Part Huiqing Zhang (S2110W0842):

Contributions: Huiqing Zhang (S2110W0842) first summarized the total work as an abstract to give a general impression of this work. And then she extracted the motivations and contributions of the article from the introduction part. Later, she introduced the problem setup for explaining what kind of a problem this paper solved and how the problem was formulated. At the same time, she introduced the evaluation datasets and evaluation protocol. Also, she explored the sources of the innovation points in this paper from related work.

Abstract

In this paper, they study a few-shot learning problem more in line with real-world scenarios:

1) Medium-sized language models can be accessed on common hardware resources.
2) Fine-tuning can be performed with a small number of samples.

The main method mainly includes:

1) A prompt-based fine-tuning together with a novel pipeline for automating prompt generation.
2) A refined strategy for dynamically and selectively incorporating demonstrations into each context.

This method dramatically outperforms standard fine-tuning procedures in this low resource setting, achieving up to 30% absolute improvement, and 11% on average across all tasks.

Motivations

First, GPT-3 has been able to achieve significant performance on downstream NLP tasks without updating the parameters of the pre-trained model by using a small number of task demonstrations as input contexts and adding natural language prompt. However, the number of parameters of GPT-3 model is so huge that it is difficult to be applied to practical applications. Therefore, this work investigates the performance improvement of prompt-based prediction and task demonstration methods for medium-sized models (e.g., BERT, RoBERT) on few-shot learning.

In addition, most of the prompt templates and label words are currently selected by hand, which relies heavily on the level of the designer, and the designed templates and selected label words are not very extensible. This paper proposed a method to automatically generate prompt templates and label words, and optimize the method for selecting task demonstrations example of GPT-3.

Contributions

Firstly, this paper proposed a fine-tuning pre-trained model method based on prompts and demonstrations.

Secondly, this paper introducing automatic prompt generation, including a pruned brute-force search to identify the best working label words, and a novel decoding objective to automatically generate templates using the generative T5 model. In this case, we can cheaply obtain effective prompts that match or outperform manually chosen ones.

Thirdly, this paper sample a single example to create demonstration sets and devise a novel sampling strategy that pairs inputs with similar examples, thereby providing the model with more discriminative comparisons.

Problem Setup

Assuming that we want to fine-tune on a task D with label space Y, using the pretrained model L. For the task, we only assume K training examples per class for the task' s training set, such that the total number of examples is K total. And we want to develop task-agnostic learning strategies that generalize well to an unseen test set $\left(x_{in}^{test}, y^{test}\right) \sim D_{test}$. Noticed that the size of develop set is the same as train set.

$$K_{tot} = K * |Y| \quad , \; D_{train} = \left\{\left(x_{in}^{i}, y^{i}\right)\right\}_{i=1}^{K_{tot}}$$

About the evaluation dataset, Here conducts a systematic study across 8 single-sentence and 7 sentence-pair English tasks.

To account for the variability of results, this work measure average performance across 5 different randomly sampled train set and develop set splits. Sampling multiple splits gives a more robust measure of performance, and a better estimate of the variance.
This work also sweeps multiple hyper-parameters for each data sample, and take the best setting as measured on the develop set of that sample, because hyper-parameters can make a significant difference.

Part Lianying Yin (S2110W0896):

Contribution: Lianying Yin (S2110W0896) introduces some methods of this paper, i.e. prompt-based fine-tuning, automatic selection of label words and automatic template generation. She explains the difference between prompt-based fine-tuning method and standard fine-tuning approach. The prompts consist of templates and labels, and the choice of templates and labels can have a great impact on the final result, and setting the suitable prompts manually is very difficult due to the expertise and understanding of the language model required. Then she presents the specific method proposed in the paper for the automatic generation of templates and label words.

Prompt-based fine-tuning

In the standard fine-tuning approach, for a classification task for example, an additional classifier is typically added to the 'CLS' section, however the number of newly initialized parameters that independent of those outside the pre-trained model can be large, which can make learning from small samples difficult.

In order not to introduce new parameters and to make full use of the knowledge of the pre-trained model, this paper takes the approach of MLM (BERT) and converts the downstream task directly into an MLM task, constructing a specific prompt template for a specific task and then "auto-completing" the prompt template through the language model.

Taking the sentiment analysis task as an example, given a sentence x_1 = "No reason to

watch it ." The prompt-based fine-tuning method first constructs a prompt template as

$$x_{\text{prompt}} = \texttt{[CLS]}\ x_1\ \textbf{It was}\ \texttt{[MASK]}\ .\ \texttt{[SEP]}$$

, with [MASK] contained in the prompt template. Then it predicts the mapping contained in [MASK], and the output of the prediction are label words, which are mapped to specific labels.

