Scalable Prompt Generation for Semi-supervised Learning with Language Models

Yuhang Zhou*
University of Maryland
College Park, MD
tonyzhou@umd.edu

Suraj Maharjan*
Amazon
Seattle, WA
mhjsuraj@amazon.com

Beiye Liu
Amazon
New York, NY
beiyeliu@amazon.com

Abstract

Prompt-based learning methods in semi-supervised learning (SSL) settings have been shown to be effective on multiple natural language understanding (NLU) datasets and tasks in the literature. However, manually designing multiple prompts and verbalizers requires domain knowledge and human effort, making it difficult and expensive to scale across different datasets. In this paper, we propose two methods to automatically design multiple prompts and integrate automatic verbalizer in SSL settings without sacrificing performance. The first method uses various demonstration examples with learnable continuous prompt tokens to create diverse prompt models. The second method uses a varying number of soft prompt tokens to encourage language models to learn different prompts. For the verbalizer, we use the prototypical verbalizer to replace the manual one. In summary, we obtained the best average accuracy of 73.2% (a relative improvement of 2.52% over even the previous state-of-the-art SSL method with manual prompts and verbalizers) in different few-shot learning settings.

1 Introduction

Pre-training large language models with huge amounts of text corpora in masked language modeling tasks and then fine-tuning the pre-trained language model (PLM) on downstream tasks have shown superior performance in many natural language processing tasks. However, the discrepancy between the pretraining task (masked language modeling objective) and the downstream fine-tuning task (task without MASK token) could lead to unexpected behaviors. Recently, there has been growing research interest in the area of prompt-tuning, where any NLU task is transformed into a cloze task to mimic the pre-training objective of a large masked language model (Kumar et al., 2016; McCann et al., 2018; Radford et al., 2018). Prompt-based learning transforms an input $\mathbf{x}$ into $\mathbf{x}'$ using a prompt function. It makes use of the vast amount of acquired knowledge of PLMs to predict a distribution of tokens at the masked position. The verbalizer then maps the predicted tokens to classes. The main advantage of this approach is that this method works well in a few-shot learning environment (Schick and Schütze, 2021). However, the main disadvantage of this method is the limitation posed by the prompt and verbalizer functions, which require human knowledge to carefully craft them. Such handcrafting work is expensive and not scalable with the increase in the variety of tasks and datasets. For example, in Alexa, there are thousands of domains and manually designing prompts and verbalizer for intent classification for each of them according to the dataset content demand human expertise, which is time consuming and not applicable. It is essential to reduce the human efforts in the process of prompt generation. Prompt-based learning requires finding the right tokens in the prompts that align with the task requirement and dataset content. However, since the objective of these prompt tokens is only for the language models to perform the task at hand, it is not necessary for them to be a sequence of words that humans can understand.

Continuous prompt-based learning alleviates the need for human intervention to determine prompt tokens. Instead, it automates the prompt design process. In the literature, there are mainly two methods: i) automatically search for discrete prompt text tokens (Shin et al., 2020a) ii) automatically learn numerical prompt embeddings (Lester et al., 2021; Li and Liang, 2021; Liu et al., 2021c,b; Hambardzumyan et al., 2021). The main difference between these two approaches is that the first searches for actual discrete tokens from the language model vocabulary, whereas the second method directly learns the embeddings for prompt tokens, which

*Equal contribution. This work was done during Yuhang's internship at Amazon, Alexa AI.

may not be human comprehensible. Similarly, automatic selection of label words (Shin et al., 2020a; Schick et al., 2020a; Gao et al., 2021), soft verbalizer (Hambardzumyan et al., 2021; Liu et al., 2021b), and prototypical verbalizer (Cui et al., 2022) are the methods proposed to eliminate the tedious process of manually defining verbalizer mapping functions.

Most of these continuous prompt and automatic verbalizer methods focus on supervised learning (SL) settings but ignore their generalization under semi-supervised learning (SSL) settings. The previous state-of-the-art (SoTA) SSL method with various manual prompts and verbalizers has shown superiority over SL language models with a single manual prompt (Schick and Schütze, 2021). In this SSL pipeline, we normally train several labeler models with different manual prompts to capture diverse information from the limited training data and make use of them to annotate a huge amount of unlabeled data. Having to design several manual prompts and verbalizer models for SSL settings and applying them across multiple datasets and tasks will exacerbate the scalability and cost problem. In this paper, we tackle the problem posed by manual prompt and verbalizer design and propose automatic methods to fully automate the design of diverse prompts and verbalizers in SSL settings. Our main contributions are as follows.

- We propose methods to generate various prompts by adding multiple demonstration examples with continuous prompt tokens for use in SSL settings.

- To the best of our knowledge, we are the first to completely eliminate human involvement in designing multiple prompts and verbalizers in SSL settings and obtain similar and even better performance than the SoTA methods with manual prompts and verbalizers.

- We empirically show that using the automatic verbalizer with manual prompts can achieve a similar performance to manual verbalizers' performance in the SSL pipeline.

2 Methodology

Our overall prompt-based SSL workflow follows Pattern-exploiting Training (PET) semi-supervised learning setting (Schick and Schütze, 2021). PET first transforms the input sequence x to a cloze question containing a single MASK token. Next, it uses PLM to fill in the value of the MASK token and applies verbalizers to map the output tokens to the class labels $y \in Y$. They devise a semi-supervised framework to produce soft labels on a large amount of unlabeled data, which are later used to train a final supervised classifier $\mathbf{F}$. They report strong performance over other supervised prompt-tuning methods and other semi-supervised approaches without prompts across multiple NLU tasks. Before this paper, the PET approach was the state-of-the-art (SoTA) framework that integrates the prompt-tuning method into the SSL pipeline.

