TOWARDS BETTER FEW-SHOT AUTOMATIC PROMPT SEARCHING

Juncheng Dong[*], **Oded Schlesinger**[*], **Yuren Zhou**[*]
Duke University
{juncheng.dong, oded.schlesinger, yuren.zhou}@duke.edu

ABSTRACT

Language models pre-trained on unsupervised corpus can be adapted for various supervised learning tasks with only a few samples (or even zero-shot). This is achieved by deft text interaction with the pre-trained language models through prompts (templates and label word mappings for classes) which require significant domain expertise and cannot be reused across tasks. In this project, we follow Gao et al. (2021) to dive into automatic prompt search under few-shot learning scenarios that are challenging but faithfully represent real-world scenarios. In this work, we propose three modifications that attempt to improve the state-of-the-art (SOTA) model from three different aspects. We explore the impact of the proposed methods on the SOTA model and evaluate their impact on the prediction results. Our proposed modifications successfully help the SOTA model to increase its performance when evaluated over tasks which were conducted on the same benchmarks used by the SOTA. Extensive experiments on various scenarios have revealed interesting behaviors of the proposed methods that can inspire future works.

1 INTRODUCTION

While GPT-3 (Brown et al., 2020) provides a powerful language model (LM) with 175 billion parameters, its enormous size could make itself impractical in common situations where only moderately-sized language models such as BERT (Devlin et al., 2018) and RoBERTa (Liu et al., 2019) can be accessed. More importantly, its prohibitively large number of parameters bring severe difficulty when fine-tuning the pre-trained model for downstream tasks. On the other hand, although large LM has demonstrated amazing capability for zero-shot tasks without fine-tuning (Radford & Narasimhan, 2018; Radford et al., 2019), prompt-based fine-tuning with only a small amount of data can lead to great performance boost. As a result, much smaller models such as BERT can have better performance than pre-trained GPT-3 on various few-shot learning tasks (Gao et al., 2021).

The prompt-based prediction is a line of research originally studied for zero-shot learning (Brown et al., 2020), then for few-shot learning (Schick & Schütze, 2020a;b). Prompt-based fine-tuning, compared to regular fine-tuning, relieves the need for a large number of extra parameters that is challenging to fine-tune in few-shot scenarios. However, the prompts in all previous work are hand-crafted, which require lots of domain expertise, while Gao et al. (2021) manages to have automatic prompt generation for both label words and templates in a two-step procedure - (1) automatic label word search with given manually selected template, followed by (2) automatic template search with the label word mapping selected in step (1). Note that the automatic prompt generation algorithm is not iterative (i.e., it ends with only one round of search for label words and template).

In this work, we attempt to improve on the SOTA model (Gao et al., 2021). We first conduct the natural yet missing ablation study to investigate the effect of iterative prompt generation. Then we embark to solve a much more pivotal problem for few-shot prompt generation: while the automatic prompt searching method proposed by Gao et al. (2021) has demonstrated significant improvements over the baseline of few-shot fine-tuning, it is still prone to the overfitting problem because only as few as 16 training examples can be used to select label words and templates in the few-shot setting. In light of the aforementioned problems, we propose the following improvements (modifications):

[*]Equal Contribution. Alphebetical Order

1. Iterative Automatic Prompt Search (section 4.1)
2. Automatic Selection of Label Words (section 4.2)
3. Bayesian Automatic Prompt Search (section 4.3)

2 Related Work

Pre-trained LMs have shown great adaptability and sample efficiency (i.e. they are natural few-shot learners) for various natural languageg processing(NLP) tasks including question answering, semantic classification, reading comprehension (Radford & Narasimhan, 2018; Radford et al., 2019). Specifically, Brown et al. (2020) has demonstrated success in applying pre-trained LMs on new tasks with only text interaction (i.e. prompts) and without any fine-tuning. However, the prompts were manually constructed, thus it requires expertise and knowledge, and cannot be reused across different NLP tasks.

