MetricPrompt: Prompting Model as a Relevance Metric for Few-shot Text Classification

Hongyuan Dong
Research Center for Social Computing and Information Retrieval
Harbin Institute of Technology
Harbin, Heilongjiang, China
hydong@ir.hit.edu.cn

Weinan Zhang
Research Center for Social Computing and Information Retrieval
Harbin Institute of Technology
Harbin, Heilongjiang, China
wnzhang@ir.hit.edu.cn

Wanxiang Che*
Research Center for Social Computing and Information Retrieval
Harbin Institute of Technology
Harbin, Heilongjiang, China
car@ir.hit.edu.cn

ABSTRACT

Prompting methods have shown impressive performance in a variety of text mining tasks and applications, especially few-shot ones. Despite the promising prospects, the performance of prompting model largely depends on the design of prompt template and verbalizer. In this work, we propose MetricPrompt, which eases verbalizer design difficulty by reformulating few-shot text classification task into text pair relevance estimation task. MetricPrompt adopts prompting model as the relevance metric, further bridging the gap between Pre-trained Language Model's (PLM) pre-training objective and text classification task, making possible PLM's smooth adaption. Taking a training sample and a query one simultaneously, MetricPrompt captures cross-sample relevance information for accurate relevance estimation. We conduct experiments on three widely used text classification datasets across four few-shot settings. Results show that MetricPrompt outperforms manual verbalizer and other automatic verbalizer design methods across all few-shot settings, achieving new state-of-the-art (SOTA) performance.

CCS CONCEPTS

• **Information systems → Clustering and classification**; Language models; • **Computing methodologies → Natural language processing**.

KEYWORDS

text mining, text classification, few-shot learning, prompt learning

ACM Reference Format:
Hongyuan Dong, Weinan Zhang, and Wanxiang Che. 2023. 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 (KDD '23), August 6–10, 2023, Long Beach, CA, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3580305.3599430

*Corresponding author.

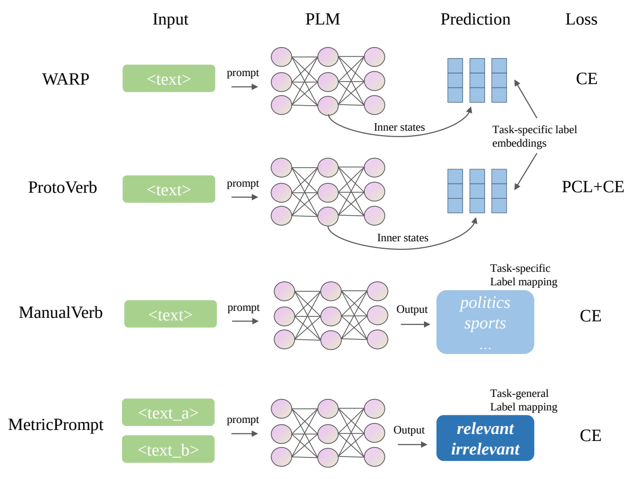

Figure 1: A comparison across verbalizer design methods. CE stands for Cross-Entropy loss and PCL is Prototypical Contrastive Learning loss [17].

1 INTRODUCTION

Since unstructured text data takes up over 80% information in our society, text mining is believed to have significant commercial value [15]. Text classification is regarded as a fundamental and essential task in text mining, and the related techniques are used in various kinds of text mining applications, such as information retrieval [10, 13], sentiment analysis [19, 23], recommendation system [1], knowledge management [36], document summarization [3], etc. Recently proposed pre-trained language models (PLMs) achieve satisfactory text classification performance under data-rich setting [2, 8, 28–30], but these models' Few-Shot Learning (FSL) ability still lags far behind human intelligence [2].

Prompting methods are proposed to better utilize PLM's general knowledge by aligning downstream tasks to its pre-training objective. Prompting method inserts sample text into prompt template to form prompted text. Prompting model takes as input the prompted text, and the result is obtained by projecting the model's output words to corresponding labels. Label mapping is conducted by *verbalizer*, which serves as a critical part of the prompting model and determines its performance.

Although achieving promising results in a wide variety of text mining tasks, prompting methods are very susceptible to sub-optimal

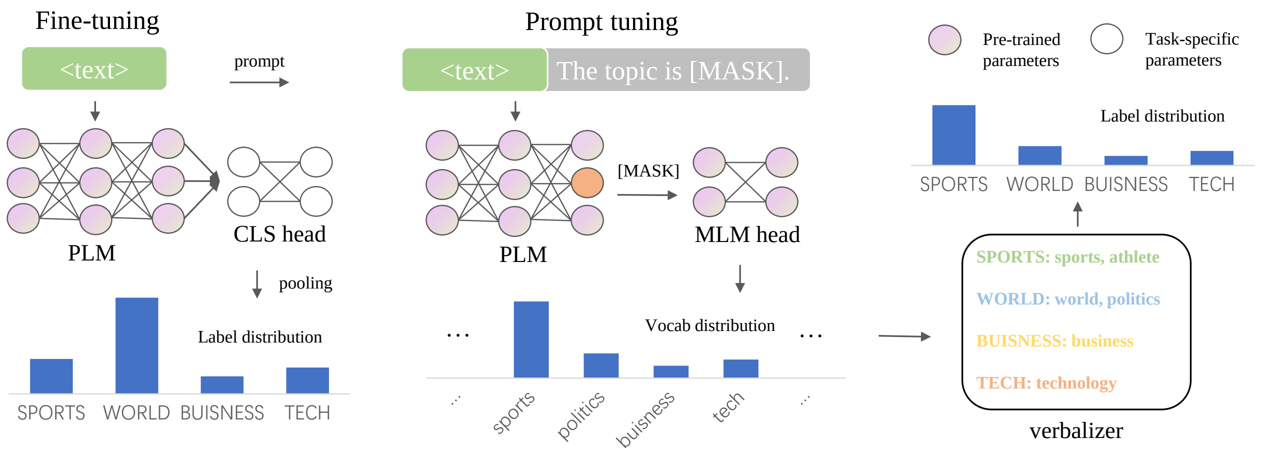

Figure 2: A comparison between vanilla fine-tuning and prompt tuning with MLM for text classification.

verbalizer design [11]. However, designing a proper verbalizer requires deep understanding of both downstream task and PLM's inner mechanism. In practice, this procedure can be extremely time and resource-consuming. To this end, automatic verbalizer design methods are proposed to ease the difficulty. These algorithms can be classified into discrete verbalizer design and soft verbalizer design [20] methods. Discrete verbalizer design methods, such as AVS [33], LM-BFF [11] and AutoPrompt [35], search each label's corresponding answer words in PLM's vocabulary to build the verbalizer. Soft verbalizer design methods like WARP [12] and ProtoVerb [6] search for proper verbalizer parameters in an infinite continuous space and thus achieve better performance.

