Effective Structured Prompting by Meta-Learning and Representative Verbalizer

Weisen Jiang [1 2] **Yu Zhang** [1 3] **James T. Kwok** [2]

Abstract

Prompt tuning for pre-trained masked language models (MLM) has shown promising performance in natural language processing tasks with few labeled examples. It tunes a *prompt* for the downstream task, and a *verbalizer* is used to bridge the predicted token and label prediction. Due to the limited training data, prompt initialization is crucial for prompt tuning. Recently, MetaPrompting (Hou et al., 2022) uses meta-learning to learn a shared initialization for all task-specific prompts. However, a single initialization is insufficient to obtain good prompts for all tasks and samples when the tasks are complex. Moreover, MetaPrompting requires tuning the whole MLM, causing a heavy burden on computation and memory as the MLM is usually large. To address these issues, we use a prompt pool to extract more task knowledge and construct instance-dependent prompts via attention. We further propose a novel soft verbalizer (RepVerb) which constructs label embedding from feature embeddings directly. Combining meta-learning the prompt pool and RepVerb, we propose MetaPrompter for effective structured prompting. MetaPrompter is parameter-efficient as only the pool is required to be tuned. Experimental results demonstrate that MetaPrompter performs better than the recent state-of-the-arts and RepVerb outperforms existing soft verbalizers.

[1]Guangdong Provincial Key Laboratory of Brain-inspired Intelligent Computation, Department of Computer Science and Engineering, Southern University of Science and Technology [2]Department of Computer Science and Engineering, Hong Kong University of Science and Technology [3]Peng Cheng Laboratory. Correspondence to: Yu Zhang <yu.zhang.ust@gmail.com>.

Proceedings of the 40^{th} International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023.

1. Introduction

In recent years, large pre-trained language models have achieved great success in solving a variety of downstream tasks (Howard & Ruder, 2018; Devlin et al., 2019; Yang et al., 2019; Conneau & Lample, 2019; Song et al., 2020; Guo et al., 2020; Raffel et al., 2020; Brown et al., 2020; Lester et al., 2021; Cui et al., 2022). Though fine-tuning the whole model (Howard & Ruder, 2018; Devlin et al., 2019) is effective and widely-used, optimizing and storing all the task-specific parameters can be compute- and memory-expensive when the model is large (e.g., GPT-3 (Brown et al., 2020) contains $100+$ billion parameters). To alleviate this issue, many approaches have been proposed. Examples include adapter tuning (Houlsby et al., 2019; Lin et al., 2020; Hu et al., 2022a) and prompt learning (Radford et al., 2019; Shin et al., 2020; Brown et al., 2020; Lester et al., 2021; Liu et al., 2021; Li & Liang, 2021; Liu et al., 2022b; Prasad et al., 2022; Liu et al., 2022a). However, prompt learning is more preferable due to its effectiveness and also that it can be easily plugged into a pre-trained MLM without invasive modification (Li & Liang, 2021; Hambardzumyan et al., 2021; He et al., 2022; Sun et al., 2022).

Prompt learning formulates the downstream task as a cloze-style MLM problem. It is useful for few-shot tasks due to its effectiveness, parameter-efficiency, and plug-and-play nature (Radford et al., 2019; Brown et al., 2020; Liu et al., 2022a). Specifically, prompt learning wraps an input text with a discrete *prompt* (e.g., "`Topic is [MASK]`") and feeds it to the MLM to predict a token at the `[MASK]` position. A *verbalizer* (Lester et al., 2021; Ding et al., 2022; Hu et al., 2022b) then maps the predicted token to the label. However, designing an effective prompt requires a good understanding of the downstream tasks.

Recently, *prompt tuning* (Lester et al., 2021; Liu et al., 2021; Zhang et al., 2022) proposes to wrap the input embedding with a *continuous* prompt. To reduce the number of parameters to be learned, the MLM is kept frozen. The continuous prompt can be further combined with discrete tokens to form a *template* (Liu et al., 2021; Schick & Schütze, 2021; Ding et al., 2022).

Prompt tuning can be sensitive to initialization (Lester et al.,

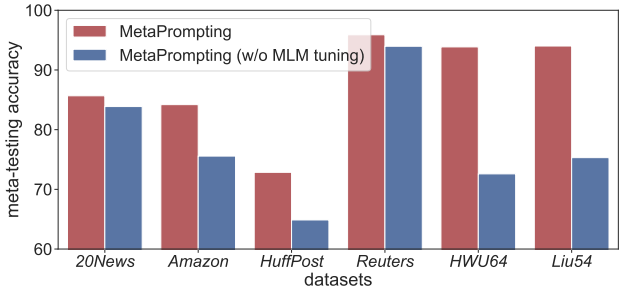


Figure 1. 5-way 5-shot classification meta-testing accuracy of MetaPrompting with or without MLM tuning on six data sets.

2021). Recently, a number of approaches have been proposed to alleviate this problem (Lester et al., 2021; Li et al., 2022; Vu et al., 2022). In particular, MetaPrompting (Hou et al., 2022) is the state-of-the-art that uses *meta-learning* (Bengio et al., 1991; Thrun & Pratt, 1998; Finn et al., 2017) to learn a meta-initialization for all task-specific prompts. However, MetaPrompting suffers from three problems. (i) When the tasks are complex, it is challenging to obtain good prompts for all tasks and samples from a single meta-initialized prompt. (ii) MetaPrompting uses a hand-crafted verbalizer. However, selecting good label tokens for the hand-crafted verbalizer is labor-intensive and not scalable for a large label set. (iii) MetaPrompting requires expensive tuning the whole MLM. Figure 1 shows a large gap in meta-testing accuracies with and without MLM tuning (experimental details are in Section 4).

In this paper, we use a pool of multiple prompts (Li et al., 2022; Wang et al., 2022a;b) to extract task knowledge from meta-training tasks, and then construct instance-dependent prompts as weighted combinations of all the prompts in the pool via attention (Vaswani et al., 2017). The attention's query vector is the instance's feature embedding. The prompt pool is the shared meta-knowledge and learned by the MAML algorithm (Finn et al., 2017). Specifically, given a task with a support set and a query set, the base learner takes the meta-parameter and the support set to build a task-specific prompt pool, then the meta-learner optimizes the meta-parameter on the query set. Meta-learning a prompt pool is more flexible than meta-learning only a single prompt initialization (as in MetaPrompting), and allows better adaptation of complex tasks. Moreover, as only the prompt pool is tuned, it is much more parameter-efficient than MetaPrompting (with $1000\times$ fewer parameters).

We also propose a novel soft verbalizer called *representative verbalizer* (RepVerb), which constructs label embeddings by averaging feature embeddings of the corresponding training samples. Unlike manually-designed verbalizers, RepVerb does not incur human effort for label token selection. Moreover, as RepVerb does not require learning any additional parameters, empirical results in Section 4.2 demonstrate that RepVerb is more effective than the soft verbalizers in WARP (Hambardzumyan et al., 2021), DART (Zhang et al., 2022), ProtoVerb (Cui et al., 2022). Besides, the feature embedding learned by RepVerb is more discriminative.

The whole procedure, which combines meta-learning the structured prompts and RepVerb, is called **MetaPrompter** in the sequel. Experiments are performed on six widely used classification data sets. Results demonstrate that RepVerb outperforms existing soft verbalizers, and is also beneficial to other prompt-based methods such as MetaPrompting. Moreover, MetaPrompter achieves better performance than the recent state-of-the-arts.

Our contributions are summarized as follows: (i) We propose a parameter-efficient algorithm MetaPrompter for effective structured prompting. (ii) We propose a simple and effective soft verbalizer (RepVerb). (iii) Experimental results demonstrate the effectiveness and parameter-efficiency of MetaPrompter.

2. Preliminaries and Related Work

2.1. Prompt Learning

Recently, it is common to use a pre-trained MLM $\mathcal{M}(\cdot;\boldsymbol{\phi})$, with parameter $\boldsymbol{\phi}$, for various downstream tasks such as language understanding (Dong et al., 2019; Yang et al., 2019; Song et al., 2020), machine translation (Conneau & Lample, 2019; Guo et al., 2020), and text classification (Brown et al., 2020; Lester et al., 2021; Liu et al., 2022b). Given a raw sentence represented as a sequence of n tokens $(x_1,\ldots,x_n)$, the MLM takes $\mathbf{x} = (\texttt{[CLS]}, x_1, \ldots, x_n, \texttt{[SEP]})$ as input (where `[CLS]` is the start token and `[SEP]` is the separator), and encodes it into a sequence of hidden representations $(\mathbf{h}_{\texttt{[CLS]}}, \mathbf{h}_1, \ldots, \mathbf{h}_n, \mathbf{h}_{\texttt{[SEP]}})$. In standard fine-tuning (Howard & Ruder, 2018; Devlin et al., 2019), an extra classifier (e.g., a fully connected layer with softmax normalization) is added on top of $\mathbf{h}_{\texttt{[CLS]}}$ to predict the label distribution. This classifier, together with $\boldsymbol{\phi}$, are tuned to maximize the probability of correct labels. As language models are large (e.g., 175 billion parameters in GPT-3 (Brown et al., 2020)), fine-tuning all parameters can cause a heavy burden on computation and memory.

On the other hand, prompt learning (Brown et al., 2020; Shin et al., 2020; Ding et al., 2022) freezes the pre-trained model and formulates the downstream task as a cloze-style MLM problem. For example, in topic classification, "`Topic is [MASK]`" can be used as the prompt, where `[MASK]` is a special token for prediction. The *discrete* tokens "`Topic is`" are also called anchor tokens. An input text $\mathbf{x}$ is wrapped with the prompt and mapped to an input embedding sequence $(\mathcal{E}(\mathbf{x}), \mathcal{E}(\texttt{Topic}), \mathcal{E}(\texttt{is}), \mathcal{E}(\texttt{[MASK]}))$, where $\mathcal{E}(\cdot)$ denotes the input embedding. Designing a suitable prompt requires domain expertise and a good understanding of the downstream tasks (Brown et al., 2020; Sanh et al.,

2022). Thus, manually-designed prompts are likely to be sub-optimal.