The PET method fine-tunes multiple PLMs with different prompts. It introduces diversity in the prompts by manually designing several prompts using domain and task knowledge. Similarly, it uses human expertise to design verbalizer mappings for each of the datasets based on the knowledge of the tasks. Here, we use continuous and automatic prompts and verbalizers, thus eliminating the need for human involvement in designing manual prompts and verbalizers.

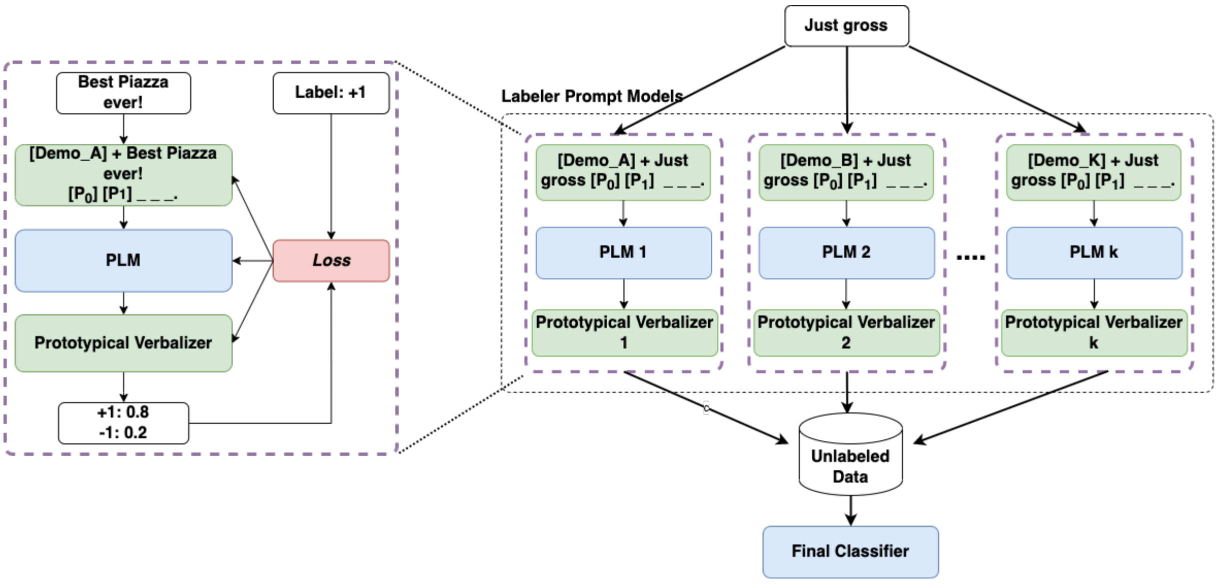
2.1 Overall Pipeline

Figure 1 shows the overall pipeline of our proposed methods. Unlike the original PET pipeline with manual prompts and verbalizers, we use a prompt generation function to generate multiple automatic prompts. Each PLM with automatic prompts serves as a labeler model. We train each of these prompts + automatic verbalizer models with a labeled dataset $\mathcal{T}$ in few-shot settings. With an input sequence $x_t \in \mathcal{T}$ and the given label y_t, we first use the prompt function P to transform x_t into a sequence $P(x_t)$ with a MASK token. The verbalizer then maps the predicted word probability at the masked position to the label probability. For each PLM m, the predicted probability $p_m(y_t|x_t)$ is defined as

$$p_m(y_t|x_t) = \frac{\exp m(y_t|x_t)}{\sum_{y' \in Y} \exp m(y'|x_t)} \quad (1)$$

where $m(y|x)$ is the raw score of PLM m in the masked position. After obtaining the probability, we minimize the cross-entropy loss $\mathcal{L}_c$ between $p_m(y|x)$ and y.

We apply trained labeler models to each sentence $x_d \in \mathcal{D}$ in the unlabeled dataset $\mathcal{D}$ and get the probability $p_m(y_d|x_d)$ for each trained model. We then take the average of these probabilities from each



Best Piazza ever!
Label: +1
[Demo_A] + Best Piazza ever! [P0] [P1] ___.
PLM
Loss
Prototypical Verbalizer
+1: 0.8
-1: 0.2
Just gross
Labeler Prompt Models
[Demo_A] + Just gross [P0] [P1] ___.
[Demo_B] + Just gross [P0] [P1] ___.
[Demo_K] + Just gross [P0] [P1] ___.
PLM 1
PLM 2
PLM k
Prototypical Verbalizer 1
Prototypical Verbalizer 2
Prototypical Verbalizer k
Unlabeled Data
Final Classifier

Figure 1: Semi-Supervised Learning (SSL) Training. Multiple diverse prompt-based learning models are trained on labeled data to soft label huge amounts of unlabeled data. The soft labels serve as ground truth to train the final classifier. $P_0, P_1, \ldots$ are continuous prompt tokens and $Demo\_A, Demo\_B, \ldots$ are demonstration examples randomly sampled from the training data.

trained model m as the ground-truth probability,

$$p_t(y_d|x_d) = \frac{1}{Z}\sum_{m\in M} p_m(y_d|x_d)$$

where Z is the total number of trained PLMs with different automatic prompts. Eventually, we fine-tune a final pre-trained language model $\mathbf{F}$ with a standard sequence classification head. We use the Kullback-Leibler (KL) divergence as our loss function. Given $p_t(y_d|x_d)$ and the predicted probability $\hat{p}(y_d|x_d)$ of the final classifier $\mathbf{F}$, the divergence loss $\mathcal{L}_{div}$ for this input is:

$$\mathcal{L}_{div}(x_d) = \sum_{y'\in Y} p_t(y'|x_d)\log\left(\frac{p_t(y'|x_d)}{\hat{p}(y'|x_d)}\right) \quad (2)$$

The final classifier $\mathbf{F}$ is then applied to the test set to obtain the results.

