Parameter-free Automatically Prompting: A Latent Pseudo Label Mapping Model for Prompt-based Learning

Jirui Qi[1], **Richong Zhang**[12]*, **Junfan Chen**[1], **Jaein Kim**[1], **Yongyi Mao**[3]
[1]SKLSDE, Beihang University, Beijing, China
[2]Zhongguancun Laboratory, Beijing, China
[3] School of Electrical Engineering and Computer Science, University of Ottawa, Canada
{qijr,zhangrc,chenjf}@act.buaa.edu.cn, jaein@buaa.edu.cn, ymao@uottawa.ca

Abstract

Prompt-based learning has achieved excellent performance in few-shot learning by mapping the outputs of the pre-trained language model to the labels with the help of a label mapping component. Existing manual label mapping (MLM) methods achieve good results but heavily rely on expensive human knowledge. Automatic label mapping (ALM) methods that learn the mapping functions with extra parameters have shown their potentiality. However, no effective ALM model comparable to MLM methods is developed yet due to the limited data. In this paper, we propose a Latent Pseudo Label Mapping (LPLM) method that optimizes the label mapping without human knowledge and extra parameters. LPLM is built upon a probabilistic latent model and is iteratively self-improved with the EM-style algorithm. The empirical results demonstrate that our LPLM method is superior to the mainstream ALM methods and significantly outperforms the SOTA method in few-shot classification tasks. Moreover, LPLM also shows impressively better performance than the vanilla MLM method which requires extra task-specific prior knowledge.

1 Introduction

With the advent of the powerful pre-trained language model (PLM) such as GPT-3 (Brown et al., 2020), prompt-based learning becomes blooming in recent years because it can effectively bridge the gap between pre-training tasks and downstream tasks (Liu et al., 2021). Several corresponding studies show better performance than the traditional fine-tuning methods in few-shot learning tasks (Cui et al., 2022), where few-shot learning is generally considered as N-way K-shot learning. Each task consists of N classes with K instances per class.

The pipeline of vanilla prompt-based learning for classification tasks is composed of a prompting phase and a prediction phase. In the prompting phase, the input sentence is reconstructed into a cloze-type sentence with a pre-defined template containing a <MASK> slot. In the prediction phase, the PLM tries to fill the <MASK> slot with corresponding words in the vocabulary based on the context. One challenge in this pipeline is mapping the fill-in word with the expected class label. To address this challenge, label mapping is introduced into the pipeline, assigning a class representative to each class to associate the PLM output with the expected class label. Typically, each class representative is one or a set of human-chosen word(s) that should be highly relevant to each class. Based on the word-occurrence probability, the model can calculate the occurrence probability of the class representatives. Finally, the label of the input sentence is inferred from the class representative with the highest occurrence probability.

As label mapping plays an important role in prompt-based learning (Liu et al., 2021), design-

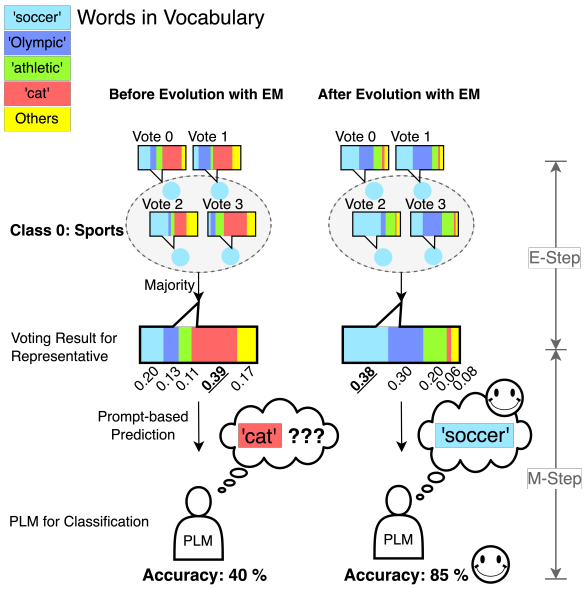

Figure 1: The evolutionary process of the selected class representative and the improvement of the overall performance by adopting the EM-style algorithm.

*Corresponding author

Findings of the Association for Computational Linguistics: EMNLP 2022, pages 3981 - 3991

ing more effective label mapping becomes significant. The recent label mapping methods can be divided into two branches: manual label mapping (MLM) and automatic label mapping (ALM). MLM method manually selects the class representative for each class, and it is proven to be very powerful in many tasks (Liu et al., 2021). However, this method is time-consuming and labor-intensive. Humans need to thoroughly understand the downstream tasks first and select the appropriate words for each class representative. Additionally, relying heavily on human subjectivity can cause different problems. Thus, ALM methods are proposed to eliminate the effects of human involvement. They can be further divided into soft label mapping, search-based label mapping, and prototypical label mapping. Soft label mapping (Hambardzumyan et al., 2021) replaces the concrete tokens with trainable tokens to generate class representatives. Search-based label mapping (Gao et al., 2021) aims to find a subset of candidate words from the vocabulary using algorithms or additional networks with additional parameters. Prototypical label mapping (Cui et al., 2022) projects the <MASK> vector onto a new embedding space with additional linear networks and fine-tunes the model with contrastive learning. However, these ALM methods involve extra parameters which require a big amount of instances to train, where in practice, the available data is scarce in few-shot scenarios, making it difficult to train the model to find optimal class representatives. Consequently, inexpressive class representatives may degrade the final classification performance.