Unlike discrete prompts, prompt tuning (Lester et al., 2021; Liu et al., 2021) uses a *continuous* prompt $\boldsymbol{\theta} \in \mathbb{R}^{L_p \times d_i}$ (of length L_p) to directly wrap the input embedding sequence as $(\mathcal{E}(\mathbf{x}), \boldsymbol{\theta}, \mathcal{E}(\texttt{[MASK]}))$. This can be further combined with anchor tokens to form a *template* (Liu et al., 2021; Schick & Schütze, 2021; Ding et al., 2022):

$$\tilde{\mathbf{x}} \equiv \mathbb{T}(\mathbf{x}; \boldsymbol{\theta}) = (\mathcal{E}(\mathbf{x}), \boldsymbol{\theta}, \mathcal{E}(\texttt{Topic}), \mathcal{E}(\texttt{is}), \mathcal{E}(\texttt{[MASK]})).$$

The MLM then outputs the hidden embedding $\mathbf{h}_{\texttt{[MASK]}}(\tilde{\mathbf{x}}) \in \mathbb{R}^{d_o}$ of `[MASK]`, and infers the token to be filled at the `[MASK]` position.

A *verbalizer* (Lester et al., 2021; Ding et al., 2022; Hu et al., 2022b) bridges the prediction at the `[MASK]` position and labels in prompt learning. Specifically, it is a *hard* mapping from each label y to a set of label-relevant tokens $\mathcal{V}_y$. For example, for $y = \texttt{SPORTS}$, we can have $\mathcal{V}_y = \{\texttt{sports, football, basketball}\}$. Prompt tuning then optimizes[1] $(\boldsymbol{\phi}, \boldsymbol{\theta})$ by maximizing the label probability:

$$\hat{\mathbb{P}}(y|\mathbf{x}; \boldsymbol{\phi}, \boldsymbol{\theta}) = \frac{1}{|\mathcal{V}_y|} \sum_{\texttt{w} \in \mathcal{V}_y} \mathbb{P}_{\mathcal{M}}(\texttt{[MASK]} = \texttt{w} | \mathbb{T}(\mathbf{x}; \boldsymbol{\theta})), \quad (1)$$

where $\mathbb{P}_{\mathcal{M}}(\texttt{[MASK]} | \mathbb{T}(\mathbf{x}; \boldsymbol{\theta}))$ is the probability distribution over vocabulary as predicted by the MLM at the `[MASK]` position.

The verbalizer is crucial to the performance of prompt learning (Lester et al., 2021; Ding et al., 2022). However, selecting label-relevant tokens requires intensive human labor. To address this problem, search-based methods (Schick et al., 2020; Shin et al., 2020; Gao et al., 2021) try to find label tokens automatically from the training data. However, searching in a *discrete* space is computationally intensive (Schick et al., 2020; Shin et al., 2020; Gao et al., 2021), especially with a large number of labels or vocabulary. Some recent works (Hambardzumyan et al., 2021; Zhang et al., 2022; Cui et al., 2022) propose *soft* verbalizers, which map each label to a *continuous* embedding and predict the label distribution based on the similarities between feature embedding and label embeddings. WARP (Hambardzumyan et al., 2021) and DART (Zhang et al., 2022) obtain this label embedding by supervised learning, while ProtoVerb (Cui et al., 2022) uses contrastive learning (Chen et al., 2020; Tian et al., 2020). However, learning the embedding $\mathbf{v}_y \in \mathbb{R}^{d_o}$ for each label y can be challenging in the few-shot learning setting (Gao et al., 2019; Bao et al., 2020; Han et al., 2021; Chen et al., 2022; Hou et al., 2022), as the number of samples per class is typically much smaller than d_o (e.g., $d_o = 768$ for BERT (Devlin et al., 2019)).

[1] $\boldsymbol{\phi}$ can be fixed for parameter-efficiency in prompt learning.

2.2. Meta-Learning for Prompt Learning

In meta-learning (Bengio et al., 1991; Thrun & Pratt, 1998), a collection $\mathcal{T}$ of tasks are used to learn a shared meta-parameter. Each task $\tau \in \mathcal{T}$ has a support set $\mathcal{S}_\tau$ and a query set $\mathcal{Q}_\tau$. Let $\mathcal{Y}_\tau$ be the label set of τ. Typical meta-learning algorithms can be metric-based (Vinyals et al., 2016; Snell et al., 2017; Bertinetto et al., 2018; Lee et al., 2019), memory-based (Santoro et al., 2016; Munkhdalai & Yu, 2017), or optimization-based (Finn et al., 2017; Rajeswaran et al., 2019; Raghu et al., 2020; Ye et al., 2021; Jiang et al., 2021; 2022; Flennerhag et al., 2022). In general, the optimization-based approach is preferred due to its simplicity and effectiveness. A representative algorithm is model-agnostic meta-learning (MAML) (Finn et al., 2017).

As prompt tuning is sensitive to prompt initialization in few-shot tasks (Lester et al., 2021), meta-learning can be used to search for a good initialization. MetaPrompting (Hou et al., 2022) uses MAML to learn a meta-initialization for the task-specific prompts. At iteration t, the base learner takes a task τ and meta-parameter $(\boldsymbol{\phi}_{t-1}, \boldsymbol{\theta}_{t-1})$, and builds a task-specific model $(\boldsymbol{\phi}_{t,J}, \boldsymbol{\theta}_{t,J})$ by performing J gradient updates on the support set with step size α and initialization $(\boldsymbol{\phi}_{t,0}, \boldsymbol{\theta}_{t,0}) \equiv (\boldsymbol{\phi}_{t-1}, \boldsymbol{\theta}_{t-1})$:

$$\begin{aligned}(\boldsymbol{\phi}_{t,j}, \boldsymbol{\theta}_{t,j}) &= (\boldsymbol{\phi}_{t,j-1}, \boldsymbol{\theta}_{\mathbf{x},j-1}) \\ &+ \alpha \nabla_{(\boldsymbol{\phi}_{t,j-1}, \boldsymbol{\theta}_{\mathbf{x},j-1})} \sum_{(\mathbf{x},y) \in \mathcal{S}_\tau} \log \hat{\mathbb{P}}(y|\mathbf{x}; \boldsymbol{\phi}_{t,j-1}, \boldsymbol{\theta}_{\mathbf{x},j-1}).\end{aligned}$$

The meta-learner then updates the meta-initialization by maximizing the log-likelihood objective on the query set with step size η:

$$\begin{aligned}(\boldsymbol{\phi}_t, \boldsymbol{\theta}_t) &= (\boldsymbol{\phi}_{t-1}, \boldsymbol{\theta}_{t-1}) \\ &+ \eta \nabla_{(\boldsymbol{\phi}_{t-1}, \boldsymbol{\theta}_{t-1})} \sum_{(\mathbf{x},y) \in \mathcal{Q}_\tau} \log \hat{\mathbb{P}}(y|\mathbf{x}; \boldsymbol{\phi}_{t,J}, \boldsymbol{\theta}_{t,J}).\end{aligned}$$

Though MetaPrompting achieves state-of-the-art performance in the few-shot classification experiments (Hou et al., 2022), it suffers from the three problems discussed in Section 1. (i) When the tasks are complex, it is challenging to use a single meta-initialized prompt for adaptation to the various tasks. (ii) MetaPrompting uses a hand-crafted verbalizer, which is labor-intensive and not scalable as discussed in Section 2.1. (iii) MetaPrompting needs to tune the MLM parameters, and thus is not parameter-efficient.

3. Proposed Method

In Section 3.1, we first propose a novel and effective soft verbalizer (representative verbalizer) without inducing additional parameters. Moreover, while MetaPrompting uses a single prompt initialization to build task-specific prompts, we propose in Section 3.2 the extraction of task knowledge

Algorithm 1 Representative Verbalizer (RepVerb).

procedure ComputeLabelEmbedding($\mathcal{S}_\tau$):
  compute $\mathbf{h}_{\texttt{[MASK]}}(\tilde{\mathbf{x}})$ for $(\mathbf{x}, \cdot) \in \mathcal{S}_\tau$;
  compute $\mathbf{v}_y$ by (2) for $y \in \mathcal{Y}_\tau$;
end procedure

procedure Predict($\mathbf{x}; \mathbf{v}_y : y \in \mathcal{Y}_\tau$)
  compute $\mathbf{h}_{\texttt{[MASK]}}(\tilde{\mathbf{x}})$ for $\mathbf{x}$;
  compute $\tilde{\mathbb{P}}(y|\mathbf{x}; \boldsymbol{\phi}, \boldsymbol{\theta})$ by (3);
end procedure

into a pool of multiple prompts, and constructs instance-dependent prompts by attention (Vaswani et al., 2017).

3.1. Representative Verbalizer (RepVerb)

Instead of explicitly learning an embedding $\mathbf{v}_y$ for each label y (Hambardzumyan et al., 2021; Cui et al., 2022; Zhang et al., 2022), we propose the *Representative Verbalizer* (RepVerb), which constructs $\mathbf{v}_y$ from feature embeddings of the corresponding training samples (Algorithm 1). It does not require learning additional parameters, and is thus more effective on limited data as in few-shot learning.

