MSBD5002 Project Report
Exploring Prompting Methods for Few-shot Text Mining

Project Type: Research

Group 11
CHEN, I Chieh (20655007)
CHEN, Jiaxuan (21018177)
CHEN, Yingan (20972900)
HAO, Siyang (21002001)
SU, Hongchang (20972649)
YIN, Zhuohao (20677990)

May 30, 2024

Declaration statement: We hereby declare that the project is independent of our own projects, and serves only for the assessment of this course.

Abstract

In the field of text mining, pre-trained language models (PLMs) have emerged as powerful tools for various downstream tasks [10, 2]. However, their performance can significantly degrade in scenarios with limited data, known as few-shot scenarios [1]. Prompt-based tuning is effective in addressing this challenge by providing a structured approach to adapt PLMs with minimal data [5]. One crucial component of prompt-based tuning is the verbalizer, which maps model outputs to their corresponding labels. In this project, we propose two techniques, Instance Mean Initialization and Shared Embedding Space, to enhance the performance of the prototypical verbalizer (ProtoVerb) in few-shot learning tasks. In our experiments, we conducted using comprehensive few-shot learning setups, which demonstrate the effectiveness of our proposed method. We evaluate our approach on various experimental settings and report classification accuracy as the evaluation metric. Our implementation, based on the PyTorch framework, the Huggingface transformers, and the OpenPrompt toolkit, uses the RoBERTa-large model as the PLM backbone. Through experimentation and analysis, we demonstrate the effectiveness of our proposed method in enhancing classification performance in few-shot scenarios, thereby contributing to the advancement of prompt-based tuning methodologies.

1 Introduction

With the increasing volume of documents, text data has become the most common data type of information. Within such a context, text mining is considered as having a great value for business [13]. Pre-trained language models (PLM), which have received attention from related parties, has achieved significant progress on text mining [10]. With huge parameters and fine-tuning techniques, PLM can perform well on various downstream tasks [10], such as question answering and reading comprehension [2]. However, when it comes to a few-shot scenario, where there are only a few data samples available for the specific task, many models may perform even worse than human [1]. Meanwhile, few-shot scenario may be common in practice, as it is highly costly to guarantee that there are always enough data samples for each class included in the data sets. Therefore, how to make PLM to maintain a satisfactory performance in a setting of lacking data becomes a meaningful problem.

Prompt-based tuning may provide a good solution to the above problem. By setting a new and appropriate prompting function, it allows the learning to adapt to those scenarios with only a few labeled data available [15]. For different tasks, the tuning method creates different strings as prompt templates, which usually include an input slot and an empty answer slot [15]. After the input text is entered into the input slot, an intermediate answer, which will be mapped to the output label, would be generated at the answer slot [15]. In this way, the task is converted to a masked problem for modeling [11]. This follows the idea that giving a textual description to elaborate the task may help to problem solving with a limited number of samples available [19]. Also, to maximize the score of the language model and get the best performance, the optimal intermediate answer is searched from the set of all possible answers [15].

A significant step in prompt-based tuning is to perform mapping between the output answer to the corresponding label. The process is conducted by verbalizer [6]. Since verbalizer determines the performance of prompt-based tuning [6] at a certain level, it is critical to select a proper method to create an effective verbalizer for prompt-based turning [3].

At the current stage, verbalizer design methods adapted by most existing works mainly includes 3 categories, which are manual verbalizer design, discrete verbalizer design, and soft verbalizer design methods [6]. However, prototypical verbalizer, which does not rely on much human efforts or a large data set, is claimed to have a much more outstanding performance than current discrete verbalizer and soft verbalizer [3]. For instances of each class, prototypical verbalizer calculates the central points of them as the prototype [3]. Then, verbalizer is trained for the objectives that instances of the same class can be put together, instances belonging to different class can be separated, and any prototype can be located on the center of all instances that belong to the same class that the prototype belongs to [3]. The idea of prototypical verbalizer may be similar to

the K-Means clustering algorithm [12] at a certain level, as both are trained to group similar instances while trying to put centroids, or prototypes, of the groups to be the center of that group.

Although it is declared that prototypical verbalizer has an excellent performance, there are also limitations existing in the mechanism of its design. One limitation is inadequate use of information in the initialization stage. Regardless of the contents of class labels, the prototypes are randomly initialized at the first step [3]. However, this may lead to more time and computing cost in training process, as the initialization of prototypes embedding, which does not make use of any semantic information of class labels in the training set, is likely to give an initialization that is far from the optimization result. Based on the understanding of the limitation, our project focuses on the improvement of prototypical verbalizer design. We propose 2 novel designs that bring performance boost. First, we propose an initialization technique called the **Instance Mean Initialization**, which utilizes the semantic information of the class labels and produce the initial values of the prototype embeddings. Second, we propose that the prototype embedding space can be **Shared** with the **Embedding Space** of the word embeddings produced by the PLM. After comprehensive experiments, both innovations demonstrate strong performance and more importantly convey key insights to the design and learning of verbalizers in general.

2 Related Work

2.1 Prompt-based Tuning in Few-shot Scenarios

In the early phase of PLM development, PLM handles specific tasks by pretraining and finetuning [5]. However, in scenarios with few data samples available, PLM may work better with prompts [5]. Normally, the pipeline of prompt may contain a string template and a verbalizer [5]. The template wraps the input text and output an intermediate answer text [15], while the verbalizer bridges the answer to the label [6]. Previous research have been conducted from various related aspects, including prompt template engineering [15]. Studies related to verbalizer will be discussed in this report.

