Prototypical Verbalizer for Prompt-based Few-shot Tuning

Ganqu Cui[1,2], **Shengding Hu**[1,2], **Ning Ding**[1,2], **Longtao Huang**[6], **Zhiyuan Liu**[1,2,3,4,5]*
[1]Dept. of Comp. Sci. & Tech., Institute for AI, Tsinghua University, Beijing, China
[2]Beijing National Research Center for Information Science and Technology
[3]Institute Guo Qiang, Tsinghua University, Beijing, China
[4]International Innovation Center of Tsinghua University, Shanghai, China
[5]Beijing Academy of Artificial Intelligence [6]Alibaba Group
cgq19@mails.tsinghua.edu.cn

Abstract

Prompt-based tuning for pre-trained language models (PLMs) has shown its effectiveness in few-shot learning. Typically, prompt-based tuning wraps the input text into a cloze question. To make predictions, the model maps the output words to labels via a *verbalizer*, which is either manually designed or automatically built. However, manual verbalizers heavily depend on domain-specific prior knowledge and human efforts, while finding appropriate label words automatically still remains challenging. In this work, we propose the prototypical verbalizer (ProtoVerb) which is built directly from training data. Specifically, ProtoVerb learns prototype vectors as verbalizers by contrastive learning. In this way, the prototypes summarize training instances and are able to enclose rich class-level semantics. We conduct experiments on both topic classification and entity typing tasks, and the results demonstrate that ProtoVerb significantly outperforms current automatic verbalizers, especially when training data is extremely scarce. More surprisingly, ProtoVerb consistently boosts prompt-based tuning even on untuned PLMs, indicating an elegant non-tuning way to utilize PLMs. Our codes are avaliable at https://github.com/thunlp/OpenPrompt.

1 Introduction

The massive-scale pre-trained language models (PLMs) (Han et al., 2021a) have been proven to be backbones for solving a variety of NLP tasks (Kowsari et al., 2019; Rajpurkar et al., 2016). To further adapt these PLMs to downstream tasks such as classification, traditional approaches fine-tune the language models through an extra classifier (Howard and Ruder, 2018). However, when task-specific data is limited (Bragg et al., 2021), training the extra classifier effectively is challenging due to the gap between pre-training tasks (e.g., masked language modeling) and fine-tuning tasks (e.g., classification and regression). This gap impedes the fast adaptation of PLMs to downstream tasks.

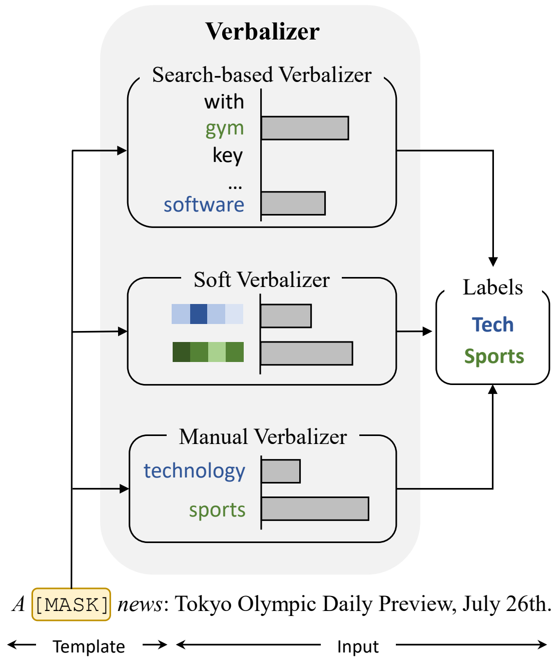

Figure 1: Illustration of three verbalizer construction methods.

Recently, prompt-based tuning (Schick and Schütze, 2021; Liu et al., 2021) has risen to be a powerful way for few-shot learning by bridging the gap between the pre-training stage and downstream task stage. In prompt-based tuning, the input texts are wrapped with task-specific templates to re-formalize the original task as a cloze-style task. For example, in topic classification task, we can use template "`<text>` This topic is about `[MASK]`", where `<text>` is the placeholder for input sentences. The PLMs are asked to infer the words to fill in `[MASK]` and the words are further mapped to corresponding labels through a *verbalizer* (e.g. "sports" for label "Sports"). Verbaliz-

[*] Corresponding author: Z.Liu (liuzy@tsinghua.edu.cn)

Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics
Volume 1: Long Papers, pages 7014 - 7024
May 22-27, 2022 ©2022 Association for Computational Linguistics

ers are of great importance in prompt-based tuning (Gao et al., 2021) since they are the bridges between model outputs and the final predictions. How to build effective verbalizers for prompt-based tuning—especially for many-class classification, is a critical issue in prompt-based tuning.

Typically, most current works adopt three kinds of verbalizers: manual verbalizers, search-based verbalizers, and soft verbalizers. We show them by an example in Figure 1. Human-designed manual verbalizers pick some label words (e.g. label names) to depict classes. These verbalizers are powerful across multiple tasks (Schick and Schütze, 2021). Despite their success, a major drawback roots in the strong assumption that we own precise understandings of downstream tasks and are able to sum up each class with several words. Without task-specific prior knowledge, selecting appropriate label words is non-trivial. Further, they also need intensive human labors when facing many classes. To mitigate these issues, search-based verbalizers aim at finding suitable label words from vocabulary with algorithms (Schick et al., 2020; Shin et al., 2020; Gao et al., 2021) and soft verbalizers use trainable tokens which are optimized during tuning (Hambardzumyan et al., 2021; Zhang et al., 2021). However, it is challenging to search or optimize adequately in a large vocabulary or embedding space under a low-data regime, making automatic verbalizers suboptimal compared with manual ones.