Specifically, let $\mathcal{S}_{\tau,y}$ be the subset of samples in $\mathcal{S}_\tau$ with label y. For an input $\mathbf{x}$, we wrap it with the template and feed $\tilde{\mathbf{x}} \equiv \mathbb{T}(\mathbf{x}; \boldsymbol{\theta})$ to the pre-trained MLM, and then obtain `[MASK]`'s embedding $\mathbf{h}_{\texttt{[MASK]}}(\tilde{\mathbf{x}})$ as its feature embedding. Similar to ProtoNet (Snell et al., 2017), we propose to construct $\mathbf{v}_y$ for each y by averaging the corresponding samples' feature embeddings, as:

$$\mathbf{v}_y = \frac{1}{|\mathcal{S}_{\tau,y}|} \sum_{(\mathbf{x},y) \in \mathcal{S}_{\tau,y}} \mathbf{h}_{\texttt{[MASK]}}(\tilde{\mathbf{x}}). \quad (2)$$

To predict the label of a given $\mathbf{x}$, we measure the cosine similarity[2] between $\mathbf{h}_{\texttt{[MASK]}}(\tilde{\mathbf{x}})$ and each $\mathbf{v}_y$ ($y \in \mathcal{Y}_\tau$):

$$\tilde{\mathbb{P}}(y|\mathbf{x}; \boldsymbol{\phi}, \boldsymbol{\theta}) = \frac{\exp(\rho \cos(\mathbf{v}_y, \mathbf{h}_{\texttt{[MASK]}}(\tilde{\mathbf{x}})))}{\sum_{y' \in \mathcal{Y}_\tau} \exp(\rho \cos(\mathbf{v}_{y'}, \mathbf{h}_{\texttt{[MASK]}}(\tilde{\mathbf{x}})))}, \quad (3)$$

where $\rho > 0$ is the temperature. When $\rho \to \infty$, $\tilde{\mathbb{P}}(y|\mathbf{x}; \boldsymbol{\phi}, \boldsymbol{\theta})$ becomes one-hot; whereas when $\rho \to 0$, $\tilde{\mathbb{P}}(y|\mathbf{x}; \boldsymbol{\phi}, \boldsymbol{\theta})$ becomes uniform. In the experiments, we set $\rho = 10$ as in Oreshkin et al. (2018).

3.2. Meta Structured-Prompting

In the following, we propose the use of MAML and attention mechanism (Vaswani et al., 2017) to meta-learn a prompt pool. While MetaPrompting uses task-specific prompts (Hou et al., 2022), we propose the construction of instance-specific prompts, which allows more flexibility.

3.2.1. Meta-Learn a Prompt Pool

While MetaPrompting uses only a single initialization for the prompt, we propose to leverage a pool of prompts to extract more task knowledge, which is particularly effective when the tasks are complex and very different prompts may be needed. A prompt pool has K learnable prompts $\{(\mathbf{k}_i, \boldsymbol{\theta}_i) : i = 1, \ldots, K\}$, with key $\mathbf{k}_i \in \mathbb{R}^{d_o}$ and value $\boldsymbol{\theta}_i \in \mathbb{R}^{L_p \times d_i}$ (Li et al., 2022; Wang et al., 2022a;b). Note that the size of the prompt pool is negligible compared with that of the MLM. For example, in our experiments, the MLM has 109.52×10^6 parameters, while the prompt pool has only $55,296$.

The prompt pool can be considered as shared meta-knowledge. Given an input $\mathbf{x}$, the attention weights between $\mathbf{x}$ and the K prompts are computed as $\mathbf{a} = \mathrm{softmax}(\frac{\mathbf{K}\mathbf{q}_\mathbf{x}}{\sqrt{d_o}})$, where $\mathrm{softmax}(\cdot)$ is the softmax function, $\mathbf{K} = [\mathbf{k}_1^\top; \ldots; \mathbf{k}_K^\top]$, and $\mathbf{q}_\mathbf{x} \in \mathbb{R}^{d_o}$ is the embedding of the `[MASK]` output by a pre-trained and frozen MLM with the wrapped input (e.g., ($\mathbf{x}$. `Topic is [MASK]`)) (Wang et al., 2022a;b). Such a mapping from $\mathbf{x}$ to $\mathbf{q}_\mathbf{x}$ is called the query function $q(\cdot)$. An instance-dependent prompt is then generated by weighted averaging over all the values ($\boldsymbol{\theta}_i$'s):

$$\boldsymbol{\theta}_\mathbf{x}(\mathbf{K}, \boldsymbol{\Theta}) = \sum_{i=1}^{K} a_i \boldsymbol{\theta}_i, \quad (4)$$

where $\boldsymbol{\Theta} = [\boldsymbol{\theta}_1; \ldots; \boldsymbol{\theta}_K]$. While Wang et al. (2022a;b) only select the top-N most similar prompts from the pool, in (4) all the prompts are used and updated simultaneously.

The proposed procedure for meta-learning the prompt pool $(\mathbf{K}, \boldsymbol{\Theta})$, which will be called MetaPrompter, is shown in Algorithm 2. The MAML algorithm (Finn et al., 2017) is used here, but other meta-learning algorithms (e.g., Reptile (Nichol et al., 2018), BMG (Flennerhag et al., 2022)) can also be used. At iteration t, the base learner takes $(\mathbf{K}_{t-1}, \boldsymbol{\Theta}_{t-1})$ and a task τ to optimize for a task-specific prompt pool by gradient descent (steps 4-15). $(\mathbf{K}_{t-1}, \boldsymbol{\Theta}_{t-1})$ is used as the initialization (step 4). For each inner iteration j, $(\mathbf{K}_{t,j-1}, \boldsymbol{\Theta}_{t,j-1})$ constructs the instance-dependent prompts $\boldsymbol{\theta}_{\mathbf{x},j}(\mathbf{K}_{t,j-1}, \boldsymbol{\Theta}_{t,j-1})$ in (4) (steps 7 and 8). Next, $\boldsymbol{\theta}_{\mathbf{x},j}$ is used to predict the label probability with a combination of the hand-crafted verbalizer (step 9) and soft verbalizer (steps 11 and 12):

$$\mathbb{P}(y|\mathbf{x}; \boldsymbol{\theta}_{\mathbf{x},j}) = (1-\lambda)\hat{\mathbb{P}}(y|\mathbf{x}; \boldsymbol{\theta}_{\mathbf{x},j}) + \lambda \tilde{\mathbb{P}}(y|\mathbf{x}; \boldsymbol{\theta}_{\mathbf{x},j}), \quad (5)$$

where $\lambda \in [0, 1]$ (in the experiments, we set $\lambda = 0.5$). Let $\mathcal{L}(\mathcal{S}_\tau; \mathbf{K}_{t,j-1}, \boldsymbol{\Theta}_{t,j-1}) = -\sum_{(\mathbf{x},y) \in \mathcal{S}_\tau} \log \mathbb{P}(y|\mathbf{x}; \boldsymbol{\theta}_{\mathbf{x},j})$ be the loss on $\mathcal{S}_\tau$ (step 13). The base learner builds a

[2] Dissimilarity measures, such as the Euclidean distance, can also be used.

Algorithm 2 MetaPrompter.

Require: prompt length L_p; size of prompt pool K; $\lambda = 0.5$; step sizes α, η; meta-parameters $(\mathbf{K}, \mathbf{\Theta})$; query function $q(\cdot)$;

1: **for** $t = 1, \ldots, T$ **do**
2: sample a task $\tau = (\mathcal{S}_\tau, \mathcal{Q}_\tau) \in \mathcal{T}$;
3: *base learner:*
4: $(\mathbf{K}_{t,0}, \mathbf{\Theta}_{t,0}) \equiv (\mathbf{K}_{t-1}, \mathbf{\Theta}_{t-1})$;
5: **for** $j = 1, \ldots, J$ **do**
6: **for** $(\mathbf{x}, y) \in \mathcal{S}_\tau$ **do**
7: compute $\mathbf{q}_\mathbf{x}$ by $q(\cdot)$;
8: $\boldsymbol{\theta}_{\mathbf{x},j}(\mathbf{K}_{t,j-1}, \mathbf{\Theta}_{t,j-1}) = \text{softmax}(\mathbf{K}_{t,j-1}\mathbf{q}_\mathbf{x})^\top \mathbf{\Theta}_{t,j-1}$;
9: feed $\tilde{\mathbf{x}} \equiv \mathbb{T}(\mathbf{x}; \boldsymbol{\theta}_{\mathbf{x},j})$ into $\mathcal{M}$, obtain $\mathbf{h}_{\texttt{[MASK]}}(\tilde{\mathbf{x}})$, and $\hat{\mathbb{P}}(y|\mathbf{x}; \boldsymbol{\theta}_{\mathbf{x},j})$ by (1);
10: **end for**
11: call ComputeLabelEmbedding($\mathcal{S}_\tau$) of Algorithm 1 to obtain $\{\mathbf{v}_y : y \in \mathcal{Y}_\tau\}$;
12: for $(\mathbf{x},y) \in \mathcal{S}_\tau$, call Predict($\mathbf{x}; \mathbf{v}_y : y \in \mathcal{Y}_\tau$) of Algorithm 1 to obtain $\tilde{\mathbb{P}}(y|\mathbf{x}; \boldsymbol{\theta}_{\mathbf{x},j})$, and compute $\mathbb{P}(y|\mathbf{x}; \boldsymbol{\theta}_{\mathbf{x},j})$ by (5);
13: $\mathcal{L}(\mathcal{S}_\tau; \mathbf{K}_{t,j-1}, \mathbf{\Theta}_{t,j-1}) = -\sum_{(\mathbf{x},y)\in\mathcal{S}_\tau} \log \mathbb{P}(y|\mathbf{x}; \boldsymbol{\theta}_{\mathbf{x},j})$;
14: $(\mathbf{K}_{t,j}, \mathbf{\Theta}_{t,j}) = (\mathbf{K}_{t,j-1}, \mathbf{\Theta}_{t,j-1}) - \alpha \nabla_{(\mathbf{K}_{t,j-1}, \mathbf{\Theta}_{t,j-1})} \mathcal{L}(\mathcal{S}_\tau; \mathbf{K}_{t,j-1}, \mathbf{\Theta}_{t,j-1})$;
15: **end for**
16: *meta-learner:*
17: **for** $(\mathbf{x}, y) \in \mathcal{Q}_\tau$ **do**
18: compute $\mathbf{q}_\mathbf{x}$ by $q(\cdot)$;
19: $\boldsymbol{\theta}_{\mathbf{x},J}(\mathbf{K}_{t,J}, \mathbf{\Theta}_{t,J}) = \text{softmax}(\mathbf{K}_{t,J}\mathbf{q}_\mathbf{x})^\top \mathbf{\Theta}_{t,J}$;
20: call Predict($\mathbf{x}; \mathbf{v}_y : y \in \mathcal{Y}_\tau$) of Algorithm 1 to obtain $\tilde{\mathbb{P}}(y|\mathbf{x}; \boldsymbol{\theta}_{\mathbf{x},J})$;
21: compute $\hat{\mathbb{P}}(y|\mathbf{x}; \boldsymbol{\theta}_{\mathbf{x},J})$ and $\mathbb{P}(y|\mathbf{x}; \boldsymbol{\theta}_{\mathbf{x},J})$ by (1) and (5), respectively;
22: **end for**
23: $\mathcal{L}(\mathcal{Q}_\tau; \mathbf{K}_{t,J}, \mathbf{\Theta}_{t,J}) = -\sum_{(\mathbf{x},y)\in\mathcal{Q}_\tau} \log \mathbb{P}(y|\mathbf{x}; \boldsymbol{\theta}_{\mathbf{x},J})$;
24: $(\mathbf{K}_t, \mathbf{\Theta}_t) = (\mathbf{K}_{t-1}, \mathbf{\Theta}_{t-1}) - \eta \nabla_{(\mathbf{K}_{t-1}, \mathbf{\Theta}_{t-1})} \mathcal{L}(\mathcal{Q}_\tau; \mathbf{K}_{t,J}, \mathbf{\Theta}_{t,J})$;
25: **end for**
26: **return** $(\mathbf{K}_T, \mathbf{\Theta}_T)$.

