Multi-Prompting Decoder Helps Better Language Understanding

Zifeng Cheng[*], **Zhaoling Chen**[*], **Zhiwei Jiang**[†], **Yafeng Yin**,
Cong Wang, Shiping Ge, Qing Gu
National Key Laboratory for Novel Software Technology, Nanjing University
{chengzf,zhaolingchen}@smail.nju.edu.cn {jzw,yafeng}@nju.edu.cn
{cw,shipingge}@smail.nju.edu.cn guq@nju.edu.cn

Abstract

Recent large Pre-trained Language Models (PLMs) usually only provide users with the inference APIs, namely the emerging Model-as-a-Service (MaaS) setting. To adapt MaaS PLMs to downstream tasks without accessing their parameters and gradients, some existing methods focus on the output-side adaptation of PLMs, viewing the PLM as an encoder and then optimizing a task-specific decoder for decoding the output hidden states and class scores of the PLM. Despite the effectiveness of these methods, they only use a single prompt to query PLMs for decoding, leading to a heavy reliance on the quality of the adopted prompt. In this paper, we propose a simple yet effective Multi-Prompting Decoder (MPD) framework for MaaS adaptation. The core idea is to query PLMs with multiple different prompts for each sample, thereby obtaining multiple output hidden states and class scores from PLMs for subsequent decoding. Such multi-prompting decoding paradigm can simultaneously mitigate reliance on the quality of a single prompt, alleviate the issue of data scarcity under the few-shot setting, and provide richer knowledge extracted from PLMs. Specifically, we propose two decoding strategies: multi-prompting decoding with optimal transport for hidden states and calibrated decoding for class scores. Extensive experiments demonstrate that our method is effective on multiple natural language understanding datasets under the few-shot setting.

1 Introduction

Pre-trained Language Models (PLMs) have recently achieved remarkable performance on various downstream language understanding tasks with the "pre-training and fine-tuning" paradigm (Liu et al., 2023). However, as the scale of PLMs continues to grow, fine-tuning the full model for each downstream task has become computationally expensive and deployment-inefficient. In light of this, Model-as-a-Service (MaaS) (Diao et al., 2022; Sun et al., 2022b) has emerged as a more practical model deployment scheme, allowing pre-trained models to serve downstream tasks by providing inference APIs. Within the MaaS scheme, users can access the outputs of PLMs (e.g., *hidden states*, *class scores*, or *textual output*) through APIs[1], but cannot access the parameters and gradients of PLMs, presenting a challenge in effectively adapting PLMs to downstream tasks.

To adapt MaaS PLMs to downstream tasks, existing methods have primarily explored both the input-side and the output-side adaptation. On the input-side adaptation, researchers focus on utilizing gradient-free methods to heuristically search a good continuous (Sun et al., 2022b,a) or discrete prompt (Diao et al., 2022; Deng et al., 2022) for the target task, such as using evolutionary algorithm and reinforcement learning. However, input-side methods need to query PLM thousands of times for optimization, resulting in significant time overhead and the optimization process is very difficult due to the huge search space and the lack of gradients. On the output-side adaptation, researchers (Hou et al., 2023; Cui et al., 2023) treat the PLM as an encoder and locally train a task-specific decoder to post-process PLM's outputs. Compared to the input-side adaptation, the output-side adaptation methods only query PLM a few times and can benefit from gradient for optimization, often resulting in better performance.

Despite the effectiveness of these output-side adaptation methods, they usually only adopt a single prompt to query PLMs for a certain downstream task, thus often struggling with the performance

[*] Both authors contributed equally to this research.
[†] Corresponding author.

[1] Currently, some famous MaaS services include GPT-3.5 Turbo, which offers APIs for accessing class scores and text output, as well as text-embedding-3-small and text-embedding-3-large, which provide APIs for accessing embedding representations (i.e., *hidden states*) of input text.

Findings of the Association for Computational Linguistics: ACL 2025, pages 192–208

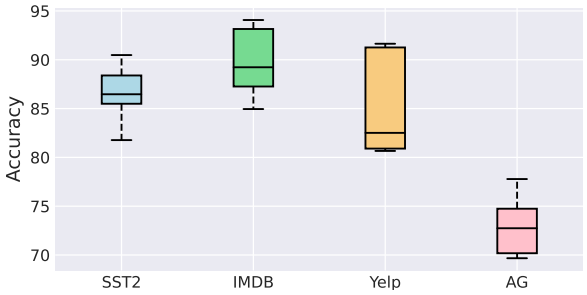

Figure 1: Performance and variation using five different prompts on four datasets.

variations brought by using different prompts (Lu et al., 2022; Chen et al., 2023b). As shown in Figure 1, on all four datasets, the maximum performance fluctuations in accuracy can exceed 8% when using different prompts. Therefore, the quality of prompt is crucial, and relying on only one prompt easily exposes them to performance risks of encountering a low-quality query.

To this end, we propose to query PLMs with multiple prompts for a certain downstream task. This multi-prompting paradigm offers several advantages. Firstly, using multiple prompts helps mitigate reliance on the quality of a single prompt, leading to better stability. Secondly, in the few-shot setting, using multiple prompts enables a single sample to obtain multiple representations, thereby addressing the issue of data scarcity. Lastly, multiple prompts can guide the extraction of knowledge from PLMs from different perspectives, providing richer knowledge for downstream tasks.

Based on the above analysis, in this paper, we propose a novel output-side PLMs adaptation framework for few-shot classification, named Multi-Prompting Decoder (MPD). This MPD framework can decode the outputs of PLMs, including hidden states and class scores, to yield classification results. In contrast to previous methods, MPD adopts a novel multi-prompting decoding paradigm, which queries PLMs with multiple prompts for a sample, thereby obtaining more representation information for decoding. Specifically, we propose two decoding strategies: multi-prompting decoding with optimal transport for hidden states and calibrated multi-prompting decoding for class scores. Extensive experiments demonstrate that our method is effective under the few-shot setting on multiple natural language understanding tasks, including sentiment analysis, topic classification, and natural language inference.

2 Related Work

Prompt Tuning Prompt tuning aims to reduce the gap between PLM pre-training and fine-tuning via predicting `[MASK]` token in the prompt for data-efficiency and achieves great performance in zero-shot and few-shot learning (Liu et al., 2023). Since prompt has a large impact on the performance of prompt tuning, many studies have focused on finding a good prompt, including discrete prompt (Jiang et al., 2020; Shin et al., 2020; Gao et al., 2021), continuous prompt (Liu et al., 2021b; Qin and Eisner, 2021; Li and Liang, 2021; Lester et al., 2021; Zhang et al., 2022), prompt selection (Sorensen et al., 2022; Chen et al., 2023b), and so on.

MaaS Adaptation Some PLMs, such as GPT-3 (Brown et al., 2020), are released as a service in the cloud. Some works try to adapt these models to downstream tasks without accessing the model parameters and the gradients (Sun et al., 2022b; Diao et al., 2022).

Some work about MaaS uses reinforcement learning (RL) (Diao et al., 2022; Deng et al., 2022) or derivative-free optimization (Sun et al., 2022b,a) to find the optimal prompt. For the first type of method, Diao et al. (2022) first use a variance-reduced policy gradient algorithm to estimate the gradient of the prompt token distribution. Deng et al. (2022) use a policy network and reward function to generate and optimize discrete prompts. For the second type of method, BBT (Sun et al., 2022b) adopts a covariance matrix adaptation evolution algorithm to optimize continuous prompt. BBTv2 (Sun et al., 2022a) further prepends continuous prompts to every layer of PLM and proposes a divide-and-conquer gradient-free algorithm to optimize them. GDFO (Han et al., 2023) uses a prompt generator to generate an extra vector as input based on the BBT framework. TEMPERA (Zhang et al., 2023) constructs query-dependent prompts through test-time prompt editing and formulates this as an RL problem. DP_2O (Li et al., 2023) first designs a prompt generation strategy through multi-round dialogue alignment on GPT-4 and prompt evaluation metric SUE. Then, DP_2O constructs RL framework based on policy gradients to match the prompts to inputs optimally.

However, searching in a large space of prompts is inefficient and difficult to optimize. Inspired by this, PromptBoosting (Hou et al., 2023) constructs a series of weak learners, learns the corresponding

verbalizer, and uses AdaBoost algorithm to ensemble them. DecT (Cui et al., 2023) is an output-side adaptation framework and optimizes class prototypes as hyperspheres with a radius parameter.

It's worth noting that parameter-efficient tuning (Houlsby et al., 2019; Hu et al., 2022; Zaken et al., 2022; He et al., 2022) requires gradients of PLMs to update the small portion of parameters, which are unavailable in the MaaS setting.

Optimal Transport OT theory (Monge, 1781) was originally used to study the movement of many items from one place to another with minimal cost and provides a distance measure method between two distributions in a closed way. Due to the excellent properties of the distance measure, OT is used in many areas, such as topic model (Zhao et al., 2021a), vision-language models (Chen et al., 2023a), and image classification (Liu et al., 2021a).