Intuitively, class proxies in verbalizers should encapsulate class-level semantic features, which are expressed implicitly by instances. To obtain these semantic representatives with few data, one promising approach is computing central points of class instances, namely *prototypes*, as approximation. To this end, we manage to estimate prototype vectors for each class to serve as verbalizer. Summarized from instances, prototypes are supposed to establish concepts similar with human-designed labels.

In this work, we introduce prototypes into this problem and propose prototypical verbalizer (ProtoVerb), which learns class prototypes from training data to build verbalizers automatically. For prototype learning, inspired by the idea of PCL (Li et al., 2021), ProtoVerb trains the prototype vectors by contrastive learning with the InfoNCE estimator (Oord et al., 2018). Specifically, our optimization objective includes two components: The first part is an instance-instance loss to cluster intra-class instances and separate inter-class instances; The second part is an instance-prototype loss which enforces the prototypes to be center points of classes. Compared with other verbalizer construction methods, ProtoVerb learns continuous vectors straight from training instances efficiently, which makes it a plug-in-and-play algorithm with high flexibility.

To verify the effectiveness of ProtoVerb, we conduct extensive experiments on topic classification and entity typing tasks. We study two different settings where ProtoVerb can work: (1) When manual verbalizers are available, ProtoVerb can play as an extra verbalizer in the inference stage. Results show that ProtoVerb consistently improves the classification performance with low cost, and even untuned PLMs benefit largely. (2) Consider a realistic setting where only a limited number of samples are provided with no manual verbalizers, ProtoVerb also produces verbalizers of high quality. Experimental results demonstrate that ProtoVerb significantly outperforms existing search-based and soft verbalizers.

2 Related Work

2.1 Prompt-based Tuning

Despite the success of PLMs (Devlin et al., 2019; Liu et al., 2019; Raffel et al., 2019) in massive NLP tasks, few-shot fine-tuning of PLMs was suboptimal due to the gap between pre-training and downstream tasks. Inspired by the "in context learning" proposed by GPT-3 (Brown et al., 2020), stimulating model knowledge with a few *prompts* has recently received much attention. A series of prompt-based work on knowledge probing (Trinh and Le, 2018; Petroni et al., 2019; Davison et al., 2019), text classification (Schick and Schütze, 2021; Gao et al., 2021), relation extraction (Han et al., 2021b), and entity typing (Ding et al., 2021a) emerge and achieve impressive progress. Typically, a piece of prompt contains a template and a verbalizer. Early prompts employ human-picked prompts which demand human knowledge and manual efforts. To alleviate this issue, later works explore automatic designing and optimizing prompts (Liu et al., 2021; Gao et al., 2021; Zhang et al., 2021). Recently research works further propose continuous prompts to replace the discrete phrases (Lester et al., 2021; Li and Liang, 2021). However, the designation of verbalizers, an important part of prompts, is less explored. In this work, we investigate the automatic verbalizer construction in prompt-based tuning.

2.2 Verbalizer Design

Verbalizers bridge between model outputs and labels and make great impact on prompt-based tuning (Gao et al., 2021). With task-specific knowledge, human-picked words are widely used and proved effective (Schick and Schütze, 2021). The major drawback of manual verbalizers is the assumption that we possess sufficient knowledge of downstream tasks, which is not always satisfied. To avoid intensive human labor and expert knowledge dependency in manual verbalizers, some works explore search-based verbalizers (Schick et al., 2020; Gao et al., 2021; Shin et al., 2020) that identify label words automatically with training data. However, with a large vocabulary and few examples, it is non-trivial to find suitable words. Another line of researches focuses on soft verbalizers (Hambardzumyan et al., 2021; Zhang et al., 2021), which insert continuous embeddings as soft labels. The label embeddings are optimized along with model tuning. Similarly, soft verbalizers require abundant data for sufficient optimization, which can not be satisfied with the few-shot setting. In contrast, our approach learns prototype vectors from scratch, hence is more effective for few-shot tuning.

2.3 Prototype-based Few-shot Learning

In few-shot learning, prototype-based metric-learning methods have been promising approaches for their simplicity and effectiveness. Prototypical Networks (ProtoNet) (Snell et al., 2017) is the pioneering work that introduces prototypes into deep learning. Specifically, ProtoNet calculates prototype vectors by taking the average of instance vectors and makes predictions by metric-based comparisons between prototypes and query instances. A set of following works concentrates on the advancement of prototype estimation (Li et al., 2021; Gao et al., 2019; Ding et al., 2021c). Among them, PCL (Li et al., 2021) achieves remarkable results on self-supervised few-shot learning by using prototypes as latent variables and inspires us in designing training objectives. The success of prototype-based models indicates that prototypes, which are representative embeddings of instances from the same classes, encapsulate some class-level semantic features. Inspired by the intrinsic similarity of prototypes and verbalizers, we find it natural and elegant to introduce prototypes into verbalizer construction for prompt-based tuning.

3 Background

Given a pre-trained language model $\mathcal{M}$, our goal is to tune it for specific downstream tasks. Take N way K shot few-shot text classification as an example, the support set for class n $\mathcal{D}_n = \{x_1^n, \cdots, x_K^n\}$ contains K sentences. We aim to predict the label $y \in \mathcal{Y}$ for each sentence, where $\mathcal{Y}$ is the label set with N distinct classes.