To resolve these problems, Schick et al. (2020) devised an automatic method to search for the label word for each class that gives similar performance to hand-crafted ones; Zhong et al. (2021) and Shin et al. (2020) researched methods to find prompt templates with abundant training samples.
The selection of such core components highly affects the prediction results. A technique that allows for probabilistic sampling of such components as in Holtzman et al. (2019) could potentially further improve the performance. Our work follows realistic, yet challenging, few-shot scenarios such as those appear in (Gao et al., 2021).

3 Prompt-based Few-Shot Learning

We here provide an overview about the state-of-the-art approaches for prompt-based few-shot learning, which loosely summarizes the methods proposed in (Gao et al., 2021).

The prompt in prompt-based few-shot learning refers to the collection of two items: the template and the label words. The template is a short paragraph with an input position and a masked position, and each label word is matched with one distinct class label. Given an input sentence, we aim at correctly classifying it with the set of labels $\mathcal{C}$. The vocabulary set is $\mathcal{V}$ and its size is denoted by $|\mathcal{V}|$. We insert the input sentence into the template, denoted as $\mathcal{T}$, and use language model $\mathcal{L}$ to fill in the masked position with label words. The conditional probability of each label word under the language model corresponds to the soft classification of the input sentence over all $|\mathcal{C}|$ classes. For example, in a positive vs. negative sentiment classification problem, the template could be "[INPUT] It was [MASK]." and the label words could be "great" for the positive sentiment class, "terrible" for the negative sentiment class. For a given classification problem, we want to develop an automatic approach for finding its prompt that doesn't require any prior knowledge of the problem. This requires the automatic selection of label words and the automatic generation of templates. In Gao et al. (2021), it is done in a two-step fashion. Given some initial fixed template, the label words are first selected. Then conditioned on the label words, the template is generated.

The automatic selection of label words is done in a three-fold screening process, using pre-trained masked language model $\mathcal{L}$ such as BERT (Devlin et al., 2018) or RoBERTa (Liu et al., 2019). Leveraging the initial template, we first pick top k words for each class that maximize their likelihood using the initial pre-trained language model. We then pick top n assignments among $k^{|\mathcal{C}|}$ that minimize the loss (misclassification error for classification problem) over the training set. In the end, the best assignment is chosen to minimize the loss over development set. For instance, in the positive vs. negative sentiment classification problem, the top k label words for the positive sentiment class picked in the first step could be {good, great, fantastic}, and the top n assignments picked in the second step could be {{positive class: good, negative class: bad}, {positive class: great, negative class: terrible}, ...}. The label words assignment {positive class: great, negative class: terrible} is chosen in the third step based on development set loss. In this work, we focus on automatic selection of the label words. We use the same manual template proposed by Gao et al. (2021) (interested readers are invited to read in Section A for details of the automatic generation of templates).

While our focus is to discuss the state-of-the-art automatic prompt generation method, we note that in-context learning is often incorporated into the model to boost performance (Gao et al., 2021). For

each input in the training and developing set, we pick a number of other instances in the training set that serve as its similar demonstrations. These similar demonstrations are chosen to maximize the cosine similarity under a pre-trained sentence embedding model SBERT (Reimers & Gurevych, 2019). Templates containing the similar demonstrations and their label words are appended to the original template as input to the masked language model $\mathcal{L}$. For instance, when classifying an input to positive and negative sentiments, given a similar demonstration [DEMO] with label word [LAB] corresponding to its class, we will be using masked language model to fill in the masked position in "[INPUT] It was [MASK]. [DEMO] It was [LAB].".