3 Preliminaries

MaaS adaptation aims to correctly predict the label $y \in \{1, \ldots, N\}$ for test sample x based on a given few-shot training set $\mathcal{D}_{\text{train}} = \{(x_i, y_i)\}_{i=1}^{NK}$ and PLM $\mathcal{M}$, where N is the number of classes and each class has K training samples. In the MaaS setting, PLM $\mathcal{M}$ is a black-box inference API without accessing the model parameters and the gradients. Therefore, we can only query the model with input x and get corresponding outputs, including hidden states and class scores.

To better utilize the PLM, we use prompt learning to wrap input samples into prompts using templates. Specifically, we first use a template $\mathcal{T}$ with a `[MASK]` token to enclose each input. Then, we can query $\mathcal{M}$ with prompt $\mathcal{T}(x)$ to get the hidden states and class scores. Hidden states can be obtained by the representation of `[MASK]` token of the final layer and class scores $\mathbf{s} \in \mathbb{R}^N$ can be obtained by using the verbalizer to map predicted tokens at the `[MASK]` token to classes. Take sentiment analysis as an example, we can use

$$\mathcal{T}(x) = x \text{ In summary, it was } \texttt{[MASK]}.$$

as prompt with verbalizer that maps "bad" and "great" as label words for negative and positive sentiment.

4 Methodology

We introduce a novel output-side MaaS adaptation framework, Multi-Prompting Decoder (MPD), for few-shot classification. This framework can decode the outputs of PLMs, including hidden states and class scores, to yield classification results. In contrast to previous decoding frameworks that query PLMs only once for a sample, we propose multi-prompting decoding paradigm, which queries PLMs with multiple prompts for a sample, thereby obtaining more representation information for decoding. Specifically, for decoding the PLMs' output hidden states, we design a multi-prompting decoding strategy based on optimal transport. For decoding the PLMs' output class scores, we additionally design a calibrated multi-prompting decoding strategy. These two decoding strategies are jointly used in the final decoding process.

4.1 Multi-Prompting Decoding with Optimal Transport

In our multi-prompting paradigm, each sample has multiple types of hidden states corresponding to multiple prompts. Building a unified classifier for these hidden states neglects the characteristics of each prompt, leading to a crude solution. Building multiple classifiers specific to each type of hidden state is also a sub-optimal solution, which will lead to an ensemble of multiple classifiers, and these classifiers cannot benefit from each other during training.

To fully exploit the advantages of the multi-prompting decoding paradigm, we design a specific multi-prompting decoder for hidden states. The core idea is to establish multiple prototypes for each class to capture the class characteristics specific to different prompts. Subsequently, through optimal transport, we identify the best matching between the sample representations and the prototypes, thereby achieving classification. Considering the limited number of training samples available in the few-shot setting, the decoder is lightweight, comprising only a linear layer and several learnable class prototypes.

4.1.1 Querying with Multiple Prompts

Given a text x_i, we first use a series of templates $\mathcal{T}_1, \cdots, \mathcal{T}_P$ with a `[MASK]` token to enclose the text and then query $\mathcal{M}$ with template-wrapped inputs $\mathcal{T}_1(x_i), \cdots, \mathcal{T}_P(x_i)$ to get a series of initial text representations $\mathbf{h}_{i,1}, \cdots, \mathbf{h}_{i,P}$. Specifically, we take hidden states of the final layer at the `[MASK]` position as the initial text representations extracted by PLM $\mathcal{M}$. Then, we use a linear layer parameterized by $\mathbf{W}$ to project the initial representations to

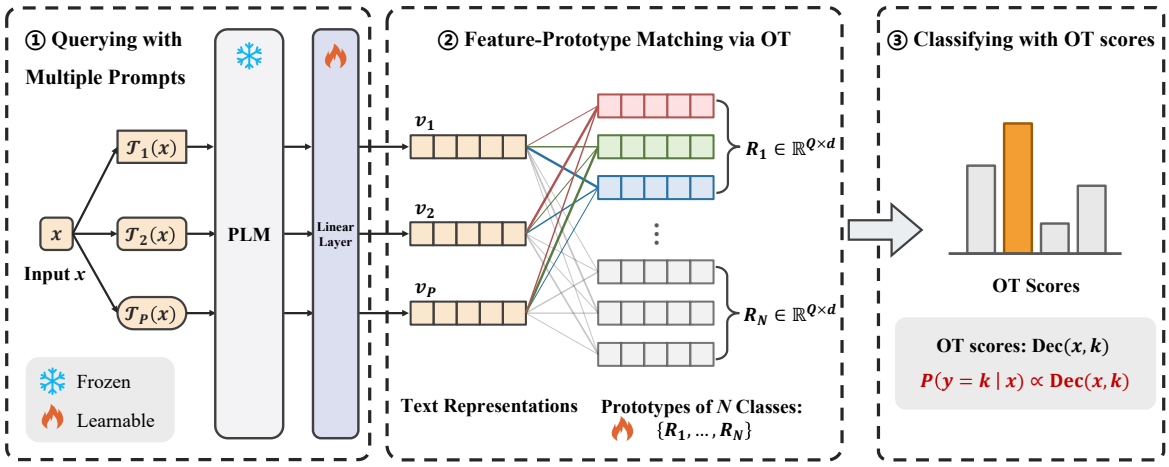

Figure 2: Illustration of the multi-prompting decoding with optimal transport.

get text representations for classification:

$$\mathbf{v} = \mathbf{W}\mathbf{h} \tag{1}$$

It is worth noting that the text representation matrix of text x_i is $\mathbf{V}_i \in \mathbb{R}^{P\times d}$.

4.1.2 Features-Prototypes Matching via OT

Since our framework contains multiple text representations and class prototypes, OT is a proper solution to better map multiple text representations and multiple prototypes of each class. OT aims to find the optimal transport plan $\boldsymbol{T} \in \mathbb{R}^{P\times Q}$ that represents the fine-grained matching flow between text representations and class prototypes, where P denotes the number of prompts and Q denotes the number of prototypes. The optimal transport plan assigns a higher weight to the pair of text representation and prototype that are close to each other. We use Sinkhorn's algorithm (Cuturi, 2013) to optimize the transport plan. Specifically, we fix text representations and class prototypes to optimize the transport plan. The details of OT and Sinkhorn's algorithm are shown in Appendix A.

4.1.3 Classifying with OT Scores

After getting the optimal transport plan, the optimal transport score between text x_i and class k can be computed according to the optimal transport plan and similarity matrix. Specifically,

$$\text{Dec}(x_i, k) = \sum_{m=1}^{P}\sum_{n=1}^{Q} \boldsymbol{T}_{m,n}^{i,k}\text{sim}(\mathbf{V}_{i,m}, \mathbf{R}_{k,n}) \tag{2}$$

where Dec(,) denotes the optimal transport score between text x_i and class k, $\boldsymbol{T}_{m,n}^{i,k}$ denotes the (m,n)-th element in the optimal transport plan between text x_i and class k, sim(,) denotes cosine similarity, $\mathbf{V}_{i,m}$ is the text representations of $\mathcal{T}_m(x_i)$, and $\mathbf{R}_{k,n}$ is the n-th prototype representation of the k-th class.

Finally, we use the softmax function over optimal transport scores between text representations and all classes to get the predicted probability distribution, and cross-entropy loss to optimize the parameters. Specifically,

$$\mathcal{L} = \sum_{i=1}^{NK} \frac{-1}{NK}\log\frac{\exp(\text{Dec}(x_i, y_i))}{\sum_{j=1}^{N}\exp(\text{Dec}(x_i, j))} \tag{3}$$

The whole training flow is shown in Algorithm 1.

4.2 Calibrated Multi-Prompting Decoding

We also decode the PLMs' output class scores as supplementary for the hidden states decoding to provide additional prior knowledge of the PLMs. Specifically, we first expand the set of label words for the verbalizer, then calibrate the class scores for each prompt separately, and average the calibrated class scores of all prompts. Finally, we combine OT scores and calibrated class scores for joint decoding.

4.2.1 Label Words Expansion

We first use a simple and effective way to expand the label words. Given the prediction layer of MLM $\mathbf{W}_{mlm} \in \mathbb{R}^{|\mathcal{V}|*d}$, we can treat each item as a representation of a word. Then, given a label word, we expand a new set of label words S based on cosine similarity between words and use the softmax function over cosine similarity to assign a weight to each element in the set. It is worth noting that due to the large number of open-sourced PLMs, getting an expanded set of label words is easy.

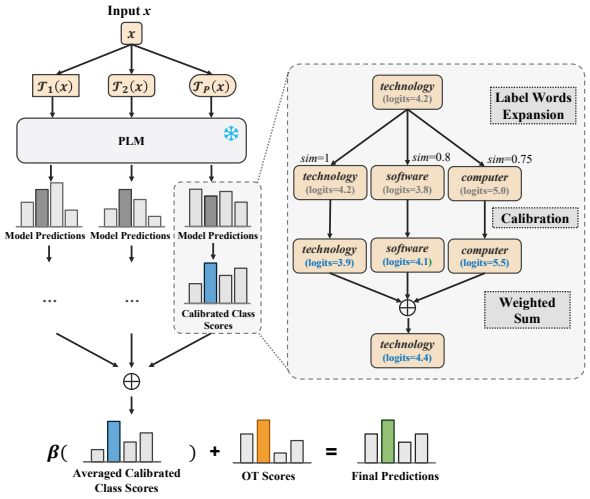

Figure 3: Illustration of the calibrated multi-prompting decoding.