To address the above problem, we propose a Latent Pseudo Label Mapping (LPLM) model. LPLM first uses a Majority Voter that shares the same parameters with PLM to automatically generate class representatives. As the class representatives produced by this method may be noisy, we build a latent variable model and introduce an EM-style algorithm to gradually reduce the noise and enhance the expressiveness of each class representative, as shown in Figure 1[1].

The novel prompt-based learning pipeline with LPLM has the advantage that no additional parameters or human knowledge are required to generate class representatives. In the E-step, the distributions of latent variables are updated with a Majority Voter and the class representatives are re-selected based on the parameters optimized in the previous M-step. In the M-step, predictions are made according to the generated class representatives from the E-step, and the parameters of the PLM are updated based on the prediction results. The two steps perform alternately to obtain increasingly optimal distributions for class representatives and to improve the overall performance. Moreover, we introduce two strategies for selecting the keywords in each class representative, which are discussed in Section 3.3.

To verify the effectiveness of LPLM, we conduct a series of experiments on widely-used classification datasets. Since our method can be applied to any classification task, we deliberately choose three different types of datasets, a sentiment classification dataset SST-2 (2-way), a well-known text classification dataset AG's News (4-way), and a popular entity typing dataset Few-NERD (66-way). The experimental results show that LPLM significantly outperforms the other three ALMs and even outperforms MLM. The contributions of this paper can be summarized as follows:

- We introduce the latent variable model in prompt-based learning and propose LPLM that optimizes the label mapping without human knowledge or extra parameters.
- We propose LPLM to automatically generate the class representative for each class using the majority voting mechanism, and alternately optimize the parameters of PLM and the distributions of latent variables by the EM-style algorithm.
- We conduct a series of experiments to compare the performance of LPLM with other baseline label mappings, which demonstrates that LPLM outperforms not only all other ALMs, including the SOTA, but also MLM.

2 Related Work

2.1 Prompt-based Learning

As the demand for effectively fine-tuning large-scale language models such as GPT-3 gets bigger, the popularity of prompt-based learning (in-context learning) has also increased. To fine-tune the model with prompt-based learning, the template with a <MASK> slot is used to reconstruct the input texts, like '*<TEXT>. The category is <MASK>.*'. In

[1]To be intelligible, we illustrate our motivation with the 4-way 4-shot classification task, which is consistent with one of the settings in our experiments.

recent works, prompt-based learning has achieved impressive performance in many downstream tasks, such as text classification (Gao et al., 2021), knowledge probing (Petroni et al., 2019), and data construction (Choenni et al., 2021).

Although prompt-based learning consists of two parts, namely template designing and label mapping, most of the existing works focus on the former due to its simplicity. However, as discussed in Section 1, label mapping is also an important part that determines the performance of prompt-based learning as it builds a bridge between the word-occurrence possibility and the class labels. To accomplish this more effectively, in this paper, we propose a novel method for ALM.

2.2 Mainstream Label Mappings

While label mapping has a crucial impact on the performance of prompt-based learning (Gao et al., 2021), each of the four mainstream label mapping approaches has its shortcoming.

Manual label mapping engages human-involvement to select the class representative for each class (Schick and Schütze, 2021). As this method is heavily dependent on human knowledge, side effects may occur. Contrary to expectations, the level of human knowledge may not always be guaranteed as each task requires different background knowledge. If the selected class representatives are not expressive enough, the overall performance could fluctuate greatly (Gao et al., 2021).

Soft label mapping directly uses trainable continuous tokens as class representatives and aims at optimizing them at the fine-tuning stage (Hambardzumyan et al., 2021). However, to achieve this, enough data is required for optimization, which is not realistic in practical few-shot scenarios.

Search-based label mapping obtains candidate words from the entire vocabulary and uses the validation set to select the best class representatives for fine-tuning the PLM. (Gao et al., 2021). However, this method also faces the same difficulty as soft label mapping. It is difficult to optimize the model to find suitable candidates from a large vocabulary when there are only a few instances available.

Prototypical label mapping takes inspiration from contrastive learning to introduce an additional contrastive loss for the fine-tuning process (Cui et al., 2022). Specifically, after projecting the instances into a new embedding space, the instances that belong to the same class are summarized and the ones belonging to different classes are separated. Nonetheless, this method also requires training additional randomly initialized parameters for the projecting function. When only a few instances are available (e.g., K=1), it is difficult to train the projecting networks and the different classes may hardly be distinguished.

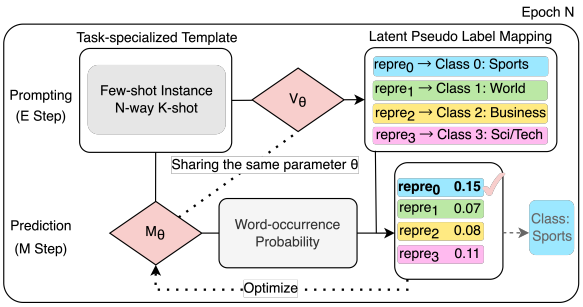

Figure 2: Pipeline of prompt-based learning with LPLM based on the EM-style algorithm. 'repre's denote the selected class representatives.

3 Latent Pseudo Label Mapping

In order to remedy the shortcomings of the methods mentioned in Section 2.2, we introduce LPLM to effectively exploit the knowledge provided by the limited instances. It is built upon a latent variable model and optimized with the EM-style algorithm. In this section, we first define the necessary notations in few-shot classification tasks and introduce the overall architecture of LPLM. Then, we explain the detailed mechanism of the E-step and the M-step with the mathematical derivation and the learning process of the EM-style algorithm.