Schick and Schütze (2021) introduce diversity in their SSL pipeline by training several models with different manual prompts and applying them to softly label a large number of unlabeled datasets. The diversity between manual prompts brings consistent improvements. We observe that diverse knowledge learned by the language model is mostly introduced by the prompts rather than manual verbalizers, since in most datasets, they prepare only one manual verbalizer but multiple prompts for experimentation. Thus, we propose replacing manual prompts with multiple automatic prompts and using the same automatic verbalizer for all labeler models.

2.2 Continuous Prompt Design

Several researchers have proposed methods to automate the prompt design process (Liu et al., 2021c; Li and Liang, 2021; Lester et al., 2021). In most of these methods, they insert the continuous trainable prompt tokens into the input sentence and learn the token embeddings during the training process. However, existing continuous prompt-based learning methods do not consider their application in the PET pipeline, which requires training several labeler models (Schick and Schütze, 2021), in order to learn diverse knowledge from the datasets. Therefore, most methods do not define strategies to compose multiple continuous prompts. We propose two scalable solutions to introduce different variables in the design of continuous prompt labeler models (various demonstration examples or varying numbers of continuous prompt tokens). We expect that with these diverse continuous prompts, trained language models can fully learn different aspects of knowledge from the training dataset.

2.2.1 Scalable Prompt Generation

Inspired by the P-tuning (Liu et al., 2021c) method, we insert multiple continuous prompt tokens p_n into the input sentence x, transforming it into

$[\mathbf{x}][p_0, p_1, \ldots, p_n]$[MASK].. Different from the original P-tuning method, we invent two scalable designs to make it suitable for the prompt-based SSL pipeline.

Add Demonstration Examples: In this method, we add different demonstration examples to construct diverse prompts. This is similar to the prompt augmentation method, in which one chooses to add additional answered prompts to demonstrate what kind of answer the language model should produce for the MASK token (Liu et al., 2021a). These additional answered prompts are called the demonstration example $[demo]$. To reduce the discrepancy between the demonstration examples and the input sentences, we also add a fixed number of continuous prompt tokens p between the demonstration sentence and its true label. Thus, given the labeled input $\mathbf{x_d}$ and its corresponding ground-truth label $\mathbf{y_d}$ from the labeled training dataset, we construct the demonstration example as $[demo] = [\mathbf{x_d}][p_0, p_1, \ldots, p_n][\mathbf{y_d}]$, where $p_0, p_1, \ldots, p_n$ are continuous prompt tokens.

After composing the demonstration examples $[demo]$, given a training input from the labeled dataset $x_t = (s_i, s_2, \ldots, s_k) \in \mathcal{T}$ and label y_t, where $s_i, s_2, \ldots, s_k$ are input tokens for the PLM m, the prompt template function $P_1(x_t)$ is formally defined as

$$\begin{aligned} &P_1(x_t)_1 = [demo_1][\mathbf{x_t}][p_0, \ldots, p_n][\text{MASK}] \\ &\cdots \\ &P_1(x_t)_k = [demo_k][\mathbf{x_t}][p_0, \ldots, p_n][\text{MASK}] \end{aligned} \tag{3}$$

We create multiple prompts by adding different demonstration examples with exactly n continuous soft tokens with the input sentence. Demonstration examples are randomly sampled from the labeled datasets. For longer input sentences, we first truncate the length of $[demo]$ to fit the PLM requirement. Our intuition is that different demonstration examples will introduce the diversity necessary for SSL experimentation.

Vary Soft Token Numbers: In this method, we vary the number of continuous prompt tokens between different labeler models. In other words, this prompt function $P_2(x_t)$ with input sentence x_t is defined as

$$\begin{aligned} &P_2(x_t)_1 = [\mathbf{x_t}][p_0, p_1, \ldots, p_{n_1}][\text{MASK}] \\ &\cdots \\ &P_2(x_t)_k = [\mathbf{x_t}][p_0, p_1, \ldots, p_{n_k}][\text{MASK}] \end{aligned} \tag{4}$$

and each of the labeler models uses different n_1 to n_k number(s) of continuous prompt tokens p. Here, we do not prepend the demonstration example. Our intuition is that given different numbers of continuous prompt tokens, the optimized learned continuous prompts may also be different. For example, for AG's News dataset (Zhang et al., 2015a) about news topics, the optimized prompts with two continuous prompt tokens could be: $[[\mathbf{x}]$[News :][MASK]], while optimized prompts with three continuous prompt tokens could be: $[[\mathbf{x}]$[the category is][MASK]]. We expect that varying the number of continuous prompt tokens will have a similar impact to manually constructing different prompts.

2.2.2 Reparameterization Block

Li and Liang (2021) and Liu et al. (2021c) empirically show that directly updating the parameters in continuous prompts leads to unstable optimization. Hence, we first feed prompt embeddings through a reparameterization block rather than directly feeding them into the PLM. Our reparametrization block uses a bidirectional LSTM (Hochreiter and Schmidhuber, 1997) network with a two-layer $ReLU$ activated multilayer perceptron (MLP) (Liu et al., 2021c; Li and Liang, 2021).