4.2.2 Calibrating Class Scores

Predictions of PLM are biased because PLM tends to predict tokens that are common in its pre-training distribution (Zhao et al., 2021b). Thus, we first calibrate PLM on a given prompt and then use calibrated class scores for classification.

Specifically, we use a template to wrap to empty input x_e ="" and query the model with $\mathcal{T}_j(x_e)$ to obtain the predictions of label words $\mathbf{s}_{ej} \in \mathbb{R}^{N|S|}$ to calibrate. Then, given a prediction $\mathbf{s}_{ij}$ for sample x_i wrapped by prompt $\mathcal{T}_j$, we can calibrate $\mathbf{s}_{ij}$ by

$$\bar{\mathbf{s}}_{ij} = \text{diag}(\mathbf{s}_{ej}/\text{mean}(\mathbf{s}_{ej}))^{-1}\mathbf{s}_{ij} \quad (4)$$

where $\bar{\mathbf{s}}_{ij} \in \mathbb{R}^{N|S|}$ is calibrated prediction over the expanded set. We can see that the calibrated predictions for context-independent inputs such as empty input become uniformly distributed.

Then, we use the weight vector for each class to aggregate the predictions of each element in the corresponding set to form the calibrated class scores $\hat{\mathbf{s}}_{ij} \in \mathbb{R}^N$. Finally, we average the class scores of multiple prompts to get averaged calibrated class scores $\hat{\mathbf{s}}_i$ for classification.

$$\hat{\mathbf{s}}_i = \frac{\sum_{j=1}^{P} \hat{\mathbf{s}}_{ij}}{P} \quad (5)$$

4.2.3 Joint Decoding

Finally, we sum OT scores and averaged calibrated class scores to jointly decode the predictions.

$$\hat{y}_i = \hat{y}_i^{OT} + \beta\hat{\mathbf{s}}_i \quad (6)$$

where $\hat{y}_i$ is the final predictions, $\hat{y}_i^{OT}$ = [Dec(x_i,1), $\cdots$, Dec(x_i,N)] $\in \mathbb{R}^N$ denotes OT scores, and β is a hyper-parameter to balance model and prior knowledge inside PLM.

5 Experiments

5.1 Datasets

We conduct experiments on several common natural language understanding tasks including sentiment analysis, topic classification, and natural language inference (NLI). For sentiment analysis, we choose SST2 (Socher et al., 2013), IMDB (Maas et al., 2011), and Yelp (Zhang et al., 2015). For topic classification, we choose AG's News (Zhang et al., 2015), DBPedia (Zhang et al., 2015), and Yahoo (Zhang et al., 2015). For NLI, we choose RTE (Dagan et al., 2005), SNLI (Bowman et al., 2015), and MNLI (Williams et al., 2018). The statistics of datasets are shown in Table 5.

We also randomly sample N = 1, 4, 16 samples for each class from the original training set to construct the few-shot training set. Following Cui et al. (2023) and Hou et al. (2023), we use the original validation set for evaluation on the GLUE (i.e., SST, RTE, and MNLI) and SNLI datasets.

5.2 Baselines

We compare with some strong MaaS methods. **In-context learning (ICL)** (Brown et al., 2020) concatenates some text-label pairs before the test samples. **BBT** (Sun et al., 2022b) optimizes soft prompt tokens with a derivative-free evolutionary algorithm. **BBTv2** (Sun et al., 2022a) further prepends continuous prompts to every layer and uses a divide-and-conquer algorithm to optimize them. **RLPrompt** (Deng et al., 2022) optimizes discrete prompts with RL and adopts soft Q-learning to find the optimal prompt. **PromptBoosting** (Hou et al., 2023) sequentially learns multiple weak learners and adopts AdaBoost to ensemble them. **DecT** (Cui et al., 2023) extracts text feature and optimizes class prototypes as hyperspheres with an additional radius parameter.

5.3 Implementation Details

For a fair comparison with baselines, we also use RoBERTa$_{\text{LARGE}}$ (Liu et al., 2019) as the backbone model. We set the text representation dimension to 128 and optimize the parameters for 30 epochs with Adam optimizer (Kingma and Ba, 2014). We set the number of templates P to 3 and the number of prototypes Q to 3. We set λ to 0.1 and threshold 0.01 in the Sinkhorn's algorithm. Following Cui et al. (2023), we directly set $\beta = 1/n$ for most datasets based on the intuition that β should decrease as the amount of training data increases. We

N	Method	SST2	IMDB	Yelp	AG	DB	Yahoo	RTE	SNLI	MNLI-m/mm	Avg.
1	**ICL**	$81.5_{3.7}$	$65.6_{11.4}$	$81.1_{10.6}$	$66.7_{4.8}$	$71.7_{2.6}$	$53.2_{6.2}$	$45.0_{4.7}$	$46.1_{5.3}$	$\mathbf{53.6_{0.5}}/53.9_{0.8}$	$61.8_{5.1}$
	BBT	$83.4_{1.3}$	$89.0_{0.1}$	$89.7_{0.1}$	$75.4_{0.8}$	$59.1_{1.7}$	$31.2_{2.7}$	$52.3_{1.4}$	$38.5_{0.8}$	$43.4_{2.5}/42.9_{3.3}$	$60.5_{1.5}$
	BBTv2	$83.3_{2.5}$	$89.0_{0.2}$	$89.9_{0.2}$	$74.3_{3.2}$	$74.2_{5.2}$	$34.0_{3.5}$	$48.2_{5.7}$	$38.6_{4.0}$	$44.2_{3.2}/44.3_{4.5}$	$62.0_{3.2}$
	RLPrompt	$63.5_{6.3}$	$65.0_{6.5}$	$66.3_{6.9}$	$72.5_{4.5}$	$65.6_{5.5}$	$38.1_{5.8}$	$53.8_{5.3}$	$36.5_{3.0}$	$40.3_{2.0}/41.0_{2.1}$	$54.3_{4.8}$
	PromptBoosting	$86.7_{2.6}$	$82.4_{6.1}$	$88.7_{2.5}$	$58.7_{11.8}$	$73.0_{4.8}$	$23.7_{7.0}$	$50.0_{5.9}$	$43.5_{6.1}$	$36.8_{1.6}/36.3_{2.3}$	$58.0_{5.1}$
	DecT	$90.8_{0.2}$	$91.2_{0.3}$	$94.8_{0.1}$	$79.9_{1.1}$	$78.8_{0.9}$	$\mathbf{55.2_{0.8}}$	$56.0_{2.7}$	$\mathbf{47.7_{4.1}}$	$52.2_{2.7}/53.3_{3.0}$	$70.0_{1.6}$
	MPD	$\mathbf{92.3_{0}}$	$\mathbf{94.4_{0.3}}$	$\mathbf{95.6_{1.1}}$	$\mathbf{83.2_{1.5}}$	$\mathbf{84.4_{3.5}}$	$53.6_{0.9}$	$\mathbf{57.6_{1.7}}$	$46.6_{4.1}$	$53.1_{1.5}/\mathbf{54.1_{1.8}}$	$\mathbf{71.5_{1.6}}$
4	**ICL**	$60.3_{9.8}$	$80.4_{6.6}$	$77.4_{14.6}$	$65.1_{5.4}$	$71.7_{6.5}$	$49.9_{9.9}$	$42.7_{3.9}$	$42.1_{3.2}$	$44.7_{5.9}/45.2_{6.0}$	$58.0_{7.2}$
	BBT	$84.5_{1.2}$	$89.8_{0.9}$	$90.2_{0.6}$	$79.0_{2.1}$	$67.7_{3.5}$	$42.9_{0.6}$	$48.4_{4.0}$	$40.5_{1.3}$	$41.2_{1.7}/40.7_{2.0}$	$62.5_{1.8}$
	BBTv2	$86.6_{2.2}$	$89.4_{0.6}$	$90.3_{0.5}$	$79.1_{2.1}$	$89.0_{1.7}$	$46.0_{1.4}$	$46.2_{2.3}$	$40.8_{4.3}$	$44.0_{0.9}/44.8_{1.6}$	$65.6_{1.8}$
	RLPrompt	$80.7_{7.5}$	$75.8_{10.1}$	$78.8_{7.3}$	$76.1_{4.8}$	$76.3_{5.9}$	$45.0_{3.1}$	$53.5_{2.9}$	$36.3_{2.6}$	$44.4_{2.9}/45.5_{3.8}$	$61.2_{5.1}$
	PromptBoosting	$88.9_{2.3}$	$83.0_{5.2}$	$92.3_{2.1}$	$78.2_{6.8}$	$90.1_{0.7}$	$36.4_{5.1}$	$53.5_{5.9}$	$53.4_{3.4}$	$39.8_{4.5}/40.3_{5.7}$	$65.6_{4.2}$
	DecT	$87.6_{1.6}$	$89.6_{0.9}$	$94.8_{0.7}$	$81.9_{2.6}$	$89.1_{0.6}$	$59.9_{2.1}$	$56.7_{2.7}$	$53.2_{2.9}$	$52.2_{2.3}/53.4_{2.4}$	$71.8_{1.9}$
	MPD	$\mathbf{92.6_{0.1}}$	$\mathbf{94.9_{0.2}}$	$\mathbf{96.3_{0.3}}$	$\mathbf{85.9_{1.0}}$	$\mathbf{92.8_{3.0}}$	$\mathbf{62.2_{1.3}}$	$\mathbf{59.2_{3.6}}$	$\mathbf{57.1_{2.4}}$	$\mathbf{54.4_{0.1}}/\mathbf{56.5_{0.6}}$	$\mathbf{75.2_{1.1}}$
16	**ICL**	$71.5_{15.8}$	$80.6_{6.0}$	$73.7_{14.5}$	$64.4_{6.0}$	$71.8_{9.1}$	$52.6_{5.7}$	$43.8_{7.0}$	$42.0_{6.3}$	$51.4_{3.0}/52.1_{3.3}$	$60.0_{7.7}$
	BBT	$89.6_{0.3}$	$89.3_{0.4}$	$91.5_{0.2}$	$81.5_{0.8}$	$87.8_{3.0}$	$48.3_{1.4}$	$52.6_{2.2}$	$46.6_{1.3}$	$40.0_{2.6}/39.9_{2.9}$	$66.7_{1.5}$
	BBTv2	$90.3_{1.7}$	$88.6_{2.1}$	$92.9_{0.6}$	$85.3_{0.5}$	$93.6_{0.7}$	$52.0_{1.4}$	$56.7_{3.3}$	$57.3_{2.3}$	$50.1_{2.4}/51.7_{3.2}$	$71.9_{1.8}$
	RLPrompt	$87.0_{2.6}$	$87.6_{2.4}$	$95.1_{1.0}$	$80.2_{0.7}$	$80.8_{3.3}$	$48.1_{2.2}$	$54.3_{2.8}$	$41.1_{5.0}$	$43.3_{3.9}/44.3_{4.5}$	$66.2_{2.8}$
	PromptBoosting	$87.6_{3.0}$	$86.2_{3.1}$	$94.7_{1.0}$	$85.2_{0.9}$	$95.0_{0.5}$	$46.6_{2.4}$	$60.0_{5.5}$	$61.3_{3.5}$	$52.5_{1.5}/50.4_{5.1}$	$72.0_{2.7}$
	DecT	$91.0_{0.5}$	$91.0_{0.9}$	$95.4_{0.3}$	$86.4_{0.4}$	$94.6_{0.5}$	$64.2_{0.7}$	$59.7_{1.8}$	$60.5_{0.8}$	$55.3_{1.3}/56.8_{1.5}$	$75.5_{0.9}$
	MPD	$\mathbf{91.9_{0.1}}$	$\mathbf{95.1_{0.1}}$	$\mathbf{96.4_{0.1}}$	$\mathbf{87.9_{0.4}}$	$\mathbf{96.7_{0.2}}$	$\mathbf{68.3_{0.3}}$	$\mathbf{61.7_{2.8}}$	$\mathbf{62.4_{1.3}}$	$\mathbf{57.5_{1.1}}/\mathbf{59.7_{1.1}}$	$\mathbf{77.8_{0.8}}$

