ProtoRectifier: A Prototype Rectification Framework for Efficient Cross-Domain Text Classification with Limited Labeled Samples

Shiyao Zhao[1,2], Zhu Wang[1*], Dingqi Yang[2*], Xuejing Li[1], Bin Guo[1], Zhiwen Yu[1]

[1]Northwestern Polytechnical University
[2]University of Macau
zhaoshyarl@mail.nwpu.edu.cn, wangzhu@nwpu.edu.cn, dingqiyang@um.edu.mo,
lixuejing1207@mail.nwpu.edu.cn, {guob, zhiwenyu}@nwpu.edu.cn

Abstract

During the past few years, with the advent of large-scale pre-trained language models (PLMs), there has been a significant advancement in cross-domain text classification with limited labeled samples. However, most existing approaches still face the problem of excessive computation overhead. While some non-pretrained language models can reduce the computation overhead, the performance could sharply drop off. To resolve few-shot learning problems on resource-limited devices with satisfactory performance, we propose a prototype rectification framework, ProtoRectifier, based on pre-trained model distillation and episodic meta-learning strategy. Specifically, a representation refactor based on DistilBERT is developed to mine text semantics. Meanwhile, a novel prototype rectification approach (i.e., Mean Shift Rectification) is put forward by making full use of the pseudo-labeled query samples, so that the prototype of each category can be updated during the meta-training phase without introducing additional time overhead. Experiments on multiple real-world datasets demonstrate that ProtoRectifier outperforms the state-of-the-art baselines, not only achieving high cross-domain classification accuracy but also reducing the computation overhead significantly.

Introduction

The analysis of textual content is of critical importance in various research areas, such as social media analysis (Jiang, Ren, and Ferrara 2023), recommendation systems (Cao et al. 2023), semantic Web, etc. However, in real-world scenarios, the scarcity of clean corpora and the expensive cost of human annotation often limits the number of labeled samples for supervised learning tasks, e.g. text classification.

The recent development of Natural Language Processing (NLP) techniques has made significant advancements in the analysis of textual content with limited labels. For example, pre-trained models achieve satisfactory performance on tasks where training and testing samples belong to the same domain. However, such models have the shortcoming of limited cross-domain transferability. Thereby, more and more attention is being paid on fusing pre-training and domain generalization for the development of cross-domain text classification models.

*Corresponding authors.
Copyright © 2024, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

One of the most remarkable Pretrained Language Models (PLMs) is the BERT-based approach (Devlin et al. 2019), which can generate high-quality textual representations by learning word embeddings on large-scale corpora. However, the encoder turns out to be heavyweight with the improvement of performance. Although the use of non-pretrained language models offers a potential solution, the model's performance usually decreases significantly due to the absence of sufficient feature extraction. Similarly, while distillation methods provide another viable solution, the model's generalization ability decreases as there are fewer parameters. Therefore, it is necessary to build pre-trained models with both high performance and low computation overhead for cross-domain text classification, which is a meaningful yet lesser-attended issue. To address this issue, we choose to develop a lightweight pre-trained model with good domain generalization ability.

Currently, there are mainly three different domain generalization approaches (Wang et al. 2022), which are data augmentation (Wang et al. 2020), representation learning (Ganin and Lempitsky 2015; Li, Liu, and Bilen 2021) and meta-learning (Finn, Abbeel, and Levine 2017). The data augmentation approach is widely used to generate extra training samples, especially in the field of computer vision (CV), based on techniques such as panning, cropping, flipping, and adding noises to images. However, due to the difference between images and text, these techniques are not suitable for the augmentation of textual data. The representation learning approach (Ganin and Lempitsky 2015; Li, Liu, and Bilen 2021) studies domain-agnostic representation from the perspective of both mathematical modeling and machine learning. A challenge is that data sampling could become inefficient or unreliable during the training process. Compared with the above two approaches, meta-learning (Finn, Abbeel, and Levine 2017) (i.e., meta-transfer learning under cross-domain conditions) achieves cross-domain parameter adaptation and fine-tuning by exploring the relationship between the query and support sets. Specifically, episodic sampling is adopted in the meta-training framework, which ensures the model's cross-domain transferability.

While meta-learning helps to fine-tune the feature extractor for independent meta-tasks, the classification accuracy decreases as the number of domains increases. A pos-

sible solution is contrastive learning (Chen et al. 2022), which has received much attention recently. However, such sophisticated models consume a lot of computation resources. Another possible approach is adopting prototype networks (Snell, Swersky, and Zemel 2017; Ma et al. 2020) in the meta-learning framework. It has been proved that prototype learning is able to produce satisfactory outcomes with a good feature extractor (Dopierre, Gravier, and Logerais 2021a), and does not lead to significant computation overhead. Specifically, it first maps data samples into an embedding space, and then adopts the center of each class as a prototype for direct classification. For example, the HyperProto model (Ding et al. 2022) represents different classes geometrically using tensor fields, where the class information is represented by hyperspheres with dynamic sizes. Furthermore, to prevent from overfitting, the ProtAugment model (Dopierre, Gravier, and Logerais 2021b) introduces unsupervised cross-entropy loss and unlabelled instances. Nevertheless, given a limited number of labeled samples, it is a challenge to build precise prototypes, which limits the model's performance.

To address the above-mentioned challenge, we propose to design a pre-trained prototype rectification framework, aiming to enable efficient text classification with both high accuracy and low overhead. On one hand, while most existing studies (Mueller et al. 2022; Luo et al. 2021) focus on improving the classification accuracy of pre-trained models, less attention has been paid to the efficiency of model training. For example, by performing a secondary pre-training on tagged phrases from different fields, the Label Semantic Awareness Pre-training (LSAP) model (Mueller et al. 2022) integrates label semantics into the pre-trained generative encoder and constructs sentence-label pairs from unlabeled samples. The Label-semantic augmented meta-learner (LaSAML) framework (Luo et al. 2021) demonstrates that label information can be used to extract more discriminative feature representations with pre-trained language models (e.g., BERT). Nevertheless, considering that large-scale re-training from scratch will lead to a huge computation overhead, we choose to design **a representation refactor based on DistilBERT (i.e., RRED)** to speed up the training process. We optimize the text representation by further extracting domain-specific features.

On the other hand, considering that none of the existing studies has made full use of the information of query samples for few-shot text classification, we put forward **mean shift rectification (i.e. MSR)** to expand the corresponding support set by leveraging query samples. Specifically, since query samples are from the task domain, we can explore them to construct more suitable prototypes by updating the key pivots of each category in the meta-training phase. As a result, more appropriate prototypes can be obtained, based on which the model's classification accuracy can be guaranteed.

To sum up, we propose to facilitate efficient cross-domain text classification with limited labeled samples from two perspectives. First, we design the RRED module to lower the computation overhead. Second, we use the MSR module to obtain more representative prototypes, based on which the classification accuracy can be improved. In such a way, we achieve a satisfactory balance between the model's generalization ability and computation overhead, which has been overlooked in previous studies. The main contributions of this work are summarized as follows:

- We propose a novel prototype rectification framework (named ProtoRectifier) by combining meta-transfer learning based on pseudo-label augmentation and pre-training distilization for efficient text classification with limited labeled samples. ProtoRectifier achieves a satisfactory balance between the model's generalization ability and computation overhead.
- We design a representation refactor based on DistilBERT to optimize the initial representation by extracting domain-specific features. Moreover, we put forward mean shift rectification, based on which the query set samples can be explored to rectify pivot points generated with support set samples. As a result, more appropriate prototypes are obtained.
- We conduct experiments on multiple real-world datasets. Results show that the proposed model outperforms the state-of-the-art baselines, achieving both high classification accuracy and low computation overhead. To the best of our knowledge, this is the first work that systematically addresses the accuracy-efficiency balance issue of cross-domain text classification with limited labeled samples.

