Assignment 2 LLM report

Introduction

Due to the success of GPT, large language model has become a hot topic nowadays. There have been more and more research and application in LLM these years. It could be helpful to learning something from these models. Large language models origin from natural language process (NLP), which mainly deals with tasks like machine translation, sentiment analysis, named entity recognition and so on. These tasks contain mainly two processes, the first one altering words to vectors so that computer can process, the second one using the vectors formed to do the following task by traditional machine learning models. In the early years, word to vectors use the techniques of bag of words like counting, TD-IDF and so on, while LLM primarily use tokenizer and embedding. The traditional machine learning models of NLP task include support vector machine and statistical method like Bayesian classifier and so on. However, with the development of neural network and the emergence of transformer, NLP finds a good way to improve and thus large language models develops. The mystery of large language model lies on the transformer structures. In an intuitive way, transformer is a neural network that guide the weights to remember the connection between two tokens. If two word or tokens occur together frequently, then they have a high connect with each other and the weights will activate the relation between them. Aftier several transformer structure, they are expected to capture the hidden semantic meaning. Transformer plays a role in feature extraction in sentence, like convolution kernel in computer vision, and hence there are more and more researchers devoted to developing feature extraction methods based on this transformer structure. However, there is a difference of transformer comparing conventional kernel networks, that is a small change of input sequence namely prompt will also make a difference in output. This is because of the self-attention mechanism. Unlike in convolution network, convolution kernel has its own receptive filed, a small change on a pixel would not affect the whole feature map. While in the self-attention mechanism, a small change will of course change all the embedding values. Thus, changing even only a simple word of the sentence in the prompt may dramatically result in different result. Due to this reason, there comes another perspective for analysis namely prompt engineering. Instead of developing a new model with high cost to train, we here in this experiment focus on the prompt engineering that is to find a better prompt to do multiple choice task.

Method

In order to better develop a prompt, it is necessary to analyze what kind of task it is and also the network we have. There are two fundamental concepts in this work, in-context-learning and multiple choices questions answer. In addition, I will also introduce the technique chain of thought used in prompt engineering.

Multiple-choice questions

The task of multiple choices question answer is different from the other language task like text generation. Text generation will take as input a query concatenated with a null string which is a mask sequence. This input contusing mask sequence will go through the network and then find some tokens to substitute the null string and then return these tokens sequentially. The input of and output are both strings. While in the task of multiple-choice question and answer, things become different. It takes a query string

and a list of candidates answer as input and output a list of corresponding likelihood. More precisely, for each candidate answer, we treat the question as source string and the candidate answer as target string, then the network first concatenate source string and the candidate as input, and finally output the likelihood of such a pair. After each pair of candidates and question go through the network, we have a list of likelihood and then take the maximum together with the corresponding candidate answer as result. This is quite different to the text generation as we care more about the likelihood of two input strings rather than output another string. This difference result in different analysis of prompts and more will be depicted in the experiment section.

In-context Learning

In my opinion, in-context learning is the same as few-shot learning, that is to include a few examples in prompt showing the LLM how to answer the question. Given a query, the examples can be chosen by the cosine similarity between query and all the sample questions in dataset. Find the most related questions and then combine their question answering pair into prompt, we can have a higher performance. Another method called RAG, seems quite similar that they are considering taking cosine similarity to choose the most relevant paragraphs or questions.

Chain of Thought

Prompt engineering is a work to clear explain the question to LLM so that they can understand the question better and then generate more correct answers. Taking more words on explanations is one way and the other way is using the chain of thought method. The idea of chain of thought is to split original problem into sequential subproblems and make the LLM solve the problem step by step. Adapting to this multiple-choice task, it may have limited help, as it is not a text generation task. However, I do also consider using this technique but only in the beginning of prompt.

Result and analysis

In this section, I will try several prompts and then show and analyze the results. I will first propose three basic prompts, and then based on these three basic prompts do some tuning. Finally, I will use accuracy on easy test set as criteria to do some analysis.