3.1 Fine-tuning

For a sentence concatenated with special tokens $x = \{\texttt{[CLS]}, t_1, \cdots, t_T, \texttt{[SEP]}\}$, language model $\mathcal{M}$ encodes it into hidden representations $\{\mathbf{h}_{\texttt{[CLS]}}, \mathbf{h}_1, \cdots, \mathbf{h}_T, \mathbf{h}_{\texttt{[SEP]}}\}$. Conventional fine-tuning trains an extra classifier F over the `[CLS]` embedding $\mathbf{h}_{\texttt{[CLS]}}$ and output the probability distribution on label set $\mathcal{Y}$.

$$P(\cdot|x) = \text{Softmax}(F(\mathbf{h}_{\texttt{[CLS]}})). \tag{1}$$

The classifier and PLM are tuned by maximizing $\frac{1}{N}\sum_{i=1}^{N} \log P(y_i|x_i)$, where y_i is the label of x_i.

3.2 Prompt-based Tuning

The vanilla prompt-based tuning converts the downstream task to a cloze-style mask language modeling problem. For example, to formulate the text classification task, we can modify the original input x with a template $\mathcal{T}(\cdot) =$ A `[MASK]` news: to get the prompt input $\mathcal{T}(x) =$ A `[MASK]` news: x. With $\mathcal{T}(x)$, $\mathcal{M}$ produces the hidden vector at the `[MASK]` position $\mathbf{h}_{\texttt{[MASK]}}$. To calculate the probability distribution over the label set, a manual verbalizer stores a set of label words $\mathcal{V}$ and the score for 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 label words of y and $g(\cdot)$ is to aggregate multiple scores.

4 Prototypical Verbalizer

In previous sections, we introduce the general pipeline of prompt-based tuning. As manually defining or automatically searching for appropriate verbalizers can be challenging, here we propose to learn prototypes directly from training instances. Inspired by PCL (Li et al., 2021), the prototypes are trained with contrastive learning. As shown in

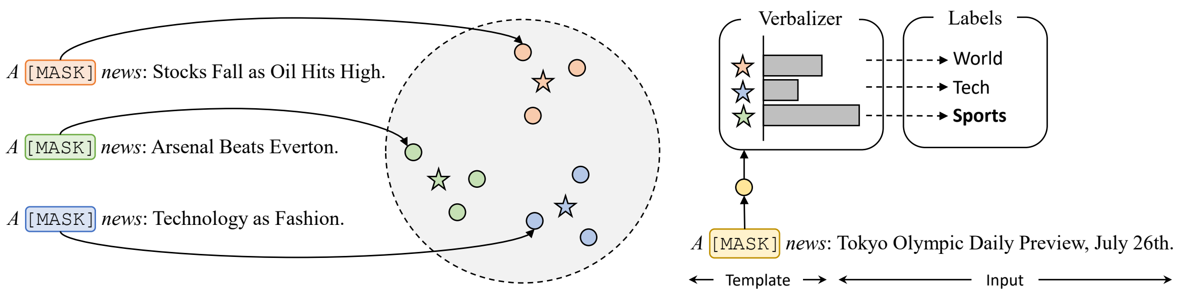
A [MASK] news: Stocks Fall as Oil Hits High.
A [MASK] news: Arsenal Beats Everton.
A [MASK] news: Technology as Fashion.
Verbalizer
Labels
World
Tech
Sports
A [MASK] news: Tokyo Olympic Daily Preview, July 26th.
Template
Input

Figure 2: Illustration of ProtoVerb. Left: We project the hidden states of [MASK] tokens to the embedding space and learn prototypes. Right: The learned prototypes constitute the verbalizer and map the PLM outputs to corresponding labels.

Figure 2, we first get the hidden states of [MASK] tokens to represent instances, then project them to another embedding space for prototype learning. The prototypes are used as verbalizers for prediction. Next, we will introduce the learning and inference stages of ProtoVerb in detail.

4.1 Instance Representation and Similarity Function

Given a piece of training text x wrapped with a template, we take the last layer's hidden state of the [MASK] token $\mathbf{h}_{\texttt{[MASK]}}$ as the initial representation of the text. With an encoder $E_\phi(\cdot)$ parameterized by ϕ, the instance representation of x is

$$\mathbf{v} = E_\phi(x) = \mathbf{W}\mathbf{h}_{\texttt{[MASK]}}. \tag{3}$$

In practice, we simply adopt a linear encoder with weight $\mathbf{W}$. To measure the similarity between instances, we adopt cosine similarity function $S(\cdot)$, where

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

4.2 Loss Function

With the instance representation and similarity function, we discuss how to define our training objective. Denote $\mathcal{C} = \{\mathbf{c}_1, \cdots, \mathbf{c}_N\}$ as the set of prototype vectors. Intuitively, there are two goals we need to achieve by optimization: (1) For instance-instance pairs, intra-class pairs should get higher similarity scores than inter-class pairs. (2) For instance-prototype pairs, the similarity scores between prototype $\mathbf{c}_n$ and instances of class n should be higher than $\mathbf{c}_n$ and other instances. To realize these two goals, we define the objective function based on the InfoNCE estimator (Oord et al., 2018), which is widely adopted in contrastive learning.

For the instance-instance objective, we minimize the following loss function

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

where $(\mathbf{v}_i^n, \mathbf{v}_j^n)$ are instance pairs of the same class. This loss function maximizes intra-class similarity and minimizes inter-class similarity between instances.

Similarly, the instance-prototype loss function is defined as

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

and $\mathbf{v}_i^n$ is of class n. This objective forces each prototype to lie at the center point of its instances.

Overall, combining the instance-instance loss and instance-prototype loss, our final training objective is

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

4.3 Inference

During inference, following the same metric, we calculate the similarity scores of query and prototypes. The probability score for class k is

$$P_{\mathcal{M}}(y_k|x) = \frac{\exp S(\mathbf{v}, \mathbf{c}_k)}{\sum_{k'}\exp S(\mathbf{v}, \mathbf{c}_{k'})}. \tag{8}$$

Then we make prediction by arg max function

$$\widetilde{y} = \arg\max_k P_{\mathcal{M}}(y_k|x). \tag{9}$$

When there are other verbalizers (e.g. manual verbalizers), we first process the logits from different verbalizers with a standard scaler (minus mean then divide by standard deviation). Then we take the mean value of the scores to get the final score.