We denote the random initialized tokens as p_i' and the real input embeddings, which are fed into the PLM, as p_i. The p_i are the output of the bidirectional LSTM network and the MLP as,

$$p_i = \text{MLP}([\text{LSTM}(p'_{0:i}), \text{LSTM}(p'_{i:n})])$$

where p_i is also the soft token used in Equations 3 and 4.We learn the optimized continuous prompt tokens $\hat{p}_{0:n}$ during the training process. With the downstream cross-entropy loss $\mathcal{L}_c$, we can differentially optimize the continuous prompts by:

$$\hat{p}_{0:n} = \underset{p}{\text{argmin}} \mathcal{L}_c(p_m(x|y), y) \tag{5}$$

2.3 Automatic Verbalizers

There are several automatic verbalizer methods that eliminate the need for human intervention and expertise to build mapping functions. We experiment with three types of automatic verbalizers: i) soft verbalizer (Hambardzumyan et al., 2021), ii) prototypical verbalizer (Cui et al., 2022), and iii) search-based verbalizer (Schick et al., 2020b).

Cui et al. (2022) experimentally show the superiority of the prototypical verbalizer in a supervised learning environment. However, they did not conduct such experiments for SSL settings. Our experiment with the SSL PET method (details in Section 3.5) with different automatic verbalizers showed that the prototypical verbalizer performed better than the soft verbalizer and the search-based verbalizer on multiple datasets. Thus, we choose to use the prototypical verbalizer as a replacement for the manual verbalizer.

With the optimized embedding of the MASK token from PLM m and the ground-truth labels y, the prototypical verbalizer learns the prototype vectors for each class using contrastive learning (Oord et al., 2018). The prototypical verbalizer first initializes a prototype embedding for each class label and then uses the embedding of the MASK token as the instance embedding. It uses instance-instance loss $\mathcal{L}_{ins}$ to maximize intra-class similarity and minimize inter-class similarity. Similarly, it uses instance-prototype loss $\mathcal{L}_{proto}$ to maximize the similarity between the prototype and instances belonging to the same class and minimize the similarity of instances belonging to other classes. The probability distribution of the MASK token for each class is calculated by the cosine similarity between the instance embedding and each optimized prototype embedding. For inference, it assigns the class of the prototype vector to the instance with the highest probability score, which is computed by taking the similarity scores of the instance vector with the prototype vectors and normalizing them.

2.4 Training and Inference Strategy

All model parameters to be optimized are randomly initialized. As mentioned in Section 2.2.2 and 2.3, we update the parameters in the continuous prompts and PLMs with the loss $\mathcal{L}_c$ and optimize the parameters in the verbalizers with the loss $\mathcal{L}_{ins}$ and $\mathcal{L}_{proto}$. Instead of summing all losses together, our training strategy is to first freeze the parameters in the prototypical verbalizer and then train the parameters in the reparameterization block and the PLM together with the cross-entropy loss $\mathcal{L}_c$. Then we freeze the learned parameters and train the parameters in the prototypical verbalizers with instance-instance loss $\mathcal{L}_{ins}$ and instance-prototype loss $\mathcal{L}_{proto}$. After training all labeler models and obtaining the class probability on the unlabeled dataset, we use $\mathcal{L}_{div}$ to fine-tune the final language model classifier. During inference, we do not rely on any prompt-based labeler models and directly use the final fine-tuned language model $\mathbf{F}$ to predict on the test dataset.

3 Experiments

To verify the effectiveness of our framework, we conduct multiple semi-supervised learning experiments with several strong baseline frameworks on the commonly-used NLU benchmarks.

3.1 Dataset Collection

We experiment with five different datasets[1]: AG's News (Zhang et al., 2015a), Yahoo Answers (Zhang et al., 2015b), MNLI (MultiNLI, Multi-Genre Natural Language Inference, Williams et al. (2018)), RTE (Recognizing Textual Entailment, Dagan et al. (2006)) and CB (Commitment-Bank, de Marneffe et al. (2019)). AG's News and Yahoo answers are topic classification (TC) datasets, while MNLI, RTE, and CB are natural language inference (NLI) datasets. In Table 1, we provide the number of distinct classes, the unlabeled dataset size used for SSL, and the test size for all five datasets. Details about the design of prompts and verbalizers can be found in Appendix A.

Dataset	Task	#Class	#Unlabeled	#Test
AG's News	TC	4	40,000	7,600
Yahoo	TC	10	100,000	60,000
CB	NLI	3	30,000	56
RTE	NLI	2	20,000	277
MNLI	NLI	3	30,000	9,815

Table 1: Data statistics. TC= Topic Classification, NLI= Natural Language Inference

We perform multiple experiments in few-shot settings for all datasets. For few-shot experiments, we use $1, 5, 10, 20$ examples per class for all datasets except for CB and RTE, where we experiment with 32 examples to align with earlier research work (Schick and Schütze, 2021). We report the average accuracy for the evaluation across three runs of each experiment with three different random seeds.

3.2 Proposed Models

Demo+Soft Tokens PET: The first method is to replace the manual verbalizer with the prototypical verbalizer and manual prompts with demonstration examples and continuous prompt tokens.

[1]We downloaded these datasets using the script provided by OpenPrompt `https://github.com/thunlp/OpenPrompt`

Vary Soft Tokens PET: The second method is to introduce diversity by varying the number of continuous prompt tokens, and we use the prototypical verbalizer across multiple labeler models.

3.3 Models for Comparison

We design several strong baseline experiments in addition to our proposed models and also perform an ablation study to show the superiority of our proposed models in multiple NLU tasks.

3.3.1 Baseline Models

Fine-tune: This is a supervised method, where we directly fine-tune the RoBERTa-large PLM with training examples in different few-shot settings. In this method, we do not leverage the unlabeled data.
Prototypical Verbalizer PET: This is a semi-supervised learning method similar to Schick and Schütze (2021), but we replace the manual verbalizer with the prototypical verbalizer and keep the manual prompts. Experiments with this setup will show the benefits of applying automatic verbalizer in the PET framework.
Manual PET: This is a semi-supervised learning method from Schick and Schütze (2021). Our main goal is to show that, with our proposed method, we can achieve similar or better results than this manual method.