Table 1: Experiment results for MaaS adaptation methods. The results (i.e., average accuracy and standard deviation (%)) over 5 runs. The best results are in **bold**.

Method	Trainable Param (K)	Query Number	SST2		AG	
			Acc	Training Time (s)	Acc	Training Time (s)
ICL	0	0	71.5	0	64.4	0
BBT	0.5	8,000	89.6	1619	87.8	5228
BBTv2	12	8,000	90.3	1624	85.3	6546
RLPrompt	3100	12,000	87.0	82286	80.2	99947
PromptBoosting	0.4	10	87.6	190	85.2	398
DecT	130	1	91.0	1.4	86.4	2.3
Ours	132	3	91.9	3.5	87.9	6.7

Table 2: Efficiency comparison for MaaS methods under 16-shot setting on two datasets.

set $\beta = 1$ on the MNLI dataset based on the validation set. We expand 10 label words for each class and we do not use label words expansion on the NLI datasets. This is because NLI datasets have a smaller number of potential or related label words that express the class compared to other datasets (i.e., sentiment analysis and topic classification). We give the templates and label words in Table 12.

5.4 Results

Table 1 shows the performance on 9 datasets under the few-shot setting. We can see that our model achieves state-of-the-art performance. Specifically, compared to DecT, our approach achieves 1.5%, 3.4%, and 2.3% average performance improvement under 1,4,16-shot settings. Among all settings, our model performs the best in 27 out of the 30. This shows our model is effective and the improvement is consistent. It is worth noting that our model performs well on both simple (i.e., sentiment analysis and topic classification) and difficult tasks that require semantic understanding (i.e., NLI). This shows that the improvement is task-independent.

By observing other baseline methods, ICL does not achieve performance gains, as the shot number increases. This is due to the length limitation of the PLM. BBTv2 exceeds BBT and RLPrompt. This shows that inserting continuous prompts to every layer of the PLM is effective. PromptBoosting and DecT usually achieve better performance than BBT and BBTv2. This is because these two methods do not need to search prompts in a large space and can benefit from gradients.

In addition, our method also achieves the minimum average standard deviation across different datasets and settings. This shows that our approach is the most robust. As the shot number increases, our model achieves the minimum standard deviation on 1, 6, and 8 datasets. This shows that compared to other methods, our model is more likely to obtain a stable result as the shot number increases.

5.5 Efficiency Comparison

We further compare our model with baselines in terms of efficiency, including the number of trainable parameters, the number of queries to access PLM, and training time. As shown in Table 2, our model is effective and efficient. The efficiency of our approach significantly outperforms all baselines except DecT. Specifically, our model only queries PLM three times per training data,

N	Method	SST2	IMDB	Yelp	AG	DB	Yahoo	RTE	SNLI	MNLI-m/mm	Avg.
64	**Fine-tuning**	$92.5_{1.9}$	$86.3_{3.8}$	$94.5_{1.4}$	$87.4_{0.6}$	$98.2_{0.2}$	$69.0_{0.7}$	$67.7_{3.2}$	$66.6_{6.4}$	$65.6_{2.9}/67.7_{4.0}$	$79.6_{2.5}$
	DecT	$92.4_{0.5}$	$91.3_{0.5}$	$94.9_{0.5}$	$89.2_{0.3}$	$97.0_{0.1}$	$69.3_{0.4}$	$65.7_{1.7}$	$67.2_{1.0}$	$62.0_{1.4}/63.3_{1.3}$	$79.2_{0.8}$
	MPD	$92.0_{0.2}$	$95.1_{0.5}$	$96.6_{0.1}$	$89.6_{0.3}$	$97.4_{0.1}$	$70.6_{0.3}$	$67.6_{1.2}$	$68.2_{0.7}$	$63.4_{0.7}/65.1_{1.1}$	$80.6_{0.5}$
256	**Fine-tuning**	$92.0_{0.9}$	$92.1_{0.2}$	$94.3_{0.3}$	$89.6_{0.3}$	$98.5_{0.2}$	$70.2_{0.4}$	$79.8_{1.0}$	$84.4_{0.4}$	$77.2_{0.2}/78.7_{0.3}$	$85.7_{0.4}$
	DecT	$92.7_{0.2}$	$92.1_{0.1}$	$95.6_{0.1}$	$90.3_{0.1}$	$97.4_{0.1}$	$71.3_{0.1}$	$69.2_{1.0}$	$69.7_{0.4}$	$68.0_{0.3}/69.4_{0.3}$	$81.6_{0.3}$
	MPD	$92.3_{0.2}$	$95.1_{0.1}$	96.7_{0}	$90.6_{0.2}$	$97.6_{0.1}$	$71.4_{0.2}$	$71.6_{1.0}$	$70.3_{0.6}$	$68.9_{0.4}/69.9_{0.7}$	$82.4_{0.4}$

Table 3: Experiment results for more training data. The results over 5 runs. Fine-tuning method requires gradients of the model to update the model.

Model Setting	Average Accuracy		
	1	4	16
Ours	**71.5**	**75.2**	**77.8**
w/o Class Scores	57.8	69.6	76.5
w/o OT Scores	68.0	68.0	68.0
w/o Expansion	70.7	74.8	77.7

Table 4: The ablation study of our model on all datasets. We run each experiment over 5 random seeds.

which is about 3× (2500×) less than PromptBoosting (BBT). In addition, the training time of our method is about 50× (400×) faster than that of PromptBoosting (BBT). Our method uses multiple prompts, so the training time is slightly longer than in DecT. Overall, the training speed of our method is still fast. In addition, our model only includes 132K trainable parameters, significantly smaller than the number of parameters in RoBERTa$_{\text{LARGE}}$ (i.e., about 0.04%). Even though some methods (i.e., BBT, BBTv2, and PromptBoosting) have fewer trainable parameters, our model takes less time to train and achieves better performance.