Despite the promising prospects of soft verbalizer, current methods' performance is still far from satisfactory. The main reason is that these methods' optimization and inference formulations are distinct from PLM's pre-training objective. As illustrated in Fig 1, WARP [12] and ProtoVerb [6] introduce task-specific label embeddings and use PLM's inner representation to make predictions. Although adopting prompting method, both methods force PLM to adapt to a distinct task formulation from its pre-training objective which only operates on output word probabilities. What's worse, these label embeddings have to be trained from scratch in downstream tasks, leading to severe over-fitting problem. As a result, PLM cannot adapt to downstream tasks smoothly.

To tackle the above issues, we propose MetricPrompt, which frees human labor from task-specific verbalizer design by reformulating few-shot text classification task into text pair relevance estimation task. We re-organize training data into text pairs, and adopt prompting model to learn the relevance metric. The learned metric is then used to evaluate each query sample's relevance with training samples, and the classification result is obtained by pooling the estimated relevance scores. As shown in Fig 1, an explicit task-specific verbalizer is no longer required in our method. Following PLM's pre-training objective, MetricPrompt only operates with PLM's output word probabilities, and thus enabling smooth adaption to downstream tasks. To produce accurate relevance estimations, MetricPrompt takes text pairs as input and aid estimation accuracy with cross-sample relevance information. Experiments on three widely used few-shot text classification datasets across four few-shot settings indicate that MetricPrompt achieves the highest few-shot text classification accuracy over previous SOTA verbalizer design baselines.

We summarize the contribution of this paper as below:

(1) We propose a novel prompting method MetricPrompt, which eases task-specific verbalizer design difficulty by reformulating few-shot classification task into relevance estimation problem and learning the relevance metric with prompting model.

(2) We conduct experiments on three widely used few-shot text classification datasets with four few-shot settings, and results show that MetricPrompt outperforms all automatic verbalizer baselines and even manual verbalizer which requires heavy human labor in task-specific verbalizer design.

(3) We provide analysis to demonstrate the extensibility and robustness of MetricPrompt, and explain its performance variance when equipped with different pooling methods.

All code and data will be publicly available at https://github.com/Dousia/MetricPrompt.

2 PRELIMINARIES AND RELATED WORK

2.1 Prompting methods

Traditionally, PLM is adapted to downstream tasks via vanilla fine-tuning. As shown in Fig 2, fine-tuning pools PLM's last layer hidden states to form a sentence representation, and makes predictions with a task-specific classification head. Fine-tuning is effective given sufficient training data, but its performance degrades significantly under few-shot scenario. The newly initialized task-specific head is prone to over-fitting problem, and the gap between downstream task formulation and the pre-training objective hinders PLM's smooth adaption.

Prompting methods are proposed to reformulate downstream tasks to enable PLM's smooth adaption to new tasks. A prompting model's pipeline is illustrated in Fig 2. Denoting $p(\cdot)$ as the prompting function which fills the input text $\mathbf{x}$ into a prompt template, prompting model takes the prompted text and produces output

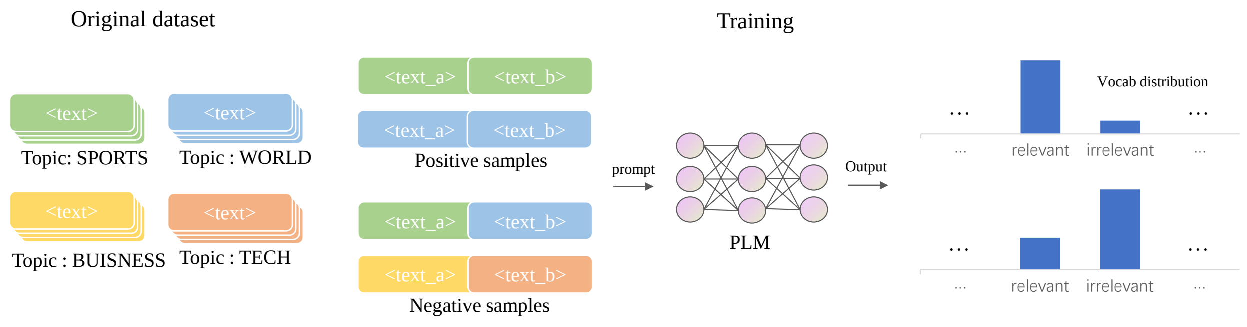

Figure 3: A demonstration of Metricprompt's data construction and training procedure. Original data is paired to form positive and negative samples, and the prompting model is optimized to map these samples to corresponding words.

word probability distribution over the vocabulary:

$$f_{vocab}(p(\mathbf{x});\theta) = P(p(\mathbf{x});\theta), \tag{1}$$

where $P(\cdot)$ is a PLM parameterized by θ. The output word probability is then mapped to final predicted probability distribution over classes with a verbalizer $v(\cdot)$:

$$f_{cls}(p(\mathbf{x});\theta) = v(f_{vocab}(p(\mathbf{x});\theta)). \tag{2}$$

First proposed by Radford et al., prompting methods have been used in a variety of text mining tasks, such as text classification [21, 27, 33, 34], text generation [4, 18, 32], named entity recognition [7, 14], knowledge probing [25, 37], etc. Prompting model alleviates the over-fitting problem under few-shot scenario significantly [2, 34, 38]. However, prompting model's performance relies largely on the selection of prompt template and verbalizer [20]. Although a number of works focus on the design of prompt [11, 18, 21, 35], less are proposed to ease verbalizer design difficulty. Verbalizer maps prompting model's output words to classification results directly, and therefore influences prompting model's performance significantly [11]. In this work, we seek to free human labor from verbalizer design with less performance loss.

2.2 Verbalizer Design Methods

Current verbalizer design methods can be classified into manual verbalizer design, discrete verbalizer design and soft verbalizer design. A carefully hand-crafted verbalizer can achieve highly competitive performance in various text mining tasks [33, 34]. However, designing such a well-performing verbalizer relies heavily on human's accurate understanding of the target task and requires heavy trial-and-error work. Discrete verbalizer design methods frees human labor from tedious and time-consuming verbalizer design process. LM-BFF [11] generates proper answer words with large PLMs such as T5 [30]. AutoPrompt [35] and AVS [33] initialize a verbalizer and optimize it to meet a predefined criteria iteratively. PETAL [31] searches for answer words that maximize the likelihood of the training data. These methods reduce human labor required by verbalizer design significantly, but their performance lags far behind carefully hand-crafted ones [6]. To achieve better performance, soft verbalizer design methods render answer words from a fixed vocabulary as differentiable label embeddings represented in an infinite continuous space. Expanding the possibilities of verbalizer design space, these methods achieve better results than discrete verbalizer design algorithms [6, 12, 40].

In this work, we ease task-specific verbalizer design difficulty by reformulating text classification task into text pair relevance estimation task. In this way, no additional task-specific parameter is introduced, and the over-fitting problem is alleviated. Adopting prompting model as the relevance metric, PLM can adapt to new tasks more smoothly. A recent work also views text classification task as natural language inference problem [26], but it focuses on zero-shot scenario and hand-craft each label's description. In comparison, our method tackle few-shot text classification tasks instead of zero-shot ones, and does not require human labor in task-specific verbalizer design.