3.1 Task Definition

In the N-way K-shot setting, a set of labeled instances is defined as X which has the size of $|X| = N * K$ as it consists of K instances for each of the N classes. The corresponding set of ground-truth labels is defined as Y which also contains $|Y| = N * K$ elements where each $y_i \in \{1, \ldots, N\}$

The proposed LPLM focuses on few-shot classification tasks and aims at predicting the label z_i for the input instance $x_i \in X$. The optimization target is to maximize

$$L(\theta) = \sum_{i=1}^{N*K} \log p_{M_\theta}(z_i|x_i, y_i, X, Y; \theta, T(\cdot)) \quad (1)$$

where $p_{M_\theta}(z_i|x_i, y_i; \theta, T(\cdot))$ is the probability that z_i is predicted to be y_i given the task-specific template $T(\cdot)$ and the pre-trained language model M_θ with the parameter θ. In total, $N * K$ few-shot instances are used by the model to optimize its parameter θ.

3.2 LPLM Architecture

The architecture of LPLM is designed on the basis of the classic principle-based EM algorithm (Dempster et al., 1977).

In prompt-based learning, M_θ only provides the probability of each word in the vocabulary $\mathcal{V}$ occurring in the <MASK> slot. To bridge the gap between the word-occurrence probability and the expected label z_i, we introduce latent variables $W = \{w_1, \ldots, w_N\}$ where each element denotes a class representative. The distribution of W and each $w_i \in W$ is updated in each E-step. The objective function is extended to

$$\begin{aligned} L(\theta) &= \sum_{i=1}^{N*K} \log p_{M_\theta}(z_i|x_i, y_i, X, Y; \theta, T(\cdot)) \\ &= \sum_{i=1}^{N*K} \log \sum_{W} p_{M_\theta}(z_i|W, x_i, y_i, X, Y; \theta, T(\cdot)) \\ &\quad \cdot p_{M_\theta}(W|x_i, y_i, X, Y; \theta, T(\cdot)) \end{aligned} \tag{2}$$

where $p(W|x_i, y_i, X, Y; \theta, T(\cdot))$ represents the probability of choosing W as the class representatives.

E-step: Each input instance is first wrapped with a template (with a <MASK> slot) and fed into the PLM M_θ after the parameters are updated in the previous M-step. M_θ outputs the distribution over all the vocabulary at the <MASK> slot. Since M_θ contains prior knowledge, and a task-specific template is used for PLM, the distribution is highly related to its contextual semantics. After averaging the distributions of all instances in the same class, LPLM updates the class representatives w_is with the keywords of the Top-k highest probabilities. The weights of the keywords are calculated by normalizing their corresponding probabilities.

M-step: For each wrapped instance, LPLM obtains the distribution at the <MASK> slot by feeding it into M_θ. The probability of each label is calculated by weighted averaging the word-occurrence probability of the keywords in each class representative. The parameters in M_θ are updated to increase the probability of predicting the correct class label for each instance.

EM-style iteration: The EM algorithm iteratively performs E-steps and M-steps to gradually mine the class-specific semantics in few-shot instances with M_θ. In the E-step of round t, the distributions of the latent variables are optimized with θ_t which is updated in the previous M-step. The class representatives W_{t-1} in round $t-1$ is updated to W_t by M_{θ_t}. Then, in the M-step, based on the classification result predicted with W_t, θ_t is updated to θ_{t+1} and M_{θ_t} is updated to $M_{\theta_{t+1}}$.

3.3 E-step Details

In E-steps, LPLM generates the class representatives, namely the latent variables, subsequent to the optimization of the distributions based on the parameter θ updated in the previous M-step. The core idea of the generation is for each instance to vote for the class representative of its own class.

3.3.1 Majority Voter

The Majority Voter V_θ is the main component in E-steps to determine the class representative for each label where θ is shared with M_θ since it also uses the PLM M_θ as its basis. The distribution of class representatives are obtained by the voter

$$W \leftarrow V_\theta(X, Y). \tag{3}$$

For each wrapped instance $x_i \in X$, V_θ outputs a $\mathcal{V}$-dimension distribution at <MASK> slot, where $\mathcal{V}$ is the vocabulary for M_θ. This distribution is the likelihood of each word in the vocabulary representing its class, namely the vote from x_i. LPLM averages the distributions (votes) of all instances in the same class according to Y and selects keywords with the highest probability using a specific selection strategy to form W.

In practice, each class representative $w_i \in W$ is a $\mathcal{V}$-dimension vector that represents a set of highly related keywords that can best express the corresponding class. Since each dimension in the averaged distribution corresponds to one keyword in $\mathcal{V}$, LPLM sets the dimensions of the unselected keywords to 0 and normalizes the distribution to get w_i.

3.3.2 Selection Strategy

After obtaining the averaged distribution with V_θ, LPLM selects the highly relevant keyword(s) as class representative for each class. For determining the class representatives, we next introduce two selection strategies, Champion selection and Top-k selection.

Champion Selection Champion selection strategy simply picks one keyword with the highest probability in the averaged distribution as the class representative. And in the next E-step, the class representatives are re-selected based on the updated distribution with the parameter-updated V_θ.

Although Champion selection strategy can effectively mine the hidden knowledge in PLM, it cannot work well when different classes have similar semantics. In this situation, LPLM is likely to select the same word for different classes as its representatives, which will further lead to a equal probability for predicting these class labels. It will disturb the model to distinguish these classes due to the unique predicting process of prompt-based learning, which we have mentioned in Section 1. Eventually, it will result in a limited improvement of the overall performance. Moreover, Champion selection strategy only considers the semantics contained in one word. Intuitively, if multiple words are selected together, LPLM can extract more semantic information, which will be more beneficial for the prediction.