4 Proposed approaches and modifications

We propose three methods with different motivations to improve the model of Gao et al. (2021). The first modification is to repeat the automatic prompt search and explore the performance of the output prompt at different rounds. The motivation is that iterative automatic search may lead to convergence with better performance. The second modification is to improve on the automatic selection of the label words, replacing the naive top-K methods with more advanced selection metrics. The third modification is to improve on the overfitting behavior of the model in few-shot learning setting by taking a Bayesian approach and construct a posterior distribution of label mappings that can improve generalization. The motivation is that Bayesian methods have known capability to help reduce overfitting (Cawley & Talbot, 2007; Hastie et al., 2009).

4.1 Iterative Automatic Prompt Search

As discussed earlier in section 3, the automatic prompt generation in (Gao et al., 2021) involves two steps, where we first select label words given an initial fixed templates and then generate templates given the selected label words assignment. We could interpret this approach as one iteration of coordinate descent algorithm in the product space of templates and label words assignments. This naturally triggers the question of whether more iterations (2 or 3) of such coordinate descent would boost performance. We refer this modification as the iterative automatic prompt search and provide the pesudocode for it in Section B (in Appendix).

4.2 Automatic Selection of Label Words

The authors of Gao et al. (2021) explore ways of automating the process for the sampling and selection of label words. The goal is to reduce the human involvement required to manually design and choose labels, which usually involves considerable domain expertise and lots of trial-and-error.

This task deals with constructing a label word mapping that improves accuracy on the development set after fine-tuning a language model, given a fixed template. Hence, we are trying to find better sampling methods for label words mapping to different classes in order to improve the prediction results. In the original paper, this is the metric that the authors used to determine the words on which they perform fine-tuning:

$$\text{Top-}k\{\sum_{x_{in}\in\mathcal{D}^c_{train}} \log\ P_{\mathcal{L}}([MASK]=v|\mathcal{T}(x_{in}))\}_{v\in\mathcal{V}} \tag{1}$$

This metric selects the top-k vocabulary words for every class based on their conditional log likelihood in the training set, using the pre-trained LM. Selections of the label words for distinct classes are independent, i.e., the vocabulary's likelihood for a class is not effected by its likelihood for another class. However, whether or not a vocabulary can serve as the label word for a class doesn't only depend on how likely it is observed in that class, but also should depend on how unlikely it is observed in other classes (the same idea shared by the famous TF-IDF model in NLP). Hence, we are interested in checking how it performs when we also consider also the other labels, and how probable the words in the vocabulary are with them:

$$\begin{aligned}\text{Top-}k\{&\sum_{x_{in}\in\mathcal{D}^c_{train}} \log\ P_{\mathcal{L}}([MASK]=v|\mathcal{T}(x_{in}))\\ -\lambda\cdot mean(&\sum_{x_{in}\notin\mathcal{D}^c_{train}} \log\ P_{\mathcal{L}}([MASK]=v|\mathcal{T}(x_{in})))\}_{v\in\mathcal{V}}\end{aligned} \tag{2}$$

In the proposed metric, the mean of the log probability of words over all other labels is being considered with a given weight λ, and being subtracted from the log probability of words over the correct label, as appeared in the original metric. This metric is a generalization of the metric in the original article, in which $\lambda = 0$. Another modification we are interested in checking is how using a different sampling method, Nucleus sampling (Holtzman et al., 2019), as covered in class, affects the results.

$$\underset{v \in \mathcal{V}}{\text{Top-}p}\{\sum_{x_{in} \in \mathcal{D}^c_{train}} \log\ P_{\mathcal{L}}([MASK] = v | \mathcal{T}(x_{in}))\} \tag{3}$$

This sampling method results in the generation of a varying number of label mappings for each class, based on a predefined cumulative probability threshold. By doing so, label mappings are being chosen, according to their probability when evaluated on the development set which leads to different combinations of label mappings, that in turn, changes the final classification results.

