An Analysis of Large Language Models' Solutions on Undergraduate Assignments

Jingbin Qian, Rongwei Peng, and Zhiying He

Abstract—**With the recent progress and development of Large Language Models (LLMs), their utilization to aid daily life, professional tasks, and academic pursuits have become increasingly prevalent. However, many students nowadays rely on LLMs to complete their assignments instead of actively engaging in self-directed learning. Using LLMs as a learning aid is acceptable for students, however, we are concerned that LLMs' availability of direct answers may lead students to mere copying without actual comprehension. To address this concern, our team aims to analyze the performance of different LLMs focused on undergraduate assignments. This paper identifies the limitations of these models through evaluating LLMs responses, compares various architectures of LLMs, elucidates the impacts of different architectures on question answering, and provides educators with effective recommendations for augmenting assignment questions. Finally, the experiment shows that almost all LLMs find it hard to generate answers when questions contain real life experience, and they fail to conduct reasoning inference and provide accurate calculations. Generally, LLMs become confused when questions are lengthy and abstract. Based on these failures, it is helpful for educators to propose assignments that prevent students from excessively relying on LLMs.**

Index Terms—**Natural Language Processing, Large Language Model, Machine Learning, Neural Network and Deep Learning**

I. INTRODUCTION

NATURAL Language Processing has emerged as a prominent subject of discussion in recent years, garnering global attention since the release of ChatGPT in 2022. While Large Language Models (LLMs) have made significant positive contributions, they have also given rise to certain negative implications. As LLMs continue to enhance their problem-solving capabilities, an increasing number of students are relying on them to complete their assignments.

This issue also raises concerns among educators as students tend to overlook the importance of serious participation in the courses and overly rely on LLMs for completing their assignments, with some even resorting to using LLMs to help with writing their own papers. Consequently, educators face challenges in accurately assessing students' academic progress, while the overall improvement of students' learning capabilities remains stagnant. Educators, faced with the formidable task of accurately evaluating student learning, must explore strategies to enhance assignment questions that cannot be easily accomplished by LLMs instead of students themselves. Consequently, there arises a pressing need for a more comprehensive understanding regarding the capabilities and limitations of LLMs in problem-solving.

To investigate the performance of LLMs' answering to undergraduate assignments and analyzing the underlying reasons behind that, we divide our work into three steps:

1. Collect the assignment questions from Beijing Normal University-Hong Kong Baptist University United International College (UIC) courses.
   The questions type includes mathematics, computer science and writing, which focus on computer science major students. These questions are fed into different LLMs to find out each LLMs' performance and their problems during answering.
2. Analyze different models' architecture including OpenAI's ChatGPT, Google's Palm2, Baidu's Ernie, iFlyTek's Spark, Alibaba's Qwen and Tencent Hunyuan, which are selected from state-of-art LLMs in the industry.
3. Find out the specific training methods that affect the performance of the answering to undergraduate assignments.
   In-depth analysis of the internal LLMs' structures is conducted in this part. After analyzing their training methods, we can give suggestions about how to adapt students' questions according to their drawbacks.

Our main contributions are as follows:

1. We find out LLMs common problems when answering undergraduate assignments. For mathematical problems, LLMs are poor at logical thinking and often give a wrong calculation result; For computer science questions, LLMs are poor at understanding longer and complex questions. They may miss characters or digits while inputting and would generate non-executable code; For writing problems, LLMs are weak at generating specific contents and long essays. They may make up facts.
2. We compare the different performances between world

This paragraph of the first footnote will contain the date on which you submitted your paper for review, which is populated by IEEE. It is IEEE style to display support information, including sponsor and financial support acknowledgment, here and not in an acknowledgment section at the end of the article. For example, "This work was supported in part by the U.S. Department of Commerce under Grant 123456." The name of the corresponding author appears after the financial information, e.g. *(Corresponding author: Jingbin Qian.)*

Jingbin Qian, Rongwei Peng and Zhiying He are with the faculty of science and technology, Beijing Normal University-Hong Kong Baptist University United International College, Zhuhai, Guangdong 519087, China (e-mail: jingbinqian@outlook.com;).

This article has supplementary downloadable material available at ___, provided by the authors.

Digital Object Identifier

leading LLMs (e.g. ChatGPT) and Chinese LLMs (e.g. Baidu Ernie) and give reviews about these state-of-art models' architectures and training methods. Readers can easily find out their architecture differences from our paper.

3. We give useful suggestions to educators to help them modify their assignments' questions. (a direction of how educators can modify their evaluation methods to students)

II. Literature Review

The development of AI large model has gone through several stages. The appearance of Convolutional Neural Network can be seen as the technology trigger stage. After that, the idea of deep learning has developed. In 2013, the proposal of Word2Vec prompted the development of word vector model and has a profound impact on natural language processing. In 2017, Google published the Transformer model, giving a new approach to process long sequence text by self-attention mechanism and parallel processing capability. In 2018, with the publishment of GPT-1 and BERT model, the era of pre-training large model began. After ChatGPT was published at the end of 2022, the outstanding performance of ChatGPT made AI become a hot topic and a lot of companies follow. 2023 became the year of LLMs explosion, many companies published their large model with architecture similar as ChatGPT (based on Transformer).

A. ChatGPT

1) GPT-1

The Initial GPT is developed using the traditional Transformer [25]. The framework is split into two parts as shown in Figure 1: Unsupervised pre-training and Supervised fine-tuning. The first stage is to train a language model with corpus, followed by fine-tuning stage where parameters adapt for specific tasks.

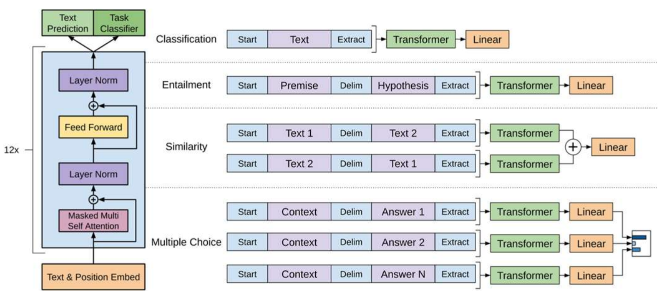


Fig. 1. (left) Transformer architecture used in GPT-1. (right) input transformations for different tasks. [18]

Unsupervised pre-training (**Fig. 1.**, Left Image):

In the pre-training part, it uses a traditional Transformer. The training data (corpus) is given to the embedding layer for tokenization and feed into a transformer model. These text features go through Attention Layer, Normalization, Feed Forward Network and use a standard language modeling objective to maximize the likelihood. The neural network parameters are trained using stochastic gradient descent.

