**Research Proposal**

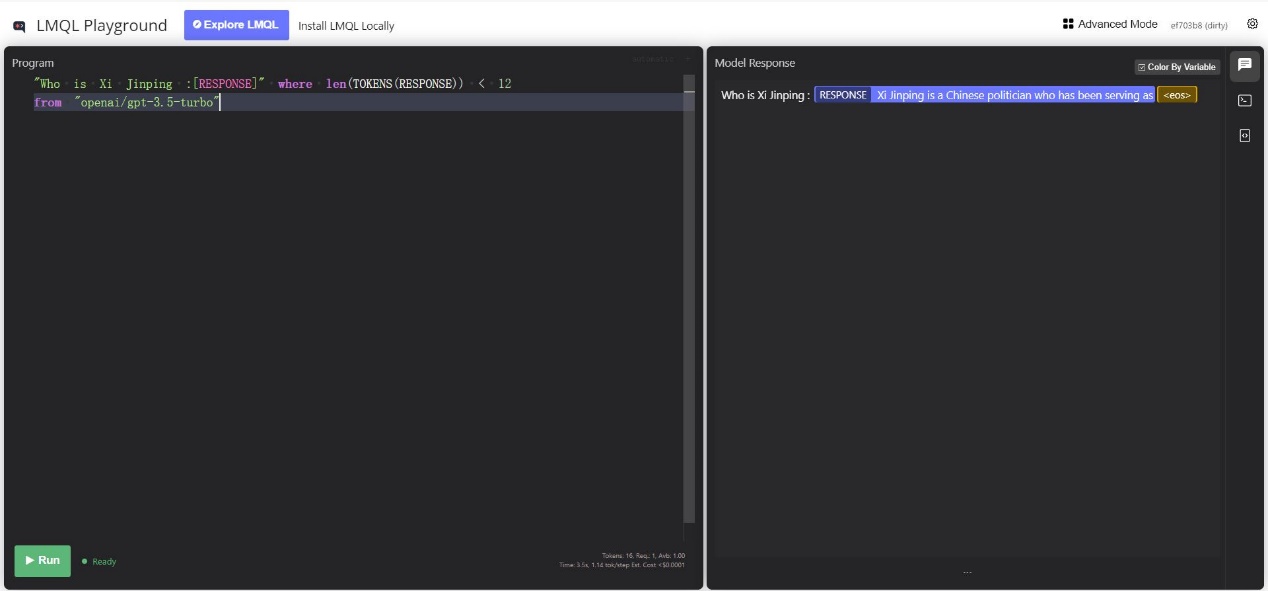
Large language models (LLMs) are increasingly being used in our everyday life. Most large language models are pre-trained models, which means that they are trained on a large corpus of general language data. For some specific problems, people often need to fine-tune LLMs to get better results. There are two methods that are currently popular, the first of which is known as fine-tuning. This method uses deep learning techniques, by modifying the structure of the model and training the model on certain tasks to optimize the model’s parameters and structure. The second method is prompting. Since LLMs are trained on a large language corpus, they already have the ability to solve certain types of problems. However, sometimes LLMs may not know the task proposed by the user. In this case, LLMs need more instruction and guidance. Prompting is about providing LLMs with the right guidance so that they can complete tasks as required.

Modern LLMs are very large. Most models have more than a billion parameters. Fine-tuning requires us to further train the model. Since training an LLM requires hardware support like GPUs, fine-tuning is not practical for most users. In addition, for closed source LLMs, users are unable to fine-tune them. Compared with fine-tuning, prompting is relatively simple, and users do not need to understand the internal structure of an LLM. In most cases, users can obtain the expected response by guiding with natural languages. Prompting is lower in cost and more suitable for the majority of users.