Top-k Selection We propose Top-k selection strategy to solve the potential issue of Champion selection strategy. Top-k strategy selects the set of keywords with top-k highest probabilities as the class representative ($k = 2, 3, \ldots, |V|$). After setting the value of unselected dimensions to 0 and normalizing the averaged distribution, LPLM calculates the class representative w_i for each class.

3.3.3 Latent Variable for Each Instance

Similar to traditional EM algorithm, LPLM arranges each instance with a latent variable according to its corresponding class,

$$w_{y_i} \leftarrow F(x_i, y_i, W) \tag{4}$$

where $F(\cdot)$ is an arrangement function. Therefore the instances from the same class should have identical latent variables, namely w_{y_i}s.

This voting system is consistent with the most basic prediction of PLM—calculating the probability that each word to be filled in the <MASK> slot. In other words, they play the same role in the internal process and this is the reason why V_θ can directly use the M_θ as its backbone and share θ with the PLM M_θ.

3.3.4 Distribution of Class Representatives

Every w_i in W has a distribution in probability space $\mathcal{R}$. For Champion selection strategy, $\mathcal{R}$ for w_i is same with $\mathcal{V}$. Therefore the averaged distribution of V_θ can be directly treated as the distribution for the latent variables as

$$p_{M_\theta}(w_i|x_i, y_i, X, Y; \theta, T(\cdot)), w_i \in \mathcal{R}^{|\mathcal{V}|}.$$

Therefore, the distribution for W is

$$p_{M_\theta}(W|x_i, y_i, X, Y; \theta, T(\cdot)), w_i \in \mathcal{R}^{(|\mathcal{V}|)^N}.$$

Similarly, for Top-k selection strategy, the size of $\mathcal{R}$ is determined by $C^k_{|\mathcal{V}|}$, where $k = 2, 3, ..|\mathcal{V}|$. Also, the occurrence probability of each element in $\mathcal{R}$ can be derived from the majority voting result and assigned to all instances in each class, namely

$$p_{M_\theta}(w_i|x_i, y_i, X, Y; \theta, T(\cdot)), w_i \in \mathcal{R}^{C^k_{|\mathcal{V}|}}.$$

Therefore, the distribution for W is

$$p_{M_\theta}(W|x_i, y_i, X, Y; \theta, T(\cdot)), w_i \in \mathcal{R}^{(C^k_{|\mathcal{V}|})^N}.$$

3.4 M-step Details

The target of M-step is to make predictions based on the class representatives, namely the latent variables w_is. In this prediction phase, the above weights are applied to calculate the weighted average of the word-occurrence probabilities of the selected words, so as to obtain the probability of classifying the input sentence x_i as the class z_i,

$$p_{M_\theta}(z_i|W, x_i, y_i, X, Y; \theta, T(\cdot)) = \frac{w_{y_i} \cdot h_M}{\sum_{w \in W} w \cdot h_M} \tag{5}$$

where h_M is the last layer's hidden state of the <MASK> slot in the wrapped x_i after LPLM feeds each wrapped instance into M_θ

$$h_M = M_\theta(T(x_i)) \tag{6}$$

3.5 Learning with the EM-style algorithm

With the introduced latent variable W, the objective function is extended as [2]

$$\begin{aligned} L(\theta) &= \sum_{i=1}^{N*K} \log p(z_i|x_i, y_i, X, Y; \theta, T(\cdot)) \\ &= \sum_{i=1}^{N*K} \log \sum_{W} p(z_i|W, x_i, y_i, X, Y; \theta, T(\cdot)) \\ &\qquad \cdot p(W|x_i, y_i, X, Y; \theta, T(\cdot)) \end{aligned} \tag{7}$$

$p(z_i|W, x_i, y_i, X, Y; \theta, T(\cdot))$ represents the probability of the predicted z_i same with the ground-truth label y_i of the input instance x_i with the help

[2] Due to the page limitation, we abbreviate p_{M_θ} in the second line to p in this section.

of W. While $p(W|x_i, y_i, X, Y; \theta, T(\cdot))$ represents the probability of choosing W as the class representatives.

Next, Q-function $Q_i(W; \theta)$ is introduced to represent the posterior probability of the latent variable W after θ is updated based on the prediction z_i of the previous M-step. Equation 7 is derived as

$$
\begin{aligned}
&L(\theta) = \sum_{i=1}^{N*K} \log \sum_{W} Q_i(W; \theta) \\
&\frac{p(z_i|W, x_i, y_i, X, Y; \theta, T(\cdot))p(W|x_i, y_i, X, Y; \theta, T(\cdot))}{Q_i(W; \theta)} \\
&\geq \sum_{i=1}^{N*K} \sum_{W} Q_i(W; \theta) \log \\
&\frac{p(z_i|W, x_i, y_i, X, Y; \theta, T(\cdot))p(W|x_i, y_i, X, Y; \theta, T(\cdot))}{Q_i(W; \theta)}
\end{aligned} \tag{8}
$$

So far, the original objective is transformed into raising the lower bound of the inequality in Equation 8. Next, we elaborate on E-step and M-step in detail.