task-specific prompt pool $(\mathbf{K}_{t,J}, \mathbf{\Theta}_{t,J})$ by taking J gradient updates ($j = 1, \ldots, J$) at step 14:

$$(\mathbf{K}_{t,j}, \mathbf{\Theta}_{t,j}) = (\mathbf{K}_{t,j-1}, \mathbf{\Theta}_{t,j-1}) - \alpha \nabla_{(\mathbf{K}_{t,j-1}, \mathbf{\Theta}_{t,j-1})} \mathcal{L}(\mathcal{S}_\tau; \mathbf{K}_{t,j-1}, \mathbf{\Theta}_{t,j-1}).$$

The meta-learner takes $(\mathbf{K}_{t,J}, \mathbf{\Theta}_{t,J})$ and $\mathcal{Q}_\tau$ to update the meta-parameters (steps 17-24). For $(\mathbf{x}, y) \in \mathcal{Q}_\tau$, we use $(\mathbf{K}_{t,J}, \mathbf{\Theta}_{t,J})$ to generate its prompt $\boldsymbol{\theta}_{\mathbf{x},J}(\mathbf{K}_{t,J}, \mathbf{\Theta}_{t,J})$ (steps 18 and 19), which is used for make prediction $\mathbb{P}(y|\mathbf{x}; \boldsymbol{\theta}_{\mathbf{x},J})$ (steps 20 and 21). Let $\mathcal{L}(\mathcal{Q}_\tau; \mathbf{K}_{t,J}, \mathbf{\Theta}_{t,J}) = -\sum_{(\mathbf{x},y)\in\mathcal{Q}_\tau} \log \mathbb{P}(y|\mathbf{x}; \boldsymbol{\theta}_{\mathbf{x},J})$ be the negative log-likelihood loss on $\mathcal{Q}_\tau$ (step 23). The meta-learner updates the meta-parameters by performing one gradient descent step on $\mathcal{L}(\mathcal{Q}_\tau; \mathbf{K}_{t,J}, \mathbf{\Theta}_{t,J})$ at step 24:

$$(\mathbf{K}_t, \mathbf{\Theta}_t) = (\mathbf{K}_{t-1}, \mathbf{\Theta}_{t-1}) - \eta \nabla_{(\mathbf{K}_{t-1}, \mathbf{\Theta}_{t-1})} \mathcal{L}(\mathcal{Q}_\tau; \mathbf{K}_{t,J}, \mathbf{\Theta}_{t,J}).$$

The meta-gradient $\nabla_{(\mathbf{K}_{t-1}, \mathbf{\Theta}_{t-1})} \mathcal{L}(\mathcal{Q}_\tau; \mathbf{K}_{t,J}, \mathbf{\Theta}_{t,J}) = \nabla_{(\mathbf{K}_{t,J}, \mathbf{\Theta}_{t,J})} \mathcal{L}(\mathcal{Q}_\tau; \mathbf{K}_{t,J}, \mathbf{\Theta}_{t,J}) \nabla_{(\mathbf{K}_{t-1}, \mathbf{\Theta}_{t-1})} (\mathbf{K}_{t,J}, \mathbf{\Theta}_{t,J})$ requires back-propagating through the entire inner optimization path, which is computationally infeasible for large models and J is large. To reduce the computational cost, we discard the second-order derivative and use the first-order approximation $\nabla_{(\mathbf{K}_{t-1}, \mathbf{\Theta}_{t-1})} \mathcal{L}(\mathcal{Q}_\tau; \mathbf{K}_{t,J}, \mathbf{\Theta}_{t,J}) \approx \nabla_{(\mathbf{K}_{t,J}, \mathbf{\Theta}_{t,J})} \mathcal{L}(\mathcal{Q}_\tau; \mathbf{K}_{t,J}, \mathbf{\Theta}_{t,J})$ (step 24) as in (Finn et al., 2017; Hou et al., 2022).

3.2.2. META-TESTING

Given an unseen task $\tau' = (\mathcal{S}_{\tau'}, \mathcal{Q}_{\tau'})$, the base learner takes $\mathcal{S}_{\tau'}$ and $(\mathbf{K}_T, \mathbf{\Theta}_T)$ to build a task-specific prompt pool $(\mathbf{K}_{T,J}, \mathbf{\Theta}_{T,J})$ as in steps 4-15. This pool is then used to construct instance-dependent prompts $\boldsymbol{\theta}_{\mathbf{x},J}$ for each $(\mathbf{x}, \cdot) \in \mathcal{Q}_{\tau'}$. The MLM receives the wrapped input $\tilde{\mathbf{x}} \equiv \mathbb{T}(\mathbf{x}; \boldsymbol{\theta}_{\mathbf{x},J})$ and predicts the label probability by (5).

3.2.3. METAPROMPTER IS PARAMETER-EFFICIENT

As MetaPrompter only tunes $(\mathbf{K}, \mathbf{\Theta})$, the total number of meta-parameters is $K(d_o + L_p d_i)$ (where d_i and d_o are the dimensions of the input and feature embeddings, respectively). This is much smaller than that of MetaPrompting (which is equal to $d_{\boldsymbol{\phi}} + L_p d_i$, where $d_{\boldsymbol{\phi}}$ is the size of $\boldsymbol{\phi}$), as it requires tuning the whole MLM. For example, in the experiments, we use BERT (with $d_o = d_i = 768$, $d_{\boldsymbol{\phi}} = 109 \times 10^6$) and $K = L_p = 8$ in MetaPrompter.

4. Experiments

4.1. Setup

Following Chen et al. (2022), we perform few-shot classification on six popularly used data sets: (i) *20News* (Lang, 1995), which contains informal discourses from news discussion forums of 20 topics; (ii) *Amazon* (He & McAuley, 2016), which consists of customer reviews from 24 products. The task is to classify reviews into product categories; (iii) *HuffPost* (Misra, 2022), which contains news headlines of 41 topics published in the HuffPost between 2012 and 2018. These headlines are shorter and less grammatical than formal sentences, thus are more challenging for classification; (iv) *Reuters* (Lewis, 1997), which is a collection of Reuters newswire articles of 31 topics from 1996 to 1997; (v) *HWU64* (Liu et al., 2019), which is an intent classification data set containing user utterances of 64 intents; (vi) *Liu54* (Liu et al., 2019), which is an imbalanced intent classification data set of 54 classes collected on Amazon Mechanical Turk. We use the meta-training/meta-validation/meta-testing splits provided in Chen et al. (2022). A summary of the data sets is in Table 1.

Following (Bao et al., 2020; Han et al., 2021; Chen et al., 2022; Hou et al., 2022), we perform experiments in the 5-way 1-shot and 5-way 5-shot settings with 15 query samples per class. The pre-trained BERT (*bert-base-uncased*) from HuggingFace (Wolf et al., 2020) is used as the pre-trained MLM as in (Chen et al., 2022; Hou et al., 2022). Experiments are run on a DGX station with 8 V100 32GB GPUs. The experiment is repeated three times with different random seeds.

Table 1. Statistics of the data sets.

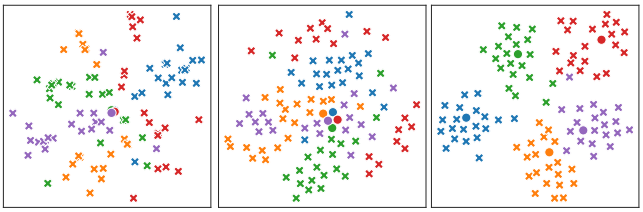
	#classes (meta-train/valid/test)	#samples	#tokens per sample (mean ± std)
20News	8/5/7	18,820	340 ± 151
Amazon	10/5/9	24,000	140 ± 32
HuffPost	20/5/16	36,900	11 ± 4
Reuters	15/5/11	620	168 ± 136
HWU64	23/16/25	11,036	7 ± 3
Liu54	18/18/18	25,478	8 ± 4

4.2. Evaluation on RepVerb

First, we compare the performance of the proposed RepVerb with state-of-the-art soft verbalizers: (i) WARP (Hambardzumyan et al., 2021)[3], and (ii) ProtoVerb (Cui et al., 2022). As the focus is on evaluating verbalizers, all methods use the same discrete prompt "`Topic is [MASK]`", and fine-tune all parameters for 5 steps with a learning rate of 0.00005 as in Cui et al. (2022).

Results. Table 2 reports the meta-testing accuracies. As can be seen, RepVerb outperforms WARP and ProtoVerb on both the 1-shot and 5-shot settings.