5 Experiments

We conduct extensive few-shot learning experiments to illustrate the effectiveness of ProtoVerb. In this section, we first introduce the experimental settings in use. Then we present and discuss the experiment results.

5.1 Datasets and Templates

Verbalizers in many-class classification tasks are difficult to get precise definitions. Hence we adopt three topic classification datasets: AG's News, Yahoo (Zhang et al., 2015), and DBPedia (Lehmann et al., 2015) and one entity typing dataset: FewNERD (Ding et al., 2021d) as benchmarks, and their statistics are summarized in Table 1.

To focus on the verbalizer and alleviate the influence of templates, we adopt multiple fixed manual templates. For topic classification, following (Hu et al., 2021), we use four templates on each dataset. For entity typing, we use three templates from (Ding et al., 2021a). Details about the templates can be found in Appendix A.

Dataset	Task	#Class	#Test
AG's News	TC	4	7,600
DBPedia	TC	14	70,000
Yahoo	TC	10	60,000
FewNERD	ET	66	96,901

Table 1: Dataset statistics. TC is for topic classification and ET is for entity typing.

5.2 Experimental Settings

Under the few-shot setting, we randomly sample $k = 1, 2, 4, 8, 16$ instances in each class from the training set and test the model on the entire test set. As for the evaluation metric, we use accuracy in all experiments. For the different usages of ProtoVerb, we consider two specific settings:

(1) ProtoVerb as a single verbalizer (§ 5.5). When manual verbalizers are not available, we can tune the model with ProtoVerb. Under this setting, we want to evaluate the performance of ProtoVerb compared with other automatic verbalizer construction methods.

(2) ProtoVerb as an extra verbalizer (§ 5.6). Naturally, we suppose that there exists a manual verbalizer and we append ProtoVerb to strengthen the performance. Under this setting, ProtoVerb is a plug-in-and-play component and does not participate in the tuning process. We compare ProtoVerb with manual verbalizers and other verbalizer ensembles.

5.3 Implementation Details

All our models and baselines are implemented with PyTorch (Paszke et al., 2019) framework, Huggingface transformers (Wolf et al., 2020), and OpenPrompt toolkit (Ding et al., 2021b). We optimize PLMs with AdamW optimizer (Loshchilov and Hutter, 2019). For prototype learning, we set the prototype dimension to 128 and optimize the loss function with Adam optimizer (Kingma and Ba, 2015). For topic classification, we use RoBERTa-large (Liu et al., 2019) as our PLM backbone and tune the model for 5 epochs. The batchsize is 2 and the learning rate is 3e-5. For entity typing, we tune a BERT-base (Devlin et al., 2019) model for 30 epochs and set the batchsize to 16. The learning rate here is 5e-5.

5.4 Baselines

The vanilla prompt-based tuning method fuses the input text with a task-specific template and maps the model outputs to labels through a verbalizer. For fair comparisons, all our baselines and proposed models are built on this pipeline and they merely differ from the verbalizers in use.

Manual verbalizers (ManualVerb) are defined by human with domain knowledge. Here we simply employ the verbalizers provided by OpenPrompt (Ding et al., 2021b).

Search-based verbalizers (SearchVerb) search for suitable words from vocabulary automatically. We adopt the implementation in PETAL (Schick et al., 2020), which finds the words that maximize the likelihood of the training data. To combine SearchVerb with ManualVerb, we merge their verbalizer words together.

Soft verbalizers (SoftVerb) introduce trainable tokens as verbalizers in prompt-based tuning. We follow the approach in WARP (Hambardzumyan et al., 2021) that applies soft tokens as a linear decoding layer, and the token embeddings are learned along with model tuning. Note that the templates in WARP are also trainable, but here we only use its soft verbalizers. In single verbalizer experiments, we initialize the token embeddings randomly for fairness. And in extra verbalizer experiments, they are initialized with label names.

K	Method	AG	DB	Yahoo	Few
0	ManualVerb	*75.13*	*67.06*	*43.11*	*20.00*
1	ManualVerb	*76.67*	*85.47*	*50.22*	*41.68*
	SearchVerb	41.50	60.06	27.39	20.88
	SoftVerb	49.79	65.35	22.72	18.78
	ProtoVerb	**64.19**	**72.85**	**36.12**	**25.00**
2	ManualVerb	*81.06*	*93.61*	*58.65*	*46.44*
	SearchVerb	65.82	78.21	40.71	31.28
	SoftVerb	56.37	80.69	30.72	32.80
	ProtoVerb	**77.34**	**85.49**	**46.30**	**35.72**
4	ManualVerb	*84.73*	*95.83*	*61.41*	*52.54*
	SearchVerb	77.43	86.40	51.58	43.10
	SoftVerb	74.38	89.12	41.62	**48.77**
	ProtoVerb	**81.65**	**90.91**	**55.08**	48.28
8	ManualVerb	*85.85*	*96.46*	*64.12*	*56.59*
	SearchVerb	82.17	88.41	58.64	50.78
	SoftVerb	79.35	93.69	46.82	53.78
	ProtoVerb	**84.03**	**95.75**	**61.40**	**56.06**
16	ManualVerb	*84.74*	*96.05*	*58.77*	*61.17*
	SearchVerb	83.40	92.00	59.66	55.49
	SoftVerb	80.57	86.90	58.20	58.87
	ProtoVerb	**84.48**	**96.30**	**64.35**	**61.29**

Table 2: Results for single verbalizer experiments. We report the mean accuracy scores (%) over 3 random seeds. *Italic*: results with task-specific knowledge. **Bold**: best results without task-specific knowledge.

