Cinarel, 2019) distinguishes the instances with incorrect prediction and correct prediction by two separate classifiers.
VAMPIRE (Gururangan et al., 2019) pretrains a variational autoencoder, and uses its internal states as features in a downstream classifier.
UDA (Xie et al., 2020) makes soft predictions with original data to train the augmented data.
MixText (Chen et al., 2020) makes predictions by mixing the original data with the augmented data.
UST (Mukherjee and Awadallah, 2020) makes predictions by selecting pseudo labels with the same prediction multiple times.
SALNet (Lee et al., 2021) makes predictions with the help of categorical dictionary.
FLiText (Liu et al., 2021) adopts A lighter framework with convolution networks, predicted pseudo-labels through knowledge distillation.
CEST (Tsai et al., 2022) proposes a certainty-driven sample selection method and a contrast-enhanced similarity graph in self-training.
SAT (Chen et al., 2022) trains a meta-learner to predict the labels with weakly-augmented data.

For fairness, we implement BERT, UDA, MixText with the code provided by Chen et al. (2020) in our experiments. For other baselines, we use the reported results provided from the original research paper to avoid re-implementation bias.

5.3 Implementation

We used PyTorch[1] for implementation. For all compared models, the maximum sentence length is set to 256. For sentences that exceed the limit, we keep the first 256 tags. All hyper-paramters are selected by grid search on the validation set. We choose 1e-5 as the learning rate for the BERT encoder and 1e-3 for the MLP. For the unlabeled data, we choose German as the intermediate language

[1] https://pytorch.org/

Model	AG News			IMDB			Yahoo! Answer			DBpedia		
	10	30	200	10	30	200	10	30	200	10	30	200
Bert	81.0	84.3	87.2	70.6	73.3	86.1	60.1	64.1	69.3	96.6	98.2	98.6
UDA	86.4	86.4	88.3	86.4	86.4	88.7	64.3	68.3	70.2	97.8	98.3	98.8
Mixtext	87.3	87.4	88.2	74.2	85.3	89.1	**67.7**	68.5	70.6	98.5	98.8	98.9
PGPL	**87.8**	**88.5**	**89.2**	**88.9**	**90.2**	**90.3**	67.4	**69.1**	**70.7**	**98.7**	**99.0**	**99.0**

Table 2: Comparison with state-of-the-arts on the AG News, DBpedia, Yahoo! Answer and IMDB test set under different partition protocols. The Wilcoxons test shows significant difference (p<0.05) between our model and baselines averaged after five runs except on Yahoo! Answer (10 labels).

for back translation augmentation using FairSeq2[2], with a random sampling temperature of 0.9. We use AdamW (Loshchilov and Hutter, 2018) to optimize model parameters. The train epoch number is set to 20. In the training process, we choose 16 unlabeled samples for each batch. And for the labeled data, the number of labeled data is the number of classes of each dataset in each batch. The factor λ is set to 0.2 in IMDB and AG News and 0.5 in Yahoo!Answer and DBpedia. The history t iteration is set to 5. The harmonic factor β_1 and β_2 are set to 1 and 0.5, respectively.

5.4 Overall Results

We first compare our model with the BERT, UDA and Mixtext. We run these models with the same setting of our model. The training set consists of 5000 unlabeled data and respectively 10, 30, and 200 labeled data for each class.

Table 2 shows the results on 5000 unlabeled data and different amount of labeled data. The results show that our method is superior to all baselines. Especially when there are only 10 labeled instances for each class, our model outperforms by 6.8% for AG News, 18.3% for IMDB, and 7.3% for Yahoo Answers, respectively. On DBpedia, due to the performance of the initially trained classifier being already outstanding (98% as shown in the table), the performance improvement is limited. Compared to the state-of-the-art methods, our method outperforms under the settings of different numbers of labeled data in AG News, IMDB, and DBpedia. On Yahoo! Answer, it also performs better than the state-of-the-art methods except under 10 labeled data. These results verify the effectiveness of our PGPL model for the SSTC task.