The rest of this paper is organized as follows. We present the related work in Section 2, followed by the details of the proposed prototype rectification framework in Section 3. We describe the experimental results in Section 4, and then conclude the paper in Section 5.

Related Work

Few-shot Learning

When solving few-shot problems, which are problems where a limited number of labeled samples are available, the construction of both training sets and test sets differs from traditional machine learning methods. Vinyals et al. (Vinyals et al. 2016) proposed an episodic strategy to simulate a real few-shot environment, thus improving the model's generalization ability by sampling the support set and the query set as meta-tasks. Specifically, the meta-learning paradigm obtains the ability of "learning to learn" (Hou et al. 2022) by generalizing the domain-agnostic features from source domains and fine-tuning models with limited labeled samples of target domains. Therefore, in this case, classes for validation and testing are invisible during the training process. In short, the training set $\mathcal{D}_{train}$ is composed of a large number of domains with limited labeled samples, whereas the test set $\mathcal{D}_{test}$ only contains a few shot of other domains. Concretely, sample labels in $\mathcal{D}_{train}$ and $\mathcal{D}_{test}$ do not intersect with each other.

Currently, meta-learning has been widely employed to solve the few-shot problem and existing studies can be classified into three categories. The first line of study is optimization-based meta-learning (Finn, Abbeel, and Levine

2017), which considers the internal task (i.e., the adaptive process) as an optimization problem and focuses on collecting the meta-knowledge necessary for performance improvement. While this line of research is effective, it suffers from the problem of memory overfitting (Rajendran, Irpan, and Jang 2020). The second line of study is model-based meta-learning (Munkhdalai and Yu 2017; Santoro et al. 2016), in which a feed-forward neural network is directly built by the meta-learning algorithm. It is inferior to optimization-based meta-learning due to the enormous instance distance. However, its performance is poor when dealing with supervised tasks. The third category is metric learning, which seeks to learn a metric space. The label of a testing sample can be predicted by simply assessing its similarity to training samples. Specifically, metric-based meta-learning techniques have demonstrated promising results, avoiding over-fitting when the class space changes.

Transfer Metric Learning

Metric learning is the process of creating a measurement space by machine learning and assessing the similarity of samples based on metrics such as Euclidean distance, cosine distance, etc. Based on the calculated similarity, an unlabeled sample can be predicted to belong to the nearest class. Metric-based transfer learning intends to improve the target metric by transferring metric information from the source domain. Thereby, a key issue is the optimization of the metric network.

Existing studies mainly focus on improving the metric network by introducing additional neural networks or external knowledge, such as siamese networks (Koch et al. 2015), matching networks (Vinyals et al. 2016), relational networks (Hu et al. 2018), induction networks (Geng et al. 2019), et al. However, most of these models consume a lot of computation resources. By contrast, prototypical networks (Snell, Swersky, and Zemel 2017) is a straightforward approach suitable for few-shot learning. To implement text classification, Fritzler et al. (Fritzler, Logacheva, and Kretov 2019) employed the prototypical network for named entity recognition by learning intermediate representations of words and aggregating them into named entity classes. Additionally, to mitigate the impact of imbalanced sample classes, Universal Prototype (Wu et al. 2021) investigated how object features could be enhanced using inherent properties that are shared across domains. PromptDA (Chen and Shu 2023) proposed a data augmentation method based on rich tag semantic information, which explored the importance of label semantics in the prompt-based learning paradigm. Even though these studies have provided a useful supplement to prototype learning, there still exists the misclassification issue when label semantics are similar to each other. In other words, it is not suitable enough to generate prototypes by simply averaging support vectors.

Unlike previous studies, we aim to minimize deviations from the expected prototype by making full use of the crucial data in query sets. Without the need for extra data sets, we alleviate the representation bias caused by sample scarcity. Based on knowledge distillation pre-training, the proposed framework leverages the query set and rectifies the prototype by mean shifting, as illustrated in Fig 3. Pseudo-labeled query samples enrich the class representation, which further enhance the classification performance. Specifically, the episodic iterative procedure increases the model's reliability without incurring additional computation overhead, and mitigates the accuracy loss due to the proposed lightweight pre-trained distillation encoder.

Methodology

In this section, we first describe the few-shot learning scenario and the related terminology. Then, we present the overall design of the proposed framework, i.e., ProtoRectifier. Finally, we provide a detailed explanation of the efficient encoder RRED and the base-learner MSR.

Problem Definition

Few-shot learning, which is also referred to as N-way K-shot problem, generates meta-tasks to stimulate the sample scarcity situation. In addition to training the model on a certain target task, meta-learning-based techniques learn meta-knowledge from different tasks to modify model parameters.

Typically, a data set is divided into three parts, including the training set $\mathcal{D}_{train}$, validation set $\mathcal{D}_{val}$, and test set $\mathcal{D}_{test}$. Before each training iteration, N labeled domains from the training set are randomly selected. Then, for each class, K samples are selected as the support set, and M samples are selected as the query set. Each episode is composed of a support set $\mathcal{S} = \{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=0}^{N\times K}$ and a query set $\boldsymbol{Q} = \{(\boldsymbol{x}_j, \boldsymbol{y}_j)\}_{j=1}^{N\times M}$, where $\boldsymbol{y}_i \in \{1, \cdots, N\}$. The validation set and test set tasks are constructed similarly. It should be noted that the label spaces of these three sets do not intersect with each other, i.e. $\mathcal{D}_{train} \bigcap \mathcal{D}_{val} \bigcap \mathcal{D}_{test} = \emptyset$. In the training stage, the classifier is trained on different meta-tasks with the loss calculated over the corresponding query set. In the testing stage, the episodic mechanism is utilized to adapt to the test set more rapidly. Under the setting of meta-learning, the model can generalize from labeled classes to unseen classes.

Framework Overview

This section introduces the general architecture and key components of the proposed framework, i.e., the $RRED$ representation module and the MSR prototype rectification module, as shown in Fig. 1. The overall design objective of ProtoRectifier is to maintain the model's generalization ability and make it more lightweight. Unlike previous studies that use large-scale PLMs directly for better encoding performance, we focus on applying an improved prototype learning method to a more lightweight pre-trained encoder, so as to achieve comparable performance as heavy models.

For each episode, the input is encoded by the $RRED$ module. A knowledge distillation-based BERT model is used to extract domain-agnostic features of each sample. Representation vectors are then fed into a recurrent neural network to obtain domain-specific features. Specifically, there exists a gap between original prototypes and target prototypes due to data scarcity. Therefore, we introduce a bias-reducing MSR module to rectify the class average proto-

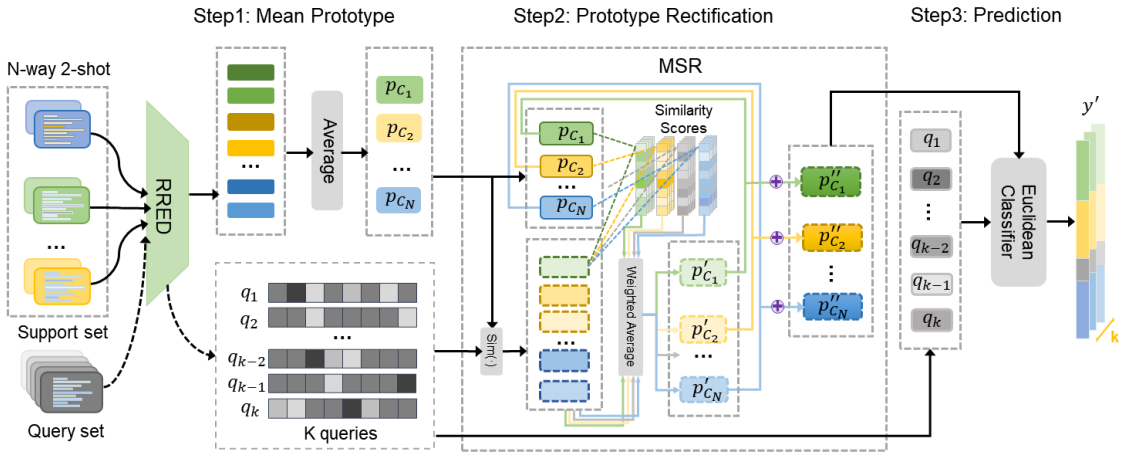


Figure 1: Framework of the prototype rectification model based on meta-learning. Function $sim(\cdot)$ denotes the similarity of the original prototype and the query samples, and " $+$ " surrounded by a circle means summation with a weighting factor.

type. Since the feature space of the query sample is the same as the corresponding support samples, it helps to construct more suitable prototypes. Adjusting the prototype through the augmented set can further improve the model's generalization ability.