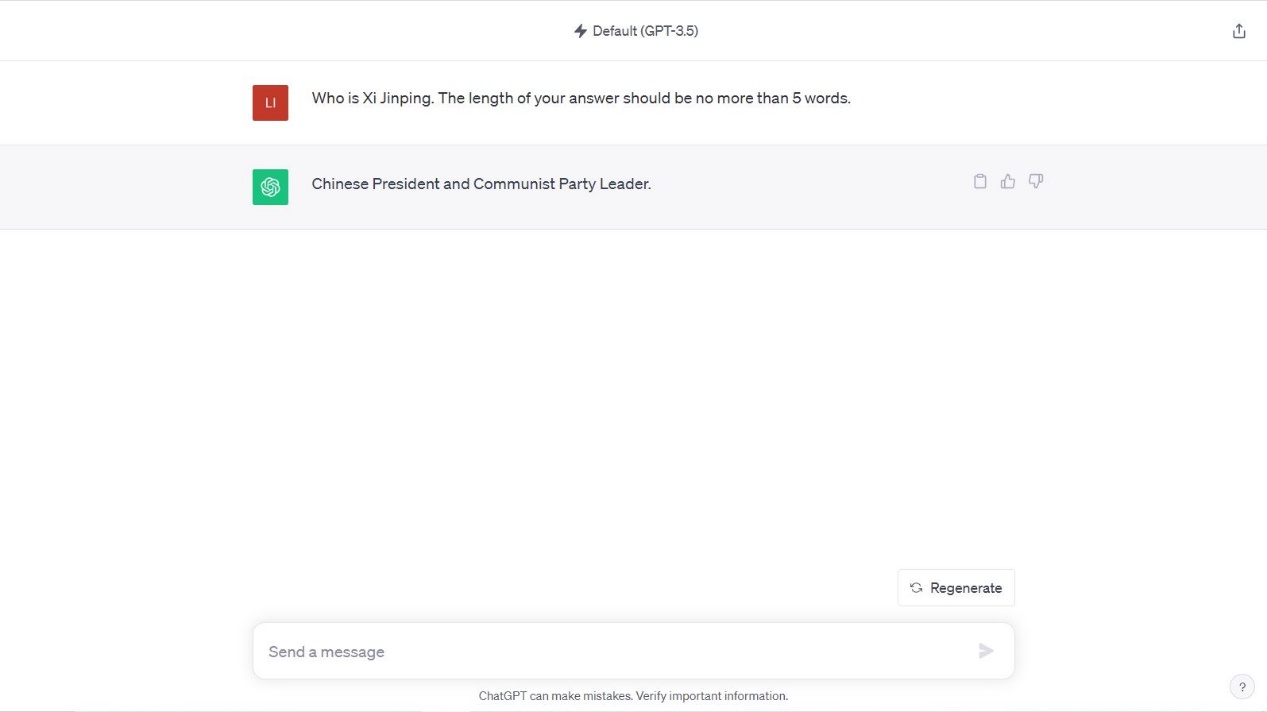
LLMs are quite sensitive to the format and sequence of prompting. Different forms of prompting can produce significant variances. Most ordinary users do not understand the prompting format needed by LLMs, so the prompts they write may not get good results. In addition, LLMs perform better with concise prompts, and lengthy sentences tend to weaken the effect of the prompt. Therefore, formatting the prompts will help to reduce the cost of LLMs and improve their performance.

There are already many programming languages that have integrated LLMs prompting. These attempts include Microsoft's Semantic Kernel, LangChain's prompt templates, and the LMQL of ETH. Taking Microsoft's Semantic Kernel as an example, we can insert other functions into the prompt to form templates. Calling the template with different function values can ultimately create different prompting effects. LMQL is more powerful, allowing us to restrict the length and content of LLM outputs to produce satisfactory results.

Most of these attempts focus on the input and output of LLMs, but sometimes they may not fully meet the user's needs. Here is an example of trying to limit the response of the LLM using a standard query in LMQL.