5.6 Performance on More Training Data

We conduct experiments with a larger training set (i.e., $N = 64$ and 256) to explore the scalability of our model beyond the few-shot setting. We compare our model with DecT and a strong baseline fine-tuning. Fine-tuning requires gradients of PLM to update full parameters of PLM.

As shown in Table 3, firstly, our model and DecT both achieve performance improvement when the shot number increases. This shows that these two methods can still fit larger datasets even with a small number of tunable parameters. In addition, as the shot number increases, the average standard deviation of two models decreases. Secondly, our model also exceeds DecT when N is large. This shows that the improvement is consistent. It is worth noting that the gap between the two methods is narrowing as the shot number increases. Thirdly, our method exceeds fine-tuning in the 64-shot setting and is exceeded by fine-tuning in the 256-shot setting. This is because our method cannot update PLM parameters, it can only update a small portion of parameters. Thus, when the shot number is larger, our model may not be able to learn more knowledge. Finally, similar to DecT, our model does not show significant performance differences compared to fine-tuning in simple tasks (i.e., sentiment analysis and topic classification). However, there are significant differences in harder tasks (i.e., SNLI). This suggests that harder tasks require more parameters to learn corresponding semantics.

6 Model Analysis

6.1 Ablation Study

We conduct ablation study to validate the effectiveness of each component. As shown in Table 4, firstly, all components achieve performance improvement. Secondly, the performance of w/o Class Scores (i.e., only using OT scores) improves rapidly as the shot number increases. This shows that shot number has a significant effect on OT scores. Thirdly, w/o OT Scores (i.e., only averaging calibrated class scores of multiple prompts) also achieves good performance, especially in the 1-shot setting. This shows that averaging the class scores of multiple prompts is also a strong baseline under 1-shot setting. Finally, label words expansion achieves more significant performance gains when the shot number is small. This is because the model relies more on class scores when the shot number is small. The results of label words expansion are shown in Appendix E.

6.2 Effects of Prompts

We explore the effects of prompts by changing the combination of prompts in Figure 4. Specifically, we use different prompts and change the number of

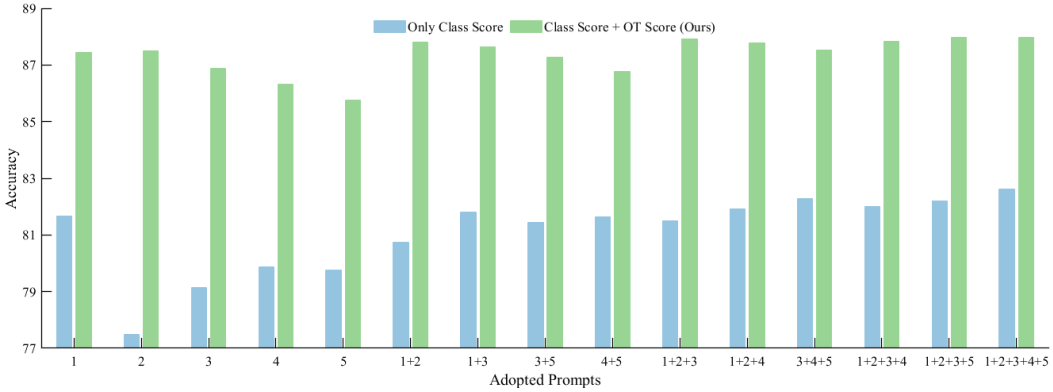

Figure 4: Effects of prompts on AG's news dataset under 16-shot setting. 1, 2, 3, 4, and 5 represent 5 different prompts. 1+2 denotes we use two prompts (i.e., 1 and 2) to train the model. The results over 5 runs.

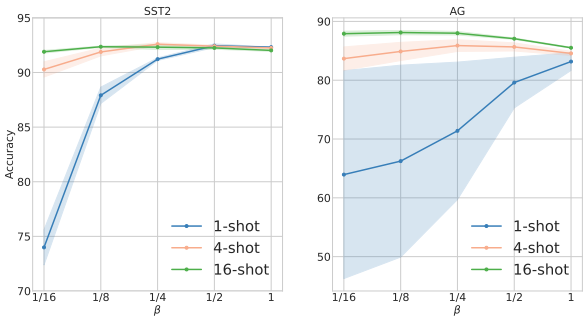

Figure 5: Effects of β over 5 runs on two datasets.

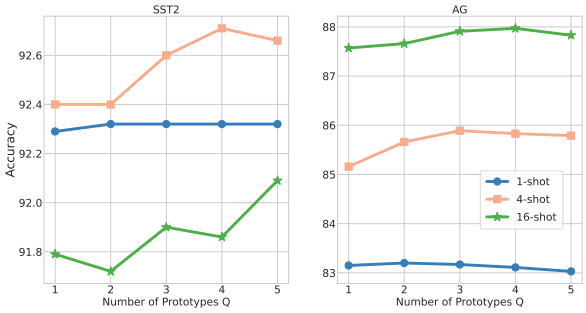

Figure 6: Effects of Q over 5 runs on two datasets.

prompts. The five templates are shown in Table 13.

Firstly, as the prompt number increases, performance will be better and more stable. When the number of prompts exceeds 3, the improvement is marginal. Secondly, there will be less performance fluctuation with multiple prompts compared to using a single prompt Specifically, when we use one/two/three/four prompts, the change in performance is 1.7%/0.3%/0.2%/0.2%. This shows that our method is robust across multiple prompts. Thirdly, the good performance of a prompt's class scores does not mean that using it can achieve better performance. This shows that there is a gap between the representation of hidden states and class scores.

6.3 Effects of β

Hyper-parameter β controls the ratio of OT scores and class scores. We explore the effects of β on two datasets in Figure 5. Firstly, our model needs a bigger β under 1-shot setting. This is because the amount of data is small and the model relies heavily on class scores for prediction rather than OT scores. Secondly, as the shot number increases, the effect of β on the results diminishes. This is because as the shot number increases, the model's capability is enhanced and relies more on the OT scores for prediction. Thirdly, as the shot number increases, the standard deviation becomes smaller. This shows that the OT scores are more stable when the training size increases.

6.4 Effects of the Number of Prototypes

We explore the effects of the number of prototypes on two datasets in Figure 6. We can see that using multiple prototypes tends to outperform using one. This shows that using multiple prototypes is valid in our multiple prompts setting. As the shot number increases, a large Q usually performs better. This is because when the shot number increases, the model needs a large Q to capture subtle differences between multiple texts of the same class.

7 Conclusion

In this paper, we propose the multi-prompting decoder framework for output-side MaaS adaptation on few-shot tasks. To achieve better decoding performance, we design two decoding strategies to decode PLMs' output hidden states and class scores, respectively. Extensive experiments show that our approach is effective and efficient.

Limitations

Although our approach achieves state-of-the-art performance, it has some limitations.

The first limitation is that our approach relies on two kinds of PLMs' outputs, namely hidden states and class scores, but these outputs are not always available for some PLMs, such as ChatGPT. More rigorous decoding, such as decoding only based on the accessing of the output tokens, is worth further exploration.

The second limitation is that we have only conducted experiments on the English datasets. The performance of our method on datasets in other languages deserves further exploration.

The third limitation is that our method relies on some manual engineering, such as the design of prompts and label words. In this paper, this problem is alleviated by using templates and label words from previous works.

The fourth limitation is that MPD may not be suitable for training with more training data. As shown in Table 1 and Table 3, the performance gap of between MPD and DecT is narrowing as the shot number N increases from 4 to 256. When N is 4, 16, 64, and 256, the average performance gap is 3.4, 2,3, 1.4, and 0.8, respectively. Considering that as the shot number N increases, the performance improvement becomes marginal, and using multiple prompts for querying incurs additional costs, we think that our MPD is not suitable in this situation.

Acknowledgments

We would like to thank the anonymous reviewers for their insightful comments. This work is supported by the National Natural Science Foundation of China under Grants Nos. 62441225, 61972192, 62172208, 61906085. This work is partially supported by Collaborative Innovation Center of Novel Software Technology and Industrialization. This work is supported by the Fundamental Research Funds for the Central Universities under Grant No. 14380001.

References

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015*, pages 632–642.

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 *NeurIPS 2020*.

Guangyi Chen, Weiran Yao, Xiangchen Song, Xinyue Li, Yongming Rao, and Kun Zhang. 2023a. PLOT: prompt learning with optimal transport for vision-language models. In *The Eleventh International Conference on Learning Representations, ICLR 2023*. OpenReview.net.

Yulin Chen, Ning Ding, Xiaobin Wang, Shengding Hu, Haitao Zheng, Zhiyuan Liu, and Pengjun Xie. 2023b. Exploring lottery prompts for pre-trained language models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2023*, pages 15428–15444.