Formally, let $\mathcal{S}'$ represent the augmented sample set and $\mathcal{S}$ represent the support set, the extended set can be denoted as $\widehat{\mathcal{S}} = \mathcal{S} \bigcup \mathcal{S}'$. Then, patterns are learned based on the extended support set $\widehat{\mathcal{S}}$, which enables the classifier to make predictions on the query set $\boldsymbol{Q}$.

Encoder Optimization Module

For this part, we begin by summarizing the encoder module from two aspects. The domain-agnostic feature extraction part learns task-agnostic representations to capture linguistic information. The domain-specific feature extraction part realizes initial representation optimization through a deep feature extraction layer. The key idea of our encoder module is to further extract the pre-trained student model, so as to preserve the reasoning effect of the teacher model as much as possible. Specifically, the gaps between different domains will become more evident after encoding through the $RRED$ module, laying a foundation for rectified prototype learning in the MSR module. Fig 2 illustrates an overview of the $RRED$ encoder.

Domain-agnostic feature extraction: Model distillation, i.e., teacher-student learning, is a technique for condensing large models. The key idea is to train small models (i.e., student models) to reproduce the output of large models given the same input. DistillBERT (Sanh et al. 2019) is a pre-trained model produced by applying the knowledge distillation method to the BERT model. It reduces the number of layers, and removes the token type embedding and next-sentence prediction tasks, while retaining the other mechanisms of BERT. It inserts a unique token [CLS] before the original text, and the encoder layer receives the

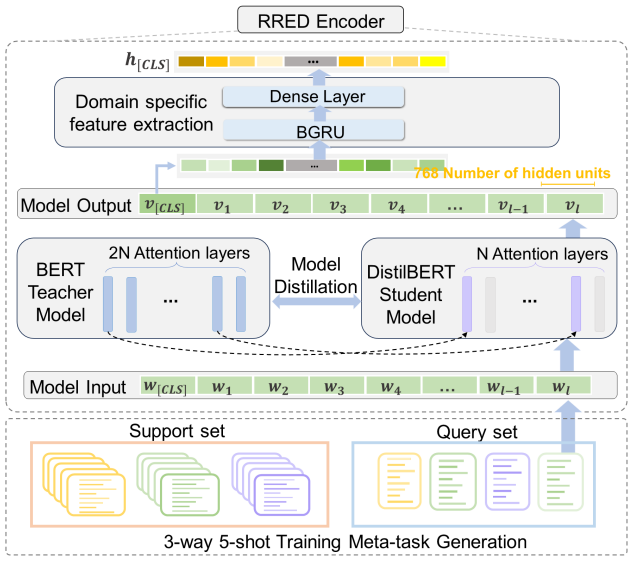


Figure 2: An illustration of the meta-task encoding process. w refers to a single word, v represents the word vector. $v_{[CLS]}$ denotes the sentence representation generated by the distilled pre-trained language model. The encoder output $h_{[CLS]}$ is a 768-dimensional vector.

token sequence as input and outputs the representation of token sequences. In this study, the text sentence is represented by the [CLS] token vector. Specifically, let $\boldsymbol{w} = \{w_{[CLS]}, w_1, \cdots, w_\ell\}$ be the original word sequence, the fundamental encoder layer outputs the symbolic representation $\boldsymbol{v} = \{v_{[CLS]}, v_1, \cdots, v_\ell\}$, where ℓ is the length of the notation sequence. The individual instances of the support set and query set are encoded separately.

Domain-specific feature extraction: Following the ex-

traction of domain-agnostic features, we introduce BGRU to further mine domain-specific features. The bidirectional gated recurrent unit provides domain-specific information to enhance the representative ability of cross-domain features. Since the process of classification is in a task-specific metric space, such configuration generates strong linkages between samples of the same type, providing a positive impact on subsequent prototype generation.

The module is composed of BGRU and a full connection layer. We set up the BGRU network with three hidden layers, where the hidden states are represented by the forward hidden unit $\overrightarrow{h_{t-1}}$ and the reverse hidden unit $\overleftarrow{h_{t-1}}$ as follows. Specifically, $\alpha^r, \beta^r, \gamma^r, U$ and ϵ represent weight parameters, $g_1(\cdot)$ and $g_2(\cdot)$ denote activation function, and h_{out} denotes the output of BGRU. The hidden size and dropout rate are set to 130 and 0.2, respectively.

$$\begin{aligned}
\overrightarrow{h_t^{(r)}} &= g_1(\overrightarrow{\alpha^r}\overrightarrow{h_t^{r-1}} + \overrightarrow{\beta^r}\overrightarrow{h_{t-1}^r}) + \overrightarrow{\gamma^r} \\
\overleftarrow{h_t^{(r)}} &= g_1(\overleftarrow{\alpha^r}\overleftarrow{h_t^{r-1}} + \overleftarrow{\beta^r}\overleftarrow{h_{t+1}^r}) + \overleftarrow{\gamma^r} \\
h_{out} &= g_2(U\left[\overrightarrow{h_t^{(r)}}; \overleftarrow{h_t^{(r)}}\right] + \epsilon)
\end{aligned} \tag{1}$$

We then obtain a 768-dimensional vector $h_{[CLS]}$ through a linear layer. In summary, the base-learner encoder is formalized as Eq. 2, where θ denotes the network parameter.

$$\boldsymbol{h}_{[CLS]} = RRED_{[CLS]}(\boldsymbol{w}; \theta) \tag{2}$$

Mean Shift Rectification

In this section, we present the details of the MSR module, which is the kernel of the proposed framework. We first introduce the metric-based prototypical network, and then describe the proposed prototype mean shift rectification method.

Euclidean distance based prototype generation: Inductive bias is used in prototypical networks (Snell, Swersky, and Zemel 2017) to map each sample onto a hyperspace. The fundamental concept is to create a prototypical vector representing each class. In particular, a prototype is usually created by averaging the embedding representations of all data samples of a certain class. We define $\boldsymbol{P}_c$ as the obtained prototype, and let $\boldsymbol{x}_s^c$ and y_s^c represent the original support set and label of domain c, respectively. We calculate the proto-vector with Eq. 3, where $\mathcal{S}_c$ is a subset corresponding to domain c of support set $\mathcal{S}$.

$$\boldsymbol{p}_c = \frac{1}{|\mathcal{S}_c|} \sum_{(\boldsymbol{x}_s^c, y_s^c) \in \mathcal{S}_c} RRED_\theta(x_s^c) \tag{3}$$

To simply the above formula, we define $f_\theta(\cdot) = RRED_\theta(\cdot)$, which represents the optimized feature extractor. Based on the obtained class prototypes, the distribution $\mathcal{P}$ of predicted labels for the query sample $\{\boldsymbol{x}_q, y_q\} \in \boldsymbol{Q}$ can be represented as the softmax values of the distances between the input vector and class centers. Given a sample x_q, its probability of belonging to class c is computed as Eq.4, where $dist(\cdot,\cdot)$ denotes the similarity function, which can