From the above analysis, we can know that prompt consists of two parts: one is Template, and another is label word mapping M(y). In order to better understand what is a good template or label word, the paper conducts a pilot study on the SST-2 and SNLI datasets, and the experimental results are shown in Table 2.

Template	Label words	Accuracy
SST-2 (positive/negative)		mean (std)
$<S_1>$ It was [MASK] .	great/terrible	**92.7 (0.9)**
$<S_1>$ It was [MASK] .	good/bad	92.5 (1.0)
$<S_1>$ It was [MASK] .	cat/dog	91.5 (1.4)
$<S_1>$ It was [MASK] .	dog/cat	86.2 (5.4)
$<S_1>$ It was [MASK] .	terrible/great	83.2 (6.9)
Fine-tuning	-	81.4 (3.8)
SNLI (entailment/neutral/contradiction)		mean (std)
$<S_1>$? [MASK] , $<S_2>$	Yes/Maybe/No	**77.2 (3.7)**
$<S_1>$. [MASK] , $<S_2>$	Yes/Maybe/No	76.2 (3.3)
$<S_1>$? [MASK] $<S_2>$	Yes/Maybe/No	74.9 (3.0)
$<S_1>$ $<S_2>$ [MASK]	Yes/Maybe/No	65.8 (2.4)
$<S_2>$? [MASK] , $<S_1>$	Yes/Maybe/No	62.9 (4.1)
$<S_1>$? [MASK] , $<S_2>$	Maybe/No/Yes	60.6 (4.8)
Fine-tuning	-	48.4 (4.8)

Table 2: The impact of templates and label words on prompt-based fine-tuning ($K = 16$).

It can be seen that different templates and label word choices actually have a great impact on the final result. using the same "template", different "label words" have different effects, such as the exchange of cat and dog; Using the same "label word", even minor changes to the "template" (such as removing punctuation) will result in different results.

The above "prompt" is designed based on artificial intuition and can better complete similar tasks. However, finding a suitable and correct "prompt" requires both expertise and a solid understanding of how the language model works internally. Therefore, the paper proposes that we should try to automate the construction of prompts: to automatically build a cheap method that is task-agnostic.

Automatic selection of label words

This paper distinguishes between the automatic construction of templates and label words. First they investigate how to construct a label word mapping M() given a fixed template T so that it can maximize the accuracy of fine-tuning. A simple search strategy is used in the paper, which can be divided into three steps:

1. At first, With the untuned pre-training model, for each class in the training set, the top-k words are selected such that the conditional probability is maximized. That is, samples from each class are first fed into the pre-training model, and the top-k words with the highest probability at [mask] are directly taken as candidate label words for that class.

$$\underset{v \in \mathcal{V}}{\text{Top-}k} \left\{ \sum_{x_{\text{in}} \in \mathcal{D}^c_{\text{train}}} \log P_{\mathcal{L}}\Big(\texttt{[MASK]} = v \mid \mathcal{T}(x_{\text{in}}) \Big) \right\},$$

Where $P_{\mathcal{L}}$ denotes the output probability distribution of language model $\mathcal{L}$.

2. And second, the candidate label words under each class are combined, and then the top-n allocation that makes the training set most accurate is found.
3. Then fine-tuning is performed to construct the label mapping relationship M by comparing the accuracy of the validation set to select the best one of the n allocation methods for label words.

Automatic template generation

The authors then investigate fixed label words, using the T5 model proposed by Google to automatically generate templates. T5 is pre-trained to fill in missing spans in its input . For example, given an input "Thank you < X > me to your party < Y > week" after the resulting mask operation, T5 will generate a string of text "< A > for inviting < B > last < C > ", meaning that "for inviting" can replace < X > and "last" can replace < Y >. The paper argues that this approach to T5 is particularly suitable for template generation, as there is no need to specify the number of tokens for the positions to be filled in advance. The main steps are as follows:

1. At first, we should fixed label word mapping relationships.
2. Then Padding tokens are added before and after the label words and then fed into the T5 model to automatically generate the template sequence. The authors list three methods of padding tokens.

$$
\begin{aligned}
&<S_1> \longrightarrow <\mathrm{X}> \mathcal{M}(y) <\mathrm{Y}> <S_1>, \\
&<S_1> \longrightarrow <S_1> <\mathrm{X}> \mathcal{M}(y) <\mathrm{Y}>, \\
&<S_1>, <S_2> \longrightarrow <S_1> <\mathrm{X}> \mathcal{M}(y) <\mathrm{Y}> <S_2>.
\end{aligned}
$$

3. Finally, the T5 model output is decoded using beam search to decode multiple templates that perform well on the training set, and then each candidate template is fine-tuned to select the best one of them.