2.2 Methods of Verbalizer Designs

Manual verbalizer, discrete verbalizer and soft verbalizer are the main three categories of verbalizer design methods [6]. Designed by human and referred to specific domain knowledge, manual verbalizer chooses label words to indicate each classes [11, 3]. Experiments have shown that carefully designed manual verbalizer can have a highly satisfactory performance over a variety of tasks [19] [6]. Nevertheless, to design a verbalizer that can output an excellent result, it requires the human designer to have a thorough and exact understanding of the

domain and the specific task[6]. Meanwhile, a considerable amount of time may be necessary to experiment and find out a satisfactory solution [6]. In addition, since one-one mapping is adapted by manual verbalizer, the coverage of label words is quite limited, thus leading to a biased prediction made referring to a limited amount of information [11].

In order to save the human efforts and time cost in constructing manual verbalizer, discrete verbalizer design methods are proposed [6]. For each label, discrete verbalizer design methods search for the word correspondingly from the model vocabulary to construct the verbalizer [6]. An example of discrete verbalizer design is automatic verbalizer search (AVS) [6], where the mapping of labels and words is randomly initialized at the first step [18]. The improvement of the mapping is performed iteratively and greedily [18] until meeting a certain condition [6]. Another design method that can save human effort is soft verbalizer design. To obtain better performance, soft verbalizer searches for suitable parameter from an infinite continuous space [6]. Word-level Adversarial ReProgramming (WARP) [9] is an example of soft verbalizing method. Proposed with reference to adversarial reprogramming, WARP enters special prompt tokens and `[MASK]` token into the input text [9]. Training parameters including the word embedding of special prompt tokens and verbalizer tokens, the model is optimized for the downstream task [9]. The soft verbalizer can perform better than discrete ones, as it optimize on a continuous thus larger space [6]. However, in real life, it is common to have a data set that is not large and balanced enough for the task to perform. In such a scenario, it is difficult to optimize enough on a discrete or continuous space, leading to the result that discrete and soft verbalizer are often less effective than manual ones [3].

Prototypical verbalizer is another design. Different from mentioned verbalizer design methods, it does not rely on much human efforts or searching in a large enough data set [3]. The method starts with the idea that prototype, which is the central point of instances in a class, can represent the semantic features of a class in a setting of few data [3]. Based on the opinions, prototype vectors, if work as verbalizers, are thought to be able to make judgements that are closed to classes determined by human [3]. Therefore, prototypical verbalizer is proposed to automatically learn prototype vectors of each class from training set by applying contrastive learning [3]. The representations of instances are hidden state of `[MASK]` token on the last layer, which are later projected to the embedding space to learn the prototypes [3]. With the goals that similarity scores of instances within the same class should be higher and similarity scores of instances belonging to different classes should be lower, as well as similarity scores between the pair of prototype and instances of the same class should be higher than similarity scores between prototype and instances of different classes, the optimization objective function is defined by contrastive learning and cosine similarity function [3]. Optimizing for the perspectives of pairs of instance and instance and pairs of instance and prototype, it is declared that prototypical verbalizer significantly performs better than existing discrete ver-

balizer and soft verbalizer [3]. However, as it is mentioned, limitations still exist for prototypical verbalizer.

3 Background

Given a pre-trained language model $\mathcal{M}$, such as BERT [4] and RoBERTa [16], we aim to tune the model $\mathcal{M}$ for specific downstream tasks, such as text classification, relation extraction and entity typing. Our project specifically focuses on n-way-k-shot text classification [7].

3.1 N-Way-K-Shot Few-Shot Learning

n-way-k-shot is a basic concept in few-shot learning (FSL). The training set for FSL picks n classes and $k+q$ samples for each class from the meta-dataset randomly to form a support set for training which consists of k samples for each class and to form a query set for testing which contains q samples for each class. So, the size of the training set should be $n \times (k+q)$. For example, 5-way 8-shot means that the task has 5 classes and the support set has 8 samples for each class.

The goal is to predict the label $y \in \mathcal{Y}$ for each sample, where $\mathcal{Y}$ is the label set with N distinct classes. FSL is useful when the labelled dataset for a specific task is scarce, as it allows the model to learn from a few samples per class, rather than requiring a large labelled dataset.

3.2 Prompt-based Tuning

The vanilla prompt-based tuning transforms the text classification into a cloze-style prediction problem by providing a prompt template, which is used to process the original text with an extra token [5]. The prompt template typically consists of a fill-in-the-blank style statement or question that provides context and guidance to the model for making predictions on the text, and is customized based on different domains.

For the original input text x, it is wrapped with a template $\mathcal{T}(\cdot)$, which contains a mask token `[MASK]` that is used to represent the position in the prompt where the answer is expected to appear, as the prompt input $\mathcal{T}(x)$. As shown in Figure 1, x = *what is the relation between speed and acceleration?*, $\mathcal{T}(\cdot)$ = *A `[MASK]` question:*, then $\mathcal{T}(x)$ = *A `[MASK]` question: what is the relation between speed and acceleration?* [11]. The prompt input is first tokenized and fed to the pre-trained language model $\mathcal{M}$ and then $\mathcal{M}$ predicts the probability of each word v to be filled in the `[MASK]` token,