As some models use a different number of labeled and unlabeled texts during training, we conduct experiments using their original settings and compare them as in Table 3. The compared models includes UST (Mukherjee and Awadallah, 2020), VAMPIRE (Gururangan et al., 2019), CEST (Tsai et al., 2022), SALNet (Lee et al., 2021), FLiText (Liu et al., 2021), SAT (Chen et al., 2022) and Delta-training (Jo and Cinarel, 2019). For consistency, we take the results from the original papers and round some of the results to one decimal place. The results show that our model outperforms other SOTA models under the same experimental set-

Datasets	Model	K	Acc.
AG News	PGPL	10	**87.8**
	SAT	20	86.4
	CEST	30	87.1
	UST	30	87.7
	VAMPIRE	200	83.9
IMDB	PGPL	30	**90.2**
	SAT	10	69.0
	CEST	30	**90.2**
	SALNet	21	75.7
	VAMPIRE	200	82.2
	Delta-training	212	75.0
Yahoo! Answer	PGPL	10	**67.4**
	SAT	20	61.5
	SALNet	34	53.7
	VAMPIRE	200	59.9
	FLiText	500	65.1
DBpedia	PGPL	10	**98.7**
	SALNet	20	98.2
	UST	30	98.6
	CEST	30	98.6

Table 3: SSTC methods with K training labels per class (UST, VAMPIRE, SALNet, FLiText, Delta-training, CEST, SAT).

[2]https://github.com/pytorch/fairseq

ting keeps its superiority when applied to different experimental settings.

5.5 Impact of Labeled and Unlabeled Data Distribution

In order to evaluate our results more accurately, we performed four experiments with different constraints for further comparison: (a) keep the original proportion of data with balanced labeled data balance and imbalanced unlabeled data; (b) keep the original proportion of balanced unlabeled data, but perform a random imbalanced operation on labeled data; (c) keep the original proportion of balanced labeled data, but perform a random imbalanced operation on unlabeled data; (d) perform a random imbalanced operation on labeled data and unlabeled data. The *balance* here means that the number of data in each class is the same. In the imbalanced setting, for convenience, the number of half classes is twice that of the other half. The experimental result is averaged after three runs on the verification set.

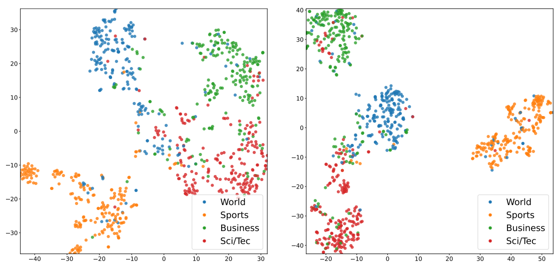
Data setting (labeled/unlabeled)	Model	Result
balanced / balanced	UDA	64.3
	Mixtext	68.1
	PGPL	**68.3**
imbalanced / balanced	UDA	64.8
	Mixtext	67.7
	PGPL	**68.8**
balanced / imbalanced	UDA	57.9
	Mixtext	67.6
	PGPL	**67.8**
imbalanced / imbalanced	UDA	64.1
	Mixtext	65.5
	PGPL	**67.3**

Table 4: Experimental results on different data balance settings, grouped by labeled/unlabeled data.

From Table 4, we find UDA show the worst result under the condition (c) and Mixtext under the condition (d) that have inconsistent distribution proportions though the two baselines have simliar results in balanced condition. Our model shows consistently good results in the four settings. Especially when only unlabeled data is imbalanced, the result is improved from 57.9% to 67.8% compared with UDA. In summary, under four different data distribution settings, our model is superior to other baselines, indicating that our model is more capable of complex data distribution scenarios.

5.6 Qualitative Evaluation

To investigate whether the PAC module helps to separate the clusters of different classes, we compare the visualized embeddings of full PGPL and PGPL without PAC model in Figure 3.



(a) PGPL$_{w/oPAC}$ (b) Full PGPL

Figure 3: Comparison of t-SNE visualization results with and without PAC module on the AG News dataset.

It is observed from the visualized results that when PGPL is implemented without PAC module (e.g., Figure 3 (a)), the boundaries of different clusters are unclear. Concretely, the clusters of different classes are separated to some extent, but not as distinct as those of full PGPL with PAC implemented. This suggests that the PAC module successfully separates the clusters of different classes and reduces the chance of data instances lying near the class boundaries.

5.7 Class Significance Evaluation

It is observed from Table 5 that PGPL with PGP applied improves the performance on all individual classes, especially in World class and Sports class.