3 METHODS

In this section, we introduce the data construction, optimization and inference procedure of MetricPrompt in detail.

3.1 Data construction

Given a few-shot text classification dataset $\mathcal{D}$, we denote training data with $\mathcal{D}_t$ and query samples with $\mathcal{D}_q$. A sample is formulated as $d = (\mathbf{x}_d, \mathbf{y}_d)$, where $\mathbf{x}_d$ stands for sample text and $\mathbf{y}_d$ represents its label. Since MetricPrompt takes as input a pair of sample text, we construct training data as follows:

$$\mathcal{D}_t^M = \bigcup_{(d_i,d_j)\in\mathcal{D}_t\times\mathcal{D}_t} \{(p(\mathbf{x}_{d_i},\mathbf{x}_{d_j}),\mathbf{y}_{ij})\}, \tag{3}$$

where $p(\cdot,\cdot)$ is MetricPromt's prompting function. It takes two pieces of sample text and produces a filled prompt template with the given text. We use "$\mathbf{x}_{d_i}$ `[SEP]` A news of `[MASK]` topic: $\mathbf{x}_{d_j}$" as the prompt template. $\mathbf{y}_{ij} = \mathbb{I}(\mathbf{y}_{d_i} = \mathbf{y}_{d_j})$ indicates whether the pair of text are of the same class.

Similarly, we build query data for MetricPrompt as follows:

$$\mathcal{D}_q^M = \bigcup_{(d_i,d_j)\in\mathcal{D}_q\times\mathcal{D}_t} \{(p(\mathbf{x}_{d_i},\mathbf{x}_{d_j}),\mathbf{y}_{ij})\}. \tag{4}$$

The overall data construction process is illustrated in Fig 3 and Fig 4. We reorganize single samples of a given few-shot text classification task into paired samples, and the original multi-class classification problem is rendered as a relevance estimation task.

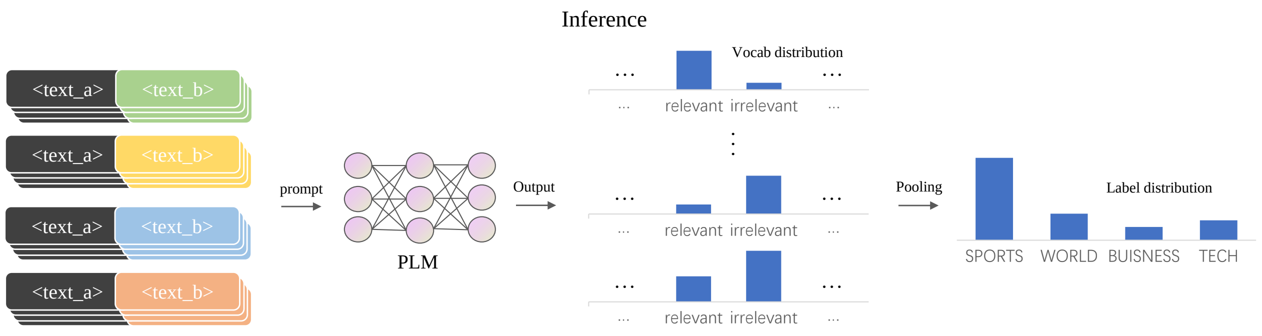

Figure 4: The inference procedure of MetricPrompt. A query sample is paired with each training sample, and the relevance scores are pooled to produce final classification probabilities.

No label's semantic representation is involved in this data construction procedure, and therefore MetricPrompt requires no human labor in task-specific verbalizer design.

3.2 Optimization

MetricPrompt differs from previous soft verbalizer design methods for we do not introduce task specific label embeddings, but instead reformulate few-shot text classification task as a text pair relevance estimation task to let loose the need of task-specific verbalizer. By reformulating the task, MetricPrompt's optimization coincides with PLM's pre-training objectives.

Let $P(\cdot;\theta)$ be an MLM model parameterized by θ and $f_{vocab}(\cdot;\theta)$ be its output word probability over the vocabulary at [MASK] position. We define the optimization objective of MetricPrompt as follows:

$$\begin{aligned}\hat{\theta} &= \arg\min_{\theta} \sum_{d^M \in \mathcal{D}_t^M} \mathcal{L}(f_{vocab}(\mathbf{x}_{d^M};\theta), \mathbf{y}_{d^M}) \\ &= \arg\min_{\theta} \sum_{d^M \in \mathcal{D}_t^M} \mathcal{L}_{CE}(v(f_{vocab}(\mathbf{x}_{d^M};\theta)), \phi(\mathbf{y}_{d^M})), \\ &= \arg\min_{\theta} \sum_{d^M \in \mathcal{D}_t^M} \mathcal{L}_{CE}(f_{cls}(\mathbf{x}_{d^M};\theta)), \phi(\mathbf{y}_{d^M})),\end{aligned} \tag{5}$$

where $\phi(\cdot)$ stands for a probability distribution over label categories. The corresponding position of the input sample's label is set as 1 while others are set to be 0. $v(\cdot)$ represents a predefined task-general meta verbalizer, which projects output word probability $f_{vocab}(\cdot;\theta)$ to a binomial distribution $f_{cls}(\cdot;\theta)$. We task this meta verbalizer to aggregate logits at {relevant, similar, consistent} to be the predicted logit of label 1, while logits at {irrelevant, inconsistent, different} are aggregated to be the logit of label 0.[1] $\mathcal{L}$ is the loss function of MetricPrompt, which is defined as the cross-entropy loss between the probability distributions produced by the meta verbalizer $v(\cdot)$ and the ground truth distribution.

MetricPrompt formulates few-shot text classification task's optimization objective to be a generalized MLM task, which is the minimization of cross-entropy loss between predicted word probability at [MASK] position and the ground truth one-hot distribution. The optimized prompting model can be used as a metric to estimate the relevance between two sample text.

[1]Note that this meta verbalizer can be used in all few-shot text classification tasks, so MetricPrompt requires no extra human labor in task-specific verbalizer design.

3.3 Inference

After optimization, the prompting model serves as a relevance metric during inference. As illustrated in Fig 4, we take an original query sample d_q colored in black and pair it with all training samples colored differently to form inference samples. Given an original training sample d_i, MetricPrompt computes its relevance score s_{d_i} with d_q as follows:

$$s_{d_i} = \Delta(f_{cls}(p(\mathbf{x}_{d_q}, \mathbf{x}_{d_i}); \hat{\theta})), \tag{6}$$

where $\Delta(\cdot)$ computes the difference between the binomial distribution's probability at 1 and 0. We denote l as a label of the few-shot text classification task. Let $\mathcal{D}_l = \{d_i | d_i \in \mathcal{D}_t, \mathbf{y}_{d_i} = l_i\}$, MetricPrompt calculates l's classification score s_l by pooling its corresponding samples' relevance with d_q:

$$s_l = \sum_{d_i \in \mathcal{D}_l} s_{d_i} / |\mathcal{D}_l|. \tag{7}$$

Finally, MetricPrompt selects the label $\hat{l}$ with the highest the relevance score as the classification result:

$$\hat{l} = \arg\max_{l} s_l. \tag{8}$$

We present MetricPrompt with sum pooling in Equation 7. This pooling function can also be replaced by max pooling and K-Nearest-Neighborhood (KNN) [5] pooling. Max pooling classifies query sample d_q into its most relevant training sample's class. MetricPrompt works with max pooling by replacing Equation 7 with:

$$s_l = \max_{d_i \in \mathcal{D}_l} s_{d_i}. \tag{9}$$

For KNN pooling, we denote $\mathcal{D}_{topk}$ as k training samples most relevant to d_q in $\mathcal{D}_t$ and reformulate Equation 7 as:

$$s_l = |\{d_i | d_i \in \mathcal{D}_{topk}, \mathbf{y}_{d_i} = l_i\}|. \tag{10}$$

We set k as half the size of training set. When several labels appear in $\mathcal{D}_{topk}$ for the same number of times, we select the sample obtaining the highest relevance score from these labels' corresponding samples, and classify d_q to its class.