Ganqu Cui, Wentao Li, Ning Ding, Longtao Huang, Zhiyuan Liu, and Maosong Sun. 2023. Decoder tuning: Efficient language understanding as decoding. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2023*, pages 15072–15087.

Marco Cuturi. 2013. Sinkhorn distances: lightspeed computation of optimal transport. In *NeurIPS*, volume 2, page 4.

Ido Dagan, Oren Glickman, and Bernardo Magnini. 2005. The PASCAL recognising textual entailment challenge. In *Machine Learning Challenges, Evaluating Predictive Uncertainty, Visual Object Classification and Recognizing Textual Entailment, First PASCAL Machine Learning Challenges Workshop, MLCW 2005*, pages 177–190.

Mingkai Deng, Jianyu Wang, Cheng-Ping Hsieh, Yihan Wang, Han Guo, Tianmin Shu, Meng Song, Eric P. Xing, and Zhiting Hu. 2022. Rlprompt: Optimizing discrete text prompts with reinforcement learning. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, EMNLP 2022*, pages 3369–3391.

Shizhe Diao, Xuechun Li, Yong Lin, Zhichao Huang, and Tong Zhang. 2022. Black-box prompt learning for pre-trained language models. *CoRR*, abs/2201.08531.

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, ACL/IJCNLP 2021, pages 3816–3830.

Chengcheng Han, Liqing Cui, Renyu Zhu, Jianing Wang, Nuo Chen, Qiushi Sun, Xiang Li, and Ming Gao. 2023. When gradient descent meets derivative-free optimization: A match made in black-box scenario. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 868–880.

Junxian He, Chunting Zhou, Xuezhe Ma, Taylor Berg-Kirkpatrick, and Graham Neubig. 2022. Towards a unified view of parameter-efficient transfer learning. In *The Tenth International Conference on Learning Representations, ICLR 2022*.

Bairu Hou, Joe O'connor, Jacob Andreas, Shiyu Chang, and Yang Zhang. 2023. Promptboosting: Black-box text classification with ten forward passes. In *International Conference on Machine Learning*, pages 13309–13324.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin de Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-efficient transfer learning for NLP. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019*, pages 2790–2799.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. Lora: Low-rank adaptation of large language models. In *The Tenth International Conference on Learning Representations, ICLR 2022*.

Minqing Hu and Bing Liu. 2004. Mining and summarizing customer reviews. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 168–177.

Zhengbao Jiang, Frank F. Xu, Jun Araki, and Graham Neubig. 2020. How can we know what language models know. *Trans. Assoc. Comput. Linguistics*, 8:423–438.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021*, pages 3045–3059.

Chengzhengxu Li, Xiaoming Liu, Yichen Wang, Duyi Li, Yu Lan, and Chao Shen. 2023. Dialogue for prompting: a policy-gradient-based discrete prompt optimization for few-shot learning. *arXiv preprint arXiv:2308.07272*.

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, ACL/IJCNLP 2021*, pages 4582–4597.

Benlin Liu, Yongming Rao, Jiwen Lu, Jie Zhou, and Cho-Jui Hsieh. 2021a. Multi-proxy wasserstein classifier for image classification. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 8618–8626.

Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. 2023. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *ACM Comput. Surv.*, 55(9):195:1–195:35.

Xiao Liu, Yanan Zheng, Zhengxiao Du, Ming Ding, Yujie Qian, Zhilin Yang, and Jie Tang. 2021b. GPT understands, too. *CoRR*, abs/2103.10385.

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

Yao Lu, Max Bartolo, Alastair Moore, Sebastian Riedel, and Pontus Stenetorp. 2022. Fantastically ordered prompts and where to find them: Overcoming few-shot prompt order sensitivity. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2022*, pages 8086–8098. Association for Computational Linguistics.

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *The 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, Proceedings of the Conference*, pages 142–150.

Gaspard Monge. 1781. Mémoire sur la théorie des déblais et des remblais. *Mem. Math. Phys. Acad. Royale Sci.*, pages 666–704.

Bo Pang and Lillian Lee. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *ACL 2005, 43rd Annual Meeting of the Association for Computational Linguistics*, pages 115–124.

Guanghui Qin and Jason Eisner. 2021. Learning how to ask: Querying lms with mixtures of soft prompts. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2021*, pages 5203–5212.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, Peter J Liu, et al. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21(140):1–67.

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 2020*, pages 4222–4235.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Y. Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, EMNLP 2013*, pages 1631–1642.

Taylor Sorensen, Joshua Robinson, Christopher Michael Rytting, Alexander Glenn Shaw, Kyle Jeffrey Rogers, Alexia Pauline Delorey, Mahmoud Khalil, Nancy Fulda, and David Wingate. 2022. An information-theoretic approach to prompt engineering without ground truth labels. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2022*, pages 819–862.

Tianxiang Sun, Zhengfu He, Hong Qian, Yunhua Zhou, Xuanjing Huang, and Xipeng Qiu. 2022a. Bbtv2: Towards a gradient-free future with large language models. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, EMNLP 2022*, pages 3916–3930.

Tianxiang Sun, Yunfan Shao, Hong Qian, Xuanjing Huang, and Xipeng Qiu. 2022b. Black-box tuning for language-model-as-a-service. In *International Conference on Machine Learning, ICML 2022*, pages 20841–20855.

Adina Williams, Nikita Nangia, and Samuel R. Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2018*, pages 1112–1122.

Elad Ben Zaken, Yoav Goldberg, and Shauli Ravfogel. 2022. Bitfit: Simple parameter-efficient fine-tuning for transformer-based masked language-models. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), ACL 2022*, pages 1–9.

Ningyu Zhang, Luoqiu Li, Xiang Chen, Shumin Deng, Zhen Bi, Chuanqi Tan, Fei Huang, and Huajun Chen. 2022. Differentiable prompt makes pre-trained language models better few-shot learners. In *The Tenth International Conference on Learning Representations, ICLR 2022*.

Tianjun Zhang, Xuezhi Wang, Denny Zhou, Dale Schuurmans, and Joseph E. Gonzalez. 2023. TEMPERA: test-time prompt editing via reinforcement learning. In *The Eleventh International Conference on Learning Representations, ICLR 2023*.

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

He Zhao, Dinh Phung, Viet Huynh, Trung Le, and Wray L. Buntine. 2021a. Neural topic model via optimal transport. In *9th International Conference on Learning Representations, ICLR 2021*. OpenReview.net.

Zihao Zhao, Eric Wallace, Shi Feng, Dan Klein, and Sameer Singh. 2021b. Calibrate before use: Improving few-shot performance of language models. In *Proceedings of the 38th International Conference on Machine Learning, ICML 2021*, volume 139, pages 12697–12706.

A Optimal Transport

OT aims to find the minimal cost transport plan between two distributions. In this paper, we focus on discrete distribution and define two discrete distributions as follows:

$$U = \sum_{p=1}^{P} u_p \delta_{\boldsymbol{f}_p} \quad \text{and} \quad V = \sum_{q=1}^{Q} v_q \delta_{\boldsymbol{g}_q}, \quad (7)$$

where $\boldsymbol{u}$ and $\boldsymbol{v}$ are the discrete probability vectors that sum to 1, and $\delta_{\boldsymbol{f}}$ is a Dirac delta function operating on $\boldsymbol{f}$ in the embedding space. Then, the cost matrix $\boldsymbol{C} \in \mathbb{R}^{P \times Q}$ in OT can be defined as $\boldsymbol{C}_{p,q} = 1 - \text{sim}(\boldsymbol{f}_p, \boldsymbol{g}_q)$ and each element in the cost matrix denotes the cost between $\boldsymbol{f}_p$ and $\boldsymbol{g}_q$, where sim(,) denotes a similarity metric function. The optimization problem of optimal transport is formulated as:

$$\begin{aligned} &\underset{\boldsymbol{T}}{\text{minimize}} \quad \sum_{p=1}^{P}\sum_{q=1}^{Q} \boldsymbol{T}_{p,q}\boldsymbol{C}_{p,q} \\ &\text{subject to } \boldsymbol{T}\mathbf{1}_Q = \boldsymbol{u},\ \boldsymbol{T}^\top \mathbf{1}_P = \boldsymbol{v},\ \boldsymbol{T} \in \mathbb{R}_+^{P \times Q}. \end{aligned} \quad (8)$$

where $\boldsymbol{T}$ is the learned transport plan denotes the matching flow between two distributions and $\boldsymbol{T}_{p,q}$ is the amount of mass that needs to move $\boldsymbol{f}_p$ to $\boldsymbol{g}_q$.