$$P_{\mathcal{M}}(\texttt{[MASK]} = v|\mathcal{T}(x)). \tag{1}$$

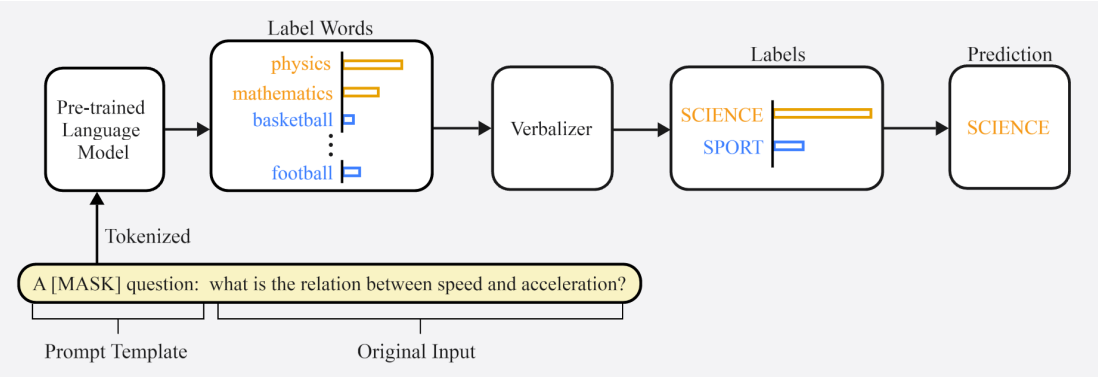


Figure 1: Illustration of prompt-based tuning.

To bridge the model output v and the final prediction, verbalizers serve as a critical part of prompt-based tuning [8]. The verbalizer, which stores a set of label words $\mathcal{V}$, maps the output word v to the corresponding original label y. Therefore, the probability of label y is

$$P_{\mathcal{M}}(y|x) = g(P_{\mathcal{M}}(\texttt{[MASK]} = v|\mathcal{T}(x))|v \in \mathcal{V}_y), \tag{2}$$

where $\mathcal{V}_y$ is the set of label words of label y and $g(\cdot)$ is to aggregate multiple scores [3].

On the whole, prompt-based tuning is a useful technique for tuning PLMs to leverage their knowledge learned during pre-training and improve their performance on downstream tasks.

3.3 Prototypical Verbalizer

As mentioned in the previous subsection, the verbalizer is a crucial component of the prompting model that projects the model outputs to their corresponding labels. In this report, we focus on prototypical verbalizer (ProtoVerb) [3], which is built directly from the training set and learns prototype vectors as verbalizers by contrastive learning.

To learn the prototype vector, the hidden state of `[MASK]` token $\mathbf{h}_{\texttt{[MASK]}}$ by the model $\mathcal{M}$ is used to represent the instance and is projected to another embedding space with a specific dimension. Encoded by a linear encoder $\mathbf{W}$, the instance representation of input text x becomes

$$\mathbf{v} = \mathbf{W}\mathbf{h}_{\texttt{[MASK]}}. \tag{3}$$

To measure the similarity between two instances, the cosine similarity function $\mathcal{S}$ is used, where

$$\mathcal{S}(\mathbf{v}_i, \mathbf{v}_j) = \frac{\boldsymbol{v}_i}{\|\boldsymbol{v}_i\|} \cdot \frac{\boldsymbol{v}_j}{\|\boldsymbol{v}_j\|}. \tag{4}$$

With the concept of instance representation and similarity function, we are trying to achieve the goals that (1) the similarity between two instances within the same class should be higher and that belonging to different classes should be lower, (2) the similarity score between the instance and the prototype of its class should be higher than that between the instance and prototype of different classes.

For the first goal, we need to minimize the loss function for the instance-instance pair, which is

$$\mathcal{L}_{ins} = \frac{-1}{N^2K^2} \sum_{n} \sum_{i,j} \log \frac{\exp S\left(\boldsymbol{v}_i^n, \boldsymbol{v}_j^n\right)}{\sum_{n',j'} \exp S\left(\boldsymbol{v}_i^n, \boldsymbol{v}_{j'}^{n'}\right)}. \tag{5}$$

And for the second goal, we denote the set of prototypes as $\mathcal{C} = \{\mathbf{c}_1, \cdots, \mathbf{c}_N\}$. Similarly, the loss function for the instance-prototype pair is defined as

$$\mathcal{L}_{proto} = \frac{-1}{N^2K} \sum_{i,n} \log \frac{\exp S\left(\boldsymbol{v}_i^n, \boldsymbol{c}_n\right)}{\sum_{n'} \exp S\left(\boldsymbol{v}_i^n, \boldsymbol{c}_{n'}\right)}. \tag{6}$$

Therefore, the final training objective is to minimize the loss function which combines the loss function for the instance-instance pair and for the instance-prototype pair,

$$\mathcal{L} = \mathcal{L}_{ins} + \mathcal{L}_{proto}. \tag{7}$$

During the inference, we measure the similarities between the query $\mathbf{q}$ and the prototype for each class. For the class k, its probability score is

$$P_{\mathcal{M}}(y_k|x) = \frac{\exp S\left(\boldsymbol{q}, \boldsymbol{c}_k\right)}{\sum_{k'} \exp S\left(\boldsymbol{q}, \boldsymbol{c}_{k'}\right)}, \tag{8}$$

and the prediction of the original input text x is defined by

$$\hat{y} = \underset{k}{\operatorname{argmax}} P_{\mathcal{M}}(y_k|x). \tag{9}$$

4 Method

As the general pipeline of prompt-based tuning has been introduced in previous sections, this section will focus on the innovation we propose on top of the vanilla

prototypical verbalizer (ProtoVerb) [3], as well as the motivation behind such alterations. In short, we propose 2 novel techniques based on ProtoVerb, which are called **Instance Mean Initialization** and **Shared Embedding Space** respectively. Details are elaborated below.