5.5 Single Verbalizer Results

Table 2 presents the performance of different verbalizers. Overall, ManualVerb is the most powerful verbalizer, which is reasonable because it is picked by human with domain knowledge. ProtoVerb outperforms SearchVerb and SoftVerb remarkably and consistently, especially when only 1 or 2 instances per class are given. The poor performances of the two baselines under extreme data scarcity corroborate the issues we claim in § 1. As the training data become sufficient, ProtoVerb gets comparable or even exceeding scores compared with ManualVerb, showing that ProtoVerb is able to learn prototypes that well represent the classes. At the same time, the gaps between ManualVerb and other verbalizers narrow, which also indicates that we can summarize data across various ways.

Across tasks, ProtoVerb gets better results on topic classification than entity typing. A possible reason is that FewNERD is a fine-grained entity typing dataset, in which the differences across classes are subtle. For example, it is hard for ProtoVerb to discriminate between "person-artist/author" and "person-director" with only a few instances. However, ProtoVerb can also catch up with ManualVerb with enough samples.

K	Method	AG	DB	Yahoo	Few
0	ManualVerb	75.13	67.06	43.11	20.00
1	Fine-tuning	25.45	10.80	10.59	7.48
	ManualVerb	76.67	85.47	**50.22**	41.68
	SearchVerb+	51.82	81.31	43.24	35.64
	SoftVerb+	76.34	85.85	49.11	37.66
	ProtoVerb+	**77.71**	**88.16**	50.08	**43.20**
	w/o tuning	76.28	78.32	45.01	29.51
2	Fine-tuning	25.78	49.01	11.26	19.03
	ManualVerb	81.06	93.61	58.65	46.44
	SearchVerb+	77.56	91.79	52.46	42.13
	SoftVerb+	79.95	93.68	55.73	42.17
	ProtoVerb+	**84.09**	**94.77**	**59.33**	**48.69**
	w/o tuning	82.13	86.11	50.34	34.44
4	Fine-tuning	28.14	94.08	26.02	20.98
	ManualVerb	84.73	95.83	61.41	52.54
	SearchVerb+	81.25	95.16	58.98	50.61
	SoftVerb+	84.22	94.90	59.01	49.45
	ProtoVerb+	**85.71**	**96.74**	**66.14**	**54.16**
	w/o tuning	83.05	89.56	55.59	35.55
8	Fine-tuning	72.78	96.83	54.76	49.77
	ManualVerb	85.85	96.46	64.12	56.59
	SearchVerb+	85.68	97.57	65.32	56.58
	SoftVerb+	86.54	97.40	63.48	54.30
	ProtoVerb+	**87.25**	**97.64**	**66.61**	**58.30**
	w/o tuning	83.79	92.61	59.42	34.37
16	Fine-tuning	84.14	**97.25**	64.27	52.66
	ManualVerb	84.74	96.05	58.77	61.17
	SearchVerb+	85.30	95.08	59.34	61.70
	SoftVerb+	85.65	96.34	58.68	59.23
	ProtoVerb+	**87.98**	97.22	**65.65**	**62.55**
	w/o tuning	84.78	93.46	60.89	33.96

Table 3: Results for multiple verbalizer experiments. We report the mean accuracy scores (%) over 3 random seeds. ProtoVerb+ w/o tuning: apply ProtoVerb to untuned PLMs. **Bold**: best results.

5.6 Multiple Verbalizer Results

Table 3 shows the experiment results when we ensemble manual verbalizers with automatic verbalizers. The ensembled versions are denoted as SearchVerb+, SoftVerb+, and ProtoVerb+ respectively. From the table, we have the following observations: (1) Basically, prompt-based tuning outperforms fine-tuning by a large margin with few samples (1∼2 per class). When sufficient training data is available, fine-tuning models will produce comparable results. (2) Overall, ProtoVerb+ certainly improves the performance of prompt-based tuning under most cases, which demonstrates the effectiveness of ProtoVerb+. At the same time, SearchVerb+ and SoftVerb+ seldom show enhancement compared with ManualVerb. As ProtoVerb+ does not introduce any external knowledge, this illustrates that ProtoVerb+ provides a better way to utilize training data.

Finally, we also present the results of applying ProtoVerb+ on untuned PLMs. It is worth noting that even for untuned models, ProtoVerb+ also boosts them considerably on all tasks. For example on DBPedia, showing only one instance per class to PLMs with ProtoVerb+ leads to 11.26% absolute accuracy improvement. On topic classification, when more training samples are given, untuned PLMs achieve competitive scores. This observation indicates a new cost-efficient way to leverage training data, which we highlight as valuable for future study of **none-tuning** methods for PLMs. Compared to the "in context learning" in GPT-3 (Brown et al., 2020), ProtoVerb+ is not limited by input length and can deal with arbitrary number of samples. We further study this "fixed model" scenario in § 6.1.

6 Analysis

In this section, we discuss several analytical topics for further understandings of ProtoVerb. For simplicity, we conduct experiments on AG's News dataset.

6.1 Fixed Model Experiments

In § 5.6, we see ProtoVerb is still powerful with fixed PLMs. For further comparisons, we conduct experiments to quantitatively evaluate verbalizers when PLMs are fixed. Figure 3 gives the results. To clarify, using ManualVerb on fixed PLMs equals the zero-shot setting, which we plot with a dashed line. Meanwhile, different from § 5.6, ProtoVerb here is a single verbalizer. From the figure we can conclude that (1) Similar with § 5.5, ProtoVerb outperforms SoftVerb and SearchVerb by a large margin under low-shot settings. Notably, ProtoVerb exceeds ManualVerb with only 2 shots per class, illustrating the experessive power of prototypes. (2) SoftVerb is also better than SearchVerb under this setting, demonstrating that tunable verbalizers could exploit training data better with PLMs fixed.