There are other SSL methods that rely on data augmentation without prompt tuning, such as UDA (Xie et al., 2020) and MixText (Chen et al., 2020). Since their performance is consistently worse than the Manual PET model across multiple datasets (Schick and Schütze, 2021), we do not choose these models for comparison in this work.

3.3.2 Model Intervention for Ablation Study

Fixed Soft Tokens PET: This semi-supervised learning method is similar to our second proposed method, where we vary the number of continuous tokens to create multiple prompts. However, here we keep the number of continuous tokens fixed and do not add demonstration examples as well. This experiment will help us to understand the importance of diversity introduced by varying continuous tokens in prompt design.
Demo+Soft in SL: This is a supervised method, where we use a prompt template to transform the input by adding a randomly selected demonstration example from the training data and a fixed number of continuous prompt tokens to the input, and we use the prototypical verbalizer for classification. We use RoBERTa-large for PLM. With this experiment, we try to understand the power of semi-supervised learning methods with multiple prompts over supervised training.

3.4 Implementation Details

We use the RoBERTa-Large model (Liu et al., 2019) as our PLM for all of our experiments. We use AdamW as our optimizer with a learning rate of $1e-5$ and a weight decay of 0.01 with linear scheduler, batch size of 2, and trained for 5 epochs. The reparameterization block contains 2-layer bidirectional LSTM and 2 linear layers with ReLU activation function. The hidden dimension of the linear layer and LSTM layer is 768, as well as the hidden dimension of Roberta-Large. We train the parameters in the reparameterization block and the PLM together. For the prototypical verbalizer, we base our implementation on the Pytorch[2], Huggingface transformer[3], and OpenPrompt[4] frameworks (Ding et al., 2021). The number of continuous prompt tokens is consistent 5. For our Vary Soft Tokens PET, we prepare 5 prompts for each dataset and the number of soft tokens in each prompt ranges from 1 to 5.

3.5 Results of Multiple Automatic Verbalizers

Datasets		**SSL PET**		
	# instances	**SoftVerb**	**SearchVerb**	**ProtoVerb**
AG's News	10	49.4	**80.5**	77.2
Yahoo	10	11.8	34.0	**51.9**
CB	32	**88.7**	73.2	85.7
RTE	32	48.2	50.2	**52.8**
MNLI	10	39.0	37.0	**50.0**

Table 2: Average accuracy on different datasets by replacing manual verbalizers with automatic verbalizers in the PET SSL setup. For CB and RTE, we use 32 training examples, whereas for other datasets, we use 10 training examples to train labeler models. The best performance is marked in bold.

To understand which automatic verbalizer is a better replacement for manual verbalizer, we first experiment with three automatic verbalizers: soft verbalizer (Hambardzumyan et al., 2021; Liu et al., 2021c,b), search verbalizer (Gao et al., 2021; Shin et al., 2020a; Schick et al., 2020a), and prototypical verbalizer (Cui et al., 2022). For all of these

[2]`https://pytorch.org/`
[3]`https://huggingface.co/`
[4]`https://github.com/thunlp/OpenPrompt`

experiments, we apply experimental setups similar to PET paper, but only replace the manual verbalizer with the automatic verbalizer (Schick and Schütze, 2021). Table 2 shows the average accuracy over three runs with three different seeds on different datasets with these verbalizers. From Table 2, the prototypical verbalizer shows better performance than other verbalizers for three (Yahoo, RTE, and MNLI) out of five datasets. The search verbalizer and soft verbalizer models perform better than the prototypical verbalizer model only on one dataset each. Since the prototypical verbalizer performs better than other verbalizers in majority of the datasets, we decided to use this as our automatic verbalizer.

3.6 Comparison with Manual PET

With the prototypical verbalizer as our automatic verbalizer, we then experiment with our proposed methods for automatic prompt design. Table 3 shows our results on different datasets and tasks in the few-shot setting. Table 3 shows that by only replacing the manual verbalizer with the prototypical verbalizer (column **Protoverb**) and keeping other aspects of the experiment the same as the PET method, we can achieve slightly lower performance (70.1 average accuracy) compared to Manual PET (71.4 average accuracy) (Schick and Schütze, 2021). This shows that to eliminate human involvement in designing verbalizers, we can simply replace the manual verbalizer with the prototypical verbalizer with only a little performance sacrifice.

For our next set of experiments, we replace manual prompts with our proposed method, automatically creating multiple prompts. The first method (Demo+Soft Tokens PET), which adds randomly sampled demonstration examples from training data with a fixed number of trainable continuous prompt tokens with input, achieves better performance than Manual PET method. The next method (Vary Soft PET), in which we vary the number of continuous trainable tokens, also achieves better performance than Manual PET method. For topic classification tasks, under multiple few-shot settings, the average accuracy of Demo+Soft and Vary Soft PET are 77.0 and 77.3, respectively, while the average accuracy of Manual PET method is 77.1. Similarly, for NLI datasets under different few-shot settings, the average accuracy of our Vary Soft PET method is 69.6 and Demo+Soft Tokens PET method is 70.7. Both of these results are better than Manual PET method (67.7). Furthermore, across all these datasets, Demo+Soft Tokens PET and Vary Soft PET achieve an average performance of 73.2 and 72.6, respectively. These results are better than Manual PET (71.4) method. This experiment shows that it is possible to completely eliminate human involvement and expertise in designing prompts and verbalizers for the SSL pipeline with even better performance.