Since direct optimization of the above processes is usually time-consuming, Cuturi (2013) use Sinkhorn distance which adds an entropic constraint for fast optimization. Thus, the new optimization objective is formulated as:

$$\begin{aligned} &\underset{\boldsymbol{T}}{\text{minimize}} \quad \sum_{p=1}^{P}\sum_{q=1}^{Q} \boldsymbol{T}_{p,q}\boldsymbol{C}_{p,q} - \lambda h(\boldsymbol{T}) \\ &\text{subject to } \boldsymbol{T}\mathbf{1}_Q = \boldsymbol{u},\ \boldsymbol{T}^\top \mathbf{1}_P = \boldsymbol{v},\ \boldsymbol{T} \in \mathbb{R}_+^{P \times Q}, \end{aligned} \quad (9)$$

where $h(\cdot)$ is entropy and $\lambda \geq 0$ is a hyperparameter. Then we can have a fast optimization solution to get optimal transport plan $\boldsymbol{T}^*$ with a few iterations as:

$$\boldsymbol{T}^* = \text{diag}(\boldsymbol{u}^{(t)}) \exp(-\boldsymbol{C}/\lambda) \text{diag}(\boldsymbol{v}^{(t)}), \quad (10)$$

where t denotes the iteration and in each iteration $\boldsymbol{u}^{(t)} = \boldsymbol{u}/(\exp(-\boldsymbol{C}/\lambda)\boldsymbol{v}^{(t-1)})$ and $\boldsymbol{v}^{(t)} = \boldsymbol{v}/\left(\exp(-\boldsymbol{C}/\lambda)^\top \boldsymbol{u}^{(t)}\right)$, with the uniform initiation $\boldsymbol{u}$ and $\boldsymbol{v}$, and $\boldsymbol{v}^{(0)} = \mathbf{1}_Q$.

B Datasets

In this section, we describe the datasets used in Table‘1.

Task	Dataset	# Class	# Test
Sentiment Analysis	SST2	2	872
	Yelp	2	38,000
	IMDB	2	25,000
Topic Classification	AG's News	4	7,600
	Yahoo	10	60,000
	DBPedia	14	70,000
NLI	RTE	2	277
	SNLI	3	9,842
	MNLI-m	3	9,815
	MNLI-mm	3	9,832

Table 5: Statistics of datasets.

The sentiment analysis task includes three datasets. The Stanford Sentiment Treebank (SST2) dataset and IMDB dataset were both extracted from movie reviews. The Yelp dataset consists of reviews from Yelp.

The topic classification task includes three datasets. AG's news dataset collects more than 1 million news articles from more than 2,000 news sources. We also only use the title and description fields to classify news. The Yahoo dataset was collected from the Yahoo Webscope program and contains questions and their answers. The fields we used include question title, question content, and best answer. The DBPedia ontology classification dataset is constructed by picking 14 non-overlapping classes from DBPedia 2014 derived from Wikipedia.

The NLI task includes three datasets. The Recognizing Textual Entailment (RTE) dataset comes from a series of textual entailment challenges. Examples on the RTE dataset are constructed based on news and Wikipedia text. The Stanford Natural Language Inference (SNLI) corpus is a collection of human-written English sentence pairs manually labeled for balanced classification with the labels entailment, contradiction, and neutral. MNLI (Multi-genre Natural Language Inference) offers ten distinct genres of written and spoken English NLI data. MNLI-m and MNLI-mm correspond to versions matched and mismatched of the MNLI dataset, respectively.

C Comparsion with TEMPERA and DP$_2$O

We further compare our model with two existing state-of-the-art methods (i.e., TEMPERA (Zhang

Model	SST2	IMDB	Yelp	AG	DB	Yahoo	RTE	SNLI	MNLI-m/mm	Avg.
T5$_{\text{Base}}$	$86.9_{0.6}$	$87.5_{0.6}$	$90.8_{0.4}$	$85.7_{0.7}$	$92.9_{0.3}$	$63.9_{0.6}$	$58.3_{0.4}$	$60.3_{2.2}$	$59.0_{1.0}/60.9_{1.1}$	$74.6_{0.8}$
T5$_{\text{Large}}$	$92.3_{0.4}$	$92.6_{0.6}$	$95.5_{0.1}$	$86.5_{0.6}$	$95.6_{0.4}$	$66.4_{0.6}$	$63.3_{2.7}$	$61.6_{1.4}$	$59.8_{2.9}/62.5_{3.0}$	$77.6_{1.3}$
T5$_{\text{3B}}$	$91.8_{0.7}$	$93.0_{0.6}$	$95.6_{0.1}$	$87.5_{0.6}$	$95.9_{0.2}$	$67.6_{0.7}$	$68.1_{1.0}$	$64.7_{1.0}$	$61.5_{1.2}/63.5_{1.5}$	$78.9_{0.8}$

Table 6: Performance (16-shot) for our method on different versions of T5 (Raffel et al., 2020). We also run each experiment over 5 random seeds and report the average accuracy and standard deviation (%).

Method	SST2	MR	CR	Yelp	Avg.
TEMPERA	$91.9_{2.0}$	$88.0_{1.1}$	$91.1_{1.6}$	$92.6_{1.7}$	$90.9_{1.6}$
DP$_2$O	$\mathbf{93.6_{0.7}}$	$88.6_{0.9}$	$90.8_{0.5}$	$94.3_{0.4}$	$91.8_{0.6}$
MPD	$92.1_{0.1}$	$\mathbf{89.6_{0.2}}$	$\mathbf{91.3_{0.3}}$	$\mathbf{96.4_{0.1}}$	$\mathbf{92.4_{0.2}}$

Table 7: Comparsion with TEMPERA and DP$_2$O following their experimental settings. The results (i.e., average accuracy and standard deviation (%)) over 4 runs. The best results are in **bold**.

Model Setting	Average Accuracy		
	1	4	16
Ours	**71.5**	**75.2**	**77.8**
Uniform Distribution	70.7	74.7	77.1
Cosine Similarity	68.3	73.8	77.4

Table 8: Performance of two variants to replace optimal transport plan. We run each experiment over 5 random seeds.

et al., 2023) and DP$_2$O (Li et al., 2023)) following their experimental settings on four sentiment classification datasets (i.e., SST2 (Socher et al., 2013), MR (Pang and Lee, 2005), CR (Hu and Liu, 2004), and Yelp (Zhang et al., 2015)). They both use RL to construct or match prompts for each input. Specifically, TEMPERA designs the agent to perform different editing techniques to construct query-dependent prompts. DP$_2$O first designs a prompt generation strategy based on GPT-4 and then designs a RL framework based on policy gradients to match suitable prompts for a single input.

Our results demonstrate that our model consistently outperforms these baselines in terms of both average performance and standard deviations, showcasing the effectiveness and robustness of our method. Moreover, we observe that MPD also significantly reduces training time compared to the two methods.

D Effectiveness of OT

We use two variants to replace $\boldsymbol{T}$ to illustrate the effectiveness of OT, including uniform distribution and cosine similarity.

We can see that OT achieves the best performance in Table 8. This shows that OT can effectively match the relationship between test representations and prototypes. The uniform distribution does not take into account the similarity between two elements. Cosine similarity only calculates the similarity between two elements and cannot take into account the similarity with other elements.

E Visualization of Label Words Expansion

We show the expanded label words in Table 11. We set the size of expanded label words to 10. We can see that the expansion method can find some relevant words and improve performance. For example, on the three sentiment datasets (i.e., SST2, IMDB, and Yelp), label word 'bad' expands 'poor', 'wrong', and 'worst', and label word 'great' expands 'good', 'perfect', and 'powerful'. While 'bad' extends 'good', we assign a weight to 'good' based on its cosine similarity to the target class word to avoid the effect of noise.

F Model Scaling

We further explore how our model applies to PLMs with different architectures and scales.

We use a encoder-decoder architecture PLM T5 (Raffel et al., 2020) with different scales, from T5$_{\text{Base}}$, T5$_{\text{Large}}$ to T5$_{\text{3B}}$ in Table 6. Firstly, MPD also achieves good performance on a encoder-decoder architecture PLM T5, illustrating the ability of MPD to transfer across PLM architectures. Secondly, a larger model often achieves better performance. This is because larger models have more representative power to get better output, including hidden states and class scores.

G Trade-offs between Training Time and Performance

In this section, we further explore the trade-offs between training time and performance in our

method.

Firstly, training time is proportional to the number of adopted prompts from Table 9. This is because we need to feed more training data into PLM. Secondly, when the number of prompts exceeds 3, the improvement becomes marginal. As a result, to balance training time and performance, we use three prompts.

Adopted Prompts	Acc	Training Time (s)
1	87.44	2.3
1+2	87.82	4.5
1+2+3	87.94	6.7
1+2+3+5	87.98	8.8
1+2+3+4+5	87.97	10.9

Table 9: Accuracy and training time using different prompts on AG's news dataset under 16-shot setting.

Adopted Prompts	Acc
Vani	87.44
Instr	86.06
E	87.47
Vani+Instr	87.49
Vani+E	88.11
Instr+E	87.66
Vani+Instr+E	87.92

Table 10: Accuracy using various types of prompts on AG's news dataset under the 16-shot setting.

H Effectiveness on Various Types of Prompts

In this section, we further show the effectiveness on various types of prompts, including instruction and example. Specifically, we use the vanilla prompt 1 in Figure 4 (denoted as Vani), instruction (denoted as Instr), and example (denoted as E, we use the first training data from the AG's news dataset to construct example prompt) on the AG's news dataset under the 16-shot setting. Three prompts as shown in Table 14.