E-step: In E-step, based on the updated parameter θ, the posterior probability distribution of W is updated. We refer to the previous work (Chen et al., 2019) for the calculation method and expand the Q-function as

$$
\begin{aligned}
&Q_i(W; \theta) = \\
&\frac{p(z_i|W, x_i, y_i, X, Y; \theta, T(\cdot))p(W|x_i, y_i, X, Y; \theta, T(\cdot))}{\sum_W p(z_i|W, x_i, y_i, X, Y; \theta, T(\cdot))p(W|x_i, y_i, X, Y; \theta, T(\cdot))} \\
&= \frac{p(W, z_i|x_i, y_i, X, Y; \theta, T(\cdot))}{p(z_i|x_i, y_i, X, Y; \theta, T(\cdot))} \\
&= p(W|z_i, x_i, y_i, X, Y; \theta, T(\cdot))
\end{aligned} \tag{9}
$$

where $\sum_W Q_i(W; \theta) = 1$.

M-step: Given the updated latent variables and computed Q-function in E-step, the parameter θ is then optimized in M-step to maximize the lower bound of the inequality in Equation 8.

$$
\begin{aligned}
&\theta = \arg\max_{\theta} L(\theta) \\
&= \arg\max_{\theta} \sum_{i=1}^{N*K} \sum_{W} Q_i(W; \theta) \log \\
&\frac{p(z_i|W, x_i, y_i, X, Y; \theta, T(\cdot))p(W|x_i, y_i, X, Y; \theta, T(\cdot))}{Q_i(W; \theta)}
\end{aligned} \tag{10}
$$

4 Experiments

We conduct a series of experiments in few-shot scenarios to demonstrate the effectiveness of LPLM.

Dataset	Training Set	Validation Set	Test Set
SST-2	2*K	2*K	1821
AG's News	4*K	4*K	7600
FewNERD	66*K	66*K	96901

Table 1: Size of the datasets used in the classification experiments.

In this section, we first introduce the experimental setups in use, then present and discuss the experimental results.

4.1 Datasets and Implementation Details

We select three well-known classification datasets: SST-2 for 2-way sentiment classification tasks, AG's News for 4-way topic classification tasks, and Few-NERD for 66-class NER tasks. The training, validation, and test set of each dataset are non-overlapping to ensure basic fairness.

In addition, we also keep the size of validation sets consistent with the training sets to further ensure unique fairness in 'few-shot' settings (Perez et al., 2021), as shown in Table 1.

Experiments are conducted on K=1/2/4/8/16 for K-shot learning tasks. For the evaluation metric, we use an average accuracy over three randomly picked seeds. In order to ensure fairness, we use a fixed template for each task to highlight the performance of different label mapping methods. For the same reason, we uniformly use RoBERTa-large as the pre-training model for all tasks with a fixed learning rate $lr = 0.00003$. The EM-style algorithm is performed in a total of 10 fine-tuning epochs.

4.2 Baselines

As introduced, we mainly compare LPLM with four mainstream label mapping methods. Among them, MLM exploits *prior knowledge of humans*. We separately highlight those results that rely on human knowledge with italics, e.g., the *Manual* method in Table 2. For more convincing results, we employ the four label mapping methods with OpenPrompt (Ding et al., 2022) using the PyTorch framework. For PLM, we use the interface provided by HuggingFace (Wolf et al., 2020) and it is optimized by AdamW optimizer (Kingma and Ba, 2015).

4.3 Template Setting

The core of prompt-based learning is to fine-tune the PLM using templates and label mappings to

Knowledge	Method	SST-2					AG's News					Few-NERD				
		K=1	K=2	K=4	K=8	K=16	K=1	K=2	K=4	K=8	K=16	K=1	K=2	K=4	K=8	K=16
With	*Manual*	*54.07*	*61.58*	*80.76*	*89.12*	*92.04*	*80.12*	*81.59*	*85.28*	*85.73*	*85.81*	*40.39*	*48.94*	*50.43*	*54.17*	*60.29*
Without	Soft	51.63	52.39	61.89	65.11	80.01	48.33	57.02	74.21	79.72	81.44	15.49	32.89	50.07	61.69	64.41
	Search	49.93	58.94	71.66	77.76	88.65	48.02	67.13	78.61	81.68	84.03	23.67	38.57	51.11	56.21	60.47
	Proto(SOTA)	52.83	59.73	78.04	86.67	92.05	73.40	79.71	81.34	83.53	85.13	25.08	38.78	54.44	62.62	65.15
	LPLM	**55.49**	**63.37**	**82.63**	**92.78**	**94.08**	**80.39**	**84.06**	**86.16**	**86.86**	**87.29**	**40.52**	**51.63**	**60.46**	**64.41**	**66.04**

Table 2: The overall performance. *Manual*, Soft, Search, and Proto represent the four baselines, *Manual Label Mapping*, Soft Label Mapping, Search-based Label Mapping, and Prototypical Label Mapping, respectively.

find the best class label for each instance. For all compared label mapping models in our experiments, we use the identical initialed PLM with the same template for each dataset. In this setting, the improvements of different label mapping methods can be observed obviously. The templates are selected from the ACL22 Best demo OpenPrompt platform as shown in Table 3.

Dataset	Template
SST-2	<TEXT> is <MASK>.
AG's News	[Category : <MASK>] <TEXT>
Few-NERD	<TEXT> In this sentence, <ENTITY> is a <MASK>.

Table 3: The selected templates for each dataset.