Supervised fine-tuning (**Fig. 1.**, Right Image):

For some tasks, the model can be directly fine-tuned to fit the output. However, some tasks have structured input like question answering and textual entailment. So, the team made some modifications to fit the input. It is shown in Figure, right image. They convert the structured inputs into an ordered sequence, and each sequence adds randomly initialized start and end tokens (<s>, <e>). If an input has different parts, they use delimiters to split them.

2) GPT-2

Current machine learning systems are good at specific tasks that are pre-trained. It is understandable because the limited training datasets affect the accuracy of prediction largely, which makes the performance of the model unstable when inputs are various. In real life situations, there is a variety of data we want to input. Supervised methods lack self-learning ability, which is not a real intelligent system. The GPT teams want to develop a system for multitask training in NLP, and they think a language model can be able to infer and perform a task demonstrated in a sequence of natural language, so they utilize Zero-Shot Learning (ZSL) [13] to test a language model. They called this approach: Language Modelling.

ZSL was created to overcome the difficulties that traditional fine-tuning faces: they need a big data set to deal with some subjective problems, but it is hard to find such an enriching training context. The idea of ZSL, in short, is to mimic what human learn new things. Humans learn new things from their experience and use previous knowledge to deduce a detailed form of a new object to recognize this object (also called top-down inference in psychology). ZSL has a similar idea to make computers be able to identify new objects.

The Figure 2 below illustrates an example: the model does not understand what Zebra is (Zebra image is not in training dataset), but the model know Zebra is a horse-like animal with stripe on its black-white skin. The model knows horses, donkeys, tigers, penguins and so on. It can extract these features on previously trained images to predict what Zebra is.

This is just a rough explanation, for detailed, please refer to the paper [13].

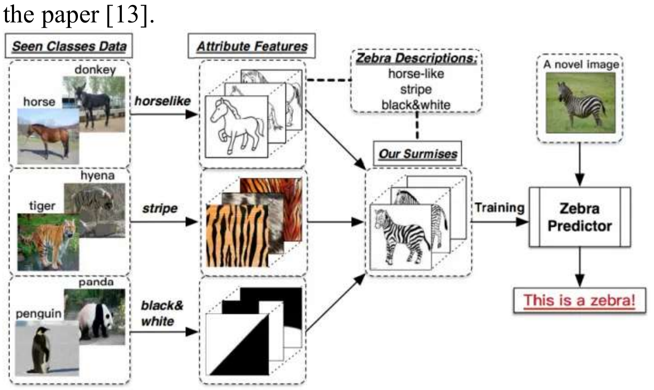


Fig. 2. An example showing the process of a Zero-Shot Learning Task. [13]

Apart from utilizing ZSL, GPT team also followed GPT-1 model to make a few modifications: Layer normalization was moved to the input of each sub-block, additional layer normalization was added after the final self-attention block and initialization was modified (scale the weights of residual layers). For training dataset, WebText dataset was created by a new website scrap which emphasized the quality of the document, use Byte-Paring instead of traditional UTF-8 to minimize the vocabulary.

This research proved that a single pretrained language model can be transferred zero-shot to perform standard NLP without the need for finetuning on a dataset. However, the performance is unsatisfactory, which only matched some supervised baselines. It is predictable. Computers nowadays are not able to think like humans because of computation power and algorithms limitations.

3) *GPT-3*

GPT-3 is based on GPT-2 but trained by a 175 billion parameters autoregressive language model. It is found that the larger the model is, the higher the performance will get (but it has an upper limit). Thus, the concept of the word "Large Language Model" appears. GPT-3 scales up the model size, dataset size, diversity and length of training. As for training dataset, it uses CommonCrawl, WebText, two internet-based books corpora and English-language Wikipedia.

In the previous part, we discussed Zero-Shot Learning. The OpenAI team is unsatisfied with this method. So, in GPT-3, they compared four types of training setting to find the best one, including Fine-Tuning, Few-Shot Learning, One-Shot Learning and Zero-Shot learning.

Fine-Tuning: update the weights of a pre-trained model to specific task, which can get high performance on many benchmarks. But it requires a large dataset for every task.

Few-Shot Learning: Given a few examples of the task at inference time but not update weights.

One-Shot Learning: Similar as few-shot learning but only given one example.

Zero-Shot Learning: Similar as few-shot learning but only given natural language description without specific examples.

After analyzing these four methods, they found that Few-Shot Learning is the best for large language model. Largen the model can make Few-Shot Learning have the same or higher accuracy than fine-tuning, as shown in Figure 3.

Fig. 3. The accuracy of different training settings on various benchmarks. [3]

B. ERNIE

Baidu's team also bases Transformer-XL [7] model to build their language model, Ernie 3.0 [23], but the innovation part is that they split the transformer block into several specific modules for specific tasks. They want to explore the effectiveness of large-scale pre-trained models. Ernie 3.0 contains two parts: Universal Representation Module and Task-specific Representation Module. The general architecture is shown as Figure 4 below.

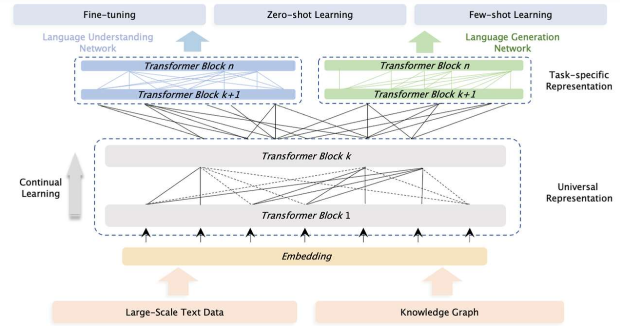


Fig. 4. Ernie 3.0 model's architecture. [23]

As for the Universal Representation Module, they set this module a large size so that the model can capture universal lexical and syntactic information. This is to make semantic information stronger.