4.3 Bayesian Automatic Prompt Search

It is widely known that the model with the "best" parameters that minimizes the training loss is a maximum a posteriori probability (MAP) estimate. Due to this reason, it very likely suffers from overfitting problem. Bayesian methods that use the complete posterior distribution of the model, although often intractable, have shown to be robust the overfitting problems. Motivated by this, we transform the original deterministic method into a probabilistic model with a Bayesian interpretation that is fully tractable. For simplicity, we only present our method for automatic search of label words. Our method can be extended toward automatic search of templates similarly.

The original work's label words search consists of multiple steps of selecting and sorting:

1. It first construct a set of candidates by selecting the top K most likely words for each label. This step reduces the possible number of label words mapping from $|\mathcal{V}|^{|\mathcal{C}|}$ to $K^{|\mathcal{C}|}$.
2. It then reduces the number of candidates using **the training dataset** by testing (on the training set) the zero-shot accuracy of every possible label word mapping ($K^{|\mathcal{C}|}$ in total) and only keeping the ones with top N accuracy. Note that until now the parameters of the model have not been changed from the pre-trained ones yet.
3. The N selected candidates from step 2 are further sorted using **the validation dataset**: for each of the N possible label word mappings, the pre-trained model is fine-tuned with the training dataset; then the N label word mappings are ranked according to the corresponding fine-tuned models' accuracy on the validation dataset.

At the end of the above automatic search, it chooses the "best" one of them (i.e., the label word mapping with the best validation dataset accuracy in step 3). Our main idea for improvement is that: **Instead of disposing the unselected candidates, we assign probabilities to all the candidates by coupling all the steps above with a probability and a Bayesian interpretation.** In the following of this manuscript, our proposed method will be called as *the Bayesian approach* for simplicity.

The Bayesian approach transforms the original method step by step:

1. The Bayesian approach share the same first step as the original method because step 1 is the essential step to have a tractable construction. We use the same K to construct a set of candidates of $K^{|\mathcal{C}|}$ label word mappings.
2. In the second step, the Bayesian approach also reduces the $K^{|\mathcal{C}|}$ candidates to N candidates using the training dataset with the same selection process as the original method. Then, the Bayesian approach assigns the N selected candidates with a probability that is proportional to the zero-shot accuracy:

$$p(\alpha) \propto exp(\frac{\text{ACC}^{tr}_{\alpha}}{\tau_1})$$

where α is a candidate label word mapping; ACC^{tr}_{α} is its corresponding zero-shot accuracy on the training data; τ_1 is the hyper-parameter to regulate the relative importance of

candidates. $p(\alpha)$ can be considered as the **prior distribution** for the selected label word mappings. τ_1 with large value assigns almost identical weights to all the candidates while τ_1 with small value increases the weights of the candidates with high accuracy and decreases the weights of the candidates with low accuracy.

3. In the third step, instead of sorting the N candidates and selecting the best one, the Bayesian approach uses the validation dataset to build a **likelihood function**. Specifically, the likelihood of a candidate label word mapping is proportional to the likelihood of observing the validation dataset:

$$p(D|\alpha) \propto exp(\frac{\text{ACC}_\alpha^{val}}{\tau_2})$$

where ACC_α^{val} is α's accuracy on the validation data; τ_2 is the hyper-parameter to regulate the relative importance of candidates.

With the constructed prior distribution $p(\alpha)$ and likelihood function $p(D|\alpha)$, we build the unnormalized posterior distribution for the candidates label word mappings accordingly. (i.e., $p(\alpha|D) \propto p(\alpha)p(D|\alpha)$). The prediction of a given language model $\mathcal{L}$ with the constructed posterior is then formulated as a weighted sum of individual predictions:

$$P(c|X) = \sum_{\alpha \in \mathcal{A}} P_{\mathcal{L}}(c|X, \alpha)p(\alpha|D) \tag{4}$$

where $\mathcal{A}$ is the set of candidate label word mappings; X is the input; $P_{\mathcal{L}}(c|X, \alpha)$ is the prediction of the language model $\mathcal{L}$ using the label word mapping α.