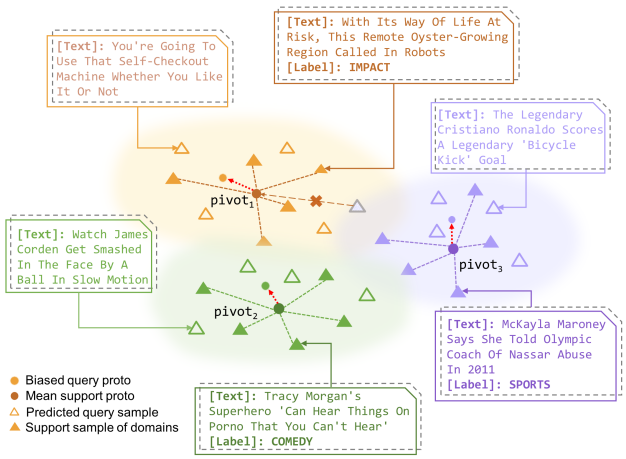


Figure 3: An illustration of prototype rectification for 3-way 5-shot task. The large solid circles that denote support prototypes will be rectified by augmented query prototypes.

be Euclidean distance or cosine distance, etc.

$$\mathcal{P}(y_q = c|\boldsymbol{x}_q, \mathcal{S}; \theta) = \frac{\exp(-dist(f_\theta(\boldsymbol{x}_q), \boldsymbol{p}_c))}{\sum_{z=1}^{N} \exp(-dist(f_\theta(\boldsymbol{x}_q), \boldsymbol{p}_z)} \tag{4}$$

Mean shift rectification for prototype: For N-way K-shot learning, as illustrated in Fig. 3, there are K samples for each class, i.e., available samples are much fewer than expected. Thus, the generated prototype tends to be biased due to data scarcity. To address this issue, we propose MSR to reduce bias using query set samples.

Since there exists distributional variance between the support set and the query set, we introduce a shifting term ξ to rectify such cross-domain bias. The shifting term ξ guarantees the lower bound when using query set data. Following the theoretical analysis of Liu et al. (Liu, Song, and Qin 2020), we propose to reduce the distributional deviation by computing ξ as Eq. 5, where $\mathcal{S}$ and $\boldsymbol{Q}$ represent the support set and the query set of N domains. By adding the shift term ξ to the query set, it will shift towards the support set and the intra-class bias can be reduced accordingly.

$$\boldsymbol{\xi} = \frac{1}{|\mathcal{S}|}\sum_{m=1}^{|\mathcal{S}|} f_\theta(\boldsymbol{x}_s^m) - \frac{1}{|\boldsymbol{Q}|}\sum_{n=1}^{|\boldsymbol{Q}|} f_\theta(\boldsymbol{x}_q^n). \tag{5}$$

First, we compute the original support prototype for each class according to Eq. 3. Then, the distance between each query sample and the support prototype pivot can be calculated. To highlight the sample similarity, we adopt an exponential function for re-scaling based on a relaxation factor ζ, which is defined by the slack of the metric. Consequently, the similarity is calculated as follows.

$$sim(\boldsymbol{p}_i, \boldsymbol{q}_j) = \exp(\zeta \cdot dist(\boldsymbol{p}_i, \boldsymbol{q}_j)), \tag{6}$$

where $sim(\cdot)$ denotes the similarity between the j^{th} query sample $\boldsymbol{q}_j$ and the i^{th} class support prototype $\boldsymbol{p}_i$. The similarity of each query sample is then normalized by softmax

to obtain a score, the highest of which is considered as the pseudo-label $\hat{y_q}$ as Eq. 7.

$$\hat{y_q} = \arg\max \frac{sim(\boldsymbol{p}_c, \boldsymbol{q}_j)}{\sum_k^N sim(\boldsymbol{p}_k, \boldsymbol{q}_j)} \tag{7}$$

Once pseudo-labels for all query samples are obtained, we can generate an expanded set $\mathcal{S}'$, which effectively increases the confidence level of the prototype. With such an "enriched" data set, the rectified prototype can be calculated accordingly.

It should be noted that the contribution of samples in the expanded set is different from samples in the original support set. Specifically, samples closer to the center of the class should receive more attention. We thus calculate the weighted summation of query samples to generate query set correction prototypes $\boldsymbol{p'}$, and obtain the rectified prototype $\boldsymbol{p''}$ by refactoring the prototype $\boldsymbol{p}$ generated from the support set to the expected class center. The corrected prototype of the query set is defined as follows.

$$\boldsymbol{p}'_c = \sum_{i=1}^{Z} score_i^c \cdot \boldsymbol{q}_i^c, \tag{8}$$

where $score_i^c$ represents the weight of each augmented sample i with pseudo-label c and $\boldsymbol{p'}$ denotes the query set correction prototype. Z is the number of augmented query samples in each class. The similarity weight score is computed as Eq. 9.

$$Score_i^c = \frac{sim(\boldsymbol{q}_i^c, \boldsymbol{p}_c)}{\sum_k^C sim(\boldsymbol{q}_i^k, \boldsymbol{p}_c)}, \tag{9}$$

where $\boldsymbol{p}_c$ is the original prototype calculated by Eq. 3. Specifically, the weighted query prototype is designed based on the consideration that samples with high similarity to the base class center should play a more significant role in prototype rectification.

Since some pseudo-labeled samples are likely to be misclassified, a simple average operation with the same weights may lead to a larger bias. Therefore, we adjust the prototype weights of support and query sets by means of applying the relaxation factor λ. Specifically, after shifting the original mean value prototype $\boldsymbol{p}$ of the support set towards the augmented prototype $\boldsymbol{p'}$, the rectified prototype $\boldsymbol{p''}$ can be obtained as Eq. 10.

$$\boldsymbol{p''} = \lambda \boldsymbol{p} + (1 - \lambda)\boldsymbol{p'} \tag{10}$$

To sum up, the MSR module updates the prototype iteratively during the meta-task training process by making full use of query samples, without introducing additional computation overhead.

The ProtoRectifier Strategy

To further improve the performance of the modified prototypical network, we put forward an end-to-end meta-training strategy in this section. The basic idea is to extract task-agnostic meta-knowledge about the rectified prototype from base classes and then apply the knowledge to new classes. Such a training strategy makes it possible to generate information for cross-domain downstream tasks, thus enhancing the efficiency of domain adaptation. The learning process of ProtoRectifier is summarized in Algorithm 1.

Algorithm 1: Learning Process of ProtoRectifier.

Input: Source domain training set $\mathcal{D}_{train}$; other domains validation set $\mathcal{D}_{val}$; hyper-parameters: Ψ.
Output: The updated network including encoder parameters θ and meta-learner parameter ϕ.

1: Initialize $\theta \leftarrow \theta_0, \phi \leftarrow \phi_0$;
2: **for all** iteration episode **do**
3: Sampling N domains from $\mathcal{D}_{train}$;
4: **for** $i = 1$ to N **do**
5: $\mathcal{S}_i, \boldsymbol{Q}_i \leftarrow$ For each class u in $\mathcal{D}_{train}$, sample k items as support set and m items as query set randomly;
6: **end for**
7: Encode $\mathcal{S}, \boldsymbol{Q}$ with $RRED$ module;
8: $\boldsymbol{x}_q \leftarrow \text{Add}(\{\boldsymbol{x}_q, \xi\})$; $\boldsymbol{p}_i \leftarrow \text{Average}(\{\boldsymbol{x}_s^i, \boldsymbol{y}_s^i\})$
9: $\mathcal{S}' \leftarrow \varnothing$ // Initialize the augmented set
10: **for** $\boldsymbol{q}_j = \{\boldsymbol{x}_q^j, \boldsymbol{y}_q^j\}$ in $\boldsymbol{Q}$ **do**
11: $\mathcal{S}_i^{'} \leftarrow \arg\max(\boldsymbol{q}_j, \{\boldsymbol{p}_1, \cdots, \boldsymbol{p}_N\})$
12: **end for**
13: $\boldsymbol{p}_i^{'} = \sum_k^{\mathcal{S}_i^{'}} \frac{sim(\boldsymbol{q}_k, \boldsymbol{p}_i)}{\sum_t^{\mathcal{S}_i^{'}} sim(\boldsymbol{q}_t, \boldsymbol{p}_i)} \cdot \boldsymbol{q}_k$
14: $\boldsymbol{p}_i^{''} = \lambda \boldsymbol{p}_i + (1 - \lambda)\boldsymbol{p}_i^{'}$
15: Update $\theta, \phi \leftarrow \text{ProtoRectifier}(\mathcal{D}_{val}, \Psi, \theta, \phi)$;
16: **end for**