Part Xiangju Chen (S2110W0813):

Contribution: Xiangju Chen (S2110W0813) introduces the remaining models of Fine-tuning with Demonstrations in this paper, and also introduces the experimental part of this paper and concludes the paper. For the Fine-tuning with Demonstrations model, she introduced the defects in GPT-3 input and how this paper can improve it. The experimental part introduces datasets and baselines used in this paper, and introduced the main results. Finally, analyze the advantages and disadvantages of this paper and make a summary.

Fine-tuning with Demonstrations

1. Defects in GPT-3 input: (1) The number of demo examples that can be added is limited by the maximum input length of the model. (2) A large number of random examples of different types are mixed together, resulting in a long context, which is not conducive to model learning.
2. In order to solve these problems, the paper proposes a simple solution: for each input sentence Xin, only one sample is randomly sampled from each class in the training dataset, and obtain the representative label words of the class, and then generate a template for it through T5 model, and concat with the sample to form a prompt. Each class will get such a prompt, all of which are spliced with the input samples, as shown in the following

figure after splicing.

$$\mathcal{T}(x_{\text{in}}) \oplus \tilde{\mathcal{T}}(x_{\text{in}}^{(1)}, y^{(1)}) \oplus \cdots \oplus \tilde{\mathcal{T}}(x_{\text{in}}^{(|\mathcal{Y}|)}, y^{(|\mathcal{Y}|)})$$

Experiment

1. Datasets: (1) |y| denotes the number of categories. (2) L denotes the average length of the sentences (pairs).

Category	Dataset	$\lvert\mathcal{Y}\rvert$	L	#Train	#Test	Type	Labels (classification tasks)
single-sentence	SST-2	2	19	6,920	872	sentiment	positive, negative
	SST-5	5	18	8,544	2,210	sentiment	v. pos., positive, neutral, negative, v. neg.
	MR	2	20	8,662	2,000	sentiment	positive, negative
	CR	2	19	1,775	2,000	sentiment	positive, negative
	MPQA	2	3	8,606	2,000	opinion polarity	positive, negative
	Subj	2	23	8,000	2,000	subjectivity	subjective, objective
	TREC	6	10	5,452	500	question cls.	abbr., entity, description, human, loc., num.
	CoLA	2	8	8,551	1,042	acceptability	grammatical, not_grammatical
sentence-pair	MNLI	3	22/11	392,702	9,815	NLI	entailment, neutral, contradiction
	SNLI	3	14/8	549,367	9,842	NLI	entailment, neutral, contradiction
	QNLI	2	11/30	104,743	5,463	NLI	entailment, not_entailment
	RTE	2	49/10	2,490	277	NLI	entailment, not_entailment
	MRPC	2	22/21	3,668	408	paraphrase	equivalent, not_equivalent
	QQP	2	12/12	363,846	40,431	paraphrase	equivalent, not_equivalent
	STS-B	$\mathcal{R}$	11/11	5,749	1,500	sent. similarity	-

2. Baselines

   (1) Majority: using the most frequent category in the training set as the prediction result. (2) Prompt-based zero-shot: a zero-sample setup, using a manually designed template and not introducing additional sample examples. (3) "GPT-3" in-context learning: zero-sample setting, using a manually designed template, as in GPT-3, with examples randomly selected from the training set and added to each input in context. (4) Fine-tuning : small-sample setup, standard fine-tuning approach, introducing new parameters. (5) Fine-tuning(full): standard fine-tuning using the full amount of annotated data. (6) Prompt-based FT(man) : the fine-tuning method of this paper, using manually designed prompt templates. (7) Prompt-based FT(auto): the fine-tuning method in this paper, which automatically builds the prompt templates. (8) +demonstrations: introduction of additional sample examples into the context.

3. Main results: In summary, the combined solution of the paper - fine tuning using automated search templates and sampled demo sets - resulted in a 30% improvement in the SNLI dataset compared to standard fine tuning, with an average improvement of 11%. While prompt-based fine-tuning can be substantially better than standard fine-tuning under small sample conditions, it still lags behind standard fine-tuning using the full amount of data.