We also observe that for the case of one-shot experiments with MNLI dataset, Demo + Soft PET method obtains an accuracy of 36.1, which is much worse than other prompt baseline models. This may be due to randomly sampled $[demo]$ examples, as previous studies have shown that the choice of examples in the few-shot setting can result in high-variance performance (Lu et al., 2021). In future work, we can utilize sentence embeddings to make intelligent decisions while selecting demonstration examples.

3.7 Ablation Study

3.7.1 Impact of Semi-supervised Learning

We compare our proposed methods with supervised learning methods: fine-tuning and prompt-based tuning methods (Demo+Soft in SL). All semi-supervised learning methods perform significantly better than supervised learning methods. Traditional fine-tuning methods perform the worst (45.1 average accuracy) on different datasets and tasks. Demo+Soft in SL method is similar to our proposed Demo+Soft Tokens PET method but does not make use of unlabeled data. Demo+Soft in SL performs better than the fine-tuning method and achieves an average accuracy of 68.7 on multiple datasets and tasks in different few-shot settings. Both of the supervised learning methods perform worse than any SSL prompting model, indicating the necessity of the SSL pipeline in NLU tasks.

3.7.2 Impact of Diversity in the Prompts

In order to understand the effect of introducing diversity through multiple prompts in SSL, we devise another experiment, where we use the SSL setup but use only **one** prompt labeler model (not adding a demonstration example but using trainable soft tokens) to label unlabeled data. We name this method as Fixed Soft Tokens PET. Table 3 shows that in most comparisons (13/14), our proposed Vary Soft PET or Demo+Soft PET method achieves better performance. When comparing with the Fixed Soft

Dataset	# Training	Semi Supervised Learning PET					Supervised	
		Demo+Soft	Vary Soft	Fixed Soft	Protoverb	Manual	Fine-Tune	Demo+Soft
Topic Classification								
AG's News	1	**83.5**	81.3	82.8	80.0	80.7	25.7	62.2
AG's News	5	87.6	**88.0**	87.3	87.3	87.8	32.6	84.9
AG's News	10	88.3	88.3	86.5	88.7	**88.8**	58.3	87.2
AG's News	20	88.8	**89.3**	88.9	89.2	89.2	86.1	88.0
Yahoo	1	61.1	**62.9**	59.6	62.0	62.3	10.7	55.6
Yahoo	5	67.4	67.9	67.1	67.8	**68.0**	12.1	65.2
Yahoo	10	68.9	69.5	69.1	**70.0**	69.5	37.8	67.0
Yahoo	20	70.7	**71.0**	70.4	70.9	70.7	66.7	66.5
TC Avg	-	77.0	**77.3**	76.5	77.0	77.1	41.2	72.1
Natural Language Inference								
MNLI	1	36.1	51.7	**52.7**	44.2	44.8	34.3	35.1
MNLI	5	51.2	**58.1**	57.7	55.3	55.2	33.5	46.9
MNLI	10	60.4	57.8	58.4	**62.3**	60.5	34.3	54.4
MNLI	20	64.0	64.7	60.5	**69.6**	68.6	35.0	41.9
CB	32	**88.7**	88.1	88.7	85.7	86.9	60.7	87.6
RTE	32	**70.4**	62.5	62.6	52.8	58.8	48.1	67.4
NLI Avg	-	**70.7**	69.6	69.5	65.5	67.7	47.7	66.5
Overall Avg	-	**73.2**	72.6	72.3	70.1	71.4	45.1	68.7

Table 3: Few-shot experiment results (average accuracy) on different datasets with our proposed methods in PET SSL setup. For CB and RTE, we use 32 training examples, whereas for other datasets we use $\{1, 5, 10, 20\}$ randomly selected examples per class for few-shot learning experiments. The best performance is marked in bold. Note that to report the average results for NLI task, we first average over the MNLI results under different few-shot settings, and then average over the three NLI datasets to give each task equal weight. The overall average results are computed following a similar approach, giving each dataset an equal weight.

PET, our proposed Demo+Soft PET shows an improvement of average accuracy from 72.3 to 73.2 ($p < 0.05$ by paired t test) (Hsu and Lachenbruch, 2014). Moreover, both Demo+Soft and Vary Soft PET methods obtain better average performance than the Fixed Soft Tokens PET in NLI and topic classification tasks. These results show the importance of diversity introduced by multiple prompt labeler models.

4 Related Work

4.1 Language Model Prompting

Cui et al. (2021) authors fine-tuned the pre-trained generative language model, BART, with a predefined template ($candidate\_span$ is a $entity\_type$ entity) for NER classification. Wang et al. (2021) proposed Entailment as Few-shot Learner (EFT) method, which transforms classification tasks into natural language textual entailment tasks and then fine-tunes the LM. The transformation also makes it easy to leverage unsupervised contrastive data augmentation methods to add pairwise examples to the limited annotated data. This setting further showed an average 2.7% improvement in 15 different NLP tasks. In addition to using the prompts for supervised learning, PET is the SoTA method to adapt the manual prompts along with semi-supervised learning to obtain strong performance across multiple NLU tasks. (Schick and Schütze, 2021).

4.2 Automatic Prompts and Verbalizers

Shin et al. (2020a) used a gradient-guided search to find the discrete tokens for prompts based on task accuracy, initialize tokens, and then fine-tune the LM. For automatic label token selection, they first train a logistic regression classifier from the contextualized embedding of the MASK token and then predict the score from MLM's output word embeddings. They select the top-k highest scoring words for each label. They showed better performance over manual prompting methods for sentiment classification and textual entailment tasks. Similarly, instead of using a gradient-guided search for prompt tokens, Li and Liang (2021) and Lester et al. (2021) attached prefix vectors and learned the embeddings for prefix vectors by keeping the LM model parameters frozen. Liu et al. (2021c) proposed P-tuning, which replaces the input embeddings of pre-trained language models with its differentiable output embeddings, using the pattern based on human design. Liu et al. (2021b) optimized and adapted the Prefix Tuning model for NLU. Vu et al. (2021) proposed to learn soft prompt embeddings from one or more source tasks and then transfer them to initialize the prompts for the target task. In addition, they also proposed an efficient retrieval approach to find task embeddings and predict the most transfarable source tasks for a given novel target task.