4.1 Instance Mean Initialization

As opposed to manual verbalizers, prototypes can be regarded as aggregated hidden representations of all the appropriate words that would have been chosen under a manual verbalizer setting. Taking the AG news dataset as an example, there are 4 classes, namely {*World, Sports, Business, Tech*}, corresponding to 4 types of news. Then a well-trained prototype embedding for the *Sports* class should aggregate the information contained in words such as *NBA, goal, baseball,* etc. Naturally, instances that belong to the same class should cluster around their corresponding prototype embedding. In other words, a good classifier is expected to map the hidden states of the `[MASK]` token to a vector that is close to the prototype embedding of its class, where the distance is measured by cosine similarity. To this end, two loss functions are used in the original ProtoVerb [3]. The first is the instance-instance loss, which utilizes the idea of contrastive learning to push instances that belong to different classes away from each other, and to pull those of the same classes near each other. The second loss function is the instance-prototype loss, which tries to minimize the total distance between the prototype and all the corresponding instances. Intuitively, it forces the prototype to be situated at the center of its instances so that it becomes a good representative of that class.

However, the vanilla ProtoVerb is not up to its full potential as it fails to leverage the semantics of the label words themselves. In the original ProtoVerb [3], prototype embeddings are randomly initialized in a sense that they are independent of the class label words. In this way, the training process will be invariant no matter the classes are labeled as {*World, Sports, Business, Tech*} or {*1, 2, 3, 4*}. This is a huge and unnecessary waste of information because the label words are inherently suitable to fill the `[MASK]` token. In fact, it is common practice to include the class labels if a manual verbalizer is to be used.

To remedy this flaw, we propose an initialization technique that we call **Instance Mean Initialization**. Specifically, for each example in the few-shot training set, we replace the `[MASK]` token in the template with its ground truth class label word and obtain its hidden representation after feeding it to the PLM. Then, for each class, we use the mean vector of the hidden states of all its training examples, instead of randomly initializing the prototype embeddings. Formally, suppose there are n classes $C = \{C_1, C_2, ..., C_n\}$, and their corresponding labels are $W = \{w_1, w_2, ..., w_n\}$. Denote the PLM as $\mathcal{M}$ and the template as $\mathcal{T}(\texttt{<LABEL>}, x)$, then given an example $x_i \in C_j$ where

$i \in \{1, 2, ..., k\},\ j \in \{1, 2, ..., n\}$, its hidden states of the label word is given by

$$\mathbf{h}^{(i)}_{\texttt{<LABEL>}} = \mathcal{M}(\mathcal{T}(w_j, x_i)). \tag{10}$$

Note that the word embeddings and the prototypes are in different vector spaces and thus an additional linear encoder $\mathbf{W}$ is needed to project word embeddings into the prototype space. Then the prototype embedding for class C_j under a k-shot setting is initialized as

$$\mathbf{c}_j = \frac{1}{k} \sum_{i=1}^{k} \mathbf{W}\mathbf{h}^{(i)}_{\texttt{<LABEL>}}. \tag{11}$$

With the instance mean initialization, the training process of the prototype embeddings can be faster and less sensitive to noise.

4.2 Shared Embedding Space

Although we are able to eliminate randomness brought by the initialization of prototype embeddings, the linear encoder $\mathbf{W}$ is still randomly initialized and thus may harm the performance of the instance mean initialization technique. Additionally, since the prototypes are learnable embeddings themselves, the extra linear encoder $\mathbf{W}$ seems redundant and we are curious to investigate its necessity to the model. To this end, we propose to remove the encoder and let word embeddings and prototype embeddings share the same vector space. For instance, if we use a RoBERTa-large [16] as the backbone PLM and set the dimension of the prototype embeddings to be 64, the original ProtoVerb [3] requires an encoder $\mathbf{W} \in \mathbb{R}^{64\times1024}$ to project word embeddings into the prototype space. We propose to remove the projection head and set the dimension of the prototype embeddings to be the same as the word embeddings, e.g., 1024. In this way, we can directly compute the cosine similarity between word embeddings and prototypes, and the model is trained by jointly optimizing the PLM and the learnable prototype embeddings.

We propose to remove the project head, i.e., the encoder for 2 obvious reasons. First, without the projection head, the overall complexity of the model can be further reduced as there are fewer trainable parameters. As a consequence, the model in general can be trained more easily with less training time and fewer computational resources. The second reason lies in the increase in dimensionality. According to the standard setting proposed in [3], the dimension of the prototype embeddings is set to 64 while they used the RoBERTa-large[16] as the PLM backbone, where the hidden dimension of word embeddings is 1024. Now that we remove the projection head and allow both word embeddings and prototype embeddings to share the same vector space, we implicitly increase the dimension of prototype embeddings to whatever the hidden dimension of the pretrained language model is. With a higher dimensionality, prototype embeddings are capable of capturing much more information than before, which

can aid the learning process in which the prototypical verbalizer tries to approximate a well-constructed manual verbalizer. However, we do hypothesize that higher dimensional embeddings require longer training time to converge so we adjust the training configurations accordingly as will be introduced below.

5 Experiments

We perform a series of comprehensive few-shot learning experiments to demonstrate the efficacy of our improved method. Specifically, we begin by introducing the experimental settings and implementation details, followed by a presentation and analysis of the experimental results.