As for the Task-specific Representation, they set each module a manageable size. Each module does a specific task like Question Answering, Text Classification. The universal text features are split into different specific modules to fit their module inputs, which is easier to fine-tune and make different applications.

This method makes the semantic information capture more powerful and makes model's parameters smaller and easy to distinguish from different task.

C. Pathways Language Model (PaLM) [5]

PaLM was released by Google in April 2022. PaLM was built based on a standard Transformer model architecture [25] in a decoder-only setup, with modifications to improve performance, training speed or computation cost. Some of the modifications are listed as follows:

1) Use Swish Gated Linear Units (SwiGLU) activation (Swish(xW) xV). Compared to standard Rectified Linear Unit (standard ReLU), Gaussian Error Linear Unit (GeLU)[21] or Swish activations, it significantly increases the quality[21].
2) Use parallel layers. In each Transformer block, use a "parallel" formulation rather than the standard "serialized" formulation, showing a faster training speed [5]. Specifically, the standard formulation can be written as:

$$y = x + MLP(\, LayerNorm\,(\, x + Attention(\, LayerNorm(x)\,)\,)$$

Whereas the parallel formulation can be written as:

$$y = x + MLP\big(LayerNorm(x)\big)$$

$$+Attention(LayerNorm(x))$$

3) Use Rotary Position Embedding (RoPE) [22] to improve the performance on long sequence lengths.
4) Not use biases terms.
   No biases terms in any of the dense kernels or layer norms, resulting in higher training stability.

D. QWEN

Qwen [2] is released by Alibaba. The current version is Qwen 14B (14 billion parameters), which is improved by previous version 7B Qwen. Qwen builds a family for different fields of use, including Qwen-Chat, Code-Qwen, Math-Qwen-Chat, and Qwen VL, with different preferences for datasets. Qwen model uses 3 trillion tokens for pre-training. It is essential to ensure that the training data is diverse and covers a wide range. The data mainly involved public web documents, encyclopedias, books, codes, etc., and the data involved multiple languages, but mainly Chinese and English. To ensure the accuracy of the data, Qwen performs manual sampling and screening from various source data. However, there is still a gap in learning performance between Qwen and GPT. In Figure 5 below, Qwen compared its learning performance with GPT on every different dataset, the result shows that Qwen is still worser than GPT-4 and GPT-3.5.

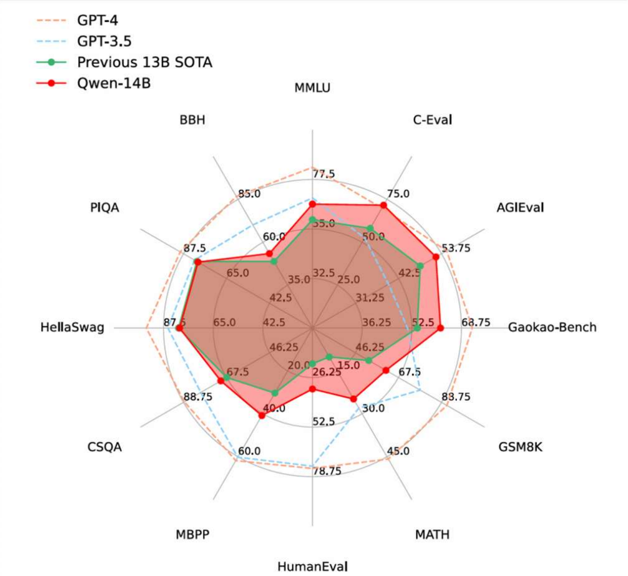


Fig. 5. Performance of GPT-4, GPT-3.5, the previous 13B SOTA, as well as QWEN-14B.

Qwen is also a Transformer finetuning model. It modified an open-source large language model LLaMA [20]. It used the most effective method of finetuning and reward model, Supervised Finetuning and Reinforce Learning with Human Feedback, which is the same as ChatGPT [16].

E. SPARK & HUNYUAN

Spark and Hunyuan are not open-source models, so the detail structure of the two models is restricted to access, Spark was released by IFLY-TEK in May this year. According to the existing information, Spark is also based on the Transformer architecture with more than 100 billion parameters. It used more than 100 billion Chinese text data for training. Hunyuan was released by Tencent in September in 2023. Hunyuan has more than 100 billion parameters and more than 2 trillion tokens in pre-training corpus, including a huge amount of Chinese text data.

III. LLMs' Performance in Solving University Assignments

A. Research Method

To find out the problems of LLMs doing undergraduate assignments, we collected 22 different types of questions from BNU-HKBU United International College, containing three categories of questions, computer science related questions, mathematical questions, and writing questions. These are the three main categories of questions for computer science students. For each category, different fields of questions were collected. For example, computer science questions contain database, webpage design, coding, and algorithms; mathematical questions contain linear algebra, calculus, and statistic; writing questions contain Chinese writing and English writing. For each single question, we have a corresponding sample answer. We graded each LLMs' response to questions according to the sample answer and calculate their accuracy. A grading scheme was introduced to ensure scoring fairness.

Grading Scheme:

1. Give a sample answer score. Divide the sample answer into each small point. For each small point, give 1 or 2 marks (depending on importance). Finally add each small point up to get the total score of this sample answer.
2. Put all the questions to different LLMs (ChatGPT, Palm2, Ernie3.5&4.0, Spark, Qwen and Hunyuan) and select their best response. Then we grade their responses based on the sample answer (compare each small point, if reach the point, give mark. Otherwise, give 0 mark).
3. Traverse all the questions using Scheme 1 and 2 to get all the scores.
4. Finally, using the formular below to calculate the performance of different types of questions for each LLMs.

$$\frac{\sum_{i=1}^{n}(m_i \times \frac{\sum_{j=1}^{n} s_j}{n s_i})}{\sum_{i=1}^{n} s_i}$$

where $\boldsymbol{n}$ is the total number of questions in each category (mat, cs and writing), $\boldsymbol{i}$ and $\boldsymbol{j}$ are the index of the questions in $\boldsymbol{n}$ number, $\boldsymbol{s}$ is the full score of a question (sample answer), $\boldsymbol{m}$ is the mark of a LLM answer the question $\boldsymbol{i}$ obtained.