Figure 2 shows the t-SNE visualization of the embeddings ($\mathbf{h}_{\texttt{[MASK]}}(\mathbf{x})$'s) of 100 samples ($\mathbf{x}$'s)[4] and learned label embeddings ($\mathbf{v}_y$'s) for a random 5-way 5-shot task from *Reuters*.[5] As can be seen, the RepVerb embedding is more discriminative and compact than those of WARP and ProtoVerb. Moreover, by design, RepVerb's label embedding is consistent with the samples' feature embeddings, while those of WARP and ProtoVerb are not.



(a) WARP. (b) ProtoVerb. (c) RepVerb.

Figure 2. t-SNE visualization of `[MASK]`'s embeddings (crosses) and label embeddings (circles) for a 5-way 5-shot task randomly sampled from *Reuters*.

4.3. Evaluation on MetaPrompter

We compare MetaPrompter with a variety of baselines. These include state-of-the-art prompt-based methods of (i) MetaPrompting (Hou et al., 2022), and its variants (ii) MetaPrompting+WARP / MetaPrompting+ProtoVerb / MetaPrompting+RepVerb, which combine MetaPrompting with the soft verbalizer of WARP / ProtoVerb / RepVerb, respectively. Moreover, we also compare with the non-prompt-based methods of: (iii) HATT (Gao et al., 2019), which meta-learns a prototypical network (Snell et al., 2017) with a hybrid attention mechanism; (iv) DS (Bao et al., 2020), which learns attention scores based on word frequency; (v) MLADA (Han et al., 2021), which uses an adversarial domain adaptation network to extract domain-invariant features during meta–training; and (vi) ContrastNet (Chen et al., 2022), which performs feature extraction by contrastive learning.

For MetaPrompter, hyperparameters K and L_p are chosen from $\{1, 2, 4, 8, 16, 32, 64\}$ using the meta-validation set. For the base learner, $\alpha = 0.1$, and $J = 5$ (resp. 15) at meta-training (resp. meta-validation or meta-testing). We train the prompt pool for $T = 3,000$ iterations using the Adam optimizer (Kingma & Ba, 2015) with a learning rate of 0.001. To prevent overfitting, we evaluate the meta-validation performance every 50 iteration and choose the checkpoint with the best meta-validation performance for meta-testing. For the hand-crafted verbalizer used in (1), label tokens are obtained by tokenizing the class name and its synonyms as in (Hou et al., 2022; Hu et al., 2022b). Following Lester et al.

[3]Note that the verbalizer of WARP is the same as that of DART (Zhang et al., 2022). Its implementation is described in Appendix A.

[4]5-way × (5 support samples + 15 query samples) = 100.

[5]Results on the other data sets are in Figure 7 of Appendix B.

Table 2. Meta-testing accuracy of various verbalizers on 5-way few-shot classification.

		20News	*Amazon*	*HuffPost*	*Reuters*	*HWU64*	*Liu54*
5-shot	WARP (Hambardzumyan et al., 2021)	61.43 ± 0.15	59.53 ± 0.20	46.31 ± 0.31	68.67 ± 0.71	68.60 ± 0.40	73.11 ± 0.26
	ProtoVerb (Cui et al., 2022)	71.33 ± 0.11	71.74 ± 0.21	57.93 ± 0.17	80.93 ± 0.54	73.43 ± 0.51	76.19 ± 0.33
	RepVerb	$\mathbf{78.81} \pm 0.08$	$\mathbf{77.56} \pm 0.16$	$\mathbf{61.90} \pm 0.08$	$\mathbf{88.33} \pm 0.40$	$\mathbf{78.37} \pm 0.49$	$\mathbf{82.14} \pm 0.23$
1-shot	WARP (Hambardzumyan et al., 2021)	49.87 ± 0.63	48.94 ± 0.34	38.21 ± 0.35	52.88 ± 0.67	53.20 ± 0.76	58.68 ± 0.64
	ProtoVerb (Cui et al., 2022)	54.13 ± 0.46	55.07 ± 0.27	41.40 ± 0.21	57.27 ± 0.73	55.17 ± 0.81	60.16 ± 0.37
	RepVerb	$\mathbf{59.86} \pm 0.38$	$\mathbf{59.18} \pm 0.31$	$\mathbf{44.65} \pm 0.20$	$\mathbf{63.63} \pm 0.41$	$\mathbf{59.83} \pm 0.71$	$\mathbf{66.17} \pm 0.40$

Table 3. Number of parameters and 5-way 5-shot classification meta-testing accuracy. Results marked with † are from Chen et al. (2022). "–" indicates that the corresponding result is not reported in Chen et al. (2022).

	#param ($\times 10^6$)	*20News*	*Amazon*	*HuffPost*	*Reuters*	*HWU64*	*Liu54*
HATT† (Gao et al., 2019)	0.07	55.00	66.00	56.30	56.20	-	-
DS† (Bao et al., 2020)	1.73	68.30	81.10	63.50	96.00	-	-
MLADA† (Han et al., 2021)	0.73	77.80	86.00	64.90	96.70	-	-
ContrastNet† (Chen et al., 2022)	109.52	71.74	85.17	65.32	95.33	92.57	93.72
MetaPrompting (Hou et al., 2022)	109.52	85.67 ± 0.44	84.19 ± 0.30	72.85 ± 1.01	95.89 ± 0.23	93.86 ± 0.97	94.01 ± 0.26
MetaPrompting+WARP	109.52	85.81 ± 0.48	85.54 ± 0.20	71.71 ± 0.72	97.28 ± 0.30	93.99 ± 0.76	94.33 ± 0.27
MetaPrompting+ProtoVerb	109.52	86.18 ± 0.51	84.91 ± 0.38	73.11 ± 0.80	97.24 ± 0.25	93.81 ± 0.81	94.38 ± 0.18
MetaPrompting+RepVerb	109.52	86.89 ± 0.39	85.98 ± 0.28	74.62 ± 0.88	97.32 ± 0.31	94.23 ± 0.67	94.45 ± 0.33
MetaPrompter	0.06	$\mathbf{88.57} \pm 0.38$	$\mathbf{86.36} \pm 0.24$	$\mathbf{74.89} \pm 0.75$	$\mathbf{97.63} \pm 0.22$	$\mathbf{95.30} \pm 0.51$	$\mathbf{95.47} \pm 0.21$

Table 4. Number of parameters and 5-way 1-shot Meta-testing classification accuracy. Results marked with † are from Chen et al. (2022). "–" indicates that the corresponding result is not reported in Chen et al. (2022).

	#param ($\times 10^6$)	*20News*	*Amazon*	*HuffPost*	*Reuters*	*HWU64*	*Liu54*
HATT† (Gao et al., 2019)	0.07	44.20	49.10	41.10	43.20	-	-
DS† (Bao et al., 2020)	1.73	52.10	62.60	43.00	81.80	-	-
MLADA† (Han et al., 2021)	0.73	59.60	68.40	64.90	82.30	-	-
ContrastNet† (Chen et al., 2022)	109.52	71.74	76.13	53.06	86.42	86.56	85.89
MetaPrompting (Hou et al., 2022)	109.52	82.46 ± 0.50	76.92 ± 0.77	68.62 ± 0.56	92.56 ± 0.77	91.06 ± 0.41	87.79 ± 0.29
MetaPrompting +WARP	109.52	82.93 ± 0.39	78.27 ± 0.72	67.78 ± 0.41	94.74 ± 0.56	91.30 ± 0.35	88.69 ± 0.26
MetaPrompting+ProtoVerb	109.52	83.15 ± 0.41	78.19 ± 0.65	68.96 ± 0.52	95.26 ± 0.40	91.27 ± 0.63	90.05 ± 0.15
MetaPrompting+RepVerb	109.52	84.13 ± 0.30	78.59 ± 0.43	$\mathbf{69.02} \pm 0.51$	95.78 ± 0.33	91.32 ± 0.44	90.13 ± 0.20
MetaPrompter	0.06	$\mathbf{84.62} \pm 0.29$	$\mathbf{79.05} \pm 0.21$	67.12 ± 0.23	$\mathbf{96.34} \pm 0.20$	$\mathbf{92.11} \pm 0.30$	$\mathbf{93.72} \pm 0.18$

(2021), prompts are initialized from input embeddings of randomly sampled label tokens for both MetaPrompting and MetaPrompter.

Results. Table 3 shows the number of parameters and meta-testing accuracy in the 5-shot setting. As can be seen, MetaPrompter is more accurate than both prompt-based and non-prompt-based baselines. Moreover, since MetaPrompter only tunes the prompt pool and keeps the language model frozen, it has much fewer meta-parameters than MetaPrompting and ContrastNet.

Furthermore, MetaPrompting+RepVerb performs better than MetaPrompting+WARP and MetaPrompting+ProtoVerb, demonstrating that the proposed RepVerb is also beneficial to MetaPrompting.

Table 4 shows the number of parameters and meta-testing accuracy in the 5-way 1-shot setting. As can be seen, the state-of-the-art prompt-based methods always achieve higher accuracies than the non-prompt-based ones. Furthermore, MetaPrompter performs the best on 5 of the 6 data sets. Besides, RepVerb is again useful to MetaPrompting on all six data sets.

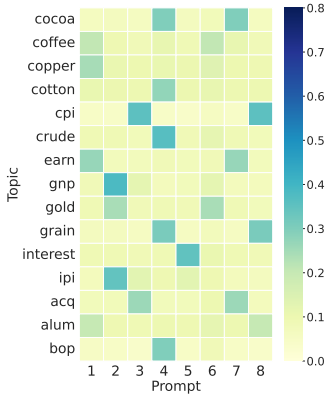


Figure 3. Distribution of attention weights on 5-way 5-shot classification of *Reuters* (15 topics).