Note that the proposed Bayesian approach is tractable because (1) the candidate set $\mathcal{A}$ is of moderate size N; (2) the normalizing constant for the posterior is relevant for the prediction since the solutions to $\arg\max_{c \in \mathcal{C}} p(c|X)$ remain unchanged under constant scaling of $p(c|X)$. The Bayesian approach only add $O(1)$ extra operation time to the original method because both $p(\alpha)$ and $p(D|\alpha)$ are built with information already computed in the original method. The Bayesian approach also include the original method as a special case with uniform prior and Dirac-δ likelihood function (only one of the selected candidates have likelihood 1 and the rest all have likelihood 0.)

Extension to Joint Search of Template and Label Words While our Bayesian approach can be easily extended to automatic searching of templates, we present a method to search template and label word mapping together. First we build the posterior for label word mapping $p(\alpha|D)$. Then we build the posterior for template $p(\mathcal{T}|D)$ similarly. The joint posterior is estimated by the product of the individual posteriors, i.e.,

$$p(\mathcal{T}, \alpha|D) = p(\alpha|D)p(\mathcal{T}|D).$$

The predictions are formulated similarly to equation 4.

5 Experiments

We split the dataset into train/validation/test datasets and average the performance over multiples runs with the reported random seeds in Gao et al. (2021). We ran experiments on several machines with varying computational resources.

5.1 Iterative Automatic Prompt Search

The automatic prompt search method in Gao et al. (2021) completes only one iteration of Algorithm B. We aim to find out here whether more iterations would boost the performance of the searched prompt. Due to limited time and computational resources, we replace the large language models in Gao et al. (2021) with their smaller versions such as T5-base, RoBERTa-base, etc, so we can not directly use the reported performance in Gao et al. (2021) as baseline. We will run Algorithm B with iteration number $T = 4$, and the performance at the end of iteration 1 would be the replicated performance from Gao et al. (2021). Our experiments are run with the SST-2 dataset, which is a dataset that classifies movie reviews into positive or negative sentiment.

In real data applications, there are cases where we have prior knowledge of the classification problem and cases where we do not, so it is important to study the algorithm's performance separately for warm initialization and cold initialization. Starting with a fixed initial template or a fixed initial label words assignment also matters. So together, we consider four groups of initializations: warm template initialization, cold template initialization, warm label words initialization, cold label words initialization. For SST-2 dataset, warm initializations could be "[INPUT] It is [MASK]." for the template and {positive class: positive, negative class: negative} for the label words, whereas cold initializations could be "[INPUT] [MASK]." for the template and {positive class: one, negative class: zero} for the label words. The test prediction accuracy at each iteration is shown in Figure 5.1.

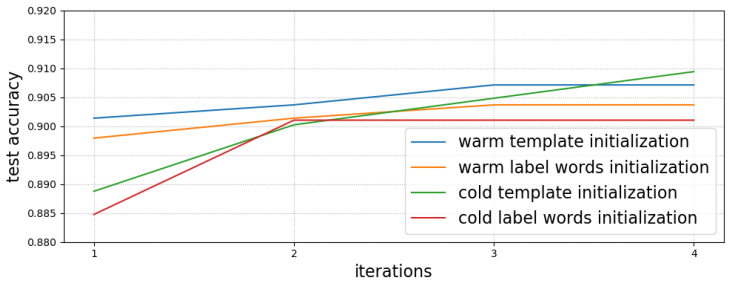

Figure 1: Test prediction accuracy for Algorithm B vs. iterations under warm/cold initialization of template/label words.

5.2 Automatic Selection of Label Words

Various attempts have been made in order to find the optimal value of λ, the coefficient associated with words probability in other labels. Giving high weight to other labels may lead to poor results by paying too little attention to the correct label. Conversely, giving low weight to other labels may lead to a too narrow view of the problem. The template used for experiments which include demonstrations is: $\mathcal{T}$ = It was [MASK]. Eventually, we empirically found $\lambda = 0.5$ to be optimal and yield the best results.