To prove the generality, the chosen questions are representative, and we also collected students' results from Neural Network and Deep Learning course's Small Project, which shows similar patterns. For detailed grading and performance of each model, please refer to the Appendix.

B. General Results of Mathematical Questions

For mathematical questions, we found almost all LLMs have the following problems:

1. Easy to calculate a wrong result (even if the formula is correct): To our surprise, we found that LLMs are poor at arithmetic operations even if they give a correct formula. For example, as shown in Figure 6, even for a simple division, it gives the wrong calculation.

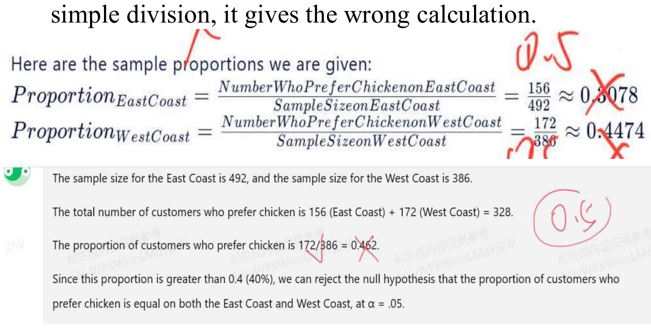


Fig. 6. Wrong calculation by Spark and Hunyuan

Other LLMs also have the same problem, especially when generating long answers, but they are better than Spark. Spark is significantly poor at handling math problems.

2. Miss digits during calculation: Sometimes, when generating long answers, LLMs may miss the digits during calculation or other descriptive answering, resulting in a wrong result. This frequently happens when the questions contain lots of arithmetic operations.
3. LLMs are difficult to select a correct formula to fit background case from several similar formulas: For some situations, LLMs should choose a correct formula from different similar formula, and they are easy to distinguish the correct formula when the question's background is complicated.
4. Poor ability of thinking and in-context learning, very bad at handling mathematical reasoning.
   Due to the limitation of computer power and deep learning algorithms, computers nowadays are not able to think like humans. Thus, for some abstract and logical mathematical questions, LLMs are poor at thinking and learning from the background. For example, in discrete structure, almost all LLMs failed to answer these questions. For some questions that need to combine the background and real-life experience, LLMs will generate messy information that makes us confused. As shown in Table 1, the scores of the Discrete Structure assignments are poor.

TABLE I

ANSWER SCORES OF DISCRETE STRUCTURE QUESTIONS

Question Type	No.		Scores							
		discribtion	Sample Answer	ChatGPT 3.5	PaLM 2	Qwen	Spark	Ernie 3.5	Ernie 4.0	Hunyuan 1.2
2.2 Discrete Structure	2.2.1	recurrence relation	7	5	0	0	0	2	3	2
	2.2.2	pigeon hole	5	2.5	0	0	0	0	1	0

We have examined 4 different types of mathematical questions and their performance of each question are showing in Table 2.

TABLE II

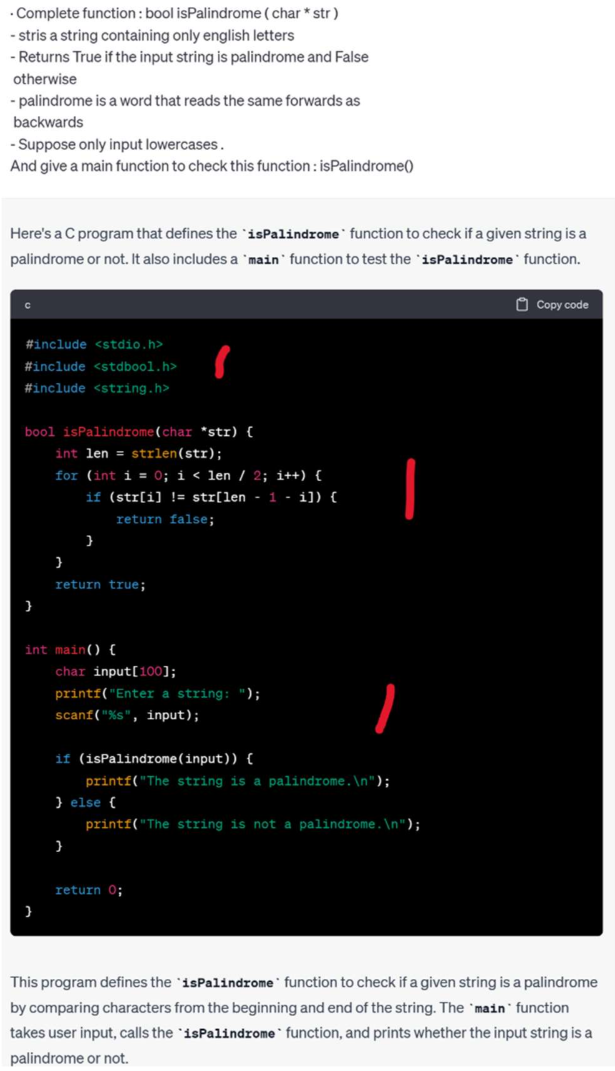
ANSWER SCORES OF MATHEMATICAL QUESTIONS

Question Type	No.		Scores							
		Description	Sample Answer	ChatGPT 3.5	PaLM 2	Qwen	SparkDesk	Ernie 3.5	Ernie 4.0	Hunyuan 1.2
2.1 Calculus	2.1.1	local max min saddle	14	14	7	6	0	7	11	0
	2.1.2	mean value theorem	5	5	5	2	1.5	3	4	2
2.2 Discrete Structure	2.2.1	recurrence relation	7	5	0	0	0	2	3	2
	2.2.2	pigeon hole	5	2.5	0	0	0	0	1	0
2.3 Statistic	2.3.1	hypothesis test	7	4	3	2	2	2.5	3	0.5
	2.3.2	Poisson distribution	5	5	4.5	5	0	1	4.5	0
2.4 Linear Algebra	2.4.1	vector basis	4	3	2.5	3	0.5	2.5	2.5	0.5
	2.4.2	linear combination	3	2.5	1	1.5	0	2	2	0