MetricPrompt renders prompting model as a relevance metric, and classify query samples according to its relevance with training samples of the few-shot task. Prompting model takes two pieces of sample text at once. Thus, MetricPrompt is able to use cross-sample information to estimate relevance and make predictions accurately.

3.4 More Efficient Inference

The cost of the inference procedure is relatively high because of the pairing strategy. To further improve the efficiency of MetricPrompt, we propose to use pivot samples to reduce the time complexity of the inference stage of MetricPrompt.

We propose to use the optimized prompting model to calculate the representativeness of each training sample. For a training sample d_i labeled with l, we use r_{d_i} to denote its representativeness, which is calculated as follows:

$$r_{d_i} = \frac{\sum_{d_j \in \mathcal{D}_l} s_{d_j}}{|\{d_j | d_j \in \mathcal{D}_l\}|} - \frac{\sum_{d_k \in \mathcal{D}_t - \mathcal{D}_l} s_{d_k}}{|\{d_k | d_k \in \mathcal{D}_t - \mathcal{D}_l\}|}. \tag{11}$$

s_{d_j} represents the relevance score between samples d_j and d_i. Based on this representativeness metric, we select the top p samples with the highest representativeness scores from the training samples corresponding to each label. These samples are marked as pivot samples. During inference, we only pair each test sample with each label's pivot samples and compute their relevance scores to make classification prediction.

Pivot samples reduce the time complexity of MetricPrompt's inference process significantly. Assume a few-shot text classification task with n labels and k samples per label. Without introducing pivot samples, each test sample needs to be paired with $n * k$ training samples and compute relevance scores, resulting into a time complexity of $O(n * k)$. In contrast, prompting methods with human-designed or automatically designed verbalizer calculate the dot product similarity between the feature representations of test samples extracted by the pre-trained model and the feature representations of each label. Since the total number of labels is n, the time complexity of these methods is only $O(n)$. By introducing pivot samples to improve the inference process, MetricPrompt only needs to estimate the relevance between each test sample and the pivot samples of each label, reducing the time complexity to $O(p * n)$. Because p is a pre-defined constant, the time complexity of MetricPrompt's inference process accelerated with pivot samples is $O(n)$, which is consistent with other commonly used prompting methods. We set the number of pivot samples per label p to 2 in the experiments.

4 EXPERIMENTS

In this section, we first introduce the datasets used in our experiments. Then we describe our experiment settings and implementation details. Finally, we present experiment results and analyze MetricPrompt's performance.

Dataset	# Class	# Test	Avg len
AG's News	4	7,600	52
DBPedia	14	70,000	68
Yahoo	10	60,000	130

Table 1: Statistics of the three text classification datasets used in our experiments.

Dataset	2-shot	4-shot	8-shot	16-shot
AG's News	120	60	30	15
DBPedia	32	16	8	4
Yahoo	36	18	9	5

Table 2: The number of training epochs for given datasets and few-shot settings.

4.1 Datasets

We conduct experiments with three widely used text classification tasks, which contain numerous classes and hence require extensive resources to build a proper manual verbalizer. We adopt AG's News, Yahoo Answers topics [41] and DBPedia [16] as our text classification tasks. The statistics of the datasets are given in Table 1.

Since the average text length of the three datasets is short, we truncate all sample text to 120 tokens for better efficiency with little semantic meaning loss.

4.2 Few-shot experiment settings

We conduct experiments under 2, 4, 8 and 16-shot settings, where corresponding number of training samples are sampled from each dataset's training set randomly. Models' performance are observed to fluctuate largely when the training set is sampled with different random seeds. Hence, we sample 10 training sets for each dataset and each few-shot setting to alleviate the influence of randomness in training set selection. All experiment results are reported as the average value of the model's performance on the 10 training sets.

4.3 Implementation Details

We implement MetricPrompt with PyTorch [24] and Huggingface [39] framework, and baselines are implemented with OpenPrompt toolkit [9]. We introduce Kernl to accelerate the inference procedure.[2]

We adopt BERT-base-uncased [8] as the backbone model for both MetricPrompt and all baseline models for fair comparison. Model parameters are optimized with AdamW optimizer [22], and the learning rate is set as 1e-5. We set total training steps proportionally to the size of training set, and the number of training epochs is adjusted accordingly. The size of training set varies across datasets and shot numbers, and the specific number of training epochs is given in Table 2.

4.4 Baselines

We select several representative verbalizer design methods for comparison. Among the listed baselines, only manual verbalizer requires human effort in task-specific verbalizer design.

[2]https://github.com/ELS-RD/kernl

Method	Dataset	Prompt template	Task-specific verbalizer
ManualVerb	AG's News	A `[MASK]` news: `<text>`	sports, politics, business, technology
	DBPedia	`<text>` In this sentence, the topic is `[MASK]`	company, school, artist, athlete, politics, transportation, building, river, village, animal, plant, album, film, book
	Yahoo	A `[MASK]` question: `<text>`	society, science, health, education, computers, sports, business, entertainment, relationships, politics
AVS	AG's News	A `[MASK]` news: `<text>`	Automatically searched label words
	DBPedia	`<text>` In this sentence, the topic is `[MASK]`	Automatically searched label words
	Yahoo	A `[MASK]` question: `<text>`	Automatically searched label words
SoftVerb	AG's News DBPedia Yahoo	`<text>` In this sentence, the topic is `[MASK]`	Soft label embeddings
ProtoVerb	AG's News	A `[MASK]` news: `<text>`	Soft label embeddings
	DBPedia	`<text>` In this sentence, the topic is `[MASK]`	Soft label embeddings
	Yahoo	A `[MASK]` question: `<text>`	Soft label embeddings
MetricPrompt	AG's News DBPedia Yahoo	`<text_a>` A news of `[MASK]` topic: `<text_b>`	–

Table 3: Prompt templates and task-specific verbalizers used by MetricPrompt and other baselines. "-" means no task-specific verbalizer is required.

Manual Verbalizer (ManualVerb) uses hand-crafted verbalizer to map PLM's output word to classification labels.