Several automatic verbalizers, such as search-based verbalizers, soft verbalizers, and prototypical verbalizers, have been proposed to automate the design of the verbalizer mapping function. Search-based verbalizers aim to find the appropriate tokens to replace human selection (Schick et al., 2020a; Shin et al., 2020b; Gao et al., 2020). Both soft verbalizers and prototypical verbalizers learn trainable class or prototyope embeddings during the training process (Cui et al., 2022; Zhang et al., 2021; Hambardzumyan et al., 2021).

Mahabadi et al. (2022) proposed a prompt-free method (PERFECT) to train the language model, which does not rely on manual commands and verbalizers. PERFECT reported performance similar to that of PET (Schick and Schütze, 2021) in a few-shot setting. However, they used a supervised learning setup and compared their results with the single labeler model with one prompt rather than the results from the final classifier. Here, we use a similar SSL setting to Schick and Schütze (2021) and report the results of the final classifier.

5 Conclusions

In this paper, we are able to successfully use automatic prompts and verbalizers in semi-supervised learning settings. We show that our proposed automatic prompt generation methods with prototypical verbalizer can eliminate human engineering in prompt-based SSL setup and achieve similar or better performance than the SoTA Manual PET method. Our methods have the added advantage of being scalable with multiple tasks and datasets. We also empirically verify the power of semi-supervised learning methods, which take advantage of large amounts of unlabeled data, over supervised methods.

In the next steps, we plan to investigate whether we would be able to achieve similar performance by freezing PLMs' parameters and only tuning verbalizer and prompt parameters. This setup will save a tremendous amount of space by making it easy to share and reuse PLMs. Moreover, we plan to explore ways to combine the two proposed methods Demo+Soft PET and Vary Soft PET, which would take advantage of both methods.

6 Limitations

Although we experiment with multiple NLU tasks and datasets, these datasets are only in the English language. Prompt-based learning relies on large language models, which have acquired knowledge through pre-training on huge corpora. With low-resource languages, it might be difficult to get PLMs trained on a huge corpus, which might make it hard to reproduce performance similar to the English corpus. The fine-tuning and inference of PLM requires multiple large GPUs, which might not be accessible to everyone.

Acknowledgments

We would like to thank the anonymous reviewers as well as Wei Ai, Paiheng Xu, Akram Almatarky, Jangwon Kim, Morteza Ziyadi, and Giannis Karamanolakis for reviewing the paper and for providing helpful comments and suggestions.

References

Jiaao Chen, Zichao Yang, and Diyi Yang. 2020. Mixtext: Linguistically-informed interpolation of hidden space for semi-supervised text classification. *arXiv preprint arXiv:2004.12239*.

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.

Leyang Cui, Yu Wu, Jian Liu, Sen Yang, and Yue Zhang. 2021. Template-based named entity recognition using BART. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 1835–1845, Online. Association for Computational Linguistics.

Ido Dagan, Oren Glickman, and Bernardo Magnini. 2006. The pascal recognising textual entailment challenge. In *Machine Learning Challenges. Evaluating Predictive Uncertainty, Visual Object Classification, and Recognising Tectual Entailment*, pages 177–190, Berlin, Heidelberg. Springer Berlin Heidelberg.

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

Tianyu Gao, Adam Fisch, and Danqi Chen. 2020. Making pre-trained language models better few-shot learners. *arXiv preprint arXiv:2012.15723*.

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.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Henry Hsu and Peter A Lachenbruch. 2014. Paired t test. *Wiley StatsRef: statistics reference online*.

Ankit Kumar, Ozan Irsoy, Peter Ondruska, Mohit Iyyer, James Bradbury, Ishaan Gulrajani, Victor Zhong, Romain Paulus, and Richard Socher. 2016. Ask me anything: Dynamic memory networks for natural language processing. In *International conference on machine learning*, pages 1378–1387. PMLR.

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*, pages 3045–3059, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4582–4597, Online. Association for Computational Linguistics.

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

Xiao Liu, Kaixuan Ji, Yicheng Fu, Zhengxiao Du, Zhilin Yang, and Jie Tang. 2021b. P-tuning v2: Prompt tuning can be comparable to fine-tuning universally across scales and tasks. *CoRR*, abs/2110.07602.

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

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

Yao Lu, Max Bartolo, Alastair Moore, Sebastian Riedel, and Pontus Stenetorp. 2021. Fantastically ordered prompts and where to find them: Overcoming few-shot prompt order sensitivity. *arXiv preprint arXiv:2104.08786*.

Rabeeh Karimi Mahabadi, Luke Zettlemoyer, James Henderson, Marzieh Saeidi, Lambert Mathias, Veselin Stoyanov, and Majid Yazdani. 2022. Perfect: Prompt-free and efficient few-shot learning with language models. *arXiv preprint arXiv:2204.01172*.

Marie-Catherine de Marneffe, Mandy Simons, and Judith Tonhauser. 2019. The commitmentbank: Investigating projection in naturally occurring discourse. *Proceedings of Sinn und Bedeutung*, 23(2):107–124.

Bryan McCann, Nitish Shirish Keskar, Caiming Xiong, and Richard Socher. 2018. The natural language decathlon: Multitask learning as question answering. *arXiv preprint arXiv:1806.08730*.