C. General Results of Computer Science Related Questions

For CS questions, we found almost all LLMs have the following problems:

1. Good for solving simple and short questions, but poor in complex ones.
   When asked to solve short and simple coding questions, all the LLMs are good for outputting common algorithms. Especially for general algorithms like sort algorithms (Quicksort, Insertion sort, Bubble sort, etc.). In one of our test examples (Figure 7), LLMs were asked to do a simple code question - write a function to check if an input word is a palindrome. All the LLMs can output a complete programing including the *isPalindrome* function and test example. Sometimes, even the requirement in the assignment question did not ask LLMs to add test case to test the function, LLMs still did that themselves.



- Complete function : bool isPalindrome (char * str)
- stris a string containing only english letters
- Returns True if the input string is palindrome and False otherwise
- palindrome is a word that reads the same forwards as backwards
- Suppose only input lowercases .

And give a main function to check this function : isPalindrome()

Here's a C program that defines the `isPalindrome` function to check if a given string is a palindrome or not. It also includes a `main` function to test the `isPalindrome` function.

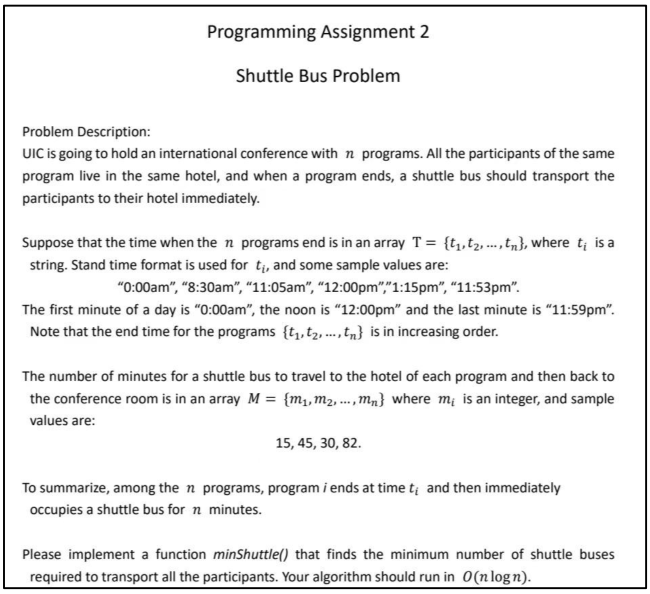
c Copy code

```c
#include <stdio.h>
#include <stdbool.h>
#include <string.h>

bool isPalindrome(char *str) {
    int len = strlen(str);
    for (int i = 0; i < len / 2; i++) {
        if (str[i] != str[len - 1 - i]) {
            return false;
        }
    }
    return true;
}

int main() {
    char input[100];
    printf("Enter a string: ");
    scanf("%s", input);

    if (isPalindrome(input)) {
        printf("The string is a palindrome.\n");
    } else {
        printf("The string is not a palindrome.\n");
    }

    return 0;
}
```

This program defines the `isPalindrome` function to check if a given string is a palindrome by comparing characters from the beginning and end of the string. The `main` function takes user input, calls the `isPalindrome` function, and prints whether the input string is a palindrome or not.

Fig. 7. isPalindrome function and test example

However, LLMs were not good at dealing with longer and complex questions. Especially when there are some specific practical examples in the questions, such as the "Shuttle Bus Problem"(Figure 8) question using more real-life information as the background. These conditions will interfere with their understanding of the questions. We have tried that when we delete all the information about real life, the performance of LLMs become better. Before deleting the background information, some LLMs like Spark and Qwen directly output that they cannot solve this question, but after deleting the background information, all LLMs can output coding even though some of them were wrong.



Programming Assignment 2

Shuttle Bus Problem

Problem Description:
UIC is going to hold an international conference with n programs. All the participants of the same program live in the same hotel, and when a program ends, a shuttle bus should transport the participants to their hotel immediately.

Suppose that the time when the n programs end is in an array $\mathrm{T} = \{t_1, t_2, \ldots, t_n\}$, where t_i is a string. Stand time format is used for t_i, and some sample values are:
"0:00am", "8:30am", "11:05am", "12:00pm","1:15pm", "11:53pm".
The first minute of a day is "0:00am", the noon is "12:00pm" and the last minute is "11:59pm". Note that the end time for the programs $\{t_1, t_2, \ldots, t_n\}$ is in increasing order.

The number of minutes for a shuttle bus to travel to the hotel of each program and then back to the conference room is in an array $M = \{m_1, m_2, \ldots, m_n\}$ where m_i is an integer, and sample values are:
15, 45, 30, 82.

To summarize, among the n programs, program *i* ends at time t_i and then immediately occupies a shuttle bus for n minutes.

Please implement a function *minShuttle()* that finds the minimum number of shuttle buses required to transport all the participants. Your algorithm should run in $O(n \log n)$.

Fig. 8. Shuttle Bus Problem

2. Code writing is constrained by comprehension of the given question.
   When the LLMs did not fully understand the meaning of the question, each LLM output code results proportional to their degree of understanding. Given the same problem, some large models understand most of the details of the problem and can output a code that is close to perfect. If the larger model understands only a few of these details, it will do only a few of them. It is interesting to found that, for the same problem, each model understands different parts of the problem, and ignores different parts of the problem, resulting in different code output. In one of our test problems, LLMs were asked to create a club webpage in HTML. Only ChatGPT noticed every requirement in the question, while the other models missed individual requirements, such as not highlighting the input field, not doing the translation task as required, incorrectly positioned text fields, etc.