4.4 Value Selection

To address the shortcomings of Champion selection strategy mentioned in Section 3.3.2, we use Top-k selection strategy in our experiments. To explore the best value of k in Top-k selection strategy, we conduct a series of experiments with different values for k on all datasets for K=1/2/4/8/16. As shown in Figure 3, as k grows, the classification performance starts to increase and peaks when k is around 50 or 500, then gradually decreases. To elaborate this, first note that we expect to mine more semantics carried in the majority voting results of the instances. For instance, if four words are contained in a class representative, it will carry about twice as much semantic information as the class representative of two-word combination. However, the value of k is not larger the better. When k is larger than a certain threshold, numerous identical words can appear in the class representatives of different classes, which will blur the distinction between the class representatives. This tendency may be related to the interpretability of the PLM, but it is not the focus of this paper. Therefore, we use the values that shows the best performance as a reference for k.

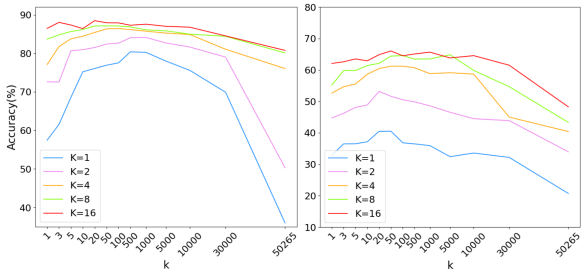

Figure 3: The fluctuation of performance when choosing the different k value for Top-k selection strategy on AG's News (left) and Few-NERD (right).

4.5 Results

4.5.1 Overall Performance

Table 2 shows the overall performance. On SST-2, one of the famous GLUE datasets (Wang et al., 2018), LPLM outperforms the SOTA ALM method by 2.66%, 3.64%, 4.59%, 6.11%, 2.03% for K=1/2/4/8/16 respectively. It even surpasses MLM that uses human prior knowledge by 1.42%, 1.79%, 1.87%, 3.66%, 2.04% respectively.

On AG's News dataset, LPLM has a significant improvement over other ALM methods and outperforms the SOTA benchmark by 6.99%, 4.35%, 4.82%, 3.33%, 2.16% respectively. Moreover, it also surpasses MLM by 0.27%, 2.47%, 0.88%, 1.13%, 1.48% respectively.

In the 66-way entity typing task, it outperforms the SOTA ALM method by 15.44%, 12.85%, 6.02%, 1.79%, and 0.89% respectively. Besides, LPLM surpasses MLM by 0.13%, 2.69%, 10.03%, 10.24%, and 5.75% respectively.

These experimental results show that our proposed LPLM successfully finds the class representatives that are rich in semantic information for the corresponding classes. The discriminativeness of the class representatives generated by LPLM not only outperforms other ALM methods but also outperforms MLM which relies on human knowledge.

Method	K=1	K=2	K=4	K=8	K=16
Manual(One)	*40.39*	*48.94*	*50.43*	*54.17*	*60.29*
Manual(Multi)	***46.63***	*50.60*	*56.78*	*59.96*	*61.98*
LPLM	40.52	**51.63**	**60.46**	**64.41**	**66.04**

Table 4: Comparison with MLM which artificially selects one or multiple words for each of the 66 classes.

4.5.2 Different Number of Words Considered in MLM Class Representatives

Intuitively, assigning one word to each class is the most common approach in MLM. However, in practice, the class representatives can also be obtained by combining multiple words.

In order to eliminate the effect of different numbers of considered words in MLM, we further carry out experiments on Few-NERD and compare the performance of MLM and LPLM under the same setting of both considering multiple words in each class representative. As shown in Table 4, MLM(Multi) only takes advantage when K=1, and it is surpassed by LPLM when K is larger than 1. This shows that as long as there are two or more instances for each class, LPLM can extract more semantic information than the human-specified way. Notably, for MLM(Multi), we manually design the combination of one to five words for each class representative, while for LPLM, each class representative considers the semantics in 500 concrete words. The fairness of the comparison is further discussed in Section 5.4

5 Analysis

In this section, we further analyze the details of LPLM. For convenience, the following experiments are performed on AG's News dataset.

5.1 LPLM vs. Search-based Label Mapping

While the voting process in LPLM may seem similar to the process of finding a subset of candidate words in search-based label mapping, the core motivations of the two mappings are of great difference.

Search-based label mapping contains two separated parts. One is to optimize class representatives from random initialization, while the other is to use them to fine-tune the PLM.

Yet, LPLM operates based on an iterative process. The initial class representatives in the E-step are obtained by PLM with θ. After the class representatives help the PLM optimization in the M-step, in the next round of the E-step, since there is already an updated θ, the new class representatives perform more distinctively and more accurately.

In summary, even though both methods aim to optimize θ, search-based label mapping has to be a blocking process $[Search]_n \rightarrow \theta$, while LPLM can be abstracted into a smooth evolutionary chain, e.g., $[LPLM \rightarrow \theta]_n$.

Method	K=1	K=2	K=4	K=8	K=16
LPLM	**80.39**	**84.06**	**86.16**	**86.86**	**87.29**
LPLM$_{-Topk}$	57.46	72.62	79.35	83.68	86.47
LPLM$_{-Topk-EM}$	48.02	67.13	78.61	81.68	84.03

Table 5: Ablation study of LPLM.

Class Name	Epoch 0	Epoch 3	Epoch 9
Class 0: Sports	‘ Sports’ ‘ News’	‘ Sports’ ‘ Football’	‘ Sports’ ‘ Teams’
Class 1: World	‘ News’ ‘ Politics’	‘ Terrorism’ ‘ Politics’	‘ Terrorism’ ‘ War’
Class 2: Business	‘ News’ ‘ Business’	‘ Finance’ ‘ Companies’	‘ Finance’ ‘ Financial’
Class 3: Sci/Tech	‘ News’ ‘ Technology’	‘ Technology’ ‘ Tech’	‘ Technology’ ‘ Technical’

Table 6: Evolutionary process based on EM-style algorithm for K=16 task on AG's News. The Top-2 words are demonstrated.