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

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

Timo Schick, Helmut Schmid, and Hinrich Schütze. 2020a. Automatically identifying words that can serve as labels for few-shot text classification. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 5569–5578, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Timo Schick, Helmut Schmid, and Hinrich Schütze. 2020b. Automatically identifying words that can serve as labels for few-shot text classification. *arXiv preprint arXiv:2010.13641*.

Timo Schick and Hinrich Schütze. 2020. It's not just size that matters: Small language models are also few-shot learners. *arXiv preprint arXiv:2009.07118*.

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.

Taylor Shin, Yasaman Razeghi, Robert L. Logan IV, Eric Wallace, and Sameer Singh. 2020a. AutoPrompt: Eliciting Knowledge from Language Models with Automatically Generated Prompts. In *Proceedings of the 2020 Conference on Empirical Methods in*

Natural Language Processing (EMNLP), pages 4222–4235, Online. Association for Computational Linguistics.

Taylor Shin, Yasaman Razeghi, Robert L Logan IV, Eric Wallace, and Sameer Singh. 2020b. Autoprompt: Eliciting knowledge from language models with automatically generated prompts. *arXiv preprint arXiv:2010.15980.*

Tu Vu, Brian Lester, Noah Constant, Rami Al-Rfou, and Daniel Cer. 2021. Spot: Better frozen model adaptation through soft prompt transfer. *arXiv preprint arXiv:2110.07904.*

Sinong Wang, Han Fang, Madian Khabsa, Hanzi Mao, and Hao Ma. 2021. Entailment as few-shot learner. *arXiv preprint arXiv:2104.14690.*

Adina Williams, Nikita Nangia, and Samuel 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, Volume 1 (Long Papers)*, pages 1112–1122, New Orleans, Louisiana. Association for Computational Linguistics.

Qizhe Xie, Zihang Dai, Eduard Hovy, Thang Luong, and Quoc Le. 2020. Unsupervised data augmentation for consistency training. *Advances in Neural Information Processing Systems*, 33:6256–6268.

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

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015a. Character-level convolutional networks for text classification. *Advances in neural information processing systems*, 28.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015b. Character-level convolutional networks for text classification. In *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc.

A Prompts and Verbalizers

A.1 Manual Prompts and Manual Verbalizers

We use the same manual prompts and manual verbalizers for our baseline experiment as used by Schick and Schütze (2021, 2020).

AG's News is a news topic classification dataset with four classes. We use the manual verbalizer that maps class 1-4 to "World", "Sports", "Business" and "Technology". For the input sentence $x = (a, b)$, where a is the news headline and b is the body of the news text, we use the manual prompts below:

$$\mathcal{P}_1(x) = \text{[MASK]} : [a]\ [b]$$
$$\mathcal{P}_2(x) = \text{[MASK]} \text{ - } [a]\ [b]$$
$$\mathcal{P}_3(x) = [a]\ (\text{[MASK]})\ [b]$$
$$\mathcal{P}_4(x) = [a]\ [b]\ (\text{[MASK]})$$
$$\mathcal{P}_5(x) = \text{[MASK] News: } [a]\ [b]$$
$$\mathcal{P}_6(x) = \text{Category : [MASK] } [a]\ [b]$$

Yahoo Questions is another dataset for topic classification with ten classes. We use the same manual prompts as AG's News, but define the manual verbalizer for the Yahoo dataset, which maps the classes 1-10 to "Society", "Science", "Health", "Education", "Computer", "Sports", "Business", "Entertainment", "Relationship" and "Politics".

MNLI is the dataset for textual entailment tasks, consisting of text pairs $x = (a, b)$. We define two manual verbalizer pairs v_1 and v_2. v_1 verbalizer maps class 0-2 to "Wrong", "Right" and "Maybe". v_2 verbalizer maps class 0-2 to "No", "Yes", "Maybe". We use the following manual prompts:

$$\mathcal{P}_1(x) = \text{“}[a]\text{” ? || [MASK], “}[b]\text{”}$$
$$\mathcal{P}_2(x) = [a] \text{ ? || [MASK], } [b]$$

RTE and **CB** are datasets for textual entailment tasks. We use v_1 as the manual verbalizer similar to MNLI task. We use the following manual prompts:

$$\mathcal{P}_1(x) = \text{“}[a]\text{” ? || [MASK], “}[b]\text{”}$$
$$\mathcal{P}_2(x) = [a] \text{ ? || [MASK], } [b]$$
$$\mathcal{P}_3(x) = [a] \text{ ? || [MASK]. } [b]$$
$$\mathcal{P}_4(x) = \text{“}[a]\text{” ? || [MASK]. “}[b]\text{”}$$

A.2 Continuous Prompts

For our proposed models: **Demo+Soft** and **Vary Soft** models, we apply continuous prompts and automatic verbalizers to ensure that the prompt-tuning SSL method can be scaled across multiple datasets. From previous works, we find that few anchor tokens help to improve the performance of NLU tasks (Liu et al., 2021c), so we design two different continuous prompts dependant on the nature of NLU tasks. For the continuous prompt for AG's News and Yahhoo Questions (text classification task), our design is:

$$\mathcal{P}(x) = [a]\ [b] \text{ Category: } [p_0, p_1, \ldots, p_n] \text{ [MASK]}$$

For continuous prompt for MNLI, CB and RTE (NLI tasks), our design is:

$\mathcal{P}(x) = [a]$ $[b]$? $[p_0, p_1, \ldots, p_n]$ answer : [MASK]

The construction of continuous prompts also follow the design of the P-tuning paper (Liu et al., 2021c). Rather than designing multiple manual prompts for different datasets, we can use our proposed methods to automate this process. This reduces human efforts and costs when we scale across multiple datasets and tasks.