Firstly, our method usually achieves performance improvement compared to using a single prompt in Table 10. Specifically, using two prompts exceeds using either prompt individually. For example, Vani+Instr exceeds Vani and Instr, respectively. It is worth noting that Vani+E exceeds all results in Figure 4. This shows that using two different types of prompts can sometimes yield better results. Secondly, the performance fluctuation of using two prompts is smaller than that of a single prompt. Thirdly, Vani+Instr+E does not exceed Vani+Instr due to the poor performance of Instr. This is because RoBERTa-large has not been fine-tuned by instruction-tuning.

Dataset	Label Words	Label Words Expansion
SST2, IMDB Yelp	bad	'bad', 'Bad', ' bad', ' Bad', 'good', 'poor', 'wrong', 'big', 'worst', ' BAD'
	great	'great', 'Great', ' great', 'small', 'huge', 'little', 'good', 'perfect', 'powerful', 'big'
AG's News	politics	'politics', 'Politics', 'political', 'Political', 'policy', ' politics', ' Politics', 'democracy', 'history', 'business'
	technology	'technology','Technology', 'tech', 'software', 'science', ' technology', 'computer', 'engineering', 'technical', 'device'
	business	'business', 'Business', ' business', ' Business', 'commercial', 'property', 'biz', 'office', 'trade', 'everything'
	sports	'sports', 'Sports', 'Sport', 'football', ' sports', ' Sports', 'music', 'gaming', 'games', 'military'
Yahoo	soc	'soc', 'Soc', 'social', 'community', 'offic', 'civil', ' soc', 'sn', 'mom', ' Soc'
	science	'science', 'Science', 'scientific', ' science', 'technology', 'biology', 'research', 'history', 'scient', 'evidence'
	health	'health', 'Health', ' health', 'medical', 'healthy', ' Health', 'hospital', 'Medical', 'cancer', 'safety'
	education	'education', 'Education', ' education', 'learning', 'training', 'educated', 'awareness', 'student', 'college', 'school'
	com	'com', 'Com', ' com', 'COM', ' Com', 'org', 'comm', 'co', 'net', 'gov'
	sports	'sports', 'Sports', 'Sport', 'football', ' sports', ' Sports', 'music', 'gaming', 'games', 'military'
	business	'business', 'Business', ' business', ' Business', 'commercial', 'property', 'biz', 'office', 'trade', 'everything'
	ent	'ent', 'ENT', 'ents', 'enting', 'ented', 'ant', 'ente', 'ento', 'ency', 'enta'
	family	'family', 'Family', ' family', 'community', ' Family', 'daughter', 'brother', 'parents', 'mom', 'small'
	politics	'politics', 'Politics', 'political', 'Political', 'policy', ' politics', ' Politics', 'democracy', 'history', 'business'
DBPedia	company	'company', 'Company', ' company', 'project', 'community', 'family', 'Companies', 'city', 'business', 'office'
	school	'school', 'School', ' school', 'college', ' School', 'chool', 'fashioned', 'education', 'student', 'program'
	artist	'artist', 'Artist', ' artist', ' Artist', 'music', 'creator', 'director', 'manager', 'editor', 'student'
	ath	'ath', 'aths', 'athe', 'ATH', 'athi', 'atha', 'athy', 'oth', 'athing', 'athan'
	politics	'politics', 'Politics', 'political', 'Political', 'policy', ' politics', ' Politics', 'democracy', 'history', 'business'
	trans	'trans', 'Trans', ' Trans', ' trans', 'rans', 'poly', 'simple', 'project', 'small', 'digital'
	building	'building', 'build', 'builder', 'Building', 'builders', ' building', 'built', ' Building', 'making', 'fighting'
	river	'river', 'River', 'rider', ' river', 'roller', 'lake', 'city', 'later', 'country', 'current'
	vill	'vill', 'Vill', ' Vill', 'vell', 'ville', 'vag', 'font', 'coll', 'christ', ' vill'
	animal	'animal', 'Animal', ' animal', 'species', 'human', 'monster', ' Animal', 'horse', 'baby', 'adult'
	plant	'plant', ' plant', ' Plant', 'bomb', ' Plants', 'flower', 'forest', 'train', 'fruit', 'print'
	album	'album', ' album', ' Album', 'music', 'movie', 'episode', 'concert', 'download', 'artist', 'film'
	film	'film', 'Film', 'movie', ' film', ' Film', 'Movie', ' Films', 'music', 'video', 'picture'
	book	'book', 'books', 'Book', ' book', ' Book', 'BOOK', 'Books', ' books', 'sheet', 'ebook'

Table 11: The results of label words expansion.

Dataset	Template	Label Words
SST2	x A [MASK] movie. x A [MASK] film. x A [MASK] piece of work.	bad, great
Yelp	x In summary, it was [MASK]. x All in all, it was [MASK]. x A [MASK] review.	bad, great
IMDB	x In summary, it was [MASK]. x All in all, it was [MASK]. x In summary, the film was [MASK].	bad, great
AG's News	[Topic : [MASK]] x_1 x_2 [Category : [MASK]] x_1 x_2 x_1 x_2 The topic is about [MASK].	politics, sports, business, technology
Yahoo	[Topic : [MASK]] x_1 x_2 A [MASK] question : x_1 x_2 x_1 x_2 The topic is about [MASK].	society, science, health, education, computers, sports, business, entertainment, family, politics
DBPedia	[Topic : [MASK]] x_1 x_2 x_1 x_2 x_1 is a [MASK]. x_1 x_2 In this sentence, x_1 is a [MASK].	company, school, artist, athlete, politics, transportation, building, river, village, animal, plant, album, film, book
RTE	x_1? [MASK], x_2	No, Yes
SNLI	x_1. [MASK], x_2	No, Maybe, Yes
MNLI-m/mm	x_1! [MASK], x_2	No, Maybe, Yes

Table 12: The templates and label words used in our experiments. Different datasets use different forms of input, such as sentence$x$ and sentence pairs x_1 x_2.

Template Id	Template
1	[Topic : [MASK]] x_1 x_2
2	[Category : [MASK]] x_1 x_2
3	x_1 x_2 The topic is about [MASK].
4	[MASK] Alert Blog Dialogue Diary Accountability x_1 x_2
5	A [MASK] news : x_1 x_2

Table 13: Five templates used in our experiments. The fourth prompt comes from Deng et al. (2022).

Template Type	Template
Vani	[Topic : [MASK]] x_1 x_2
Instr	Classify the news articles into the categories of Politics, Sports, Business, and Technology. x_1 x_2 This topic is about [MASK].
E	Article: Wall St. Bears Claw Back Into the Black. (Reuters), Reuters - Short-sellers, Wall Street's dwindling of ultra-cynics, are seeing green again. Answer: Business. Article: x_1 x_2 Answer: [MASK].

Table 14: Three different types of templates (i.e., vanilla prompt, instruction prompt, and example prompt) used in our experiments.

Algorithm 1 The training flow of our method

Input: Training set and PLM $\mathcal{M}$.

Output: The parameters of linear layer $\mathbf{W}$ and prototypes $\mathbf{R}$

▷ Extract features and initialize class prototypes:
Randomly initialize class prototypes $\boldsymbol{R}$
Obtain a feature set $\boldsymbol{H}$ via PLM $\mathcal{M}$
▷ Training:
for epoch $= 1, 2, \ldots, T$ **do**
    Sample a data x_i and use linear layer to get the text representations $\boldsymbol{V}_i$
    ▷ Computing OT scores via Sinkhorn's algorithm:
    for $k = 1, 2, \ldots, N$ **do**
        Calculate the cost matrix $\boldsymbol{C}_k = \mathbf{1} - \boldsymbol{V}\boldsymbol{R}_k^\top$ of each class
        for $t_{in} = 1, 2, \ldots, T_{in}$ **do**
            $Z = \exp(-\boldsymbol{C}/\lambda)$
            $\boldsymbol{u}^{(t_{in})} = \boldsymbol{u}/(Z\boldsymbol{v}^{(t_{in}-1)})$
            $\boldsymbol{v}^{(t_{in})} = \boldsymbol{v}/(Z^\top \boldsymbol{u}^{(t_{in})})$
            $\Delta_v = \sum |\boldsymbol{v}^{(t_{in})} - \boldsymbol{v}^{(t_{in}-1)}|/N$
            if $\Delta_v < \delta$ **then**
                break
            end if
        end for
        Obtain optimal transport plan as $\boldsymbol{T}_k^* = \text{diag}(\boldsymbol{u}^{(t)}) \exp(-\boldsymbol{C}_k/\lambda) \text{diag}(\boldsymbol{v}^{(t)})$
        Calculate the OT scores by Eq. (2)
    end for
    Update the parameters with cross-entropy loss by Eq. (3)
end for
return The well-trained linear layer $\mathbf{W}$ and prototypes $\mathbf{R}$