5.1 Datasets and Prompt Templates

In our experiments, we adopt three text classification datasets: **AG's news**, **DBPedia**[14], and **Yahoo Answers**[22]. The information of their test sets are shown in Table 1.

Dataset	**#Class**	**#Example**
AG'S News	4	7600
DBPedia	14	70000
Yahoo Answers	10	60000

Table 1: Statistics of datasets.

In order to prioritize the verbalizer and mitigate the impact of templates, we utilize several fixed manual templates. We use the default templates in **OpenPrompt**[5]. The details are as follows:

AG's News. Each sample contains a headline x and a body y. We use the following templates.

$$
\begin{aligned}
T_1(x, y) &= \text{A } \texttt{[MASK]} \text{ news: } x\ y \\
T_2(x, y) &= x\ y \text{ This topic is about } \texttt{[MASK]}. \\
T_3(x, y) &= [\text{Category: } \texttt{[MASK]}]\ x\ y \\
T_4(x, y) &= [\text{Topic: } \texttt{[MASK]}]\ x\ y
\end{aligned}
$$

DBPedia. Each sample contains an article title x and an article content y. We use the following templates.

$$
\begin{aligned}
T_1(x, y) &= x\ y \text{ is a } \texttt{[MASK]}. \\
T_2(x, y) &= x\ y \text{ In this sentence,} x \text{ is a } \texttt{[MASK]}. \\
T_3(x, y) &= x\ y \text{ The type of } x \text{ is } \texttt{[MASK]}. \\
T_4(x, y) &= x\ y \text{ The category of } x \text{ is } \texttt{[MASK]}.
\end{aligned}
$$

Yahoo Answers. Each sample contains a question x and a answer y. We use the following templates.

$$T_1(x, y) = \text{A } \texttt{[MASK]} \text{ question: } x\ y$$
$$T_2(x, y) = x\ y \text{ This topic is about } \texttt{[MASK]}.$$
$$T_3(x, y) = [\text{Category: } \texttt{[MASK]}]\ x\ y$$
$$T_4(x, y) = [\text{Topic: } \texttt{[MASK]}]\ x\ y$$

5.2 Experiment Settings

In the few-shot scenario, we randomly select k = 1, 2, 4, 8, 16 instances from each class in the training set, and evaluate the model's performance on the complete test set. To be more specific, the final size of the training set is $k \cdot c$, where k is the number of few-shot examples and c is the number of classes in total. In all experiments, we measure classification accuracy as the evaluation metric. To reduce the impact of randomness in the initialization of the experiments and ensure that the reported performance is representative and stable across different runs, we report the mean accuracy scores over 3 random seeds.

5.3 Implementation Details

We implement all of our models and baselines using the PyTorch framework[17], the Huggingface transformers[21], and the OpenPrompt toolkit[5]. We utilize the AdamW optimizer to optimize the performance of pretrained language models. We use the RoBERTa-large[16] as our PLM backbone. When training the models that are equipped with the projection head, we set the dimension of the prototype embeddings to be 64, and fine-tune the model for 5 epochs with a batch size of 2 and a learning rate of 3e-5. As for the training configurations of the prototype embeddings, we use a learning rate of 0.01 and train the embeddings for 30 epochs. As mentioned in the previous section, we hypothesize that higher dimensional embeddings require longer training time to converge. To this end, for experiments with shared embedding space, we adjusted the epochs to tune the PLM and the epochs to train the prototype embeddings to 20 epochs and 60 epochs respectively while the learning rates are kept the same.

5.4 Results & Analysis

We present experiment results in Table 2, together with baseline performance reported in [3]. It can be seen that manual verbalizer achieves the highest accuracy across all datasets under all few-shot settings since domain knowledge is involved to determine the choice of words in the manual verbalizer. Still, our ProtoVerb$_{\text{mean-init}}$ outperforms the original prototypical verbalizer on all datasets, indicating that the instance mean initialization technique has significant benefits to the training of prototypes. More notably, as more training examples become available, i.e., under 8-shot and 16-shot settings, our method

even outperforms the sophisticatedly constructed manual verbalizer, which coincides with our hypothesis that our method is able to learn and aggregate information that would be contained in a well-designed manual verbalizer with human efforts.

K	Method	AG	DB	Yahoo
0	ManualVerb	*75.13*	*67.06*	*43.11*
1	ManualVerb	*76.67*	*85.47*	*50.22*
	ProtoVerb	64.19	72.85	36.12
	$\text{ProtoVerb}_{\text{mean-init}}$	**68.71**	**89.66**	**42.00**
2	ManualVerb	*81.06*	*93.61*	*58.65*
	ProtoVerb	77.34	85.49	46.30
	$\text{ProtoVerb}_{\text{mean-init}}$	**79.45**	**92.23**	**57.32**
4	ManualVerb	*84.73*	*95.83*	*61.41*
	ProtoVerb	81.65	90.91	55.08
	$\text{ProtoVerb}_{\text{mean-init}}$	**84.19**	**95.35**	**61.14**
8	ManualVerb	*85.85*	*96.46*	*64.12*
	ProtoVerb	84.03	95.75	61.40
	$\text{ProtoVerb}_{\text{mean-init}}$	**86.73**	**96.97**	**67.47**
16	ManualVerb	*84.74*	*96.05*	*58.77*
	ProtoVerb	84.48	96.30	64.35
	$\text{ProtoVerb}_{\text{mean-init}}$	**88.06**	**96.45**	**65.97**

Table 2: Mean accuracy scores (%) of experiments under different few-shot settings for different datasets. Each accuracy is obtained by taking average over 3 different random seeds. Manual verbalizer results are italic as they require domain knowledge. Bold results are the best without domain knowledge. **ProtoVerb** represents the original ProtoVerb [3] and its results are directly retrieved from the paper. $\textbf{ProtoVerb}_{\textbf{mean-init}}$ corresponds to ProtoVerb with the Instance Mean Initialization technique.