4.4. Visualization

In this section, we visualize the meta-knowledge in the prompt pool learned from the 5-way 5-shot classification task on *Reuters*. Table 5 shows the nearest tokens to each of the K ($= 8$) learned prompts. Figure 3 shows the average attention weights between the K prompts and meta-training

Table 5. Nearest tokens to the learned prompts for *Reuters*.

prompt id	nearest tokens
1	copper, steel, trading, gas, fx, aluminum, earn, coffee
2	gross, ship, index, money, gold, tin, iron, retail
3	product, cpi, industrial, acquisitions, jobs, supplying, orange, sugar
4	cocoa, production, grain, livestock, wholesale, cotton, bop, crude
5	oil, national, rubber, nat, interest, price, reserves, regional
6	nat, wholesale, sugar, golden, reserves, drinks, production, product
7	chocolate, sugar, cheat, orange, trade, fx, cash, acquiring
8	aluminum, livestock, cpc, tin, shops, wheat, petrol, supply

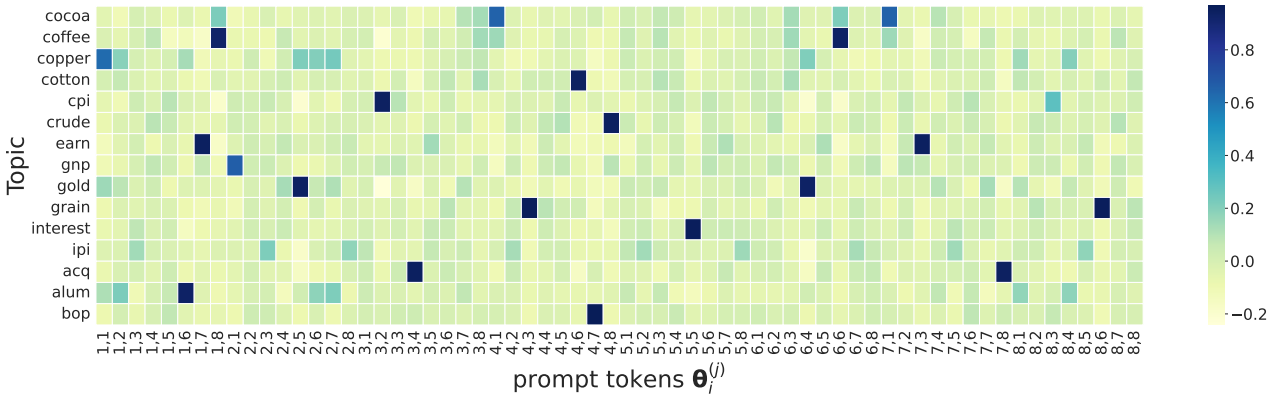


Figure 4. Cosine similarities between learned prompt tokens and topic embeddings on 5-way 5-shot classification of *Reuters*. In the x-axis, (i, j) stands for the jth row of $\boldsymbol{\theta}_i$ (i.e., $\boldsymbol{\theta}_i^{(j)}$)

samples belonging to class (topic) y:

$$\frac{1}{|\mathcal{T}_y|}\sum_{\tau\in\mathcal{T}_y}\frac{1}{|\mathcal{S}_{\tau,y}|}\sum_{(\mathbf{x},y)\in\mathcal{S}_{\tau,y}}\text{softmax}\left(\frac{\mathbf{K}_{T,J}\mathbf{q}_{\mathbf{x}}}{\sqrt{d_o}}\right),$$

where $\mathcal{T}_y$ is the subset of tasks in $\mathcal{T}$ having class y. As can be seen, samples from each target class prefer prompts whose tokens are related to that class. For example, samples from the topic *cocoa* tend to use the 4th and 7th prompts (whose tokens are close to words like *cocoa*, *chocolate* as can be seen from Table 5), while samples from the topic *coffee* tend to use the 1st and 6th prompts (whose tokens are close to words like *coffee* and *sugar*.

Recall that the prompt pool has K learnable prompts $\{(\mathbf{k}_i, \boldsymbol{\theta}_i) : i = 1, \ldots, K\}$, with key $\mathbf{k}_i \in \mathbb{R}^{d_o}$ and value $\boldsymbol{\theta}_i \in \mathbb{R}^{L_p \times d_i}$. Let $\boldsymbol{\theta}_i^{(j)}$ be the jth row of $\boldsymbol{\theta}_i$. Moreover, let $\frac{1}{|\mathcal{V}_y|}\sum_{\text{w}\in\mathcal{V}_y}\mathcal{E}(\text{w})$ be the embedding of topic (class) y, where $\mathcal{V}_y$ is a set of tokens relevant to label y (obtained from Hou et al. (2022)), and $\mathcal{E}(\cdot)$ is the input embedding. Figure 4 shows the cosine similarities between the learned prompt tokens $\{\boldsymbol{\theta}_i^{(j)} : i = 1, \ldots, K, j = 1 \ldots, L_p\}$ and topic embeddings. As can be seen, embedding of *cocoa* is close to $\boldsymbol{\theta}_4^{(1)}$ and $\boldsymbol{\theta}_7^{(1)}$. Thus, samples from *cocoa* prefer the 4th and 7th prompts (Figure 3). Similarly, embedding of *coffee* is close to $\boldsymbol{\theta}_1^{(8)}$ and $\boldsymbol{\theta}_6^{(6)}$. Thus, samples from *coffee* prefer the 1st and 6th prompts (Figure 3).

4.5. Ablation Study

In this section, we perform ablation study using the 5-way 5-shot setting in Section 4.3.

4.5.1. Effect of K

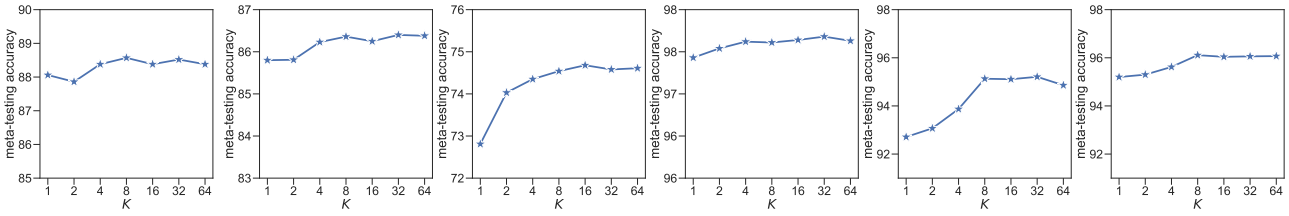
Figure 5 shows the 5-way 5-shot meta-testing accuracy of MetaPrompter with varying K. As K increases, more task knowledge can be extracted and the meta-testing accuracy increases. However, using a very large K (e.g., 64) is unnecessary and the accuracy flattens.

4.5.2. Effect of L_p

Figure 6 shows the 5-way 5-shot meta-testing accuracy of MetaPrompter with varying L_p. As L_p increases, the meta-testing accuracy increases as the expressive power of the prompt pool is enhanced. However, using a very large L_p is again unnecessary and the accuracy flattens.

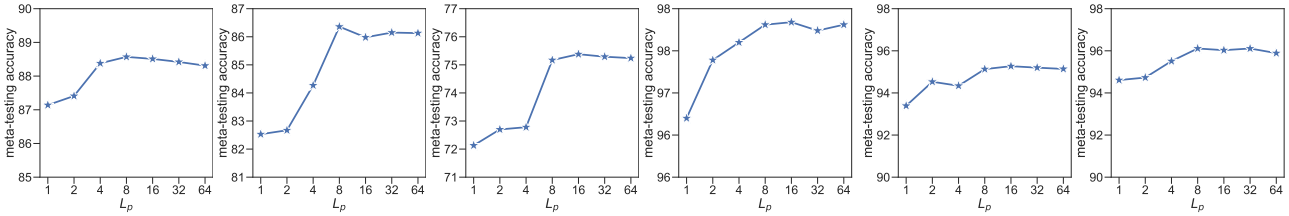
4.5.3. Effect of Verbalizer

Table 6 shows the number of parameters and meta-testing accuracy of MetaPrompter with hand-crafted verbalizer (used in (5)) and RepVerb. As can be seen, RepVerb is better than the hand-crafted verbalizer, and combining both yields the best result.



(a) *20News*. (b) *Amazon*. (c) *HuffPost*. (d) *Reuters*. (e) *HWU64*. (f) *Liu54*.

Figure 5. Effect of K (in log-scale) on 5-way 5-shot classification ($L_p = 8$).



(a) *20News*. (b) *Amazon*. (c) *HuffPost*. (d) *Reuters*. (e) *HWU64*. (f) *Liu54*.

Figure 6. Effect of L_p (in log-scale) on 5-way 5-shot classification ($K = 8$).

Table 6. 5-way 5-shot classification meta-testing accuracy of MetaPrompter with different verbalizers.

verbalizer		*20News*	*Amazon*	*HuffPost*	*Reuters*	*HWU64*	*Liu54*
hand-crafted	RepVerb						
✓	✗	85.91	81.96	70.37	95.91	91.89	90.32
✗	✓	87.12	86.05	72.63	96.69	95.25	93.35
✓	✓	**88.57**	**86.36**	**74.89**	**97.63**	**95.30**	**95.47**

Table 7. 5-way 5-shot classification meta-testing accuracy by using BMG to learn the prompt pool.

	#param ($\times 10^6$)	*20News*	*Amazon*	*HuffPost*	*Reuters*	*HWU64*	*Liu54*
MetaPrompting+BMG	109.52	85.71	83.47	73.92	96.27	93.31	93.04
MetaPrompter+BMG	0.06	**87.91**	**86.45**	**74.99**	**98.01**	**95.41**	**94.52**

4.5.4. Integration with Other Meta-learning Algorithms

While the MAML algorithm (Finn et al., 2017) is used in Algorithm 2, other meta-learning algorithms can also be used to learn the prompt pool in MetaPrompter or the meta-initialized prompt in MetaPrompting. In this experiment, we replace MAML with the state-of-the-art BMG (Flennerhag et al., 2022). Table 7 shows the meta-testing accuracy and number of parameters. As can be seen, MetaPrompter+BMG consistently outperforms MetaPrompting+BMG.