Based on the original prompt, I have designed two other prompts, one simpler and the other one more specific shown at the following table. In addition, the number of examples (hyperparameter N) may also affect the result and sometimes 3 or 4 would be helpful enough so that I do also set 8 to 4 to have a try. As for hyperparameter ‘max_len’, 1024 would be enough for these short prompts, then I did not make change on it.

Type	Prompt
V1	`Question: {question}\nCandidate answers: {candidate_answers}\nGold answer: {answer}`
V2	`Question: {question}\nGold answer: {answer}`
V3	`Given a question below, choose one candidate answer\nQuestion: {question}\nThe candidate answers are:\n{candidate_answers}\nThe gold answer should be:\n {answer}`

However, a prompt may not only include examples and questions, but also include some constrains or other system massages. I have also designed three prompt prefixes in the beginning of prompts to show the system massages. Unfortunately, for normal ablation

test, to show which prompt type is better, we should have other variables like this prompt prefix as fixed, but it is hard to find a general and sufficient prefix for theses three prompts. Thus, I designed the prefix respectively.

Type	Prompt prefix
V1	`Given a question with multiple candidate answer choice, you should answer the question using one of candidate answers. The following are {N} examples you can follow. The Question at last is the problem you should answer to.`
V2	`You are a smart robot, carefully focus on the relationship between question and gold answer during following {N} examples. Finally, give a gold answer to the question after examples.`
V3	`You are a smart robot, in the following {N} examples, carefully focus on the relationship between question and gold answer, and separate the gold answer from other candidate answers. Finally, give a gold answer to the question after examples.`

As said before, chain of thought would also help for prompt engineering so that I do also design two chain of thought prompts prefix for prompt type v1 and v2, as v3 has already been thoroughly described. These chain of thought prompts are shown below:

Type	Chain of thought prompt prefix
V1	`Given a question with multiple candidate answer choice, you should answer the question using one of candidate answers. You should first identify what the question is talking about, then calculate the correlation between question and each candidate answer, finally output the candidate answer with the highest correlation. The following are {N} examples you can follow. The Question at last is the problem you should answer to.`
V2	`You should first identify and separate {N} examples and the question. Then pay attention to the relationship between Question and Gold answer respectively. Finally give the answer to the question you have identified.`

Then test all the prompts in the easy validation set, the results are presented in the following table (N4: denote four examples; Pre: prompts including prefix; cot: chain of thought prompts).

	Base	N4	Pre1	cot
V1	0.6298	0.6386	0.6157	0.6105
V2	0.80	0.7789	0.8070	0.8052
V3	0.4544	0.4578	0.4561	

We can see from the results that prompt v2 performs better than the other two obviously. It is amazing that eliminating candidate answers could make a better performance. However, after a little analysis, it becomes not so mysterious anymore. Suppose we

have query Q, and the candidate answers are denoted by A, B, C and D. Without loss of generality, we assume that A is the correct answer. As discussed before, for this multi-choice question answering, the model is to estimate the likelihood of "Q ABCD: {i}" for v1 (or v3), and likelihood of "Q {i}" for v2, where i is one of the candidate answers ABCD. For v2, candidate answer i is directly evaluated to the query Q, while for v1 i has additional relations to ABCD which may result in some disruption. For example, if the wrong answers BCD are highly correlated and are all independent to correct answer A, then the correlation for candidate answer A to "Q ABCD" of course will be less than the correlation for BCD to "Q ABCD". Hence, the results will highly depend on the question answers set.

Notice that all expect N4 of prompt v2 performs well and close, then we should take a further look at their performance on challenge set, and also we change the model to phi-2 to see whether there is a significant improvement.

V2	Phi-1.5		Phi-2	
	Easy	Challenge	Easy	Challenge
Base	0.80	0.5184	0.8544	0.5853
Pre	0.8070	0.5117	0.8561	0.5819
Cot	0.8052	0.5117	0.8421	0.5786

The results show that phi-2 can make a difference and the best prompt should be prompt v2 without prefix. Then we apply this prompt and phi-2 to test set, we have accuracy of easy test 0.8502, and accuracy of challenge test 0.5956.