As shown in Algorithm 1, we first sample the N-way K-shot meta-tasks from the training and validation process to generate the support set $\mathcal{S}$ and query set $\boldsymbol{Q}$ from $\mathcal{D}_{train}$ and $\mathcal{D}_{val}$ (Lines 3-6). Then, the generated meta-task is encoded by the $RRED$ module (Line 7). Next, we make label predictions on samples in the rectified query set and add them to the corresponding support set $\mathcal{S}$ to generate the augmented support set $\hat{\mathcal{S}} = \mathcal{S} \bigcup \mathcal{S}'$ (Lines 8-12). The weighted incremental correction prototype is calculated using the extended support set S' according to Eq. 8 (Line 13). Finally, we rectify the prototype based on Eq. 10 (Lines 14-15). In such a way, we effectively utilize the data in the query set to correct the prototypes.

Experiment

Experimental Setup

Dataset Description To demonstrate the effectiveness of ProtoRectifier, we evaluate its performance on three public datasets in the few-shot scenario. Table 2 summarises the statistics of the used datasets.

ARSC[1] **(Blitzer, Dredze, and Pereira 2007):** ARSC is a multi-domain sentiment classification dataset, which contains Amazon product reviews for 23 products. Each domain contains three classification tasks with different rating thresholds. In this paper, we select 12 (4×3) tasks from four domains (including books, DVDs, electronics, and kitchen

[1] https://github.com/Gorov/DiverseFewShot_Amazon

20News	Attention-BiLSTM+Meta-Learner		BERT+Meta-Learner		LaSAML	ProtoRectifier
	Other Metric Leaner	Induction	Other Metric Leaner	Induction		
1-shot	2112MiB	5596MiB	5198MiB	7394MiB	18534MiB	3512MiB
5-shot	3354MiB	9418MiB	6174MiB	8066MiB	18612MiB	3808MiB

Table 1: Comparison of FLOPs on the 20news dataset.

Datasets	sent_num	train/val/test	avg_sent_len
ARSC	206,913	19/19/4	98
HuffPost	36,900	27/6/8	11
20News	18,820	9/5/6	340

Table 2: The detailed dataset statistics.

Method	w/ distil	Accuracy	Test time	Variance
w/o MSR	✓	84.85	8.19	0.33
w/o $RRED$	✓	86.42	8.62	0.11
ProtoRectifier	✓	87.56	8.65	0.15

Table 3: Ablation study on the ARSC dataset.

housewares) as test tasks and use the remaining 57 tasks as the training set.

HuffPost (Misra 2018; Misra and Grover 2021): It is a dataset containing HuffPost news topics. For a fair comparison, we allocate 27, 6, and 8 tasks of different topics to training, validation, and test sets, respectively. Specifically, there are eight selected test domains, which are topics of "HEALTHY LIVING", "DIVORCE", "WORLDPOST", "STYLE", "MONEY", "ENVIRONMENT", "CULTURE&ARTS" and "EDUCATION".

20News (Lang 1995): It contains newsgroup documents of 20 different topics, such as politics, sports, science, etc. Some of the topics are completely unrelated and therefore suitable for the evaluation of cross-domain text analysis. The topic selected for test are "rec.sport.baseball", "misc.forsale", "sci.space", "comp.sys.ibm.pc.hardware", "soc.religion.christian" and "talk.politics.mideast".

Baselines To evaluate ProtoRectifier, a set of baseline models are explored for performance comparison, which can be divided into three categories. First, for all three datasets, four metric learning methods (i.e., matching network, prototype network, relation network, and induction network) are selected to construct classification models with both the non-pretrained language model (i.e., ATTBI) and the pretrained language model (i.e., BERT). Second, for the ARSC dataset, several other models are used for experimental evaluation, including ATTBI+NTL, BERT+NTL, MEDA-PN, and MemIML, as these models are not suitable for the other two datasets. Specifically, NTL can be used to demonstrate the significance of the metric learning module. Third, for the 20News and HuffPost datasets, several latest models are used for experiments, including DS+RRML, LasAML, LEA, MetaPrompting, and TART.

NTL: NTL (i.e., neural tensor layer) is a fundamental classifier, which makes classification predictions by simply constructing a single-layer feed-forward neural network.

Matching Network (Vinyals et al. 2016): A metric learning model that adopts the attention mechanism to analyze the similarity between feature vector pairs.

Prototype Network (Snell, Swersky, and Zemel 2017): A metric learning model that maps samples into a high-dimensional space and generates the average value of each class as the class prototype.

Relation Network (Hu et al. 2018): The relation network applies a non-linear classifier to measure the similarity between the class center and the sample.

Induction Network (Geng et al. 2019): It is a combination of meta-learning and dynamic routing (Sabour, Frosst, and Hinton 2017), which aims to improve the text classification performance.

MEDA-PN (Sun et al. 2021): It is proposed to compute the minimum enclosing ball of the support set and synthesize the samples for data augmentation.

MemIML (Zhao et al. 2022): It is a memory imitation meta-learning approach that enhances the model's dependence on the support set when adapting to a new task.

RRML (Bertinetto et al. 2019): RRML computes class vectors by solving the ridge regression problem on the support set.

LaSAML (Luo et al. 2021): The Label-semantic Augmented Meta-Learner attaches class names to the input sentence and investigates the potential of using class name information for few-shot text classification.

LEA (Hong and Jang 2022): LEA is an embedding transfer method as a way to gain task-level attention through a meta-learning framework.

MetaPrompting (Hou et al. 2022): MetaPrompting is proposed to address the problem that soft prompt learning relies heavily on good initialization, by designing a generalized soft prompt framework to improve the model's generalization ability.

TART (Lei et al. 2023): Task adaptive networks are proposed to improve the discrimination of similar semantics by mapping samples into a task-relevant space.

Implementation Details To compare the performance of different models, we conduct experiments for 2-way 5-shot tasks with the ARSC sentiment classification dataset. Meanwhile, with the 20news and HuffPost datasets, we conduct experiments for 1-shot and 5-shot tasks. We randomly generate 20,000 training episodes, 2,000 validation episodes and 5,000 test episodes.