5.8 Ablation Study

In order to investigate the impact of each component in the model, we conduct variants on PGPL.

Table 6 shows the results of PGPL by removing each component at a time, averaged after three runs on valid set with 10 labeled data and 5,000 unlabeled data per class. It is observed that both the PAC and PGP individually improve the final model performance, and the combination of PAC and PGP achieves the best performance. TSA module is also advantageous for improving the performance, while applying to DBpedia is an exception because its initial classifier is already outstanding as explained previously. This confirms the effectiveness of our proposed approach.

Class	Model	Accuracy
World	PGPL	**91.2**
	-w/o PGP	88.6
Sports	PGPL	**84.0**
	-w/o PGP	81.1
Business	PGPL	**96.1**
	-w/o PGP	96.0
Sci/Tech	PGPL	**81.9**
	-w/o PGP	80.3
All categories	PGPL	**88.3**
	-w/o PGP	86.5

Table 5: Experimental results on different data balance settings, grouped by labeled/unlabeled data.

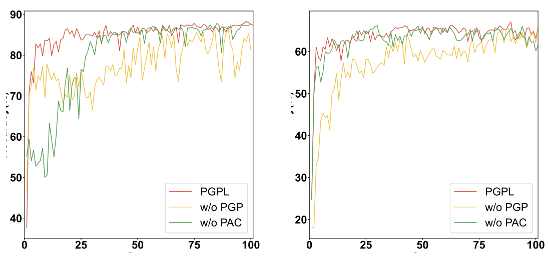
Data	AG News	IMDB
PGPL	**88.3**	**89.7**
w/o PGP	86.5	88.2
w/o PAC	86.2	88.9
w/o TSA	87.2	87.9
Data	**Yahoo!Answer**	**DBpedia**
PGPL	**68.3**	98.4
w/o PGP	65.7	98.2
w/o PAC	67.4	98.2
w/o TSA	68.0	**98.6**

Table 6: Ablation analysis of PGPL with different modules on valid data with 10 labeled data.

5.9 Stability Evalution

To investigate the efficiency of different models, we illustrate their training progresses on AG News and Yahoo! Answer datasets. From the results on the AG News dataset shown in Figure 4 (a), we observe that the model w/o PGP performs very poorly at the early stage of training but converges at about the 40th step. That is, the model w/o PAC shows a good start but continues fluctuating, and it cannot reach the convergence even after 100 training steps. The performance at early stage of PGPL is good and the training progress is also stable.

The training process on Yahoo! Answer dataset shown in Figure 4 (b) shows the similar tendency with the results in Figure 4 (a), except that the model w/o PGP and the full PGPL model show similar training stability and convergence speed. These results suggest that the Prototypical Cluster Separation module is an indispensable component in keeping the training stable and guaranteeing the convergence speed of the PGPL model. They also indicate that including both PAC and PGP modules not only improves the test accuracy of the model but also improves its training stability and training speed. Lastly, we conclude that including PAC in the early training stage provides more reliable pseudo-labeled texts and helps the latter training stage be more stable.



(a) AG News (b) Yahoo!Answer

Figure 4: The comparison of training stability and convergence speed among different models..

6 Conclusion

In this paper, we propose a semi-supervised model PGPL for SSTC task that applies PAC and PGP strategies. Specifically, PAC constructs prototypes to cluster text representations belongs to the same class by forcing them to be high-density distributed, thus alleviating the underfitting near decision boundary problem. PGP selects reliable pseudo-labeled data nearby prototypes to address the training bias from the imbalanced data. Experimental results demonstrate PGPL is effective and outperforms previous state-of-the-art methods.

Limitations

Although PGPL is proven to be effective according to our intensive experiments, the current design of the PGP module may not be optimal and could be improved in the future. Specifically, on the one hand, to choose the proper number of nearest text instances k_c for each class c, we need to obtain the statistic values μ_t^y, $\mu_{<t}^y$ and computed values μ_t^c, $\mu_{<t}^c$, γ_t for each dataset, which makes it somewhat not easy to scale to many different datasets. On the other hand, we heuristically filter pseudo-labeled texts by the top-k selection, which may also be improved with more systematical approach.

Acknowledgements

This work is supported partly by the National Key R&D Program of China under Grant 2021ZD0110700, partly by the National Natural Science Foundation of China (No.62106059), partly by the Fundamental Research Funds for the Central Universities, and partly by the State Key Laboratory of Software Development Environment.

