Exploring the Effects of Fine-Tuning for BERT and Prompt Engineering for Zephyr

Jin Yan
The Division of Statistics and Machine Learning (STIMA)
Department of Computer and Information Science (IDA)
Linköping University
jinya425@student.liu.se

Abstract

This article surveys the effects of fine-tuning and prompt-engineering on LLMs (Large Language Models). The task used in this research is sentiment analysis. Dataset is 'imdb' on HuggingFace community. BERT (Devlin et al., 2019) is the model for fine-tuning. By comparison, Zephyr (Jiang et al., 2023) is for prompt engineering research. In order to make it easier, the article used PyTorch Trainer to realize the fine-tuning. In terms of prompt engineering, Zero-shot and Extremely Few-shot strategies are implemented.

Code for fine-tuning and Prompt Engineering: https://github.com/yj313155521/LLM-Project.git

1 Introduction

The advent of transformer architecture enables people not rely on the complex recurrent or convolutional neural, since it is a simpler one. At the same time, this structure can be more parallelizable and thus save a lot of traning time. Mostly importantly, the performance of this model is much better than previous ones. (Vaswani et al., 2023) Based on transformer, two new ones come up. The first one is BERT, which is introduced by Google AI Language. This model was designed to pre-train deep bidirectional representations from unlabeled test by jointly conditioning on both left and right context in all layers. This unique architecture makes it possible to fine-tune the model without substantial modifications.(Devlin et al., 2019) It is because of this reason, we can now make use of BERT to do the fine-tuning process. Another LLM used in this article is Zephyr and this is a fine-tuned version of Mistral-7B-v0.1 model according to the model card in huggingface community. In terms of Mistral, it can be utilized for generating human-like text and is also based on transformer architecture.(Jiang et al., 2023) At the same time, owing to its grouped-query attention(GQA) and sliding window attention(SWA), this model can achieve high performance and meanwhile keep efficient in processing tasks. Given that, it is a good candidate for testing different prompt engineering strategies. In order to improve the performance of these LLMs for downstream tasks like sentiment analysis in this article, we can adopt fine-tuning and prompt engineering. For BERT model, we use PyTorch Trainer to realize the fine-tuning process. For Zephyr, considering the cost of fine-tuning LLMs in reality, this article explores the potential of using prompt engineering with zero of few slots to improve classification accuracy.

2 Theory

In order to help beginners better understand the relationship among these different models. I will briefly review these different structures and some special mechanisms.

2.1 Transformer

The main structure is shown in Figure 1. From the figure, we can clearly see that this structure is composed of two parts. On the left bottom left corner, the structure first converts the input tokens to vectors of dimension d_{model} and then these vectors("word embeddings") are added to "positional encodings". The the result are then put into "encoder". A stack of N = 6 identical layers compose the encoder. Each layer has two sub-layers. The first is a multi-head self-attention mechanism, and the second is a simple, position-wise fully connected feed-forward network. To simplify our explanations, we mainly discuss the one-head self-attention mechanism. In this mechanism, the input for one token will be replicated into three identical ones and then these three ones are projected to three vectors with same dimension (64). They represent "Query", "Key", and "Value". The output of the

self-attention mechanism(self-attention value) for one token is the weighted summation of different values of all tokens "Value". The Weights in the calculation is derived by calculating the distances between the token's "Query" and all token's "Values", including its own "Value". The advantage of using this mechanism is it can reduce computational complexity and the path between long-range dependencies in the network. Also, it can increase the amount of computation that can be parallelized.

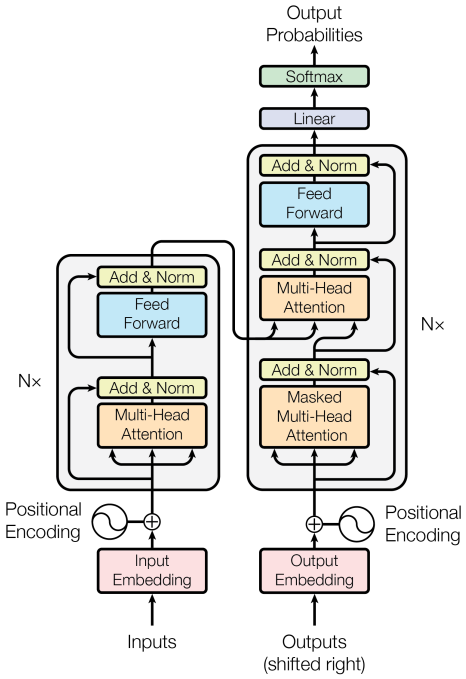

Figure 1: The Transformer - model architecture.