Method	Metric	Accuracy(%)				Test time(μs/10 queries)				Variance			
		20News		HuffPost		20News		HuffPost		20News		HuffPost	
		1 shot	5 shot	1 shot	5 shot	1 shot	5 shot	1 shot	5 shot	1 shot	5 shot	1 shot	5 shot
ATTBI+Matching	cosine	35.64	44.01	34.36	48.5	57.17	61.12	16.34	17.88	6.28	8.5	1.32	2.39
	euclid.	36.87	45.30	31.06	44.49	45.57	101.90	30.58	32.16	8.20	1.98	0.69	2.19
ATTBI+Prototype	euclid.	37.86	46.96	32.79	45.09	61.72	69.00	18.59	19.09	4.76	6.65	2.23	1.52
	cosine	36.97	48.11	38.96	46.12	48.13	59.62	29.95	34.84	9.00	2.78	0.68	1.02
ATTBI+Relation	-	33.93	43.66	21.72	28.63	77.10	78.35	23.43	38.05	1.18	1.36	1.85	0.08
ATTBI+Induction	-	30.34	35.91	30.76	37.07	27.94	42.74	24.02	24.86	0.89	0.23	1.09	0.65
BERT+Matching	cosine	60.54	64.98	42.23	48.59	16.70	18.01	18.97	19.33	0.67	0.22	0.7	0.8
	euclid.	61.53	65.80	45.98	69.52	16.23	16.8	18.16	18.81	0.16	0.28	1.27	0.73
BERT+Prototype	euclid.	64.26	76.91	46.93	68.05	14.49	17.08	18.03	18.62	0.13	0.04	0.77	0.29
	cosine	63.11	74.34	43.10	57.82	17.52	19.70	16.81	17.46	0.29	0.15	0.11	0.33
BERT+Relation	-	53.86	67.78	41.29	54.88	17.14	18.78	20.81	21.15	0.65	0.15	0.33	0.14
BERT+Induction	-	58.22	60.03	40.62	57.16	26.61	52.58	24.19	24.97	0.97	**0.02**	0.85	0.22
BERT+RRML	-	32.93	49.78	41.36	61.53	1039.56	1416.08	66.47	68.08	0.07	0.08	0.09	0.10
DS+RRML(2020)	-	52.36	68.99	42.41	62.29	20.43	26.69	16.21	16.87	**0.01**	0.09	0.08	0.09
LaSAML(2021)	-	51.84	66.93	62.53	70.10	77.3	139.34	30.41	32.53	0.13	0.67	0.13	0.11
LEA(2022)	-	53.47	65.88	48.43	71.6	-	-	-	-	-	-	-	-
MetaPrompting(2022)	-	68.83	82.95	**71.93**	76.32	-	-	-	-	-	-	-	-
TART(2023)	-	67.0	83.2	46.9	66.8	34.62	30.81	30.43	31.47	0.08	0.08	0.10	0.11
ProtoRectifier(Ours)	euclid.	**70.56**	**85.48**	62.10	**77.67**	**8.48**	**8.51**	**7.76**	**7.93**	0.04	0.05	**0.07**	**0.08**

Table 4: Experiment results of news datasets (20News & HuffPost) on 5-way k-shot tasks, where k is set to 1 and 5.

ProtoRectifier is implemented using PyTorch with smooth gradient descent training and iterative parameter updating based on the Adam optimizer (Loshchilov and Hutter 2019) and weight decay (0.01). Specifically, we set the relaxation factor λ to 0.7, the initial learning rate to 7e-5, the gradient step to 32, the warm-up learning rate to 0.06, and the dropout to 0.4 to prevent over-fitting. The experiment is done using a single NVIDIA GeForce RTX 3090, if not specified otherwise. On a hold-out validation set, hyper-parameters are further fine-tuned.

Experimental Results

We analyze the experimental results from the perspective of classification accuracy and computational overhead in this section.

Classification Accuracy The experimental results are shown in Tables 5 and 4. Accordingly, we can observe that ProtoRectifier outperforms the classification accuracy of other baseline methods in most cases, especially on the 20news dataset. Such a result demonstrates that ProtoRectifier is superior in few-shot learning. Specifically, key findings are summarized as follows.

(1) Compared with baseline methods, the classification accuracy of the proposed model improves approximately 2% over the best baseline on the ARSC dataset. Similarly, there are 2.28% improvement on the 20news dataset and 1.35% improvement on the HuffPost dataset for the 5-shot task. Specifically, even though the accuracy decreases slightly for the 1-shot task on the HuffPost dataset, it is still higher than the latest model TART and is competitive with LaSAML.

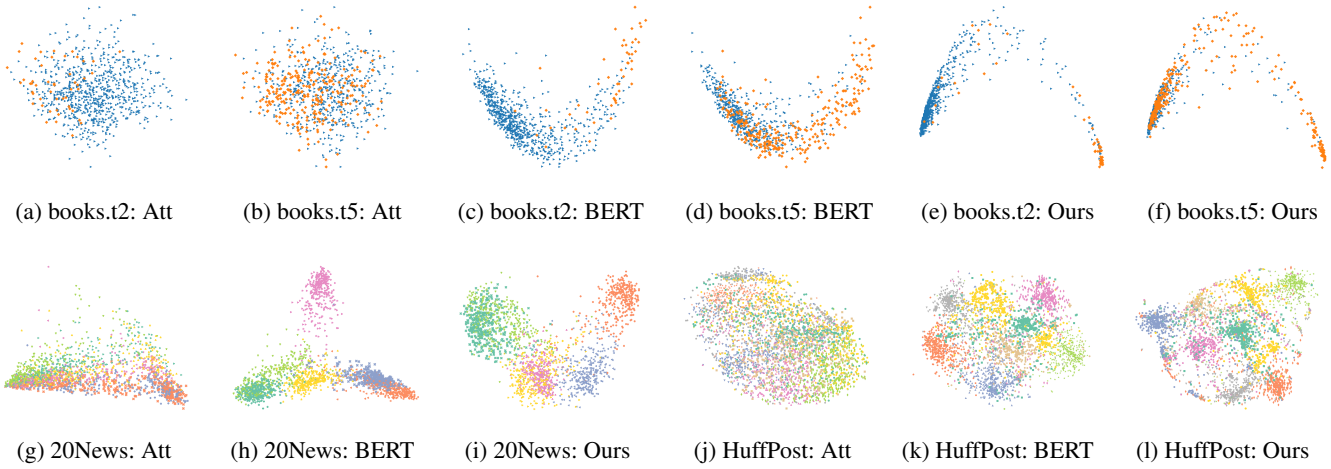
(2) In general, the BERT pre-trained language model effectively improves the performance of the meta-leaner, indicating that traditional meta-classifiers can achieve competitive results together with a transformer-based encoder. In particular, the prototypical network outperforms other classifiers, which does not depend on the weight matrix and parameters. Moreover, since BERT handles contextual classification better than keyword-based classification, "DS+RRML" performs better than "BERT+RRML" on the 20news and HuffPost datasets.

Time Consumption We evaluate the time consumption of different models by comparing the calculation time for the prediction of 10 queries. The results are illustrated in Tables 5 and 4. We can see that, compared with the baselines, ProtoRectifier has the lowest test time on all three datasets. Specifically, we have the following observations.

(1) Compared with baseline models, ProtoRectifier achieves a significant improvement in the inference speed. For example, it is 12x faster than LaSAML on the 20news dataset and 2x faster than the BERT-based prototype network, demonstrating the effectiveness of model distillation. In particular, due to the specific requirements of the DS+RRML model for text feature extraction, we chose not to compare the corresponding time overhead.

(2) Compared with the BERT pre-training models, the attention-based encoding models result in much higher time overhead. Meanwhile, there is no significant difference in time consumption between different BERT-based models, indicating that the meta-learner has less impact on the model's time complexity than the encoding module.

(3) In general, the variance of time consumption of pre-trained language models is smaller than attention-based models, where a lower variance indicates the performance is more stable. Such a result suggests that the proposed method



(a) books.t2: Att (b) books.t5: Att (c) books.t2: BERT (d) books.t5: BERT (e) books.t2: Ours (f) books.t5: Ours

(g) 20News: Att (h) 20News: BERT (i) 20News: Ours (j) HuffPost: Att (k) HuffPost: BERT (l) HuffPost: Ours

Figure 4: Representation visualization of query texts. Subfigures (a-f) represent the visualization results of Attention ProtoNet, BERT ProtoNet, and ProtoRectifier on the books.t2 and books.t5 domains of the ARSC dataset. Subfigures (g-i) and (j-l) are the visualization result of the 20News and HuffPost datasets.

is stable in time consumption when used in new domains.

Memory Consumption We present the memory consumption (i.e., FLOPs) of different models on the 20news dataset in Table 1. It can be seen that BERT-based models consume much higher memory resources. Additionally, when configured with the same encoding module, complex classifiers such as the induction network result in larger FLOPs, due to the fact that they have much more parameters. Based on distillation compression, the FLOPs of the proposed model have dropped 5x compared with that of the LaSAML model. Such a performance is comparable to non-pretrained language models based on the attention mechanism.