5. Conclusion

In this paper, we proposed MetaPrompter, an effective and parameter-efficient algorithm for prompt tuning. It combines structured prompting and a novel verbalizer called RepVerb. A prompt pool structure is used to construct instance-dependent prompts by attention, while RepVerb builds label embedding by averaging feature embeddings of the corresponding training samples. The pool of prompts is meta-learned from the meta-training tasks. Experimental results demonstrate the effectiveness of the proposed MetaPrompter and RepVerb.

One limitation is that MetaPrompter is based on meta-learning, and so requires the availability of a set of meta-training tasks.

Acknowledgements

This work was supported by NSFC key grant 62136005, NSFC general grant 62076118, and Shenzhen fundamental research program JCYJ20210324105000003. This research was supported in part by the Research Grants Council of the Hong Kong Special Administrative Region (Grant 16200021).

References

Bao, Y., Wu, M., Chang, S., and Barzilay, R. Few-shot text classification with distributional signatures. In *International Conference on Learning Representations*, 2020.

Bengio, Y., Bengio, S., and Cloutier, J. Learning a synaptic learning rule. In *International Joint Conference on Neural Networks*, 1991.

Bertinetto, L., Henriques, J. F., Torr, P., and Vedaldi, A. Meta-learning with differentiable closed-form solvers. In *International Conference on Learning Representations*, 2018.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., and Amodei, D. Language models are few-shot learners. In *Neural Information Processing Systems*, 2020.

Chen, J., Zhang, R., Mao, Y., and Xu, J. ContrastNet: A contrastive learning framework for few-shot text classification. In *AAAI Conference on Artificial Intelligence*, 2022.

Chen, T., Kornblith, S., Norouzi, M., and Hinton, G. A simple framework for contrastive learning of visual representations. In *International Conference on Machine Learning*, 2020.

Conneau, A. and Lample, G. Cross-lingual language model pretraining. In *Neural Information Processing Systems*, 2019.

Cui, G., Hu, S., Ding, N., Huang, L., and Liu, Z. Prototypical verbalizer for prompt-based few-shot tuning. In *Annual Meeting of the Association for Computational Linguistics*, 2022.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. BERT: Pre-training of deep bidirectional transformers for language understanding. In *North American Chapter of the Association for Computational Linguistics*, 2019.

Ding, N., Hu, S., Zhao, W., Chen, Y., Liu, Z., Zheng, H.-T., and Sun, M. OpenPrompt: An open-source framework for prompt-learning. In *Annual Meeting of the Association for Computational Linguistics*, 2022.

Dong, L., Yang, N., Wang, W., Wei, F., Liu, X., Wang, Y., Gao, J., Zhou, M., and Hon, H.-W. Unified language model pre-training for natural language understanding and generation. In *Neural Information Processing Systems*, 2019.

Finn, C., Abbeel, P., and Levine, S. Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning*, 2017.

Flennerhag, S., Schroecker, Y., Zahavy, T., van Hasselt, H., Silver, D., and Singh, S. Bootstrapped meta-learning. In *International Conference on Learning Representations*, 2022.

Gao, T., Han, X., Liu, Z., and Sun, M. Hybrid attention-based prototypical networks for noisy few-shot relation classification. In *AAAI Conference on Artificial Intelligence*, 2019.

Gao, T., Fisch, A., and Chen, D. Making pre-trained language models better few-shot learners. In *Annual Meeting of the Association for Computational Linguistics*, 2021.

Guo, J., Xu, L., and Chen, E. Jointly masked sequence-to-sequence model for non-autoregressive neural machine translation. In *Annual Meeting of the Association for Computational Linguistics*, 2020.

Hambardzumyan, K., Khachatrian, H., and May, J. WARP: Word-level adversarial reprogramming. In *Annual Meeting of the Association for Computational Linguistics*, 2021.

Han, C., Fan, Z., Zhang, D., Qiu, M., Gao, M., and Zhou, A. Meta-learning adversarial domain adaptation network for few-shot text classification. In *Annual Meeting of the Association for Computational Linguistics*, 2021.

He, R. and McAuley, J. Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In *International Conference on World Wide Web*, 2016.

He, Y., Zheng, S., Tay, Y., Gupta, J., Du, Y., Aribandi, V., Zhao, Z., Li, Y., Chen, Z., and Metzler, D. HyperPrompt: Prompt-based task-conditioning of transformers. In *International Conference on Machine Learning*, 2022.

Hou, Y., Dong, H., Wang, X., Li, B., and Che, W. MetaPrompting: Learning to learn better prompts. In *International Conference on Computational Linguistics*, 2022.

Houlsby, N., Giurgiu, A., Jastrzebski, S., Morrone, B., De Laroussilhe, Q., Gesmundo, A., Attariyan, M., and Gelly, S. Parameter-efficient transfer learning for NLP. In *International Conference on Machine Learning*, 2019.

Howard, J. and Ruder, S. Universal language model fine-tuning for text classification. In *Annual Meeting of the Association for Computational Linguistics*, 2018.

Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., and Chen, W. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022a.

Hu, S., Ding, N., Wang, H., Liu, Z., Wang, J., Li, J., Wu, W., and Sun, M. Knowledgeable prompt-tuning: Incorporating knowledge into prompt verbalizer for text classification. In *Annual Meeting of the Association for Computational Linguistics*, 2022b.

Jiang, W., Kwok, J., and Zhang, Y. Effective meta-regularization by kernelized proximal regularization. In *Neural Information Processing Systems*, 2021.

Jiang, W., Kwok, J., and Zhang, Y. Subspace learning for effective meta-learning. In *International Conference on Machine Learning*, 2022.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015.

Lang, K. NewsWeeder: Learning to filter netnews. In *International Conference on Machine Learning*, 1995.

Lee, K., Maji, S., Ravichandran, A., and Soatto, S. Meta-learning with differentiable convex optimization. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

Lester, B., Al-Rfou, R., and Constant, N. The power of scale for parameter-efficient prompt tuning. In *Empirical Methods in Natural Language Processing*, 2021.

Lewis, D. Reuters-21578 text categorization test collection. *Distribution 1.0, AT&T Labs-Research*, 1997.

Li, J., Tang, T., Nie, J.-Y., Wen, J.-R., and Zhao, X. Learning to transfer prompts for text generation. In *North American Chapter of the Association for Computational Linguistics*, 2022.

Li, X. L. and Liang, P. Prefix-tuning: Optimizing continuous prompts for generation. In *Annual Meeting of the Association for Computational Linguistics*, 2021.

Lin, Z., Madotto, A., and Fung, P. Exploring versatile generative language model via parameter-efficient transfer learning. In *Empirical Methods in Natural Language Processing*, 2020.

Liu, J., Shen, D., Zhang, Y., Dolan, W. B., Carin, L., and Chen, W. What makes good in-context examples for GPT-3? In *Proceedings of Deep Learning Inside Out*, 2022a.

Liu, X., Eshghi, A., Swietojanski, P., and Rieser, V. Benchmarking natural language understanding services for building conversational agents. In *International Workshop on Spoken Dialogue Systems Technology*, 2019.

Liu, X., Zheng, Y., Du, Z., Ding, M., Qian, Y., Yang, Z., and Tang, J. GPT understands, too. Preprint arXiv:2103.10385, 2021.

Liu, X., Ji, K., Fu, Y., Tam, W., Du, Z., Yang, Z., and Tang, J. P-Tuning: Prompt tuning can be comparable to fine-tuning across scales and tasks. In *Annual Meeting of the Association for Computational Linguistics*, 2022b.

Misra, R. News category dataset. Preprint arXiv:2209.11429, 2022.

Munkhdalai, T. and Yu, H. Meta networks. In *International Conference on Machine Learning*, 2017.

Nichol, A., Achiam, J., and Schulman, J. On first-order meta-learning algorithms. Preprint arXiv:1803.02999, 2018.

Oreshkin, B., López, P. R., and Lacoste, A. TADAM: Task dependent adaptive metric for improved few-shot learning. In *Neural Information Processing Systems*, 2018.

Prasad, A., Hase, P., Zhou, X., and Bansal, M. GrIPS: Gradient-free, edit-based instruction search for prompting large language models. Preprint arXiv:2203.07281, 2022.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., and Sutskever, I. Language models are unsupervised multitask learners. OpenAI Blog, 2019.

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., and Liu, P. J. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 2020.

Raghu, A., Raghu, M., Bengio, S., and Vinyals, O. Rapid learning or feature reuse? Towards understanding the effectiveness of MAML. In *International Conference on Learning Representations*, 2020.

Rajeswaran, A., Finn, C., Kakade, S. M., and Levine, S. Meta-learning with implicit gradients. In *Neural Information Processing Systems*, 2019.

Sanh, V., Webson, A., Raffel, C., Bach, S., Sutawika, L., Alyafeai, Z., Chaffin, A., Stiegler, A., Raja, A., Dey, M., Bari, M. S., Xu, C., Thakker, U., Sharma, S. S., Szczechla, E., Kim, T., Chhablani, G., Nayak, N., Datta, D., Chang, J., Jiang, M. T.-J., Wang, H., Manica, M., Shen, S., Yong, Z. X., Pandey, H., Bawden, R., Wang, T., Neeraj, T., Rozen, J., Sharma, A., Santilli, A., Fevry, T.,

Fries, J. A., Teehan, R., Scao, T. L., Biderman, S., Gao, L., Wolf, T., and Rush, A. M. Multitask prompted training enables zero-shot task generalization. In *International Conference on Learning Representations*, 2022.

Santoro, A., Bartunov, S., Botvinick, M., Wierstra, D., and Lillicrap, T. Meta-learning with memory-augmented neural networks. In *International Conference on Machine Learning*, 2016.

Schick, T. and Schütze, H. Exploiting cloze-questions for few-shot text classification and natural language inference. In *European Chapter of the Association for Computational Linguistics*, 2021.

Schick, T., Schmid, H., and Schütze, H. Automatically identifying words that can serve as labels for few-shot text classification. In *International Conference on Computational Linguistics*, 2020.

Shin, T., Razeghi, Y., Logan IV, R. L., Wallace, E., and Singh, S. AutoPrompt: Eliciting knowledge from language models with automatically generated prompts. In *Empirical Methods in Natural Language Processing*, 2020.