窗体顶端

**Fig 1A.** Response from LMQL

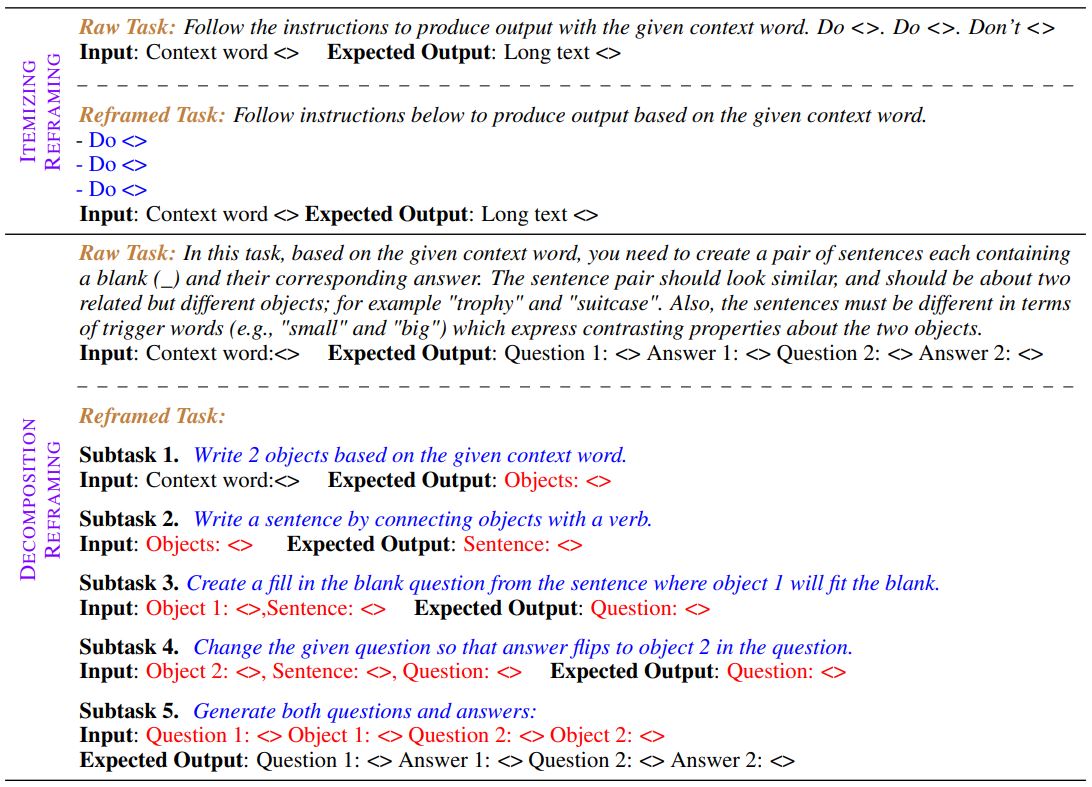


**Fig 1B.** Response from ChatGPT

As demonstrated by the outputs, both of which were generated using GPT-3.5, it is clear that the latter result is more closely aligned with what the user is prompting for.

The goal of our project is to implement new language features to allow for the limitation and modification of prompt output as a feature of the programming language itself. This would prevent the user from having to include these constraints in the prompt itself and would allow for easier modification of prompt parameters. The above example indicates that, besides input and output, introducing other prompting rules into these languages may achieve better results. The following section will describe more of these types of features that we would like to implement that will improve upon prompt output.

The article *Reframing Instructional Prompts to GPTk’s Language* introduces some methods of reframe prompting, which have been proven to improve the effectiveness of prompting.



**Fig 2.** Methods of Reframing Prompting

The two methods mentioned above belong to the reframing methods introduced in the article. These methods have very clear formats, which makes them particularly well-suited for formatting by using programming language methods. By providing specific syntax and semantics, current research on prompting programming can be expanded. Beyond controlling inputs and outputs, we can define more operations. For example, in the decomposition reframing method, we can define a data type, where each accepts input and provides output. A task can be composed of multiple sub-tasks, which work together to complete an overall task. By defining its syntax and semantics, we can define which is a good prompt and which is not. Then, we can give suggestion the user.

We plan to demonstrate our idea by comparing queries in our modified language to the result of queries using the raw prompting methods and showing the circumstances in which the modified prompting techniques produce better results. The paper *Prompting Is Programming: A Query Language for Large Language Models* introduces evaluation methods for such prompting methods. These include:

* **Expressiveness**: Can we easily implement common and advanced prompting techniques with simple and concise query logic, especially in the case of interactive prompting?
* **Performance**: Can it be used to effectively lower the required number of model queries and thereby reduce the computational or API-related cost of using LMs?
* **Accuracy**: Does its constrained decoding affect the task accuracy of LMs when evaluated on standard benchmarks?

These evaluation methods are the methods that will primarily be applied to evaluate our research and the performance of the modified language. They provide a framework for assessing how well prompting methods and languages perform, which is critical for development and optimization. We can quantitatively assess performance based on number of queries, and accuracy based on the percentage of tasks performed correctly. Expressiveness will be evaluated qualitatively based on the complexity of created queries. In summary, we hope to demonstrate that programming languages that utilize LLM-prompting can be further improved upon by introducing prompting techniques into the programming language itself for an improved user experience and better performance.