Automatic Verbalize Search (AVS) [33] is a search-based verbalizer design method. It initializes the verbalizer with random words, and improves answer words iteratively.

Soft Verbalizer (SoftVerb) [12] represents each label with a trainable embedding. In the original work WARP, the prompt template is also represented as trainable embeddings. In this work, however, we follow Cui et al. to use manual template for fair comparison.

Prototypical Verbalizer (ProtoVerb) [6] also represents classification labels as soft embeddings and samples as features encoded by PLM. ProtoVerb adopts prototypical contrastive learning loss [17] instead of vanilla cross-entropy loss to optimize model parameters.

We hand-craft a task-general prompt template and verbalizer for MetricPrompt. For other baselines, we make minor modifications to the default template and verbalizer used in OpenPrompt to unify the input format. An overview of prompt template and verbalizer used for our experiments is given in Table 3.

4.5 Main Results

We conduct experiments on three text classification datasets with different text styles under four few-shot settings. Experiment results for 2 and 4-shot settings are listed in Table 4, while 8 and 16-shot's experiment results are shown in Table 5. MetricPrompt outperforms previous SOTA automatic verbalizer design method ProtoVerb by a large margin, improving 2-shot accuracy by 5.88 (11.91% ↑), 4-shot accuracy by 11.92 (19.59% ↑), 8-shot accuracy by 6.80 (9.45% ↑) and 16-shot accuracy by 1.56 (1.96% ↑). MetricPrompt's performance even surpasses that of ManualVerb under all few-shot settings without any human labor involved in task-specific verbalizer design.

Meanwhile, we have following observations:

(1) MetricPrompt is the only prompting method outperforming ManualVerb without any human labor involved in task-specific verbalizer design. MetricPrompt benefits from paired input text, which enables the model to use cross-sample information to make up the lack of extra human knowledge and trial-and-error work.

(2) MetricPrompt achieves the highest score over automatic verbalizer design methods. Compared with AVS, MetricPrompt does not restrict each class's representation to several sub-optimal words from the vocabulary, but instead represent it with corresponding training samples, leading to more accurate semantic representation. For the comparison with SoftVerb and ProtoVerb, we attribute MetricPrompt's leading performance to the smaller gap between its task formulation and PLM's pre-training objective. Unlike SoftVerb and ProtoVerb, MetricPrompt does not operate on PLM's inner representations, but functions with only the output word probability distribution at `[MASK]` position, enabling PLM to adapt to few-shot text classification task more smoothly. Moreover, MetricPrompt does not introduce task-specific parameters to be trained from scratch, avoiding over-fitting problem under few-shot scenarios.

(3) MetricPrompt achieves comparable results with mean pooling and max pooling, but a performance drop is witnessed with KNN pooling. We ascribe the performance gap to the incompatibility between KNN pooling and the non-uniform distribution of MetricPrompt's predicted relevance scores. We further discuss this phenomenon in Section 5.3.

5 ANALYSIS

In this section, we further evaluate MetricPrompt from different aspects to illustrate its effectiveness.

5.1 Extensibility with Out-Of-Domain Data

Since it is impractical to always obtain sufficient training data for downstream few-shot text classification tasks, we further evaluate MetricPrompt's extensibility with Out-Of-Domain (OOD) data. We use 16-shot training sets of other datasets to aid each task's 1, 2 and 4-shot classification, where models' performance remains to

Method	AG's News		DBPedia		Yahoo		Average	
	2-shot	4-shot	2-shot	4-shot	2-shot	4-shot	2-shot	4-shot
ManualVerb	*45.87*	*76.22*	*69.81*	*84.15*	*34.82*	*55.56*	*50.17*	*71.98*
AVS [2021a]	44.93	57.49	32.22	53.55	21.88	28.44	33.01	46.49
SoftVerb [2021]	48.86	61.15	53.98	76.19	22.63	33.43	41.82	56.92
ProtoVerb [2022]	58.38	65.04	60.89	74.49	28.80	43.01	49.36	60.85
MetricPrompt$_{\text{KNN}}$	62.69	73.17	66.27	86.06	26.02	50.90	51.66	70.04
MetricPrompt$_{\text{MAX}}$	65.64	76.12	**71.28**	**88.44**	**28.85**	52.99	**55.26**	72.52
MetricPrompt$_{\text{MEAN}}$	**65.77**	**76.33**	71.20	**88.44**	28.76	**53.54**	55.24	**72.77**
MetricPrompt$_{\text{PIVOT}}$	65.76	74.53	71.20	86.12	28.76	51.32	55.24	70.66

Table 4: Experiment results in terms of accuracy under 2-shot and 4-shot settings. Italic score means human labor is involved in task-specific verbalizer design, and bold number indicates the best result among methods requiring no human labor.

Method	AG's News		DBPedia		Yahoo		Average	
	8-shot	16-shot	8-shot	16-shot	8-shot	16-shot	8-shot	16-shot
ManualVerb	*78.94*	*83.66*	*94.24*	*97.27*	*58.30*	*62.42*	*77.16*	*81.12*
AVS [2021a]	71.37	77.81	75.91	85.36	46.53	57.68	64.60	73.62
SoftVerb [2021]	73.28	80.61	90.34	96.93	45.01	59.09	69.54	78.88
ProtoVerb [2022]	75.57	80.31	87.45	**97.16**	52.87	61.57	71.96	79.68
MetricPrompt$_{\text{KNN}}$	80.64	84.43	94.25	96.55	58.09	62.05	77.66	81.01
MetricPrompt$_{\text{MAX}}$	81.03	84.27	94.28	96.55	59.68	62.66	78.33	81.16
MetricPrompt$_{\text{MEAN}}$	**82.04**	**84.69**	**94.57**	96.59	**59.68**	**62.45**	**78.76**	**81.24**
MetricPrompt$_{\text{PIVOT}}$	81.19	84.15	94.13	96.22	58.63	61.78	77.98	80.72

Table 5: Experiment results in terms of accuracy under 8-shot and 16-shot settings. Italic score means human labor is involved in task-specific verbalizer design, and bold number indicates the best result among methods requiring no human labor.

Method	AG's News			Method	DBPedia			Method	Yahoo		
	1-shot	2-shot	4-shot		1-shot	2-shot	4-shot		1-shot	2-shot	4-shot
ProtoVerb	46.79	58.38	65.04	ProtoVerb	45.86	60.89	74.49	ProtoVerb	21.60	28.80	43.01
+DBPedia	57.43	65.72	71.27	+AG's News	49.00	63.29	75.56	+AG's News	28.93	39.15	49.96
+Yahoo	63.63	71.84	75.34	+Yahoo	$\underline{54.33}$	66.78	77.56	+DBPedia	30.13	39.57	51.39
MetricPrompt	39.16	65.77	76.33	MetricPrompt	32.31	71.20	88.44	MetricPrompt	18.80	28.76	53.54
+DBPedia	$\underline{66.95}$	$\underline{71.40}$	$\underline{77.34}$	+AG's News	$\underline{53.23}$	$\underline{74.21}$	$\underline{88.34}$	+AG's News	$\underline{32.10}$	**$\underline{44.27}$**	$\underline{52.29}$
+Yahoo	**$\underline{71.00}$**	**$\underline{73.99}$**	**$\underline{79.57}$**	+Yahoo	53.03	**$\underline{76.41}$**	**$\underline{89.47}$**	+DBPedia	**$\underline{32.77}$**	$\underline{43.63}$	**$\underline{53.78}$**

Table 6: The performance of MetricPrompt and ProtoVerb with additional OOD training data. Underlined number indicates the best result under the same few-shot and OOD setting. Bold number represents the best result on the few-shot task.

be improved. We adopt mean pooling and simply mix up training samples of the original dataset and OOD ones to train the model. For ProtoVerb baseline, we first train the model on OOD training data, and then re-initialize prototype parameters for the training on the original dataset.