Different Distance Metrics Considering that the performance of both the matching network and the prototype network rely on the distance metric used for feature similarity estimation, we adopt two metrics which are cosine distance and Euclidean distance for experiments, as illustrated in Table 5 and 4. Results show that there exist performance differences between the two metrics, indicating that the choice of distance metric has an indispensable impact on the model's performance. Specifically, although the matching network is designed to use the cosine distance, it achieves better performance when applying the Euclidean distance for similarity calculation.

Ablation Study To further validate the effectiveness of ProtoRectifier, a set of ablation experiments are conducted. Specifically, we consider two ablated models of ProtoRectifier. The first one removes the proto-refactor from the framework, and is named as w/o MSR. The second one removes the BGRU optimization layer, and is represented as w/o $RRED$. The corresponding results are shown in Table 3.

We find that both of the ablated models lead to worse performance than the full model, which confirms the necessity of the respective modules. In particular, removing the MSR module leads to a larger variance in time consumption, indicating that the model's stability declines. Meanwhile, the $RRED$ module is capable of improving the model's classification accuracy while bringing in a slight time overhead.

Method		Metric	Accuracy	Test time	Variance
ATTBI	NTL	-	59.95	13.04	2.86
	Matching	cosine	70.17	17.41	4.03
		euclid.	69.43	32.69	10.93
	Prototype	euclid.	71.81	12.97	5.66
		cosine	72.84	31.53	10.48
	Relation	-	68.18	40.52	11.27
	Induction	-	72.26	42.08	5.74
BERT	NTL	-	62.35	17.80	1.51
	Matching	cosine	81.34	15.44	0.19
		euclid.	77.87	16.03	0.17
	Prototype	euclid.	82.27	15.68	0.14
		cosine	81.69	15.72	**0.11**
	Relation	-	83.12	20.54	0.13
	Induction	-	85.18	28.22	0.19
MEDA-PN(2021)		-	85.68	-	-
MemIML(2022)		-	85.69	-	-
ProtoRectifier(Ours)		euclid.	**87.56**	**8.65**	0.15

Table 5: Experimental results of the ARSC dataset on 2-way 5-shot tasks, including the classification accuracy (%) and the test time (μs/10 queries) as well as its variance.

Text Representation Visualization

We visualize the text representation results of the query set based on PCA (Hotelling 1933), as shown in Figure 4. Specifically, three models are compared, which are prototype network with attention-based word vector encoder, Protonet with BERT encoder, and the proposed ProtoRectifier. The generated text embeddings are mapped into 2-

dimensions, and different colored dots are used to represent samples from different domains.

First, we find that vectors generated by the Attention ProtoNet model fail to distinguish samples with different labels. Second, the BERT ProtoNet model produces more discriminative representations of query texts, even though the boundaries of domains with similar meanings are blurred. Third, compared with these two models, the proposed ProtoRectifier has much better discriminative capability. Specifically, reducing the distance between similar samples makes the discrepancy between different classes more significant. To sum up, the visualization result demonstrates the effectiveness of ProtoRectifier in generating discriminative representations for few-shot text classification tasks.

Conclusion

To solve the limited labeled text classification problem in cross-domain scenarios, we proposed a prototype rectification framework (i.e., ProtoRectifier) based on meta-learning. On one hand, an encoder optimization module is designed to generate better text representations as well as improve the inference speed. On the other hand, a mean shift rectification module is developed to acquire precise prototypes of distinct domains. To verify the effectiveness of the proposed framework, We conducted extensive experiments based on three real-world datasets. Results demonstrated that the proposed framework achieves not only high classification accuracy but also low computation overhead, which significantly outperforms state-of-the-art baselines.

Limitations

Due to the limitations of the available dataset, only text data is considered in this study. In future research work, the generalization ability of multimodal data can be further investigated. Moreover, the computation efficiency is not tested on some complicated models such as MetaPrompting due to resource constraints.

Acknowledgments

This work is partially supported by the National Natural Science Foundation of China (No. 61960206008, 62032018, 62072375, 62102322), the Science and Technology Development Fund, Macau SAR (0047/2022/A1, 001/2024/SKL), and the University of Macau (SRG2021-00002-IOTSC).

References

Bertinetto, L.; Henriques, J. F.; Torr, P. H. S.; and Vedaldi, A. 2019. Meta-learning with differentiable closed-form solvers. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

Blitzer, J.; Dredze, M.; and Pereira, F. 2007. Biographies, Bollywood, Boom-boxes and Blenders: Domain Adaptation for Sentiment Classification. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*. Prague, Czech Republic: Association for Computational Linguistics.

Cao, Q.; Yin, Q.; Song, Y.; Wang, Z.; Chen, Y.; Xu, R. Y. D.; and Yang, X. 2023. RTANet: Recommendation Target-Aware Network Embedding. 17: 84–94.

Chen, C.; and Shu, K. 2023. PromptDA: Label-guided Data Augmentation for Prompt-based Few Shot Learners. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*. Dubrovnik, Croatia: Association for Computational Linguistics.

Chen, J.; Zhang, R.; Mao, Y.; and Xu, J. 2022. Contrastnet: A contrastive learning framework for few-shot text classification. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, 10492–10500.

Devlin, J.; Chang, M.-W.; Lee, K.; and Toutanova, K. 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*, 4171–4186. Minneapolis, Minnesota: Association for Computational Linguistics.

Ding, N.; Chen, Y.; Cui, G.; Wang, X.; Zheng, H.; Liu, Z.; and Xie, P. 2022. Few-shot Classification with Hypersphere Modeling of Prototypes. *CoRR*, abs/2211.05319.

Dopierre, T.; Gravier, C.; and Logerais, W. 2021a. A Neural Few-Shot Text Classification Reality Check. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*. Association for Computational Linguistics.

Dopierre, T.; Gravier, C.; and Logerais, W. 2021b. ProtAugment: Intent detection meta-learning through unsupervised diverse paraphrasing. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing*, 2454–2466. Association for Computational Linguistics.

Finn, C.; Abbeel, P.; and Levine, S. 2017. Model-agnostic meta-learning for fast adaptation of deep networks. In *International conference on machine learning*, 1126–1135. PMLR.

FORCE11. 2020. The FAIR Data principles. https://force11.org/info/the-fair-data-principles/.

Fritzler, A.; Logacheva, V.; and Kretov, M. 2019. Few-shot classification in named entity recognition task. In *Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing*, 993–1000.

Ganin, Y.; and Lempitsky, V. 2015. Unsupervised domain adaptation by backpropagation. In *International conference on machine learning*, 1180–1189. PMLR.

Gebru, T.; Morgenstern, J.; Vecchione, B.; Vaughan, J. W.; Wallach, H.; Iii, H. D.; and Crawford, K. 2021. Datasheets for datasets. *Communications of the ACM*, 64(12): 86–92.

Geng, R.; Li, B.; Li, Y.; Zhu, X.; Jian, P.; and Sun, J. 2019. Induction Networks for Few-Shot 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)*. Hong Kong, China: Association for Computational Linguistics.

Hong, S. K.; and Jang, T. Y. 2022. LEA: Meta Knowledge-Driven Self-Attentive Document Embedding for Few-Shot Text Classification. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 99–106. Seattle, United States: Association for Computational Linguistics.

Hotelling, H. 1933. Analysis of a complex of statistical variables into principal components. *Journal of educational psychology*, 24(6): 417.

Hou, Y.; Dong, H.; Wang, X.; Li, B.; and Che, W. 2022. MetaPrompting: Learning to Learn Better Prompts. In *Proceedings of the 29th International Conference on Computational Linguistics*, 3251–3262. Gyeongju, Republic of Korea: International Committee on Computational Linguistics.

Hu, H.; Gu, J.; Zhang, Z.; Dai, J.; and Wei, Y. 2018. Relation networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 3588–3597.