	SST-2 (acc)	**SST-5** (acc)	**MR** (acc)	**CR** (acc)	**MPQA** (acc)	**Subj** (acc)	**TREC** (acc)	**CoLA** (Matt.)
Majority†	*50.9*	*23.1*	*50.0*	*50.0*	*50.0*	*50.0*	*18.8*	*0.0*
Prompt-based zero-shot‡	83.6	35.0	80.8	79.5	67.6	51.4	32.0	2.0
"GPT-3" in-context learning	84.8 (1.3)	30.6 (0.9)	80.5 (1.7)	87.4 (0.8)	63.8 (2.1)	53.6 (1.0)	26.2 (2.4)	-1.5 (2.4)
Fine-tuning	81.4 (3.8)	43.9 (2.0)	76.9 (5.9)	75.8 (3.2)	72.0 (3.8)	90.8 (1.8)	88.8 (2.1)	**33.9** (14.3)
Prompt-based FT (man)	92.7 (0.9)	47.4 (2.5)	87.0 (1.2)	90.3 (1.0)	84.7 (2.2)	91.2 (1.1)	84.8 (5.1)	9.3 (7.3)
+ demonstrations	92.6 (0.5)	**50.6** (1.4)	86.6 (2.2)	90.2 (1.2)	**87.0** (1.1)	**92.3** (0.8)	87.5 (3.2)	18.7 (8.8)
Prompt-based FT (auto)	92.3 (1.0)	49.2 (1.6)	85.5 (2.8)	89.0 (1.4)	85.8 (1.9)	91.2 (1.1)	88.2 (2.0)	14.0 (14.1)
+ demonstrations	**93.0** (0.6)	49.5 (1.7)	**87.7** (1.4)	**91.0** (0.9)	86.5 (2.6)	91.4 (1.8)	**89.4** (1.7)	21.8 (15.9)
Fine-tuning (full)†	*95.0*	*58.7*	*90.8*	*89.4*	*87.8*	*97.0*	*97.4*	*62.6*
	MNLI (acc)	**MNLI-mm** (acc)	**SNLI** (acc)	**QNLI** (acc)	**RTE** (acc)	**MRPC** (F1)	**QQP** (F1)	**STS-B** (Pear.)
Majority†	*32.7*	*33.0*	*33.8*	*49.5*	*52.7*	*81.2*	*0.0*	-
Prompt-based zero-shot‡	50.8	51.7	49.5	50.8	51.3	61.9	49.7	-3.2
"GPT-3" in-context learning	52.0 (0.7)	53.4 (0.6)	47.1 (0.6)	53.8 (0.4)	60.4 (1.4)	45.7 (6.0)	36.1 (5.2)	14.3 (2.8)
Fine-tuning	45.8 (6.4)	47.8 (6.8)	48.4 (4.8)	60.2 (6.5)	54.4 (3.9)	76.6 (2.5)	60.7 (4.3)	53.5 (8.5)
Prompt-based FT (man)	68.3 (2.3)	70.5 (1.9)	77.2 (3.7)	64.5 (4.2)	69.1 (3.6)	74.5 (5.3)	65.5 (5.3)	71.0 (7.0)
+ demonstrations	**70.7** (1.3)	**72.0** (1.2)	**79.7** (1.5)	**69.2** (1.9)	68.7 (2.3)	77.8 (2.0)	**69.8** (1.8)	73.5 (5.1)
Prompt-based FT (auto)	68.3 (2.5)	70.1 (2.6)	77.1 (2.1)	68.3 (7.4)	**73.9** (2.2)	76.2 (2.3)	67.0 (3.0)	75.0 (3.3)
+ demonstrations	70.0 (3.6)	**72.0** (3.1)	77.5 (3.5)	68.5 (5.4)	71.1 (5.3)	**78.1** (3.4)	67.7 (5.8)	**76.4** (6.2)
Fine-tuning (full)†	*89.8*	*89.5*	*92.6*	*93.3*	*80.9*	*91.4*	*81.7*	*91.9*

Conclusion

1. Advantages and Disadvantages

   Advantages: (1) Proposes a prompt-based fine-tuning strategy and automates the generation of prompt templates. (2) Dynamic selection of sample examples for in-context training.

   Disadvantages: (1) LM-BFF still lags behind standard fine-tuning methods based on the full amount of a notated data. (2) Only applicable to a small number of categories, limited sentence template length, etc.

2. Conclusion

   This approach is more suitable for certain tasks that (1) can naturally be used as 'fill-in-the-blank' problems; (2) have relatively short input sequences; and (3) do not contain many output classes. Problems (2) and (3) may be improved by longer contextual language models, while for tasks that cannot be formulated directly in prompts, such as structured prediction, this is a larger problem that needs to be addressed in future work.