We have also conducted an ablation study to investigate the effectiveness of the proposed ideas. Specifically, we are curious about the importance of the linear encoder. Mean accuracy scores on the AG's News dataset are shown in Table 3. As one can observe, when the available training data are limited, e.g., 1-shot or 2-shot, the performance of the model without an encoder is worse than that with an encoder. However, as the number of training examples gets larger, the encoder-free model outperforms the model with an encoder and its performance is almost comparable with $\text{ProtoVerb}_{\text{mean-init}}$.

Our conjuncture is as follows. When the encoder is removed so that the word embeddings and the prototype embeddings reside in the same vector space, the PLM takes over the encoder's job to produce meaningful word embeddings for the `[MASK]` token. In other words, the PLM is expected to output an infor-

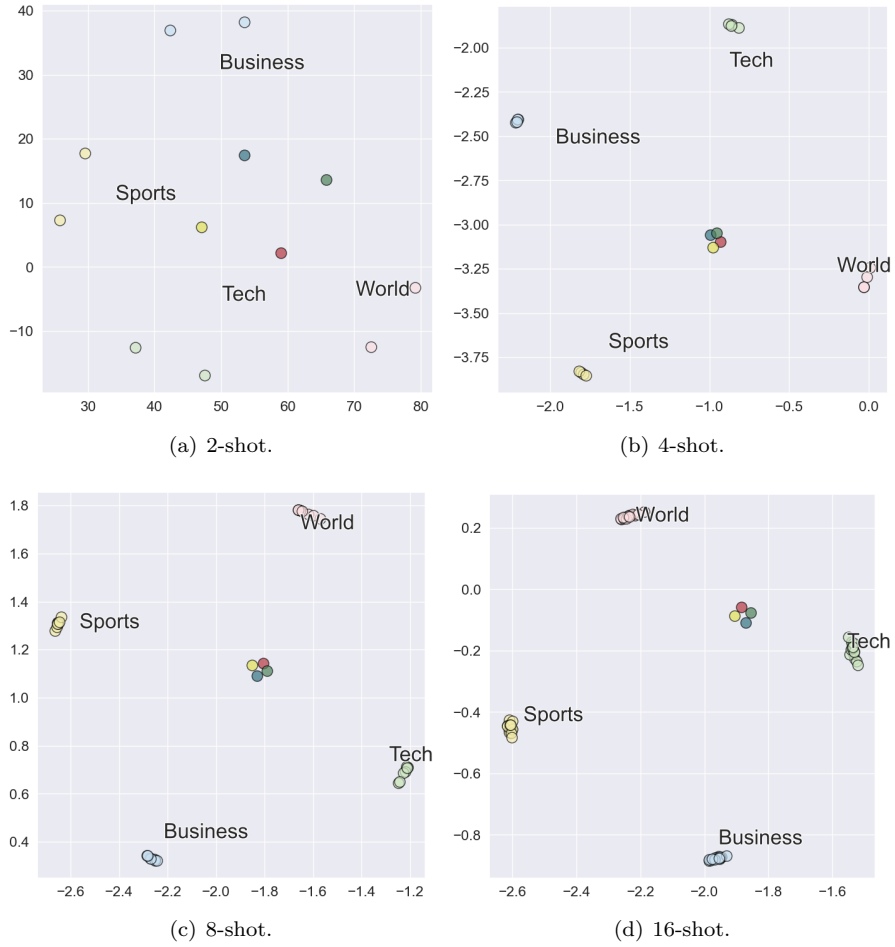
mative embedding that well summarize the input text. Meanwhile, it should be as close as possible to the prototype embedding of the corresponding class. This poses a challenging task in terms of fine-tuning the PLM, who has a large amount of parameters and thus is immensely data-hungry. This explains the ineffectiveness of sharing embedding space when the data scarcity problem is severe under 1-shot or 2-shot settings. However, as data become more abundant, sharing embedding space is more comparable with manual verbalizers because a higher dimension is capable of capturing more information. As a result, the key message is that when adequate data are provided (e.g., not less than 4-shot), sharing the embedding space and not using the encoder may achieve better results. However, extra efforts will be necessary to be spent on searching for the optimal set of hyperparameters in order to unveil its full potential.

K	Encoder	Mean Init	Accuracy (%)
1	✓	✓	68.71
	✗	✓	58.54
	✓	✗	*64.19*
2	✓	✓	79.45
	✗	✓	77.34
	✓	✗	*77.16*
4	✓	✓	84.19
	✗	✓	81.65
	✓	✗	*78.74*
8	✓	✓	86.73
	✗	✓	84.03
	✓	✗	*85.37*
16	✓	✓	88.06
	✗	✓	84.48
	✓	✗	*88.04*

Table 3: Ablation study of the 2 proposed ideas over the original ProtoVerb [3], i.e., instance mean initialization and shared embeddings space. Encoder: whether or not to use the linear encoder to project word embeddings. Mean Init: whether or not to use the instance mean initialization technique. The vanilla ProtoVerb uses the encoder but not Mean Init and its accuracy scores are in italic as a reference.