3. Miss digits during inputting and poor in giving detailed steps of repeated operations.
   All the LLMs were not good at dealing with data that appears repeatedly, whether it's a single number or the same formula. And if they were asked to give the details of each step for a specific problem, they would miss some data, and even if the algorithm is correct, they will make mistakes in the repeated operation part. In one of our test examples, LLMs were asked to solve one sample t-test question in R. When the LLMs meet a sequence of digits in a problem, such as five consecutive digits "3", the LLMs will recognize the five digits "3" as two digits "3" or three digits "3" (see figure 9 and figure 10). Almost every LLMs miss digits when processing the digits inputting, causing errors in their later calculations. In addition, when LLMs were

asked to compute KNN algorithm with a bench of data in every detailed step. Even if the algorithm is correct, the output steps still become wrong after several same operations. And LLMs always hide the process of showing this part of the error, showing only the beginning calculation and the final calculation results.

A data scientist is analyzing 31-day transaction records from a department store. Below is a table that contains the number of purchases customers made during the period from the transaction records of 30 randomly selected customers.

1	3	6	2	3	4	4	2	6	3
3	1	8	4	3	3	3	3	3	4
5	3	1	4	4	2	3	4	2	3

Fig. 9. There are some consecutive "3" in the data of the question. The correct input should be five digits "3", like this"...8 4 3 3 3 3 3 4 5..."

```
R                                                     Copy code

# Given data
data <- c(1, 3, 6, 2, 3, 4, 4, 2, 6, 3, 3, 1, 8, 4, 3, 3, 3, 3, 4, 5, 3, 1,

# Calculate the sample mean
sample_mean <- mean(data)
```

Fig. 10. ChatGPT only recognized four digits "3", which is "...8 4 3 3 3 3 4 5..."

We have examined 8 different types of CS questions, and their performance of each question is shown in Table 3, and we found that when dealing with CS problems, ChatGPT has the best performance among all the models, followed by Ernie4.0, and Spark has the worst performance.

TABLE III
ANSWER SCORES OF COMPUTER SCIENCE QUESTIONS

Question Type	No.		Scores							
		description	Sample Answer	ChatGPT 3.5	PaLM2	Qwen	Spark	Ernie 3.5	Ernie 4.0	Hunyuan 1.2
1.2 Database	1.2.1	Database relational schema	3	3	2	1.5	0.5	3	3	1
	1.2.2	Database relational algebra expression	3	3	1.5	1.5	0	0	3	2
1.3 programing and coding (C,java, python, html)	1.3.1	Check Words Palindrome (C)	3	3	2	2	2	2	3	2
	1.3.2	A simple website (html)	5	4.5	3	5	3.5	2.5	5	2
	1.3.3	Shuttle Bus Problem (C)	6	3.5	2.5	0.5	0	1	0.5	2
1.4 Machine Learning ((Un-)Supervised, Data Minning)	1.4.1	k-means clustering	12	1.5	0.5	0.5	0.5	1.5	1.5	1.5
	1.4.2	Determine Tree	5	5	4	5	5	5	5	5
1.5 data analysis (R)	1.5.1	one sample t-test (critical value & the p-value approach)	6	5	3	3	1	4	2	3
Total			43	28.5	18.5	19	12.5	19	23	18.5

D. General Results of Writing Questions

We found that LLMs are good at writing grammar and article structure. However, it is unsatisfied when asking them to provide detailed supporting materials. What's more, although LLMs can understand and complete some requirements in one step, they do not perform well when the requirements are mixed. Based on our tests of LLMs, we found they had the following weaknesses:

1. Missing detailed requirements

   We found that when we input simple requirements, LLMs can understand it and give good output. However, when the requirements are complex, LLMs may miss some of them. For example, we tried to ask LLMs simple questions like "Write a rationale to the topic 'Explain the main effects of increased access to higher education' in 1 or 2 sentences". The results were that LLMs could generate a strong and clear rationale. When we asked LLMs to contain a rationale in writing an essay, lots of LLMs could not do that (Figure 12).

CA **You**

1.Explain the meaning of a rationale in the introduction of an essay 2. Write a rationale to the topic "Explain the main effects of increased access to higher education" in 1 or 2 sentences

ChatGPT

1. **Rationale in the Introduction of an Essay:**
   The rationale in the introduction of an essay refers to a clear and logical explanation for the chosen topic or subject matter. It is a statement that outlines the reasons behind the selection of the particular theme and serves as a bridge between the general context and the specific focus of the essay. The rationale provides readers with a glimpse into the significance of the topic, its relevance, and why it is worthy of exploration.
2. **Rationale for "Explain the Main Effects of Increased Access to Higher Education":**
   In recent decades, the global landscape of education has witnessed a remarkable surge in the accessibility of higher education. This phenomenon demands a nuanced exploration to comprehend its multifaceted impacts on individuals, societies, and economies. Understanding the main effects of increased access to higher education is crucial for informed discourse on the evolving nature of education and its broader implications.

Fig. 11. A simple prompt about writing a rationale and the output of GPT

Write an essay to explain the main effects of increased access to higher education. The words should be no less than 750 and no more than 1000. Please write 5 paragraphs, including 1 introduction paragraph, 3 body paragraphs and 1 conclusion paragraph.

Please follow the detailed requirements:

Conventions: The introduction contains a relevant introductory statement that gains the reader's interest, clear and interesting background information, a strong rationale, and a complete, clear, relevant thesis. Body paragraphs contain clear and accurate topic sentences which are connected to the thesis and introduce the purpose of the paragraph. Conclusion uses different language to give a clear and concise summary of purpose and key points of the essay. The concluding sentence is clear and appropriate.

Effectiveness: Content: The writer paraphrases and cites source material frequently and accurately. The essay is focused, and ideas are developed with detailed explanations and relevant evidence. Organization: Relationships between ideas are clearly expressed.

Vocabulary: Writer uses a wide range of accurate vocabulary. Writing nearly always maintains an appropriate level of formality. Writing contains few word choice / usage errors.

Grammar: Writer uses a wide range of structures. Writing contains few errors in grammar, punctuation, or spelling.

Fig. 12. A complex writing prompt.