As shown in Table 6, MetricPrompt achieves much higher accuracy when boosted by OOD training data. Compared with previous SOTA baseline ProtoVerb, MetricPrompt obtains better prediction accuracy in 17 out of 18 few-shot and OOD data settings (underlined numbers in the table), showing remarkable extensibility with OOD samples. Although OOD data improves the model's performance under most settings, AG's News training set fail to aid DBPedia 4-shot and Yahoo 4-shot performance. We ascribe this failure to the abundance of training samples under these settings. DBPedia and Yahoo contains 4096 and 1600 training samples under 4-shot setting, which is sufficient for the model to adapt to the relevance estimation task. OOD data serves more as noises and therefore harms the model's performance.

It is worth noticing that MetricPrompt's performance improvement under 1-shot setting is abnormally high. MetricPrompt underperforms ProtoVerb on the three datasets without OOD data, because MetricPrompt only takes two identical pieces of text as positive sample under 1-shot setting, leading to severe over-fitting problem. The model is optimized to only produce high relevance score when given two identical pieces of text. Diversified OOD data

Method	1 wrong		2 wrong		4 wrong		Average	
	8-shot	16-shot	8-shot	16-shot	8-shot	16-shot	8-shot	16-shot
ProtoVerb	2.79	0.83	4.95	1.85	11.31	3.71	6.35	2.13
MetricPrompt$_{\text{KNN}}$	5.74	2.61	5.66	3.08	12.38	3.38	7.93	3.02
MetricPrompt$_{\text{MAX}}$	**1.59**	0.59	3.12	**0.89**	**7.01**	**1.21**	3.91	**0.90**
MetricPrompt$_{\text{MEAN}}$	1.81	**0.55**	**2.72**	1.04	7.06	1.52	**3.86**	1.04

Table 7: Models' performance drop under AG's News 8 and 16-shot settings with 1, 2 and 4 noisy samples. Bold number indicates the least drop among all methods.

alleviate the over-fitting problem effectively, and therefore improve MetricPrompt's 1-shot performance by a large fraction.

5.2 Robustness against Noisy Samples

Noisy samples harm few-shot text classification model's performance severely due to the lack of supervision signal. In this section, we evaluate MetricPrompt's robustness against noisy samples on AG's News dataset. Following Cui et al., we conduct experiments under 8 and 16-shot settings for training stability. We replace 1, 2 and 4 training samples' labels randomly to introduce noises, and evaluate MetricPrompt's performance when equipped with mean, max and KNN pooling. We compare our method with previous SOTA baseline ProtoVerb, which shows the best robustness against noisy samples among automatic verbalizer design methods [6]. The performance drop caused by noisy samples is displayed in Table 7.

Compared with ProtoVerb, MetricPrompt suffers from less performance drop and achieves higher classification accuracy with mean and max pooling, while KNN pooling leads to worse performance. We attribute the large performance drop of KNN pooling to its susceptibility to the variance of each class's training sample number, which is introduced by noisy samples. We provide a detailed analysis in Section 5.3.

5.3 Comparison across Pooling Methods

In this part, we analyze different pooling methods with clean and noised training data to explain their performance gap.

Firstly, we focus on clean data scenario without noisy samples. We choose AG's News 2-shot setting and compute the average relevance score of each query sample's most relevant training sample, second most relevant training sample, etc. As shown in Fig 5, the distribution of relevance scores is highly non-uniform. The highest relevance score is much larger than others, so the max relevance score serves as a decisive factor for MetricPrompt with mean pooling. Therefore, mean pooling shows similar behavior with max pooling. KNN pooling, however, adopts voting strategy which ignores score value information, leading to deteriorated performance.

We then analyze MetricPrompt's performance when noisy samples are introduced. We first categorize classes of AG's News dataset 8-shot training sets according to the number of their corresponding training samples. Then we collect the statistics of the average predicted query sample number for each type of class and show them in Figure 6. KNN pooling shows significantly stronger preference to classes with more training samples than mean pooling and max pooling do. Since the distribution of MetricPrompt's relevance score except the top ones is relatively even, samples from classes

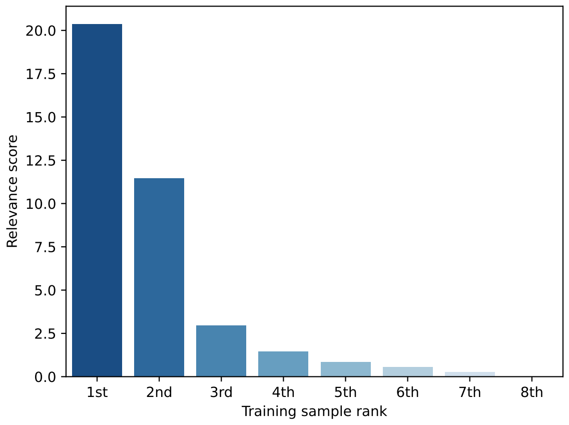

Figure 5: Average relevance scores between each query sample and all training samples under AG's News 2-shot setting. The scores are sorted and shifted to non-negative region.

with large training set are more likely to become the majority of KNN pooling's top k relevant samples. Without considering relevance score value, a large drop in KNN pooling's performance is witnessed. On the contrary, mean pooling and max pooling take each training sample's relevance score value into consideration, so the influence of training sample number is mitigated. As a result, they suffer from smaller performance drops than KNN pooling do.

5.4 Influence of Pivot Sample Number

In this part, we investigate the influence of pivot sample numbers to the performance of MetricPrompt. We conduct experiments on the three datasets across four few-shot settings with pivot sample number p set as 1, 2 and 4, the performance of MetricPrompt under different few-shot settings are displayed Table 8 and Table 9.

As shown in the tables, the performance of MetricPrompt correlates with the number of pivot samples positively. As the number of pivot samples increase, MetricPrompt captures each label's semantic meaning more accurately and therefore achieves better performance. It is worth noticing that even if only one pivot sample is selected for each class, MetricPrompt still outperforms ProtoVerb under the four few-shot settings. The selection of pivot sample number p serves as a trade-off between classification accuracy and efficiency. In real world applications, MetricPrompt can adapt to varying scenarios with different requirements by adjusting the number of pivot samples.