Additionally, we have adopted the t-SNE algorithm [20] to visualize the learned prototype embeddings, as well as the projected word embeddings, as shown in Figure 2. The dataset used is the AG's News dataset, consisting of news of 4 categories, i.e., {*World, Sports, Business, Tech*}. The dimension of the prototypes is set to 64, and projected onto the 2D space using the t-SNE algorithm. Note that we do not perform the t-SNE visualization under 1-shot setting as the result is unlikely to convey meaningful insights with only one example for each class. We can see that under all few-shot settings, the matched prototypes and word embeddings are the closest in terms of Euclidean distance, indicating that

our proposed method is effective. However, we do observe that there exists a distinct gap between the positions of the prototypes and the word embeddings as they lie quite away from each other. We conjecture that this heterogeneity is caused by the training objectives. Since the cosine similarity is used in the training objectives as the distance metric, the prototypes and word embeddings are only trained to closer in terms of vector direction instead of the Euclidean distance. By this token, it is normal to see such significant discrepancies on the 2D plane.



(a) 2-shot. (b) 4-shot.

(c) 8-shot. (d) 16-shot.

Figure 2: t-SNE visualizations for the trained prototype embeddings and encoded word embeddings of the training examples under 2-shot, 4-shot, 8-shot and 16-shot respectively. Red, yellow, blue and green represent the 4 classes in the AG News dataset, i.e., *World, Sports, Business* and *Tech*. The points with darker colors are the trained prototype embeddings. The points with lighter colors are the encoded label word embeddings obtained by filling the template with the ground truth label word for each training example. Even though there exists significant gaps between the prototypes and the word embeddings after projecting them onto the 2D space, one can see evidently that under the 4 few-shot settings, prototypes and encoded label word embeddings that correspond to the same class are the closest.

6 Conclusion

By investigating in previous works of prompt-based tuning methods, our work further explored on the field of prototypical verbalizers methods. We proposed Instance Mean Initialization technique for prototypical verbalizers, which achieves better accuracy than the original ProtoVerb model and is also compatible with the manual crafted verbalizers which requires domain knowledges and heavy laborforce. Our work also proposed the removal of linear encoder and bringing the prototype embeddings and word embeddings to the same dimenson, and our results have shown that our model can capture more information and achieve better computation efficiency with our architecture.

7 Contribution

CHEN, I Chieh (20655007): In the project proposal, I introduced the backgrounds, techniques, and discussed about the advantages of two verbalizers: Soft Verbalizer and Prototypical Verbalizer. I also assisted in training the proposed models and generating the results of our experiments. I helped organized the final results in the final report, and I examined the whole report for possible language mistakes and I wrote the main proposed ideas and the contribution of our work in the Conclusion part in our final report. I will also be responsible in organizing the flow of our presentation and designing the PowerPoint for our final presentation.

CHEN, Jiaxuan (21018177): In the Background section of the project proposal, I introduced the basic concepts of the prompt-based fine-tuning. I participated in writing the source codes to implement both the proposed methods, and I conducted most of the experiments for the performance evaluation of both the proposed methods and the baseline method. For the final project, I wrote the first three parts of the Experiments section, which introduces the statistics of the datasets and prompt templates, experiments setting and the implementation details for the source code. I will also participate in the preparation for the presentation.

CHEN, Yingan (20972900): In the Verbalizers section of the project proposal, I introduced two approaches for generating verbalizers: manual verbalizer and automatic verbalizer search, and compared their pros and cons. In the final report, I wrote the section of Background to provide a clear and comprehensive overview of the theoretical concepts and methods we are based on, including n-way-k-shot few-shot learning, the prompt-based tuning and prototypical verbalizer, with examples of n-way-k-shot and prompt template, a picture for illustrating the pipeline of prompt-based tuning and formulas used in the prototypical verbalizer, for readers to understand the concepts more clearly before we present our innovations.

HAO, Siyang (21002001): I wrote the section of problem statement for the project proposal. I also participated in formatting the project proposal. For the final report, I read papers related to prompt-based tuning and verbalizer design methods to better understand the concepts of prompt-based tuning and multiple verbalizer designs, as well as the advantages and disadvantages of different existing verbalizer design methods. I wrote the section of introduction (excluding the part of summary of novel design) and the section of related work for the final report. In these two parts, I discussed about the characteristics of prompt-based tuning and different verbalizers, the limitation of prototypical verbalizer (which brings up the problem to solve) and other related contents. For the teamwork, I organized zoom meetings and actively participated in group discussions. I also emailed the professor to ask our group's questions related to the project. I will prepare scripts for presentation and will participate in the presentation.

SU, Hongchang (20972649): In the Preliminary Methods section of the project proposal, I proposed the first basic idea, which involves selecting a subset of representative instances from the dataset by clustering to improve efficiency without sacrificing accuracy. In the final report, I summarized the project backgrounds and the project's focus on improving the performance of prompt-based tuning in few-shot scenarios and wrote them into the Abstract section. I am mainly in charge of producing the project slides. I will summarize the key points from each section, and I will generate graphs that help express the idea of our project effectively.

YIN, Zhuohao (20677990): I wrote the second and third part of the Preliminary Methods section in the project proposal. I actively participated in group meetings and proposed the 2 novel ideas that we have implemented in this project, i.e., Instance Mean Initialization and Shared Embedding Space. I participated in writing the source codes to implement both the proposed methods. I wrote the whole Method section in the final report and the Results & Analysis subsection in the Experiments section, including texts, equations and the table that presents the results of the ablation study. I wrote the codes for generating the t-SNE visualizations. In the following week, I will be participating in the preparation for the presentation.