Jiang, J.; Ren, X.; and Ferrara, E. 2023. Retweet-BERT: Political Leaning Detection Using Language Features and Information Diffusion on Social Networks. *Proceedings of the International AAAI Conference on Web and Social Media*, 17(1): 459–469.

Koch, G.; Zemel, R.; Salakhutdinov, R.; et al. 2015. Siamese neural networks for one-shot image recognition. In *ICML deep learning workshop*, volume 2. Lille.

Lang, K. 1995. Newsweeder: Learning to filter netnews. In *Machine learning proceedings 1995*, 331–339. Elsevier.

Lei, S.; Zhang, X.; He, J.; Chen, F.; and Lu, C.-T. 2023. TART: Improved Few-shot Text Classification Using Task-Adaptive Reference Transformation. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics*, 11014–11026. Toronto, Canada: Association for Computational Linguistics.

Li, W.-H.; Liu, X.; and Bilen, H. 2021. Universal representation learning from multiple domains for few-shot classification. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 9526–9535.

Liu, J.; Song, L.; and Qin, Y. 2020. Prototype rectification for few-shot learning. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part I 16*, 741–756. Springer.

Loshchilov, I.; and Hutter, F. 2019. Decoupled Weight Decay Regularization. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

Luo, Q.; Liu, L.; Lin, Y.; and Zhang, W. 2021. Don't miss the labels: Label-semantic augmented meta-learner for few-shot text classification. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, 2773–2782.

Ma, D.; Wang, Z.; Xie, J.; Yu, Z.; Guo, B.; and Zhou, X. 2020. Modeling Multivariate Time Series via Prototype Learning: a Multi-Level Attention-based Perspective. In *2020 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, 687–693.

Misra, R. 2018. News Category Dataset.

Misra, R.; and Grover, J. 2021. Sculpting data for ml: The first act of machine learning. *University of California San Diego: La Jolla, CA, USA*.

Mueller, A.; Krone, J.; Romeo, S.; Mansour, S.; Mansimov, E.; Zhang, Y.; and Roth, D. 2022. Label Semantic Aware Pre-training for Few-shot Text Classification. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics*. Dublin, Ireland: Association for Computational Linguistics.

Munkhdalai, T.; and Yu, H. 2017. Meta networks. In *International conference on machine learning*, 2554–2563. PMLR.

Rajendran, J.; Irpan, A.; and Jang, E. 2020. Meta-learning requires meta-augmentation. *Advances in Neural Information Processing Systems*, 33: 5705–5715.

Sabour, S.; Frosst, N.; and Hinton, G. E. 2017. Dynamic routing between capsules. *Advances in neural information processing systems*, 30.

Sanh, V.; Debut, L.; Chaumond, J.; and Wolf, T. 2019. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. *CoRR*, abs/1910.01108.

Santoro, A.; Bartunov, S.; Botvinick, M.; Wierstra, D.; and Lillicrap, T. 2016. Meta-learning with memory-augmented neural networks. In *International conference on machine learning*, 1842–1850. PMLR.

Snell, J.; Swersky, K.; and Zemel, R. 2017. Prototypical networks for few-shot learning. *Advances in neural information processing systems*, 30.

Sun, P.; Ouyang, Y.; Zhang, W.; and Dai, X.-y. 2021. MEDA: Meta-Learning with Data Augmentation for Few-Shot Text Classification. In Zhou, Z.-H., ed., *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*, 3929–3935. International Joint Conferences on Artificial Intelligence Organization.

Vinyals, O.; Blundell, C.; Lillicrap, T.; Wierstra, D.; et al. 2016. Matching networks for one shot learning. *Advances in neural information processing systems*, 29.

Wang, J.; Lan, C.; Liu, C.; Ouyang, Y.; Qin, T.; Lu, W.; Chen, Y.; Zeng, W.; and Yu, P. 2022. Generalizing to unseen domains: A survey on domain generalization. *IEEE Transactions on Knowledge and Data Engineering*.

Wang, Y.; Yao, Q.; Kwok, J. T.; and Ni, L. M. 2020. Generalizing from a few examples: A survey on few-shot learning. *ACM computing surveys (csur)*, 53(3): 1–34.

Wu, A.; Han, Y.; Zhu, L.; and Yang, Y. 2021. Universal-prototype enhancing for few-shot object detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 9567–9576.

Zhao, Y.; Tian, Z.; Yao, H.; Zheng, Y.; Lee, D.; Song, Y.; Sun, J.; and Zhang, N. 2022. Improving Meta-learning for Low-resource Text Classification and Generation via Memory Imitation. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 583–595. Dublin, Ireland: Association for Computational Linguistics.

Checklist

1. For most authors...
   (a) Would answering this research question advance science without violating social contracts, such as violating privacy norms, perpetuating unfair profiling, exacerbating the socio-economic divide, or implying disrespect to societies or cultures? Yes
   (b) Do your main claims in the abstract and introduction accurately reflect the paper's contributions and scope? Yes
   (c) Do you clarify how the proposed methodological approach is appropriate for the claims made? Yes, please see section 1.
   (d) Do you clarify what are possible artifacts in the data used, given population-specific distributions? NA, the data we use is publicly accessed in NLP.
   (e) Did you describe the limitations of your work? Yes, please see section 6.
   (f) Did you discuss any potential negative societal impacts of your work? NA
   (g) Did you discuss any potential misuse of your work? NA
   (h) Did you describe steps taken to prevent or mitigate potential negative outcomes of the research, such as data and model documentation, data anonymization, responsible release, access control, and the reproducibility of findings? Yes, please see section 4.1.
   (i) Have you read the ethics review guidelines and ensured that your paper conforms to them? Yes
2. Additionally, if your study involves hypotheses testing...
   (a) Did you clearly state the assumptions underlying all theoretical results? NA
   (b) Have you provided justifications for all theoretical results? NA
   (c) Did you discuss competing hypotheses or theories that might challenge or complement your theoretical results? NA
   (d) Have you considered alternative mechanisms or explanations that might account for the same outcomes observed in your study? NA
   (e) Did you address potential biases or limitations in your theoretical framework? NA
   (f) Have you related your theoretical results to the existing literature in social science? NA
   (g) Did you discuss the implications of your theoretical results for policy, practice, or further research in the social science domain? NA
3. Additionally, if you are including theoretical proofs...
   (a) Did you state the full set of assumptions of all theoretical results? NA
   (b) Did you include complete proofs of all theoretical results? NA
4. Additionally, if you ran machine learning experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? Yes, public datasets are used in our research and we also provide detailed instruction. The data splits and hyperparameters are given to reproduce the main experimental results.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? Yes, please see section 4.1.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? Yes, we evaluate the result variance in the result table.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? Yes
   (e) Do you justify how the proposed evaluation is sufficient and appropriate to the claims made? Yes, please see section 3.2.
   (f) Do you discuss what is "the cost" of misclassification and fault (in)tolerance? NA, the misclassification cause no other damage to property, et al.
5. Additionally, if you are using existing assets (e.g., code, data, models) or curating/releasing new assets, **without compromising anonymity**...
   (a) If your work uses existing assets, did you cite the creators? Yes
   (b) Did you mention the license of the assets? NA
   (c) Did you include any new assets in the supplemental material or as a URL? No
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? NA, the data we applied is open-source materials.
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? NA
   (f) If you are curating or releasing new datasets, did you discuss how you intend to make your datasets FAIR (see FORCE11 (2020))? NA
   (g) If you are curating or releasing new datasets, did you create a Datasheet for the Dataset (see Gebru et al. (2021))? NA
6. Additionally, if you used crowdsourcing or conducted research with human subjects, **without compromising anonymity**...
   (a) Did you include the full text of instructions given to participants and screenshots? NA
   (b) Did you describe any potential participant risks, with mentions of Institutional Review Board (IRB) approvals? NA
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? NA
   (d) Did you discuss how data is stored, shared, and deidentified? NA