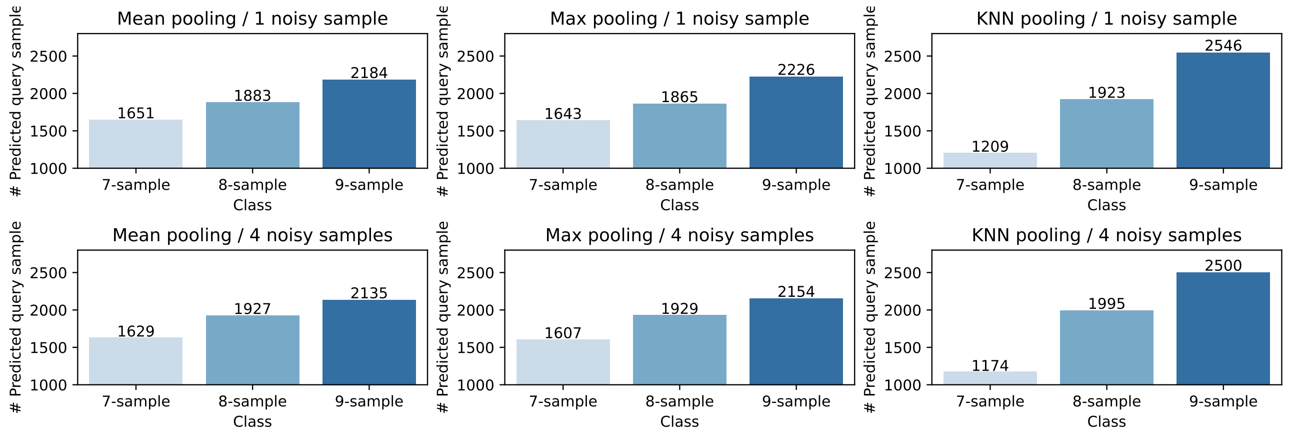

Figure 6: Average query sample number classified to classes with 7, 8 and 9 training samples under AG's News 8-shot setting. "# Predicted query sample" indicates the average number of query samples predicted to the class. KNN pooling shows stronger preference to 9-sample classes than classes with fewer training samples.

Method	AG's News		DBPedia		Yahoo		Average	
	2-shot	4-shot	2-shot	4-shot	2-shot	4-shot	2-shot	4-shot
ProtoVerb	58.38	65.04	60.89	74.49	28.80	43.01	49.36	60.85
MetricPrompt$_{1\text{PIVOT}}$	61.77	71.41	65.01	84.07	25.92	48.42	50.90	67.97
MetricPrompt$_{2\text{PIVOT}}$	**65.76**	74.53	**71.20**	86.12	**28.76**	51.32	**55.24**	70.66
MetricPrompt$_{4\text{PIVOT}}$	**65.76**	**76.33**	**71.20**	**88.44**	**28.76**	**53.54**	**55.24**	**72.77**

Table 8: Experiment results with pivot samples under 2-shot and 4-shot settings. Bold number indicates the best result.

Method	AG's News		DBPedia		Yahoo		Average	
	8-shot	16-shot	8-shot	16-shot	8-shot	16-shot	8-shot	16-shot
ProtoVerb	75.57	80.31	87.45	**97.16**	52.87	61.57	71.96	79.68
MetricPrompt$_{1\text{PIVOT}}$	79.23	84.17	93.38	96.04	57.20	61.15	76.60	80.45
MetricPrompt$_{2\text{PIVOT}}$	**81.19**	84.15	94.13	96.22	58.63	61.78	77.98	80.72
MetricPrompt$_{4\text{PIVOT}}$	81.13	**84.62**	**94.42**	96.49	**59.21**	**61.93**	**78.25**	**81.01**

Table 9: Experiment results with pivot samples under 8-shot and 16-shot settings. Bold number indicates the best result.

6 CONCLUSIONS AND FUTURE WORK

In this work, we propose MetricPrompt, which frees human labor from task-specific verbalizer design by reformulating few-shot text classification task into a text pair relevance estimation problem. MetricPrompt prompts query-training text pair to fulfill relevance estimation, which coincides with PLM's pre-training objective and thus enables smooth adaption to downstream tasks. Taking a pair of sample text simultaneously, MetricPrompt introduces cross-sample information for better accuracy. Experiments on three widely used text classification datasets under four few-shot settings indicate MetricPrompt's leading performance over previous SOTA baselines. Our analysis further demonstrates MetircPrompt's promising extensibility and robustness, and explains its performance variance with different pooling methods and pivot sample numbers.

Although MetricPrompt achieves satisfactory few-shot text classification performance in our experiments, its performance with large language models remains to be explored. Current large language models achieve impressive performance in few-shot text classification tasks with prompting methods, but they still suffer from prompting methods' susceptibility to the design of verbalizers. When given classes which are difficult to be described with several words, these models' performance deteriorates significantly. The proposed MetricPrompt is not bounded with specific backbone model, and can be easily generalized to large language models. We look forward to using MetricPrompt to ease human effort from verbalizer design and further improve the few-shot text classification accuracy of large language models.

ACKNOWLEDGEMENTS

This work was supported by the National Key R&D Program of China via grant 2020AAA0106501 and the National Natural Science Foundation of China (NSFC) via grant 62236004 and 61976072.

REFERENCES

[1] Charu C Aggarwal and Charu C Aggarwal. 2016. Content-based recommender systems. *Recommender systems: The textbook* (2016), 139–166.

[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. 2020. Language Models are Few-Shot Learners. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin (Eds.). https://proceedings.neurips.cc/paper/2020/hash/1457c0d6bfcb4967418bfb8ac142f64a-Abstract.html

[3] Ziqiang Cao, Wenjie Li, Sujian Li, and Furu Wei. 2017. Improving Multi-Document Summarization via Text Classification. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA*, Satinder P. Singh and Shaul Markovitch (Eds.). AAAI Press, 3053–3059. http://aaai.org/ocs/index.php/AAAI/AAAI17/paper/view/14525

[4] Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. 2021. Evaluating large language models trained on code. *ArXiv preprint* abs/2107.03374 (2021). https://arxiv.org/abs/2107.03374

[5] Thomas Cover and Peter Hart. 1967. Nearest neighbor pattern classification. *IEEE transactions on information theory* 13, 1 (1967), 21–27.

[6] Ganqu Cui, Shengding Hu, Ning Ding, Longtao Huang, and Zhiyuan Liu. 2022. 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)*. Association for Computational Linguistics, Dublin, Ireland, 7014–7024. https://doi.org/10.18653/v1/2022.acl-long.483

[7] Leyang Cui, Yu Wu, Jian Liu, Sen Yang, and Yue Zhang. 2021. Template-Based Named Entity Recognition Using BART. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*. Association for Computational Linguistics, Online, 1835–1845. https://doi.org/10.18653/v1/2021.findings-acl.161

[8] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Association for Computational Linguistics, Minneapolis, Minnesota, 4171–4186. https://doi.org/10.18653/v1/N19-1423

[9] Ning Ding, Shengding Hu, Weilin Zhao, Yulin Chen, Zhiyuan Liu, Haitao Zheng, and Maosong Sun. 2022. OpenPrompt: An Open-source Framework for Prompt-learning. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*. Association for Computational Linguistics, Dublin, Ireland, 105–113. https://doi.org/10.18653/v1/2022.acl-demo.10

[10] Sanjay K Dwivedi and Chandrakala Arya. 2016. Automatic text classification in information retrieval: A survey. In *Proceedings of the second international conference on information and communication technology for competitive strategies*. 1–6.