References

Sercan Ö Arik and Tomas Pfister. 2020. Protoattend: Attention-based prototypical learning. *The Journal of Machine Learning Research*, 21(1):8691–8725.

Jacob Bien and Robert Tibshirani. 2011. Prototype selection for interpretable classification. *The Annals of Applied Statistics*, 5(4):2403–2424.

Ming-Wei Chang, Lev-Arie Ratinov, Dan Roth, and Vivek Srikumar. 2008. Importance of semantic representation: Dataless classification. In *Aaai*, volume 2, pages 830–835.

Olivier Chapelle and Alexander Zien. 2005. Semi-supervised classification by low density separation. In *International workshop on artificial intelligence and statistics*, pages 57–64. PMLR.

Hui Chen, Wei Han, and Soujanya Poria. 2022. SAT: Improving semi-supervised text classification with simple instance-adaptive self-training. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 6141–6146, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Jiaao Chen, Zichao Yang, and Diyi Yang. 2020. MixText: Linguistically-informed interpolation of hidden space for semi-supervised text classification. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2147–2157, Online. Association for Computational Linguistics.

Danilo Croce, Giuseppe Castellucci, and Roberto Basili. 2019. Kernel-based generative adversarial networks for weakly supervised learning. In *International Conference of the Italian Association for Artificial Intelligence*, pages 336–347. Springer.

Danilo Croce, Giuseppe Castellucci, and Roberto Basili. 2020. GAN-BERT: Generative adversarial learning for robust text classification with a bunch of labeled examples. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2114–2119, Online. Association for Computational Linguistics.

Hongyan Cui, Gangkun Wang, Yuanxin Li, and Roy E Welsch. 2022. Self-training method based on gcn for semi-supervised short text classification. *Information Sciences*, 611:18–29.

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

Xiaowei Gu. 2020. A self-training hierarchical prototype-based approach for semi-supervised classification. *Information Sciences*, 535:204–224.

Suchin Gururangan, Tam Dang, Dallas Card, and Noah A. Smith. 2019. Variational pretraining for semi-supervised text classification. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5880–5894, Florence, Italy. Association for Computational Linguistics.

Hwiyeol Jo and Ceyda Cinarel. 2019. Delta-training: Simple semi-supervised text classification using pretrained word embeddings. 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 3458–3463, Hong Kong, China. Association for Computational Linguistics.

Ju-Hyoung Lee, Sang-Ki Ko, and Yo-Sub Han. 2021. Salnet: Semi-supervised few-shot text classification with attention-based lexicon construction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 13189–13197.

Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo N Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick Van Kleef, Sören Auer, et al. 2015. Dbpedia–a large-scale, multilingual knowledge base extracted from wikipedia. *Semantic web*, 6(2):167–195.

Chen Li, Xutan Peng, Hao Peng, Jianxin Li, and Lihong Wang. 2021. Textgtl: Graph-based transductive learning for semi-supervised text classification via structure-sensitive interpolation. In *IJCAI*, pages 2680–2686.

Junnan Li, Pan Zhou, Caiming Xiong, and Steven CH Hoi. 2020. Prototypical contrastive learning of unsupervised representations. *arXiv preprint arXiv:2005.04966*.

Shujie Li, Min Yang, Chengming Li, and Ruifeng Xu. 2022. Dual pseudo supervision for semi-supervised text classification with a reliable teacher. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 2513–2518.

Hu Linmei, Tianchi Yang, Chuan Shi, Houye Ji, and Xiaoli Li. 2019. Heterogeneous graph attention networks for semi-supervised short text classification. 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 4821–4830, Hong Kong, China. Association for Computational Linguistics.

Chen Liu, Zhang Mengchao, Fu Zhibing, Panpan Hou, and Yu Li. 2021. FLiText: A faster and lighter semi-supervised text classification with convolution networks. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 2481–2491, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Ilya Loshchilov and Frank Hutter. 2018. Decoupled weight decay regularization. In *International Conference on Learning Representations*.

Andrew 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.

Subhabrata Mukherjee and Ahmed Hassan Awadallah. 2020. Uncertainty-aware self-training for text classification with few labels. *arXiv preprint arXiv:2006.15315*.