TABLE IV
SUMMARY OF LLMS' ANSWERS TO THE REQUIREMENT OF CONVENTION

The requirements for convention \ Models	Chat-GPT3.5	Palm2	Qwen	Spark	Ernie 3.5	Ernie 4.0	Tencent-Hunyuan 1.2
A relevant introductory statement that gains the leader's interest.	1	0	1	1	1	0	1
Clear and interesting background information.	1	1	0	0	0	1	1
A strong rationale.	0	0		0	0	0	0
A complete, clear, relevant thesis.	1	1	1	1	0	1	0
Body paragraphs contain clear and accurate topic sentences which are connected to the thesis and introduce the purpose of the paragraph.	1	1	1	1	0	1	1
Conclusion uses different language to give a clear and concise summary of key points of the essay.	0	0	1	1	0	1	1
Conclusion mention of the purpose of the essay.	0	0	0	0	0	0	0
The concluding sentence is clear and appropriate.	0	0	1	0	0	1	1
Total marks for the convention requirements	4	3	5	4	1	5	5

2. Generating spurious data

We found that LLMs would generate fake data. The generated data may appear semantically coherent but lacks consistency with authentic sources. For example, we asked LLMs to generate references and found that most of the references were not true.

Ernie 3.5 output:

Kross, Ethan, et al. "Facebook Use Leads to Decreases in Subjective Well-being in Social Comparisons with Others." Journal of Computer-Mediated Communication 18.5 (2013): 1007-1026. Print.

A real reference:

Kross, Ethan, et al. (2013). Facebook use predicts declines in subjective well-being in young adults. *PLoS ONE, 8*(8), Article e69841.

Fig. 13. An instance of LLMs generating spurious data.

3. Not enough length

We tried to ask LLMs to write long essays and found that they cannot do that in one prompt. For example, we asked LLMs to write no less than 750 words on several topics. It came out that LLMs can only output around 500 words in general.

4. Lack of specific supporting information

In most cases, LLMs could not refer to real source materials. They cannot generate output according to actual details of academic research. When LLMs are asked to add personal experience, the answers showed similarity in different LLMs, and the content is not in depth. LLMs tend to generate general expansions on the topics. Although they sometimes do refer to source materials, the reference content is not always true.

TABLE V
SUMMARY OF LLMS' ANSWERS TO THE REQUIREMENT OF EFFECTIVENESS

The requirements for effectiveness \ Models	Chat-GPT3.5	Palm2	Qwen	Spark	Ernie 3.5	Ernie 4.0	Hunyuan 1.2
The writer paraphrases and cites source material frequently and accurately.	0	0	3	0	0	4	0
The essay is focused, and ideas are developed with detailed explanations and relevant evidence.	4	2	3	2	3	3	3
Relationships between ideas are clearly expressed.	4	3	3	3	2	3	2
Total marks for the convention requirements	8	5	9	5	5	10	5

5. Confused logic

Confused logic means that the connection of main ideas is hard to follow. This is a seldom occurred problem, but it still may happen.

Firstly, we must consider the positive impact of increased access to higher education. The most significant benefit is the widening of opportunities for individuals. The increase in enrolment has led to a larger pool of skilled and highly trained individuals, who have become an asset to society. These educated individuals contribute to the growth of knowledge, technology, and innovation, leading to economic development. In addition, increased access to education has also reduced income inequality, as education is a key determinant of income. As more individuals gain access to higher education, the gap between the rich and the poor is slowly but surely closing.

However, increased access to higher education has not always had positive outcomes. One of the main concerns is the rise in student debt. As universities expand their intake, the cost of education continues to rise, resulting in many students taking on loans to finance their studies. This has led to a surge in student debt, which has had a knock-on effect on young people's ability to buy homes, start families, and pursue their dreams.

Moreover, the increase in higher education has not necessarily led to better job opportunities for graduates. While educated individuals may have an edge in the job market, the competition for jobs has also increased, leading to high unemployment rates among new graduates. In some cases, the skills and knowledge gained from higher education do not always translate into employment, resulting in a mismatch between educational qualifications and job requirements.

Finally, one of the broader effects of increased access to higher education is its impact on society's cultural and social fabric. The expansion of higher education has led to a more diverse and inclusive society, as marginalized groups have had greater access to education. This has led to a richer tapestry of ideas, cultures, and perspectives that have added depth and richness to our social landscape.

Fig. 14. Giving a prompt to write about the effects of increased access to higher education. Ernie 3.5 generated a messy answer.

IV. REASONS OF LLMS' LIMITATIONS

A. Mathematics Problems

In a traditional view, computers are good at calculation. But after testing all the LLMs, we found that almost all of them have problems when doing calculations. They may have wrong results (even if given a simple formula, it may calculate a wrong result, especially for fraction calculation). After analyzing their models' architecture, we found that they may not use mathematical tools to validate their results. They just use training methods to find the most likely answer. We take GPT as an example to show the reasons:

Reasons for limitation of LLM on dealing with mathematical problems:

GPT-3 itself can do simple arithmetic operations without specific training by using few-shot learning. Because in practical use, many new patterns have not been trained before, using few-shot learning can adapt quickly to an unusual task. In arithmetic expressions, few-shot learning can be based on previous examples and experience on calculation to do complex operations.

The performance of GPT-3 is below:

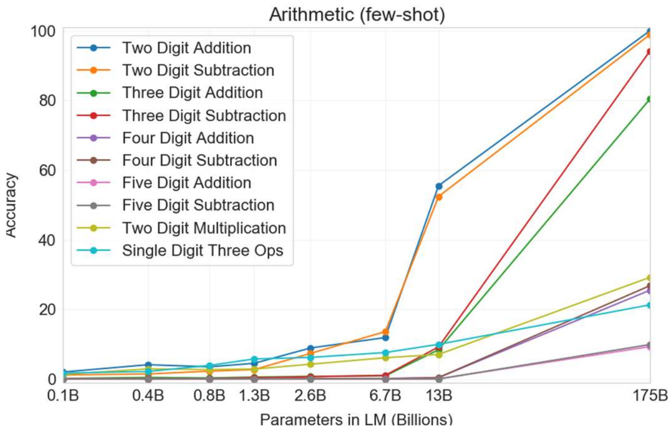


Fig. 15. GPT 3's arithmetic accuracy. [3]

We can see that from Figure 15, the bigger models' parameters, the higher the arithmetic performance. It suggests that we can make the model larger to obtain a higher performance. This is also one of the reasons why so many companies increase their model's size.