5.2 Ablation Study

To show the effectiveness of Top-k selection strategy, we conduct an ablation study. In this subsection, three different settings of models, the original LPLM, LPLM without Top-k strategy, LPLM without Top-k and the EM-style algorithm are compared. For the model without Top-k, Champion selection strategy is used instead, and denoted as LPLM$_{-Topk}$. For LPLM without the EM-style algorithm, it is in fact difficult to concretely exclude the EM-style algorithm because the E-step and the M-step together form a tightly locked process in the model. Though, as discussed in Section 5.1, it may not be very accurate, the search-based label mapping can be used as an alternative. This model is referred to as LPLM$_{-Topk-EM}$ where Top-k strategy is also not applied.

The experimental results of the ablation study are concluded in Table 5. It can be observed that when K is relevantly small, the Top-k selection strategy can greatly improve the performance, and when K is larger, the EM-style algorithm grows more helpful for performance improvement.

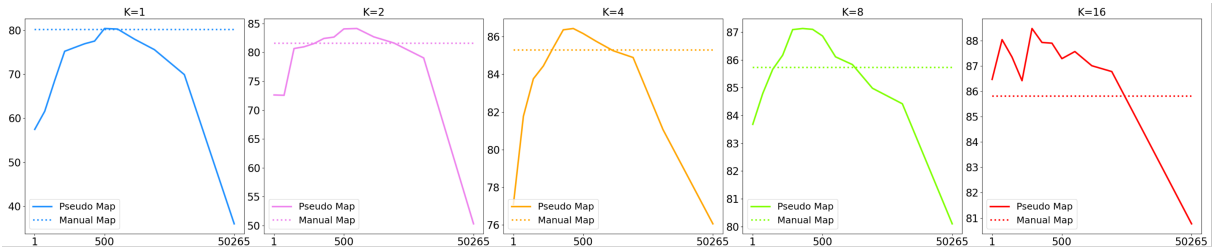

Figure 4: Comparison between MLM and LPLM with different values for k in Top-k on AG's News, where the solid lines represent the performance of LPLM and the dashed lines represent the performance of the MLM.

.

5.3 Evolutionary Process of the EM-style algorithm

To further demonstrate the effectiveness of the EM-style algorithm, we show the evolution of class representatives during optimization. As shown in Table 6, at the beginning, all initialized class representatives are noisy because they contain the same word 'news'. After three rounds of the E-step and M-step process, the dissimilarity gradually increases between class representatives of different classes. After all rounds are performed, each class representative further enhances its distinctions and carries more prominent semantics. This demonstrates that our EM-style algorithm can effectively find the distinct class representative with unique semantics for each class.

5.4 Fairness of Top-k and Breakthrough Point of Label Mapping

The class representative of each class in MLM (Multi) is selected by combining one to five manually-selected words. In LPLM, the number of selected words for each class representative can reach 500 or bigger. This may raise concerns that the comparison between MLM and LPLM is unfair. However, the ultimate goal of ALM is to improve the overall few-shot classification performance by making the model automatically generate N class representatives for N classes without using human prior knowledge. Therefore, the focus should lie on how differentiated the class representatives are and how much semantic information they can contain. MLM helps N class representatives to contain strong semantics from the beginning by introducing human prior information, while LPLM extracts the discriminative semantic information of each class by selecting Top-k words and integrating them into each class representative. The SOTA ALM (Cui et al., 2022) also incorporates the semantics of all words in the vocabulary into their proposed 'class prototypes' when calculating class representatives. Therefore, the core of label mapping research is no more than to find a semantically rich class representative for each class. In conclusion, compressing more words with different semantics into one class representative is of great significance, especially on few-shot tasks.

Moreover, while human ability is limited, there is another advantage of LPLM where it effectively summarizes a big amount of words. That is, the semantic information of hundreds of words can be automatically integrated into the class representatives by the Top-k strategy, which is beyond the reach of humans.As shown in Figure 4, for all settings of K, LPLM always has a range of k values over which it outperforms MLM. Therefore, in practice, it is not necessary to traverse all possible k values. Instead, simply choosing a value of k between 50 and 1000 is sufficient to obtain a better performance than MLM.

6 Conclusion

In this paper, we propose a novel automatic label mapping model that excludes the reliance on human knowledge in manual label mapping methods by automatically generating a set of keywords as each class representative. To increase the distinction of the class representatives, we further introduce an EM-style algorithm to optimize the distribution of latent variables, namely the class representatives, to discover better class representatives and improve the overall classification performance.

Acknowledgements

This work was supported in part by the National Key R&D Program of China under Grant 2021ZD0110700, in part by the Fundamental Research Funds for the Central Universities, in part by the State Key Laboratory of Software Development Environment.

Limitations

Although LPLM achieves remarkable results in our experiments with 2-way, 4-way and 66-way datasets without using prior human knowledge or additional parameters, it may not necessarily be the most appropriate model for datasets where representatives have a large probability space $\mathcal{R}$ or for tasks where N*K is large. According to Equation 10, we need to compute Q_i for each instance, which requires $O(N * K)$ computation complexity. And computing each Q_i need sum over all $W \in \mathcal{R}^{(C^k_{|\mathcal{V}|})^N}$ with a computation complexity of $O((C^k_{|\mathcal{V}|})^N)$, which results in total $O(N * K * (C^k_{|\mathcal{V}|})^N)$ computation complexity at E-step. This property of the EM-style algorithm may lead to an efficiency degradation of LPLM. In our implementation, we adopt to simplify the calculation, incorporating the distribution update process of latent variables into $p_{M_\theta}(W|x_i, y_i, X, Y; \theta, T(\cdot))$ and adopt an approximation of the Q-function with value 1 for the W consistent with the voting result and 0 for the other Ws so that the computation complexity can be reduced to $O(N * K)$.