2.2 BERT

BERT stands for Bidirectional Encoder Representations from Transformers. Basically, it is a multi-layer bidirectional Transformer encoder based on the Transformer Model mentioned above. It is worth noting that it is mainly based on the "encoder" part of Transformer. The parameters used in this model is different from Transformer. Take the model used in the article, $\text{BERT}_{\text{base}}$, as an example. The number of layers for encoder is 12, the hidden size is 768, and self-attention heads is 12. By contrast, in Transformer model, these figures are 6, 512, and 8 respectively. In order to get such pre-trained model, the researcher used two tasks. One is "Masked LM". That is masking some percentage of the input tokens at random, and then predict those masked tokens. The final hidden vectors corresponding to the mask tokens are fed into an output softmax over the vocabulary. Another task for pre-training is sentence prediction. After getting the pre-trained model, in order to improve the performance of downstream tasks, the model need to be fine-tuned for each task. In Fine-tuning Procedure, most model hyperparameters are the same as in pre-training, except for batch size, learning rate, and number of training epochs.

2.3 Mistral 7B

Mistral 7B is based on Transformer architecture.(Jiang et al., 2023) This model outperforms Llama 1(the best open 13B model) across all evaluated benchmarks and Llama2(the best released 34B model) in reasoning, mathematics, and code generation. Both Llama 2 and Llama 1 are from Meta. Also, this model makes use of "GQA" and "SWA" and thus can effectively handle sequences of arbitrary length with a reduced inference cost.

2.4 prompt engineering

As for dealing with NLP tasks, there are three main stages. The first stage is characterized by feature engineering and architecture engineering. The key point is to find the salient features and the appropriate inductive bias. The second stage is famous for the pre-train and fine-tune paradigm. In this paradigm the model is pre-trained as a language model (LM) and then adapted to all kinds of downstream tasks. The process may involve introducing the new parameters and fine-tuning them using task-specific objective functions. The third one is "pre-train, prompt and predict" paradigm. In stead of using objective engineering, this stage is famous for "prompt engineering". When we want to find the sentiment of a film review, we can input "[review], and it contains [ ] emotion." and ask the model to fill in the square bracket. At the same time, there are already different training strategies, like Tuning-free strategy and Fix-prompt LM Tuning. The strategy used in this article, namely Tuning-free strategy, would avoid the risk of catastrophic forgetting, since it not change the parameters of the pre-trained models.

3 Data

The dataset I used in this article is 'imdb', which is a large dataset for sentiment analysis and contains 25,000 reviews for training and 25,000 reviews for testing.

In the case of fine-tuning the BERT model, the size of the training Data set and evaluation data set

used are both 1000. Meanwhile, since the limitation of the input for BERT model is 512, I truncated the input sentences and ensured the length of the input is within 250. This way, the number of tokens created can satisfy this requirement.

By comparison, in the case of prompt engineering, the size of the test data set is 10, since this model would take a longer time to create an answer including a desired label and this size can ensure a appropriate test time span(about 30 minutes). Like BERT, Zephyr also has its own input length limitation. This way, the length of truncated sentence is within 200 to make sure that even few slots are added to reviews, the length requirement can be satisfied.

4 Method

In this section, I will show the the methodology employed in this thesis project. The first one is Fine-tuning. In order to realize this, I used the class transformers.Trainer. The process is much easier and the trained model [1] is saved in Huggingface community. The second one is the usage of prompt engineering or called Tuning-free Prompting. (Luo, 2023). In this process, we use three different prompt templates to explore the results. These templates are shown below.