We compared the performance of this modification in several different scenarios over the SST-2 dataset, as used in the original paper. Table 1 lists key experiments results over 100 mappings that are chosen according to their performance on the development set:

Setting \ Accuracy(%)	Original metric	Our metric
Zero-shot	79.5 (3.88)	**79.9 (3.4)**
Prompt-based fine-tuning	89.35 (1.05)	**89.58 (0.84)**
Zero-shot + demonstrations	**66.99 (11.88)**	65.94 (11.31)
Prompt-based fine-tuning + demonstrations	89.97 (0.87)	**90.11 (0.72)**

Table 1: Accuracy in different scenarios over the SST-2 dataset with RoBERTa-base. Our metric uses $\lambda = 0.5$. Original metric results are not taken from the main reference paper, but appear as computed in our experiments. Numbers in ( ) denotes the standard deviation.

Nucleus sampling for label word mappings produced poor results when evaluated in an initial, limited prediction scenario. Therefore, it was decided to focus on the other methods that produced more promising results in the initial evaluation stage.

5.3 Bayesian Automatic Prompt Search

We focus on automatic searching of label word mapping in our experiments. We test the Bayesian method on two established benchmark datasets - SST-2 (single sentence classification task) and MNLI (sentences-pair classification task) with the same setting of 16-shot (16 training examples and 16 validation examples). Below in this manuscript, we will call the method of Gao et al. (2021)

as the baseline model. We use the same number of candidates for label word mapping and the same hyper-parameters of the baseline model. (Please note that due to limitation of computational power and time, we didn't search the best hyper-parameters for our method in large scale. We tried multiple values of large difference for each hyper-parameter in the pilot experiment and found that the parameters used by the baseline model also have good performance. Hence, the reported accuracy is only a coarse estimation of the best accuracy that can be achieved.) We use the same hyper-parameter for both prior and likelihood function in Bayesian method (i.e., $\tau = \tau_1 = \tau_2$). We search τ in four values: $[0.1, 0.25, 0.5, 1.0]$ and choose the τ with the highest validation accuracy.

In order to test the robustness of our method, we compare with the baseline in various different setting: (1)Zero-shot/Fine-tuning ; (2) Large Model (Roberta-large) / Small Model (Roberta-base); (3) With Demonstration / Without Demonstration. The key results are summarized in Tables 2, 3.

Setting \ Accuracy (%)	**Baseline (SOTA)**	**Bayesian Method**
Zero-shot (base)	77.8	86.7(+11.4%)
Prompt-based fine-tuning (base)	88.6	89.2(+0.7%)
Zero-shot (large)	83.6	90.7(+8.5%)
Prompt-based fine-tuning (large)	93.2	93.2(+0.0%)
Zero-shot + demo (base)	86.6	78.1(-9.8%)
Prompt-based fine-tuning + demo (base)	89.0	89.0(+0.0%)
Zero-shot + demo (large)	91.7	81.4(-11.2%)
Prompt-based fine-tuning + demo (large)	92.8	93.7(+0.96%)

Table 2: Comparison of the proposed Bayesian method with the original method (state-of-the-art) as baseline on the SST-2 dataset. base/large after the setting name corresponds to Roberta-base/Roberta-large model. *demo* means with demonstration. Number in ( ) after the performance of Bayesian is the percentage of change with respect to the performance of baseline.

Setting \ Accuracy (%)	**Baseline (SOTA)**	**Bayesian Method**
Zero-shot (base)	48.0	49.3(+2.7%)
Prompt-based fine-tuning (base)	59.1	62.3(+5.4%)
Zero-shot (large)	50.9	55.7(+9.4%)
Prompt-based fine-tuning (large)	69.8	70.1(+0.4%)
Zero-shot + demo (base)	50.5	45.9(-9.1%)
Prompt-based fine-tuning + demo (base)	63.7	63.2(-0.8%)
Zero-shot + demo (large)	55.7	47.4(-14.9%)
Prompt-based fine-tuning + demo (large)	71.7	72.2(+0.7%)

Table 3: Comparison of the proposed Bayesian method with the original method (state-of-the-art) as baseline on the MNLI dataset.