Amir Najafi, Shin-ichi Maeda, Masanori Koyama, and Takeru Miyato. 2019. Robustness to adversarial perturbations in learning from incomplete data. *Advances in Neural Information Processing Systems*, 32.

Yingwei Pan, Ting Yao, Yehao Li, Yu Wang, Chong-Wah Ngo, and Tao Mei. 2019. Transferrable prototypical networks for unsupervised domain adaptation. *CoRR*, abs/1904.11227.

Mengye Ren, Eleni Triantafillou, Sachin Ravi, Jake Snell, Kevin Swersky, Joshua B. Tenenbaum, Hugo Larochelle, and Richard S. Zemel. 2018. Meta-learning for semi-supervised few-shot classification. *CoRR*, abs/1803.00676.

Jake Snell, Kevin Swersky, and Richard Zemel. 2017. Prototypical networks for few-shot learning. *Advances in neural information processing systems*, 30.

Kihyuk Sohn, David Berthelot, Nicholas Carlini, Zizhao Zhang, Han Zhang, Colin A Raffel, Ekin Dogus Cubuk, Alexey Kurakin, and Chun-Liang Li. 2020. Fixmatch: Simplifying semi-supervised learning with consistency and confidence. *Advances in Neural Information Processing Systems*, 33:596–608.

Austin Cheng-Yun Tsai, Sheng-Ya Lin, and Li-Chen Fu. 2022. Contrast-enhanced semi-supervised text classification with few labels.

Ximei Wang, Jinghan Gao, Mingsheng Long, and Jianmin Wang. 2021. Self-tuning for data-efficient deep learning. In *International Conference on Machine Learning*, pages 10738–10748. PMLR.

Qizhe Xie, Zihang Dai, Eduard Hovy, Thang Luong, and Quoc Le. 2020. Unsupervised data augmentation for consistency training. *Advances in Neural Information Processing Systems*, 33.

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

Dataset	Supervised (all labels)		Supervised (10 labels)		Semi-supervised (10 labels)	
	BERT	RoBERTa	BERT	RoBERTa	PGPL(BERT)	PGPL(RoBERTa)
AG News	91.2	92.4	80.2	80.7	87.8	88.4
IMDB	90.4	93.5	70.9	71.2	88.9	91.2
Yahoo!Answer	73.7	74.2	60.1	61.0	67.4	67.8
DbPedia	99.1	99.1	96.6	96.1	98.7	98.8
Average	88.6	89.8	76.9	77.3	85.7	86.6

Table 7: Comparison with state-of-the-arts on the AG News, DBpedia, Yahoo! Answer and IMDB test set under different partition protocols. The results are averaged after three runs.

A Hyper-parameter Settings

For reproduction, we report our hyper-parameter settings in Table 8. The initial learning rate is tuned in [$1e^{-6}$, $1e^{-5}$] for BERT parameters and [$1e^{-4}$, $1e^{-3}$] for other parameters. The threshold β_2 and threshold λ are tuned in [0.1, 1]. Note that the hyper-parameter settings are tuned on the validation data by grid search with 3 trials.

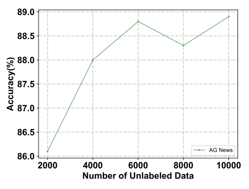
Hyper-parameter	PGPL
type embedding dimension d	768
Bert attention dropout	0.1
Bert hidden dropout	0.1
MLP hidden dimension	128
Sequence Length	256
batch size on labeled data	class numbers
batch size on unlabeled data	16
training epoch	20
initial learning rate of BERT	$1e^{-5}$
learning rate of MLP	$1e^{-3}$
threshold λ (AG News,IMDB)	0.2
threshold λ(DBpedia,Yahoo!Answer)	0.5
threshold β_1	1
threshold β_2	0.5
t memory set	5

Table 8: Hyper-parameter settings of PGPL.

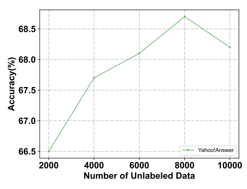
B Evaluation Results with other pre-trained models

In order to evaluate our model on different pre-training language models, we conduct three different experiments on supervised and semi-supervised settings: (1) supervised learning with all labels; (2) supervised learning with 10 labels; (3) and seimi-supervised with 10 labels. The experimental results are shown in Table 7. It is observed that all results with RoBERTa is better that those with BERT. In semi-supervised setting, PGPL is better than supervised setting with 10 available labels, and also not far behind the results of full supervised setting with all labels. In other words, PGPL is very efficient when exploiting few labeled data and achieves almost similar results to the supervised setting that fully exploit all labeled data. This indicates that PGPL is more practical in real-world scenarios where annotated data is limited.