[11] Tianyu Gao, Adam Fisch, and Danqi Chen. 2021. Making Pre-trained Language Models Better Few-shot Learners. 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)*. Association for Computational Linguistics, Online, 3816–3830. https://doi.org/10.18653/v1/2021.acl-long.295

[12] Karen Hambardzumyan, Hrant Khachatrian, and Jonathan May. 2021. 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)*. Association for Computational Linguistics, Online, 4921–4933. https://doi.org/10.18653/v1/2021.acl-long.381

[13] Doris Hoogeveen, Li Wang, Timothy Baldwin, Karin M Verspoor, et al. 2018. Web forum retrieval and text analytics: A survey. *Foundations and Trends® in Information Retrieval* 12, 1 (2018), 1–163.

[14] Yutai Hou, Cheng Chen, Xianzhen Luo, Bohan Li, and Wanxiang Che. 2022. Inverse is Better! Fast and Accurate Prompt for Few-shot Slot Tagging. In *Findings of the Association for Computational Linguistics: ACL 2022*. Association for Computational Linguistics, Dublin, Ireland, 637–647. https://doi.org/10.18653/v1/2022.findings-acl.53

[15] Vandana Korde and C Namrata Mahender. 2012. TEXT CLASSIFICATION AND CLASSIFIERS: A SURVEY. *International Journal of Artificial Intelligence & Applications* 3, 2 (2012), 85.

[16] 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* (2015).

[17] Junnan Li, Pan Zhou, Caiming Xiong, and Steven C. H. Hoi. 2021. Prototypical Contrastive Learning of Unsupervised Representations. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net. https://openreview.net/forum?id=KmykpuSrjcq

[18] Xiang Lisa Li and Percy Liang. 2021. Prefix-Tuning: Optimizing Continuous Prompts for Generation. 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)*. Association for Computational Linguistics, Online, 4582–4597. https://doi.org/10.18653/v1/2021.acl-long.353

[19] Bing Liu and Lei Zhang. 2012. A survey of opinion mining and sentiment analysis. In *Mining text data*. Springer, 415–463.

[20] Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. 2021. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *ArXiv preprint* abs/2107.13586 (2021). https://arxiv.org/abs/2107.13586

[21] Xiao Liu, Yanan Zheng, Zhengxiao Du, Ming Ding, Yujie Qian, Zhilin Yang, and Jie Tang. 2021. GPT Understands, Too. *ArXiv preprint* abs/2103.10385 (2021). https://arxiv.org/abs/2103.10385

[22] Ilya Loshchilov and Frank Hutter. 2019. Decoupled Weight Decay Regularization. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net. https://openreview.net/forum?id=Bkg6RiCqY7

[23] Bo Pang, Lillian Lee, et al. 2008. Opinion mining and sentiment analysis. *Foundations and Trends® in information retrieval* 2, 1–2 (2008), 1–135.

[24] 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, 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 *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett (Eds.). 8024–8035. https://proceedings.neurips.cc/paper/2019/hash/bdbca288fee7f92f2bfa9f7012727740-Abstract.html

[25] Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander Miller. 2019. Language Models as Knowledge Bases?. 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)*. Association for Computational Linguistics, Hong Kong, China, 2463–2473. https://doi.org/10.18653/v1/D19-1250

[26] Flor Miriam Plaza-del Arco, María-Teresa Martín-Valdivia, and Roman Klinger. 2022. Natural Language Inference Prompts for Zero-shot Emotion Classification in Text across Corpora. In *Proceedings of the 29th International Conference on Computational Linguistics*. International Committee on Computational Linguistics, Gyeongju, Republic of Korea, 6805–6817. https://aclanthology.org/2022.coling-1.592

[27] Raul Puri and Bryan Catanzaro. 2019. Zero-shot text classification with generative language models. *ArXiv preprint* abs/1912.10165 (2019). https://arxiv.org/abs/1912.10165

[28] Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. 2018. Improving language understanding by generative pre-training. (2018).

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

[30] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *Journal of Machine Learning Research* 21 (2020), 1–67.

[31] Timo Schick, Helmut Schmid, and Hinrich Schütze. 2020. Automatically Identifying Words That Can Serve as Labels for Few-Shot Text Classification. In *Proceedings of the 28th International Conference on Computational Linguistics*. International Committee on Computational Linguistics, Barcelona, Spain (Online), 5569–5578. https://doi.org/10.18653/v1/2020.coling-main.488

[32] Timo Schick and Hinrich Schütze. 2020. Few-shot text generation with pattern-exploiting training. *ArXiv preprint* abs/2012.11926 (2020). https://arxiv.org/abs/2012.11926

[33] Timo Schick and Hinrich Schütze. 2021. Exploiting Cloze-Questions for Few-Shot Text Classification and Natural Language Inference. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*. Association for Computational Linguistics, Online, 255–269. https://doi.org/10.18653/v1/2021.eacl-main.20

[34] Timo Schick and Hinrich Schütze. 2021. It's Not Just Size That Matters: Small Language Models Are Also Few-Shot Learners. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, Online, 2339–2352. https://doi.org/10.18653/v1/2021.naacl-main.185

[35] 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 the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, Online, 4222–4235. https://doi.org/10.18653/v1/2020.emnlp-main.346

[36] KL Sumathy and M Chidambaram. 2013. Text mining: concepts, applications, tools and issues-an overview. *International Journal of Computer Applications* 80, 4 (2013).

[37] Alon Talmor, Yanai Elazar, Yoav Goldberg, and Jonathan Berant. 2020. oLMpics-On What Language Model Pre-training Captures. *Transactions of the Association for Computational Linguistics* 8 (2020), 743–758. https://doi.org/10.1162/tacl_a_00342

[38] Derek Tam, Rakesh R. Menon, Mohit Bansal, Shashank Srivastava, and Colin Raffel. 2021. Improving and Simplifying Pattern Exploiting Training. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Online and Punta Cana, Dominican Republic, 4980–4991. https://doi.org/10.18653/v1/2021.emnlp-main.407

[39] 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 the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. Association for Computational Linguistics, Online, 38–45. https://doi.org/10.18653/v1/2020.emnlp-demos.6

[40] 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* abs/2108.13161 (2021). https://arxiv.org/abs/2108.13161

[41] Xiang Zhang, Junbo Jake Zhao, and Yann LeCun. 2015. Character-level Convolutional Networks for Text Classification. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, Corinna Cortes, Neil D. Lawrence, Daniel D. Lee, Masashi Sugiyama, and Roman Garnett (Eds.). 649–657. https://proceedings.neurips.cc/paper/2015/hash/250cf8b51c773f3f8dc8b4be867a9a02-Abstract.html