6 Analysis

6.1 Iterative Prompt Search

As shown in Figure 5.1, we can observe that the improvement of running more iterations in Algorithm B mostly depends on the quality of initialization, that is, whether we have good prior knowledge of the problem. When we are using random initializations of label words or template, running more iterations could significantly boost the performance of the generated prompt. But since running every iteration demands an enormous amount of time and computation resources, when we have some descent idea of what label words or template could be (i.e. warm initialization), running one iteration of Algorithm B already generates adequately good results. However, detailed experiment results also show that running more iterations allow a broader exploration of the prompt space, which gives us more diversed prompt candidates even under warm initializations. For instance, the prompt obtained after one iteration typically has template "[INPUT] It's [MASK]!" and label words {positive class: incredible, negative class: terrible}, while the prompt obtained after three or four

iterations could become "Very [MASK]. [INPUT]" for the template and {positive class: special, negative class: confusing} for the label words.

6.2 Label Words Selection

The label words selection experiments as appear in Table 1 emphasise the efficacy and importance of prompt-based fine-tuning, as zero-shot experiments' accuracy is lower and their standard deviation is higher than these of prompt-based fine-tuning results, with both metrics. Overall, the proposed method leads to improvement in accuracy, and lower standard deviation over all but one of the tested scenarios, highlighting its impact on prompt-based fine-tuning. It yielded several surprising, yet accurate, mappings. Among them: laughable/infectious and hideous/welcome, which achieved accuracy of 91.51 when applying prompt-based fine-tuning and 91.28 when applying prompt-based fine-tuning and utilizing demonstrations, respectively. Although these are not clearly associated to positive and negative labels, they outperformed other mappings that are strongly associated to the same labels.

6.3 Bayesian Prompt Search

The Bayesian method has revealed consistent but complicated behavior on the two benchmark datasets. We first consider the model without demonstrations. It is clear that the Bayesian method has helped improve the performance significantly in the zero-shot setting on both small and large language models (see Table 2 and Table 3). Improvements can still be observed but the gain is much lesser in the fine-tuning setting. Surprisingly, demonstrations can severely damage the contribution of the Bayesian method. In the setting of zero-shot learning (where the Bayesian method has brought significant gain for models without demonstrations), applying demonstration decreases the performance significantly ($\geq 10\%$ accuracy). However, this phenomenon disappear in the fine-tuning setting with demonstrations, especially on large models. On both the SST-2 dataset and the MNLI dataset, combination of the Bayesian method with fine-tuning large model and demonstration outperforms the SOTA (as demonstrated by the last rows of Table 2 and Table 3). These behavior are baffling yet consistently observed on both benchmark datasets. These results are even more surprising to the authors because the experiments were finished in the reverse order of the experiments listed in the table, i.e., we first defeat the SOTA, then we start the experiments on various settings. A reasonable conjecture for these observations are difficult to present at current time (one possible reason is the lack of hyper-parameter search). Further research and experiments are necessary to understand these behaviors.

7 Conclusion

The problem of automatic prompt searching was investigated in this project. Three modifications (iterative search, automatic label search, Bayesian search) were proposed to improve on the performance of the SOTA. While the proposed methods have successfully outperformed the SOTA, further testing and evaluation are necessary as part of future work to better understand the success and failure of the proposed methods.