References

[1] Jonathan Bragg, Arman Cohan, Kyle Lo, and Iz Beltagy. "Flex: Unifying evaluation for few-shot nlp". In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 15787–15800.

[2] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen

Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. *Language Models are Few-Shot Learners*. 2020. arXiv: 2005.14165 [cs.CL].

[3] Ganqu Cui, Shengding Hu, Ning Ding, Longtao Huang, and Zhiyuan Liu. "Prototypical Verbalizer for Prompt-based Few-shot Tuning". In: *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Ed. by Smaranda Muresan, Preslav Nakov, and Aline Villavicencio. Dublin, Ireland: Association for Computational Linguistics, May 2022, pp. 7014–7024. DOI: 10.18653/v1/2022.acl-long.483. URL: https://aclanthology.org/2022.acl-long.483.

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

[5] Ning Ding, Shengding Hu, Weilin Zhao, Yulin Chen, Zhiyuan Liu, Hai-Tao Zheng, and Maosong Sun. *OpenPrompt: An Open-source Framework for Prompt-learning*. 2021. arXiv: 2111.01998 [cs.CL].

[6] Hongyuan Dong, Weinan Zhang, and Wanxiang Che. "Metricprompt: Prompting model as a relevance metric for few-shot text classification". In: *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 2023, pp. 426–436.

[7] Chelsea Finn, Pieter Abbeel, and Sergey Levine. "Model-agnostic meta-learning for fast adaptation of deep networks". In: *International conference on machine learning*. PMLR. 2017, pp. 1126–1135.

[8] Tianyu Gao, Adam Fisch, and Danqi Chen. "Making pre-trained language models better few-shot learners". In: *arXiv preprint arXiv:2012.15723* (2020).

[9] Karen Hambardzumyan, Hrant Khachatrian, and Jonathan May. "WARP: Word-level Adversarial ReProgramming". In: *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Ed. by Chengqing Zong, Fei Xia, Wenjie Li, and Roberto Navigli. Online: Association for Computational Linguistics, Aug. 2021, pp. 4921–4933. DOI: 10.18653/v1/2021.acl-long.381. URL: https://aclanthology.org/2021.acl-long.381.

[10] Xu Han, Zhengyan Zhang, Ning Ding, Yuxian Gu, Xiao Liu, Yuqi Huo, Jiezhong Qiu, Yuan Yao, Ao Zhang, Liang Zhang, et al. "Pre-trained models: Past, present and future". In: *AI Open* 2 (2021), pp. 225–250.

[11] Shengding Hu, Ning Ding, Huadong Wang, Zhiyuan Liu, Jingang Wang, Juanzi Li, Wei Wu, and Maosong Sun. "Knowledgeable prompt-tuning: Incorporating knowledge into prompt verbalizer for text classification". In: *arXiv preprint arXiv:2108.02035* (2021).

[12] Xin Jin and Jiawei Han. "K-Means Clustering". In: *Encyclopedia of Machine Learning*. Ed. by Claude Sammut and Geoffrey I. Webb. Boston, MA: Springer US, 2010, pp. 563–564. ISBN: 978-0-387-30164-8. DOI: 10.1007/978-0-387-30164-8_425. URL: https://doi.org/10.1007/978-0-387-30164-8_425.

[13] Vandana Korde and C Namrata Mahender. "Text classification and classifiers: A survey". In: *International Journal of Artificial Intelligence & Applications* 3.2 (2012), p. 85.

[14] Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo N Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick Van Kleef, Sören Auer, et al. "Dbpedia–a large-scale, multilingual knowledge base extracted from wikipedia". In: *Semantic web* 6.2 (2015), pp. 167–195.

[15] Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. "Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing". In: *ACM Computing Surveys* 55.9 (2023), pp. 1–35.

[16] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. *RoBERTa: A Robustly Optimized BERT Pretraining Approach*. 2019. arXiv: 1907.11692 [cs.CL].

[17] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. *PyTorch: An Imperative Style, High-Performance Deep Learning Library*. 2019. arXiv: 1912.01703 [cs.LG].

[18] Timo Schick, Helmut Schmid, and Hinrich Schütze. "Automatically Identifying Words That Can Serve as Labels for Few-Shot Text Classification". In: *Proceedings of the 28th International Conference on Computational Linguistics*. Ed. by Donia Scott, Nuria Bel, and Chengqing Zong. Barcelona, Spain (Online): International Committee on Computational Linguistics, Dec. 2020, pp. 5569–5578. DOI: 10.18653/v1/2020.coling-main.488. URL: https://aclanthology.org/2020.coling-main.488.

[19] Timo Schick and Hinrich Schütze. "Exploiting cloze questions for few shot text classification and natural language inference". In: *arXiv preprint arXiv:2001.07676* (2020).

[20] Laurens Van der Maaten and Geoffrey Hinton. "Visualizing data using t-SNE." In: *Journal of machine learning research* 9.11 (2008).

[21] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. "Transformers: State-of-the-Art Natural Language Processing". In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. Ed. by Qun Liu and David Schlangen. Online: Association for Computational Linguistics, Oct. 2020, pp. 38–45. DOI: 10.18653/v1/2020.emnlp-demos.6. URL: https://aclanthology.org/2020.emnlp-demos.6.

[22] Xiang Zhang, Junbo Zhao, and Yann LeCun. *Character-level Convolutional Networks for Text Classification*. 2016. arXiv: 1509.01626 [cs.LG].