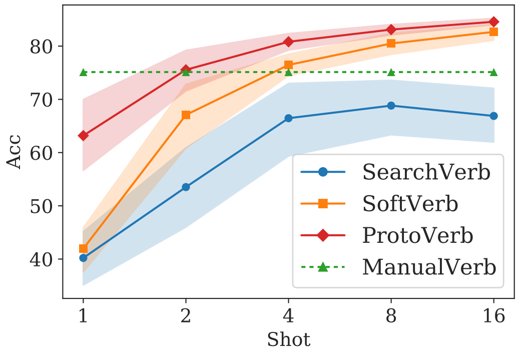

Figure 3: Experiment results with fixed PLMs. We report the mean accuracy (%) with 95% confidence interval on AG's News.

Method	$K=2$	$K=4$	$K=8$
$\mathcal{L}_{\text{ins}} + \mathcal{L}_{\text{proto}}$	**77.34**	**81.65**	**84.03**
$\mathcal{L}_{\text{proto}}$	76.37	81.06	82.91
Instance Mean	73.36	77.76	82.57

Table 4: Ablation study of ProtoVerb on AG's News. Instance Mean: using the mean embeddings of instances as prototype embeddings. **Bold**: best results

K	Method	# Noisy Samples		
		1	2	3
8	SearchVerb	4.86	5.96	5.19
	SoftVerb	4.84	7.80	11.71
	ProtoVerb	**2.34**	**3.11**	**4.37**
16	SearchVerb	0.80	2.93	5.18
	SoftVerb	2.01	4.17	4.58
	ProtoVerb	**0.04**	**2.13**	**3.16**

Table 5: Accuracy drop (%) with noisy samples. Lower is better. **Bold**: best results.

6.2 Ablation Study

To validate the effect of each part in the loss function, we conduct an ablation study on AG's News dataset. For comparison, we consider two variants of prototype calculation methods: (1) ProtoVerb with $\mathcal{L}_{\text{proto}}$ only. (2) Following ProtoNet (Snell et al., 2017), take the average of instance embeddings for prototype embeddings. Table 4 shows the results. Compared to taking the mean embedding

Class	$K = 1$	$K = 16$
World	Qaida, Syria, Iraq, Nusra, TPP	Taliban, Iraq, Afghan, militants, rebellion
Sports	Steelers, Raptors, Knicks, Dodgers	ball, ESPN, baseball, Fifa, Sports
Business	cash, earnings, Securities, NYSE	Dow, dividend, investing, markets
Tech	LTE, Tel, Huawei, Mbps, VPN	Vault, IBM, Qualcomm, Technologies

Table 6: Words that are most similar with prototypes of each class on AG's News.

vectors directly, optimizing the embedding vectors of prototypes using our loss functions leads to better performances and stability. Adding $\mathcal{L}_{\text{ins}}$ is also beneficial, meaning that $\mathcal{L}_{\text{ins}}$ helps ProtoVerb in learning instance embeddings.

6.3 Robustness on Noisy Samples

Noisy data are commonly seen as threats in real-world datasets for few-shot learning systems. For automatic verbalizers, noisy data are more harmful because of the effect on both the quality of verbalizers and the training process. In this section, we evaluate the robustness of different automatic verbalizers against noisy samples on AG's News. For training stability, we set $K = 8, 16$. Table 5 presents the accuracy drop when there are 1, 2, or 3 samples having wrong labels. It is clearly seen that a limited number of noisy samples will hinder the performance greatly, showing the vulnerability of automatic verbalizers. Meanwhile, we can also find that ProtoVerb is more robust than baseline methods when facing noisy samples.

6.4 Prototype Discretization

Since ProtoVerb learns continuous prototype vectors, their meanings are implicit. Here we manage to investigate which words are most similar to the learned prototypes. Due to word embeddings and prototype vectors lying in different embedding spaces, we can not directly calculate their similarity. Hence we use the vocabulary as the input texts (one word at a time) to get the top-scored word for each class. On AG's News dataset, we collect some most similar words for each class and list them in Table 6.

To investigate the property of prototypes learned with different numbers of samples, we present words for $K = 1$ and $K = 16$. With the table, we see that: (1) Even when only one example is available, the learned prototypes are meaningful. Most of the similar words are proper nouns and entity names closely related to class topics. For example, "Steelers", "Raptors", "Knicks", and "Dodgers" are all baseball or basketball teams that appear frequently in sports news. We attribute this to prompt mechanism that allows PLMs to extract the most conclusive information and fill the [MASK] with it. Then the relevant words are also included. (2) With more training instances, prototypes show diverse interests. Despite entity names, more "conceptual" words show up on the list, such as "ball" and "Sports" for class Sports. We interpret this as the summarization and abstraction ability of prototypes. Given many instances, prototypes are enforced to capture their common features, hence some abstract concepts are found automatically. In this way, ProtoVerb encapsulates class-level, rather than entity-level, semantics, which leads to better performance on unseen data.