Snell, J., Swersky, K., and Zemel, R. Prototypical networks for few-shot learning. In *Neural Information Processing Systems*, 2017.

Song, K., Tan, X., Qin, T., Lu, J., and Liu, T.-Y. MPNet: Masked and permuted pre-training for language understanding. In *Neural Information Processing Systems*, 2020.

Sun, T., He, Z., Qian, H., Zhou, Y., Huang, X., and Qiu, X. BBTv2: Towards a gradient-free future with large language models. In *Empirical Methods in Natural Language Processing*, 2022.

Thrun, S. and Pratt, L. Learning to learn: Introduction and overview. In *Learning to Learn*. 1998.

Tian, Y., Sun, C., Poole, B., Krishnan, D., Schmid, C., and Isola, P. What makes for good views for contrastive learning? In *Neural Information Processing Systems*, 2020.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. Attention is all you need. In *Neural Information Processing Systems*, 2017.

Vinyals, O., Blundell, C., Lillicrap, T., and Wierstra, D. Matching networks for one shot learning. In *Neural Information Processing Systems*, 2016.

Vu, T., Lester, B., Constant, N., Al-Rfou, R., and Cer, D. SPoT: Better frozen model adaptation through soft prompt transfer. In *Annual Meeting of the Association for Computational Linguistics*, 2022.

Wang, Z., Zhang, Z., Ebrahimi, S., Sun, R., Zhang, H., Lee, C.-Y., Ren, X., Su, G., Perot, V., Dy, J., and Pfister, T. DualPrompt: Complementary prompting for rehearsal-free continual learning. In *European Conference on Computer Vision*, 2022a.

Wang, Z., Zhang, Z., Lee, C.-Y., Zhang, H., Sun, R., Ren, X., Su, G., Perot, V., Dy, J., and Pfister, T. Learning to prompt for continual learning. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2022b.

Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., and Brew, J. HuggingFace's transformers: State-of-the-art natural language processing. In *Empirical Methods in Natural Language Processing*, 2020.

Yang, Z., Dai, Z., Yang, Y., Carbonell, J., Salakhutdinov, R. R., and Le, Q. V. XLNet: Generalized autoregressive pretraining for language understanding. In *Neural Information Processing Systems*, 2019.

Ye, F., Lin, B., Yue, Z., Guo, P., Xiao, Q., and Zhang, Y. Multi-objective meta learning. In *Neural Information Processing Systems*, 2021.

Zhang, N., Li, L., Chen, X., Deng, S., Bi, Z., Tan, C., Huang, F., and Chen, H. Differentiable prompt makes pre-trained language models better few-shot learners. In *International Conference on Learning Representations*, 2022.

A. Implementation of WARP

WARP (Hambardzumyan et al., 2021) is developed for supervised learning with limited samples. The proposed RepVerb (Algorithm 1) is also designed for the supervised learning setting. In the meta-learning procedure in Algorithm 2, it is used in the inner level (steps 4-15) which is also supervised.

Given a meta-testing task $\tau' = (\mathcal{S}_{\tau'}, \mathcal{Q}_{\tau'})$ with label set $\mathcal{Y}_{\tau'}$, let $\mathbf{V} \equiv \{\mathbf{v}_y : y \in \mathcal{Y}_{\tau'}\}$ be τ''s learnable label embeddings, and $\boldsymbol{\phi}$ be the MLM parameter. For an input $\mathbf{x}$, the distribution for labels $y \in \mathcal{Y}_{\tau'}$ is predicted as:

$$\mathbb{P}(y|\mathbf{x}; \boldsymbol{\phi}, \mathbf{V}) = \frac{\exp(\mathbf{v}_y^\top \mathbf{h}_{\texttt{[MASK]}}(\tilde{\mathbf{x}}))}{\sum_{y' \in \mathcal{Y}_\tau} \exp(\mathbf{v}_{y'}^\top \mathbf{h}_{\texttt{[MASK]}}(\tilde{\mathbf{x}}))}, \tag{6}$$

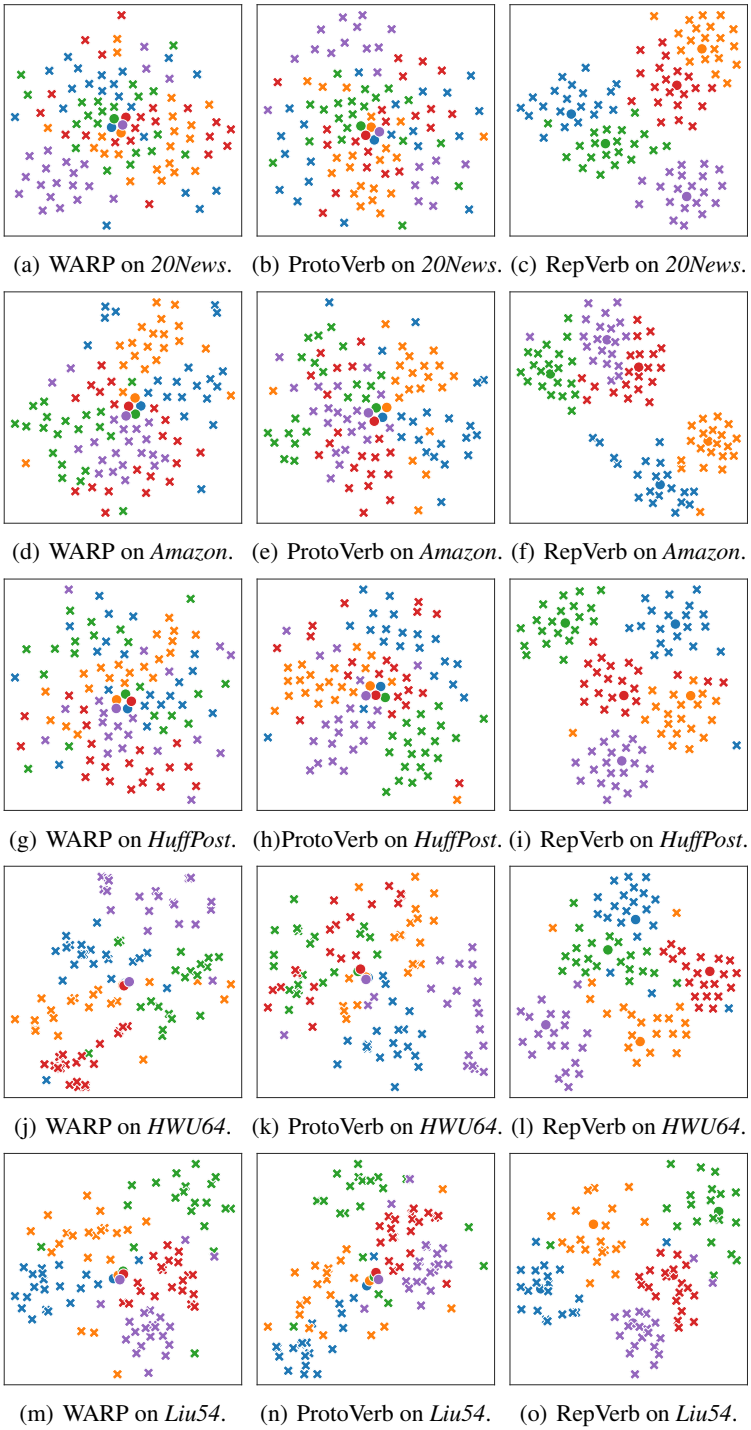
where $\mathbf{h}_{\texttt{[MASK]}}(\tilde{\mathbf{x}})$ is the `[MASK]`'s embedding of wrapped input $\tilde{\mathbf{x}}$. $(\boldsymbol{\phi}, \mathbf{V})$ is learned by performing $T = 5$ gradient updates to minimize the negative log-likelihood loss on the support set $\mathcal{S}_{\tau'}$:

$$\sum_{(\mathbf{x},y) \in \mathcal{S}_{\tau'}} -\log \mathbb{P}(y|\mathbf{x}; \boldsymbol{\phi}, \mathbf{V}).$$

$\boldsymbol{\phi}$ is initialized by the pre-trained MLM, while $\mathbf{V}$ is initialized randomly. The learned $(\boldsymbol{\phi}, \mathbf{V})$ is then evaluated on $\mathcal{Q}_{\tau'}$. For a test sample $(\mathbf{x}^\star, \cdot) \in \mathcal{Q}_{\tau'}$, its prediction is given by (6). We run the WARP algorithm on all meta-testing tasks and report the average meta-testing accuracy in Table 2.

B. Visualization for Verbalizers

Figure 7 shows the t-SNE visualization of the embeddings ($\mathbf{h}_{\texttt{[MASK]}}(\mathbf{x})$'s) of 100 samples ($\mathbf{x}$'s) and learned label embeddings ($\mathbf{v}_y$'s) of a 5-way 5-shot task randomly from *20News*, *Amazon*, *HuffPost*, *HWU64*, and *Liu54*. As shown, the RepVerb embedding is more discriminative and compact than WARP and ProtoVerb. Furthermore, RepVerb's label embedding is consistent with the samples' feature embeddings, while those of WARP and ProtoVerb are not.



(a) WARP on *20News*. (b) ProtoVerb on *20News*. (c) RepVerb on *20News*.

(d) WARP on *Amazon*. (e) ProtoVerb on *Amazon*. (f) RepVerb on *Amazon*.

(g) WARP on *HuffPost*. (h)ProtoVerb on *HuffPost*. (i) RepVerb on *HuffPost*.

(j) WARP on *HWU64*. (k) ProtoVerb on *HWU64*. (l) RepVerb on *HWU64*.

(m) WARP on *Liu54*. (n) ProtoVerb on *Liu54*. (o) RepVerb on *Liu54*.

Figure 7. t-SNE visualization of `[MASK]`'s embeddings (crosses) and label embeddings (circles) for a 5-way 5-shot task randomly sampled from *20News*, *Amazon*, *HuffPost*, *HWU64*, and *Liu54*.