In brief, how to keep efficiency with large representative selecting space or numerous of tasks becomes an interesting direction for future work.

References

Stephanie Brandl, Ruixiang Cui, and Anders Søgaard. 2022. How conservative are language models? adapting to the introduction of gender-neutral pronouns. *arXiv preprint arXiv:2204.10281*.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Boxi Cao, Hongyu Lin, Xianpei Han, Fangchao Liu, and Le Sun. 2022. Can prompt probe pretrained language models? understanding the invisible risks from a causal view. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5796–5808, Dublin, Ireland. Association for Computational Linguistics.

Boxi Cao, Hongyu Lin, Xianpei Han, Le Sun, Lingyong Yan, Meng Liao, Tong Xue, and Jin Xu. 2021. Knowledgeable or educated guess? revisiting language models as knowledge bases. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1860–1874, Online. Association for Computational Linguistics.

Junfan Chen, Richong Zhang, Yongyi Mao, Hongyu Guo, and Jie Xu. 2019. Uncover the ground-truth relations in distant supervision: A neural expectation-maximization framework. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 326–336, Hong Kong, China. Association for Computational Linguistics.

Junfan Chen, Richong Zhang, Yongyi Mao, and Jie Xue. 2022. Contrastnet: A contrastive learning framework for few-shot text classification.

Rochelle Choenni, Ekaterina Shutova, and Robert van Rooij. 2021. Stepmothers are mean and academics are pretentious: What do pretrained language models learn about you? In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 1477–1491, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Kevin Clark, Urvashi Khandelwal, Omer Levy, and Christopher D. Manning. 2019. What does BERT look at? an analysis of BERT's attention. In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 276–286, Florence, Italy. Association for Computational Linguistics.

Ganqu Cui, Shengding Hu, Ning Ding, Longtao Huang, and Zhiyuan Liu. 2022. Prototypical verbalizer for prompt-based few-shot tuning. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7014–7024, Dublin, Ireland. Association for Computational Linguistics.

Arthur P Dempster, Nan M Laird, and Donald B Rubin. 1977. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society: Series B (Methodological)*, 39(1):1–22.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Ning Ding, Shengding Hu, Weilin Zhao, Yulin Chen, Zhiyuan Liu, Haitao Zheng, and Maosong Sun. 2022. OpenPrompt: An open-source framework for prompt-learning. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 105–113, Dublin, Ireland. Association for Computational Linguistics.

Oliver Eberle, Stephanie Brandl, Jonas Pilot, and Anders Søgaard. 2022. Do transformer models show similar attention patterns to task-specific human gaze? In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4295–4309, Dublin, Ireland. Association for Computational Linguistics.

Tianyu Gao, Adam Fisch, and Danqi Chen. 2021. Making pre-trained language models better few-shot learners. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 3816–3830, Online. Association for Computational Linguistics.

Karen Hambardzumyan, Hrant Khachatrian, and Jonathan May. 2021. WARP: Word-level Adversarial ReProgramming. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4921–4933, Online. Association for Computational Linguistics.

Victor Petrén Bach Hansen, Atula Tejaswi Neerkaje, Ramit Sawhney, Lucie Flek, and Anders Søgaard. 2022. The impact of differential privacy on group disparity mitigation. *arXiv preprint arXiv:2203.02745*.

Pride Kavumba, Ryo Takahashi, and Yusuke Oda. 2022. Are prompt-based models clueless? In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2333–2352, Dublin, Ireland. Association for Computational Linguistics.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. 2021. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *arXiv preprint arXiv:2107.13586*.

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)*, pages 8086–8098, Dublin, Ireland. Association for Computational Linguistics.

Ethan Perez, Douwe Kiela, and Kyunghyun Cho. 2021. True few-shot learning with language models. *Advances in Neural Information Processing Systems*, 34:11054–11070.

Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander Miller. 2019. Language models as knowledge bases? In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2463–2473, Hong Kong, China. Association for Computational Linguistics.

Vinit Ravishankar, Mostafa Abdou, Artur Kulmizev, and Anders Søgaard. 2022. Word order does matter (and shuffled language models know it). *arXiv preprint arXiv:2203.10995*.

Timo Schick and Hinrich Schütze. 2021. Exploiting cloze-questions for few-shot text classification and natural language inference. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 255–269, Online. Association for Computational Linguistics.

Jake Snell, Kevin Swersky, and Richard Zemel. 2017. Prototypical networks for few-shot learning. *Advances in neural information processing systems*, 30.

Taylor Sorensen, Joshua Robinson, Christopher Rytting, Alexander Shaw, Kyle Rogers, Alexia 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)*, pages 819–862, Dublin, Ireland. Association for Computational Linguistics.

Kai Sun, Richong Zhang, Samuel Mensah, Yongyi Mao, and Xudong Liu. 2022. A transformational biencoder with in-domain negative sampling for zero-shot entity linking. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 1449–1458.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. 2018. Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*.

Albert Webson and Ellie Pavlick. 2021. Do prompt-based models really understand the meaning of their prompts? *arXiv preprint arXiv:2109.01247*.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.