C Evaluation Results When Varying Number of Unlabeled Data



(a) AG News



(b) Yahoo!Answer

Figure 5: The accuracy of PGPL when varying the number of unlabeled data on AG News and Yahoo!Answer Datasets.

We also conduct experiments to evaluate our model performances with fixed number of labeled data and different number of unlabeled data. Specifically, the former uses 10 labels and the latter uses between 2,000 and 10,000 labels on AG News and Yahoo! Answer. The results in Figure 5 show that with more unlabeled data, the accuracy increases significantly on both datasets, which further verify the effectiveness of utilizing unlabeled data.

ACL 2023 Responsible NLP Checklist

A For every submission:

☐ A1. Did you describe the limitations of your work?
Left blank.

☐ A2. Did you discuss any potential risks of your work?
Left blank.

☐ A3. Do the abstract and introduction summarize the paper's main claims?
Left blank.

☐ A4. Have you used AI writing assistants when working on this paper?
Left blank.

B ☐ Did you use or create scientific artifacts?

Left blank.

☐ B1. Did you cite the creators of artifacts you used?
Left blank.

☐ B2. Did you discuss the license or terms for use and / or distribution of any artifacts?
Left blank.

☐ B3. Did you discuss if your use of existing artifact(s) was consistent with their intended use, provided that it was specified? For the artifacts you create, do you specify intended use and whether that is compatible with the original access conditions (in particular, derivatives of data accessed for research purposes should not be used outside of research contexts)?
Left blank.

☐ B4. Did you discuss the steps taken to check whether the data that was collected / used contains any information that names or uniquely identifies individual people or offensive content, and the steps taken to protect / anonymize it?
Left blank.

☐ B5. Did you provide documentation of the artifacts, e.g., coverage of domains, languages, and linguistic phenomena, demographic groups represented, etc.?
Left blank.

☐ B6. Did you report relevant statistics like the number of examples, details of train / test / dev splits, etc. for the data that you used / created? Even for commonly-used benchmark datasets, include the number of examples in train / validation / test splits, as these provide necessary context for a reader to understand experimental results. For example, small differences in accuracy on large test sets may be significant, while on small test sets they may not be.
Left blank.

C ☐ Did you run computational experiments?

Left blank.

☐ C1. Did you report the number of parameters in the models used, the total computational budget (e.g., GPU hours), and computing infrastructure used?
Left blank.

The Responsible NLP Checklist used at ACL 2023 is adopted from NAACL 2022, with the addition of a question on AI writing assistance.

☐ C2. Did you discuss the experimental setup, including hyperparameter search and best-found hyperparameter values?
Left blank.

☐ C3. Did you report descriptive statistics about your results (e.g., error bars around results, summary statistics from sets of experiments), and is it transparent whether you are reporting the max, mean, etc. or just a single run?
Left blank.

☐ C4. If you used existing packages (e.g., for preprocessing, for normalization, or for evaluation), did you report the implementation, model, and parameter settings used (e.g., NLTK, Spacy, ROUGE, etc.)?
Left blank.

D ☐ Did you use human annotators (e.g., crowdworkers) or research with human participants?
Left blank.

☐ D1. Did you report the full text of instructions given to participants, including e.g., screenshots, disclaimers of any risks to participants or annotators, etc.?
Left blank.

☐ D2. Did you report information about how you recruited (e.g., crowdsourcing platform, students) and paid participants, and discuss if such payment is adequate given the participants' demographic (e.g., country of residence)?
Left blank.

☐ D3. Did you discuss whether and how consent was obtained from people whose data you're using/curating? For example, if you collected data via crowdsourcing, did your instructions to crowdworkers explain how the data would be used?
Left blank.

☐ D4. Was the data collection protocol approved (or determined exempt) by an ethics review board?
Left blank.

☐ D5. Did you report the basic demographic and geographic characteristics of the annotator population that is the source of the data?
Left blank.