Template 1 (with Detailed Instruction, zero-shot): System: You are an AI assistant that follows instruction extremely well. Human: [text] Given the above review we have two two classes. class_0: In such film reviews, customers complain about something they do not like and think this film is not satisfactory. class_1: In such film reviews, customers show positive emotions, they might mention something that they like. Please classify this film review into one class out of these two classes,and just output the label without anymore word. Assistant:

Template 2 (with demonstration examples, extremely few-shot): System: You are an AI assistant that follows instruction extremely well. Human: First review: Brilliant and moving performances by Tom Courtenay and Peter Finch. Second review: This is a great movie. Too bad it is not available on home video. Third review: Primary plot!Primary direction!Poor interpretation. Fourth review: Read the book, forget the movie! Above, the first two reviews belong to 'class_1'; the last two reviews belong to 'class_0'. Please classify the following film review into one class out of these two classes,and the output format should be the same as 'This review belongs to 'class_X'. New review:[text] Assistant:

Template 3 (Baseline prompt): System: You are an AI assistant that follows instruction extremely well. Human: [text] Given the above review we have two two classes. 'class_0' and 'class_1'. Please classify this film review into one class out of these two classes,and just output the label without anymore word. Assistant:

In template 1, I at first introduce a review and then give the task description and this would make the model understand the task very well. Finally the model is asked to classify the review I mention first.

In template 2, instead of presenting the task descriptions, I show the model few demonstration examples. In each example, the model can learn how to label film reviews. This process is also called augmentation method.(Liu et al., 2023) or "in-context learning".

In template 3, I do not give much information about the tasks, and just let model classify the input review according to its understanding. And thus, the result created can be used as a baseline to assess the effects of the other two prompt engineering strategies.

5 Results

This section contains two parts. The first part shows the comparison between the Bert Base model the fine-tuned model. The result is shown in table 1. As for the second part, the comparison among different strategies are shown in table 2.

6 Discussion

It is clear that with the implement of the fine-tuning method, the accuracy and F1-score for the sentiment analysis increased a lot. This further proves the power of fine-tuning for LLM method.

Although LLM after fine-tuning can do a good job, we cannot ignore a fact that training such a model needs a lot of computational resources and time. Most importantly, we cannot ensure that enough labeled dataset can be offered each time. Without enough data points, it is hard for us to reach the same goal.

In order to solve the drawbacks mentioned above, we can consider another generative pre-trained model, Zephyr in this case. From the final result

[1] jinya425/bert-base-cased_for_sentiment_analysis

Name of Models	Accuracy/Macro F1-score
`bert-base-cased`	0.51 / 0.34
`fine-tuned version of Bert`	0.87 / 0.87

Table 1: Results of different Bert Models for sentiment analysis

Template No.	Accuracy/Macro F1-score
1	0.85 / 0.85
2	0.90 / 0.90
3	0.50 / 0.48

Table 2: Results of different prompt engineering tactics for sentiment analysis

we can draw a conclusion that compared to the baseline prompt template 3, template 1 (Zero-shot) and template 2 (Extremely-few-shots) can provide exciting results.

In this experiment, Extremely-few-shots strategy reaches the best result. However, it is different from the result in (Luo, 2023). In that paper, zero-shot strategy achieves better result. I think the main reason is the differences between the two classification tasks. In that paper, the data set consists of 14 labels. Among them there are 13 specific labels and another label that representing those conversations that cannot be categorized into the first 13 ones. By comparison, in this article, there are only two labels for the model to distinguish.

In terms of the limitation of this work, I think it would better if more models can be imported in this article. So, we can do more comparisons among the models especially for the prompt engineering strategy part. Also, Prompting method is a large topic and there are many prompt-based training strategies besides tuning-free prompting mentioned in this article. For the fine-tuning part, it would be better if more training details are involved, readers would get more knowledge about how to fine-tune.

7 Conclusion

In general, this article clearly shows that the fine-tuning for model with encoder part like BERT and prompt-engineering for model with decoder part can really improve the performance of models. With more training data points, we can implement fine-tuning. If not, prompt engineer can also help us get the similar results.

After this survey writing process, besides the knowledge I got about LLM, the ability of searching for relevant information and grouping them together has improved. Also, it prepares me very well for exploring more in NLP field.

References

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. ArXiv:1810.04805 [cs].

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. Mistral 7B. ArXiv:2310.06825 [cs].

Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. 2023. Pre-train, Prompt, and Predict: A Systematic Survey of Prompting Methods in Natural Language Processing. *ACM Computing Surveys*, 55(9):195:1–195:35.

Hengyu Luo. 2023. *Prompt-learning and Zero-shot Text Classification with Domain-specific Textual Data*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2023. Attention Is All You Need. ArXiv:1706.03762 [cs].