Author Contributions

All group members met together in order to discuss and get a deep understanding of the paper, related methods, as well as successfully running the provided code. After brainstorming, Juncheng came up with the idea of inference using a Bayesian model while Oded came up with the ideas for using different sampling methods for label word mappings and template generation. Each member of the group was assigned an idea for improvement, its implementation, and integration with the existing code. Juncheng worked on inference using a Bayesian model, Yuren worked on iterative automatic prompt search and Oded worked on label word mappings selection. We would like to thank the course team for grouping us together so that this fruitful collaboration can happen.

REFERENCES

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 1877–1901. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf.

Gavin C. Cawley and Nicola L. C. Talbot. Preventing over-fitting during model selection via bayesian regularisation of the hyper-parameters. *J. Mach. Learn. Res.*, 8:841–861, may 2007. ISSN 1532-4435.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

Tianyu Gao, Adam Fisch, and Danqi Chen. 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)*, pp. 3816–3830, Online, August 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.acl-long.295. URL https://aclanthology.org/2021.acl-long.295.

Trevor Hastie, Robert Tibshirani, Jerome H Friedman, and Jerome H Friedman. *The elements of statistical learning: data mining, inference, and prediction*, volume 2. Springer, 2009.

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. The curious case of neural text degeneration. *arXiv preprint arXiv:1904.09751*, 2019.

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

Alec Radford and Karthik Narasimhan. Improving language understanding by generative pre-training. 2018.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. 2019.

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

Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084*, 2019.

Timo Schick and Hinrich Schütze. Exploiting cloze questions for few shot text classification and natural language inference. *arXiv preprint arXiv:2001.07676*, 2020a.

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

Timo Schick, Helmut Schmid, and Hinrich Schütze. Automatically identifying words that can serve as labels for few-shot text classification. In *Proceedings of the 28th International Conference on Computational Linguistics*, pp. 5569–5578, Barcelona, Spain (Online), December 2020. International Committee on Computational Linguistics. doi: 10.18653/v1/2020.coling-main.488. URL https://aclanthology.org/2020.coling-main.488.

Taylor Shin, Yasaman Razeghi, Robert L. Logan IV, Eric Wallace, and Sameer Singh. AutoPrompt: Eliciting Knowledge from Language Models with Automatically Generated Prompts. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 4222–4235, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.346. URL https://aclanthology.org/2020.emnlp-main.346.

Zexuan Zhong, Dan Friedman, and Danqi Chen. Factual probing is [mask]: Learning vs. learning to recall. In *NAACL-HLT*, pp. 5017–5033, 2021. URL https://doi.org/10.18653/v1/2021.naacl-main.398.

A Automatic Genenration of Templates

The automatic generation of template uses pre-trained language model $\mathcal{L}'$ for filling in missing spans such as T5 model (Raffel et al., 2020). A initial form of templates is given, such as "[INPUT] [PLACEHOLDER] [LABEL WORD] [PLACEHOLDER]", we fill in the placeholders with the language model and use beam search with beam size k to keep a variety of top template candidates. Then for each template candidate, we fine-tune the masked language model $\mathcal{L}$ with the training set ($\mathcal{L}$ is the language model for filling in label words e.g. BERT or RoBERTa). We then pick the best template the minimizes the loss over the development set. In the case of positive vs. negative sentiment classification, for instance, the beam could include template candidates like "[INPUT] It was [LABEL WORD]." and "[INPUT] A [LABEL WORD] one." where the label words are "great" and "terrible".

B Pesudocode for Iterative Search

Algorithm 1 Iterative Automatic Prompt Search

Fix the number of iterations T
Initialize the template $temp_0$
for $t = 1, 2, \ldots, T$ **do**
    Select the best label words assignment lab_t given the template $temp_{t-1}$
    Generate the best template $temp_t$ given the label words lab_t
end for

C Computational requirements

It is worth noting that experiments involving the generation and sampling of labels and templates require heavy computational resources, and required an exceedingly long run-time. Therefore, we did not compare our modifications with the numerous experiments detailed in the original paper, but instead focused on several experiments that we believe capture the performance of the proposed methods.