Answering High-precision Problems for LLMs by Combining Text2code

Tong Guo 1

1Affiliation not available

January 8, 2024

Abstract

The current large language models (LLMs) mainly lacks this capability: answering high-precision questions/prompts. LLMs is actually a powerful fuzzy memory system that makes it difficult to answer high-precision questions. The results of code execution is a kind of high-precision answer. And expert system applications need to answer these kinds of questions. In this paper, to solve the above problem, We propose a design of LLMs combined with the text2code approach.
Answering High-precision Problems for LLMs by Combining Text2code

Tong Guo
779222056@qq.com

Abstract. The current large language models (LLMs) mainly lacks this capability: answering high-precision questions/prompts. LLMs is actually a powerful fuzzy memory system that makes it difficult to answer high-precision questions. The result of code execution is a kind of high-precision answer. And expert system applications need to answer these kinds of questions. In this paper, to solve the above problem, We propose a design of LLMs combined with the text2code approach.

Keywords: Deep Learning · Text-To-Code · Expert System · Large Language Models

1 Introduction

Transformers-based models [10] become the best models in most deep learning [2] tasks. The transformers-based models also have the extremely excellent storage capabilities [6–8]. Data centric methods [9] [11] [12] become the main methods to improve model performance. These data-centric methods become the cornerstones of industrial level NLP system. On the other hand, reward-based methods [13] formulate the world into a reward based framework. And we need a lot of engineering works to make a reward based closed-loop system.

The development of LLMs/ChatGPT has made it possible to build more advanced expert systems. Current LLMs can be viewed as an AI system with strong fuzzy memory capabilities, based on the Turing computer architecture. As an expert system, Current LLMs lacks the ability to answer high-precision questions, such as code execution results. The examples are shown in Table 1 and Table 2.

In this paper we propose a design of AI system, called LLM-text2code system, that try to answer the high-precision questions for LLMs more accurately.

The LLM-text2code system has these modules:

1) LLMs that is pretrained on text and code.
2) An intent recognition module that determines whether to return code execution results or only natural language results.
3) A final answer generation model to combine the results of code execution and natural language results into a final answer.
Prompt and LLMs Answer

Prompt
Break down 2024 into all the possibilities of multiplying two numbers together.

Answer
Sure, here are all the possible pairs of integers that can multiply together to give 2024:
1 x 2024
2 x 1012
3 x 674
4 x 506
6 x 338
7 x 289
8 x 253
9 x 224
12 x 169
14 x 144

Table 1. The example that is wrong answer by LLMs, which prompt is called a high-precision question in this paper.

Prompt and LLMs Answer

Prompt
How much aspirin should a person weighing 75kg take at one time.

Answer
Generally, the dosage of aspirin depends on the reason it's being taken.
For example, for pain relief, the typical dose for an adult is between 300-600mg every 4-6 hours,
not exceeding 4g in a 24 hour period.
However, for heart-related concerns, the dosages are typically much lower, often around 75-100mg per day.

Table 2. The example that is insufficiently precise answer by LLMs, which prompt is called a high-precision question in this paper.
2 Related Work

There are great code-LLM works [1, 3–5, 14, 15] that are pretrained on code and text. Our paper presents some contributions more from a perspective of applications such as expert system.

3 The LLM-text2code System

In this section we illustrate the detail of our design.

![Diagram](image)

**Fig. 1.** The overall framework of our design

3.1 Common LLMs Module

This module includes natural language generation model and natural language to code generation model.

3.2 Intent Recognition Module

This module determines whether to use the text2code module or not. If the user only needs a relatively vague answer, we can just return the results of the text, without code execution results.

3.3 Answers Mixing Module

This module assembles the results of code execution and natural language intermediate results into a final result.
4 Discussion

The text2code approach has a lot of potential, because, in fact, all natural languages have their logical forms. We can design specific code for the application, for example SQL is a kind of code for the specific database-based application.

Letting LLMs to memorise all the correct answers is very time-consuming and difficult, so joining text2code methods with LLMs is a promising way forward.

Returning fuzzy answers for fuzzy questions by LLMs and returning high-precision answers for high-precision questions by text2code are ideal solution.

5 Future Works

In our design, we view that data is part of the code. That is, models trained on the data, viewed as part of the code.

In the future, we will try to design an AI system contains more modules: code2text, code2code, to build a better expert system like a better AI-doctor/AI-lawyer/AI-teacher.

References