6.5 Is ProtoVerb Similar with ManualVerb?

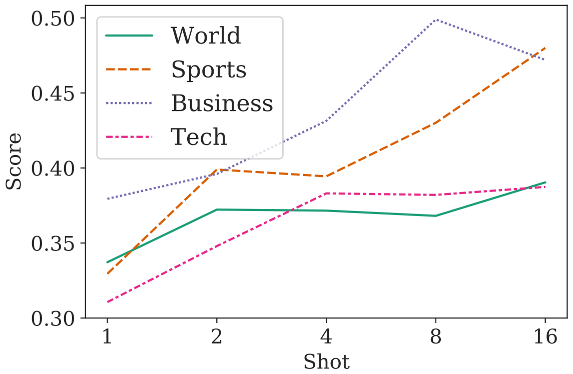

Figure 4: Similarity scores between ProtoVerb and ManualVerb on AG's News.

To give further analyses for the inner workings of prototypes, we measure the similarity between ProtoVerb and ManualVerb to see whether ProtoVerb is able to learn abstract concepts as humans do. On AG's News dataset, we calculate the similarity scores between prototypes and manual verbalizers and normalize the scores using the softmax function across the four classes. In Figure 4 we plot the scores with various shots. It is clearly seen that the similarity of prototypes and corresponding verbalizers are above average (0.25). As shot increases, the scores also gradually grow, which illustrates that prototypes can capture the conceptual information better from more instances. This observation matches our findings in § 6.4. Among the four classes, Business and Sports get higher scores than World and Tech. A reasonable guess is that World and Tech news includes diverse sub-topics that are hard to summarize.

7 Conclusion

In this paper, we propose a novel approach for automatic verbalizer construction in prompt-based tuning. The proposed ProtoVerb learns class prototypes from training instances using contrastive learning. We explore the performance of ProtoVerb on few-shot topic classification and entity typing tasks. As a single verbalizer, ProtoVerb outperforms state-of-the-art automatic verbalizers considerably. Working together with manual verbalizers, ProtoVerb can also consistently improve prompt-based tuning with minor effort. The results validate the effectiveness of ProtoVerb. Our analysis further reveals the intrinsic properties of prototypes. For future work, we will focus on extending ProtoVerb for effective non-tuning algorithms of PLMs and prompt-tuning with soft templates. Moreover, we are finding proper ways to combine label words and prototypes for verbalizer construction.

Acknowledgements

This work is supported by the National Key R&D Program of China (No. 2020AAA0106502), Institute for Guo Qiang at Tsinghua University, Beijing Academy of Artificial Intelligence (BAAI), and International Innovation Center of Tsinghua University, Shanghai, China. Ganqu Cui and Shengding Hu conducted the experiments. Ganqu Cui, Shengding Hu, Ning Ding and Zhiyuan Liu wrote the paper. Longtao Huang provided valuable advices to the research.

Ethical Considerations

Our work explores how to stimulate large PLMs with few samples. We conduct experiments under the few-shot setting, where requires less training time and fewer resources than normal full-data setting. Also, we open up our codes and hyperparameters to facilitate future reproduction without repeated energy cost.

References

Jonathan Bragg, Arman Cohan, Kyle Lo, and Iz Beltagy. 2021. Flex: Unifying evaluation for few-shot nlp. *arXiv preprint arXiv:2107.07170*.

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. 2020. Language models are few-shot learners. In *Proceedings of NeurIPS*.

Joe Davison, Joshua Feldman, and Alexander M. Rush. 2019. Commonsense knowledge mining from pretrained models. In *Proceedings of EMNLP-IJCNLP*, pages 1173–1178.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: pre-training of deep bidirectional transformers for language understanding. In *Proceedings of NAACL-HLT*, pages 4171–4186.

Ning Ding, Yulin Chen, Xu Han, Guangwei Xu, Pengjun Xie, Hai-Tao Zheng, Zhiyuan Liu, Juanzi Li, and Hong-Gee Kim. 2021a. Prompt-learning for fine-grained entity typing. *arXiv preprint arXiv:2108.10604*.

Ning Ding, Shengding Hu, Weilin Zhao, Yulin Chen, Zhiyuan Liu, Hai-Tao Zheng, and Maosong Sun. 2021b. Openprompt: An open-source framework for prompt-learning. *arXiv preprint arXiv:2111.01998*.

Ning Ding, Xiaobin Wang, Yao Fu, Guangwei Xu, Rui Wang, Pengjun Xie, Ying Shen, Fei Huang, Hai-Tao Zheng, and Rui Zhang. 2021c. Prototypical representation learning for relation extraction. In *Proceedings of ICLR*.

Ning Ding, Guangwei Xu, Yulin Chen, Xiaobin Wang, Xu Han, Pengjun Xie, Haitao Zheng, and Zhiyuan Liu. 2021d. Few-nerd: A few-shot named entity recognition dataset. In *Proceedings of ACL*, pages 3198–3213.

Tianyu Gao, Adam Fisch, and Danqi Chen. 2021. Making pre-trained language models better few-shot learners. In *Proceedings of ACL*, pages 3816–3830.

Tianyu Gao, Xu Han, Zhiyuan Liu, and Maosong Sun. 2019. Hybrid attention-based prototypical networks for noisy few-shot relation classification. In *Proceedings of AAAI*, pages 6407–6414.

Karen Hambardzumyan, Hrant Khachatrian, and Jonathan May. 2021. WARP: word-level adversarial reprogramming. In *Proceedings of ACL*, pages 4921–4933.

Xu Han, Zhengyan Zhang, Ning Ding, Yuxian Gu, Xiao Liu, Yuqi Huo, Jiezhong Qiu, Liang Zhang, Wentao Han, Minlie Huang, et al. 2021a. Pre-trained models: Past, present and future. *AI Open*.

Xu Han, Weilin Zhao, Ning Ding, Zhiyuan Liu, and Maosong Sun. 2021b. Ptr: Prompt tuning with rules for text classification. *arXiv preprint arXiv:2105.11259*.

Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. *arXiv preprint arXiv:1801.06146*.

Shengding Hu, Ning Ding, Huadong Wang, Zhiyuan Liu, Juanzi Li, and Maosong Sun. 2021. Knowledgeable prompt-tuning: Incorporating knowledge into prompt verbalizer for text classification. *arXiv preprint arXiv:2108.02035*.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of ICLR*.

Kamran Kowsari, Kiana Jafari Meimandi, Mojtaba Heidarysafa, Sanjana Mendu, Laura Barnes, and Donald Brown. 2019. Text classification algorithms: A survey. *Information*, 10(4):150.

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

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. In *Proceedings of EMNLP*.

Junnan Li, Pan Zhou, Caiming Xiong, and Steven C. H. Hoi. 2021. Prototypical contrastive learning of unsupervised representations. In *Proceedings of ICLR*.

Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. In *Proceedings of ACL/IJCNLP*, pages 4582–4597.

Xiao Liu, Yanan Zheng, Zhengxiao Du, Ming Ding, Yujie Qian, Zhilin Yang, and Jie Tang. 2021. Gpt understands, too. *arXiv preprint arXiv:2103.10385*.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *Proceedings of ICLR*.

Aaron van den Oord, Yazhe Li, and Oriol Vinyals. 2018. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*.

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 Z. Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. Pytorch: An imperative style, high-performance deep learning library. In *Proceedings of NeurIPS*, pages 8024–8035.

Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick S. H. Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander H. Miller. 2019. Language models as knowledge bases? In *Proceedings of EMNLP-IJCNLP*, pages 2463–2473.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2019. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv preprint arXiv:1910.10683*.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. In *Proceedings of EMNLP*, pages 2383–2392.

Timo Schick, Helmut Schmid, and Hinrich Schütze. 2020. Automatically identifying words that can serve as labels for few-shot text classification. In *Proceedings of COLING*, pages 5569–5578.

Timo Schick and Hinrich Schütze. 2021. Exploiting cloze-questions for few-shot text classification and natural language inference. In *Proceedings of EACL*, pages 255–269.

Taylor Shin, Yasaman Razeghi, Robert L. Logan IV, Eric Wallace, and Sameer Singh. 2020. Autoprompt: Eliciting knowledge from language models with automatically generated prompts. In *Proceedings of EMNLP*, pages 4222–4235.

Jake Snell, Kevin Swersky, and Richard S. Zemel. 2017. Prototypical networks for few-shot learning. In *Proceedings of NIPS*, pages 4077–4087.

Trieu H Trinh and Quoc V Le. 2018. A simple method for commonsense reasoning. *arXiv preprint arXiv:1806.02847*.

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. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of EMNLP*, pages 38–45, Online.

Ningyu Zhang, Luoqiu Li, Xiang Chen, Shumin Deng, Zhen Bi, Chuanqi Tan, Fei Huang, and Huajun Chen. 2021. Differentiable prompt makes pre-trained language models better few-shot learners. *arXiv preprint arXiv:2108.13161*.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Proceedings of NIPS*, volume 28, pages 649–657.

A Templates

For topic classification, we use the default templates and verbalizers in OpenPrompt (Ding et al., 2021b).

AG's News is a news' topic classification dataset. There are four categories: World, Sports, Business, and Tech. We use the following templates.

$$\mathcal{T}_1(x) = \text{A } \texttt{[MASK]} \text{ news: } x$$
$$\mathcal{T}_2(x) = x \text{ This topic is about } \texttt{[MASK]}.$$
$$\mathcal{T}_3(x) = [\text{ Category : } \texttt{[MASK]} \text{] } x$$
$$\mathcal{T}_4(x) = [\text{ Topic : } \texttt{[MASK]} \text{] } x$$

DBPedia is an ontology classification dataset. Each sample contains an article title x and abstract y extracted from Wikipedia, and the task is to classify the subject's ontology class. There are 14 classes in total. We employ four templates shown below:

$$\mathcal{T}_1(x, y) = x \; y \; x \text{ is a } \texttt{[MASK]}.$$
$$\mathcal{T}_2(x, y) = x \; y \text{ In this sentence,} x \text{is a } \texttt{[MASK]}.$$
$$\mathcal{T}_3(x, y) = x \; y \text{ The type of} x \text{is } \texttt{[MASK]}.$$
$$\mathcal{T}_4(x, y) = x \; y \text{ The category of} x \text{is } \texttt{[MASK]}.$$

Yahoo is a question classification dataset with 10 classes. Each piece of text consists of a question and an answer. We use the templates in AG's News where "news" is replaced with "question" in $\mathcal{T}_1(\cdot)$

$$\mathcal{T}_1(x) = \text{A } \texttt{[MASK]} \text{ question: } x$$
$$\mathcal{T}_2(x) = x \text{ This topic is about } \texttt{[MASK]}.$$
$$\mathcal{T}_3(x) = [\text{ Category : } \texttt{[MASK]} \text{] } x$$
$$\mathcal{T}_4(x) = [\text{ Topic : } \texttt{[MASK]} \text{] } x$$

FewNERD is a large-scale fine-grained entity typing dataset with 66 types and we use the official split of its supervised setting. Following (Ding et al., 2021a), we employ 3 templates as below

$$\mathcal{T}_1(x) = x \; \texttt{[ENT]} \text{ is } \texttt{[MASK]}.$$
$$\mathcal{T}_2(x) = x \; \texttt{[ENT]} \text{ is a } \texttt{[MASK]}.$$
$$\mathcal{T}_3(x) = x \text{ In this sentence, } \texttt{[ENT]} \text{ is a } \texttt{[MASK]}.$$

where [ENT] copies the entity mention in the sentence.