However, the more digits in one calculation, the easier it is to lose performance. It suggests that LLMs may be easier to calculate incorrectly if the questions have many digits and operators. This also proves our findings.

Other LLMs excluding GPT may have the same problems. From the architecture of Ernie 3.0, we can see that it also uses few-shot learning. Other LLMs, due to the company secretes, we do not know the detailed architecture, but we can convince that they just use different settings (fine-tuning or few-shot) to do the calculation. The basic theory is just to find similarity and probability, so it will make mistakes.

However, a question pops out. We know that different LLMs may use similar methods to adapt the learning process, but why is ChatGPT's performance much higher than other LLMs?

There are three possible reasons:

1. Training dataset

   ChatGPT expanded a large training dataset including the Common Crawl dataset (after filtered), WebText dataset (scrap web pages which have been curated/filtered by humans), two internet-based books corpora and Wikipedia. However, Baidu's Ernie3.0 only uses dataset from (a Chinese text corpora constructed by themselves), Chinese multi-modal pre-training data, WuDaoCorpus, PanGu Corpus and several other corpuses from Ernie2.0 (baike, wikipedia, feed and etc.), Baidu Search (Baijiahao, Zhidao, Tieba). We cannot ensure the quality of each dataset since most of their datasets are not published, but the GPT team mentioned that their dataset is filtered by human to obtain a high quality. Additionally, we know that Baidu's Tieba, Zhidao are Chinese online forum where anyone can put anything on. It may affect the quality of Baidu's dataset.

2. Training Verifiers to Solve Math Word Problems. [6]

   To obtain a higher performance in mathematical problem, GPT team purposed a method called Verifiers to judge the correctness of solutions.

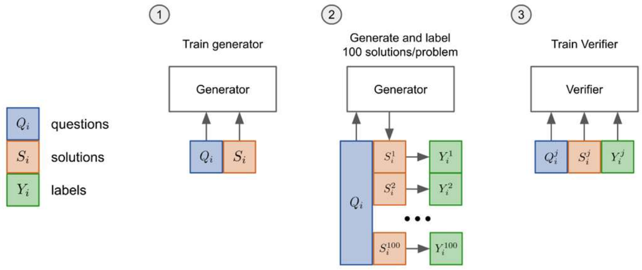


Fig. 16. The process of verification method pipeline. [6]

The above figure shows a pipeline of a verification training process. First, finetune a model ("generator") for 2 epochs on training set, and use the question and its corresponding solution to train this model. Then, input the same question and sample 100 answers generated by the model and label them as correct or incorrect. Finally, use these tuples of data to train the verifier. After introducing this method, GPT team found that verification provides a significant performance increase than just fine-tuning (figure 17).

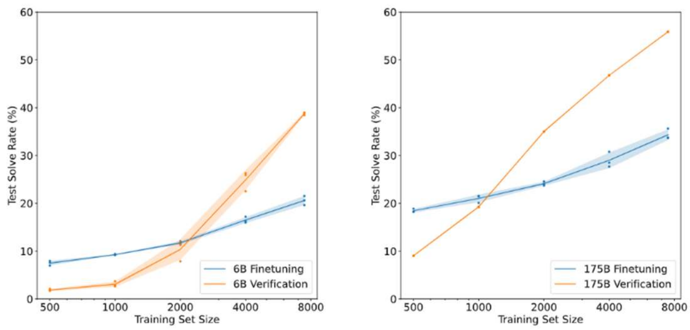


Fig. 17. The performance after introducing verification. [6]

However, this method also has some drawbacks. The training dataset is GSM8K (high quality grade school math problems), but our research is for undergraduate assignments. So, for some complicated university questions, it may appear mistakes. The second is that their method frequently failed to accurately perform calculations (which also prove our findings), and they introduced a calculator annotations method to mitigate this issue.

To mitigate the calculation issue, ChatGPT introduces a calculator by injecting calculation annotations "<< >>" into the training sets. After identifying the special token, the models will do the calculation (Figure 17). This method splits the arithmetic expression from input text to do the calculation, but it only has a slight improvement. However, this method is easy to ignore some digits that should be annotated, and GPT team also admitted that. From our findings, ChatGPT sometimes will miss some digits when generating long answers. It may be due to the generation of incorrect annotations or the ignorance of some lines that should be annotated because this method is not provided by humans but the combination of hard-code logic and fine-tune language model.

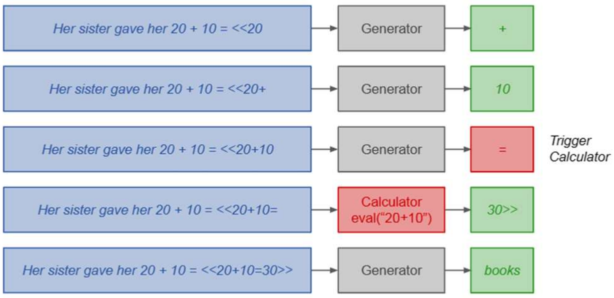


Fig. 18. A diagram of a calculation annotation procedure. [6]

3. Used reinforcement learning and trained from human feedback. [15]

   OpenAI fine-tuned the GPT-3 by introducing a reward model. They collected and ranked the preference of labelers for different outputs which are in response to the given prompt. Used this data to train a reward model and optimized by using reinforcement learning. So, after GPT gives a wrong result, we tell its mistakes and it can make a correction and output a desired result. Even if it is not always correct, its performance is much higher than other models (contrary example: Ernie 3.0 always sticks to its wrong point).

Advice for teachers to modify mathematical questions:

- Ask students to give details of the calculation process, like in data mining, give each clustering steps, decision trees steps, etc.
- Give more digits in calculation as possible and choose the formula that is easy to confuse.
- Apply more logical thinking in question. Not just simply followed the algorithm. Create more special situations and conditions.

B. Computer Science Related Problems

Code generating is based on model's training and learning on large amounts of code data. LLMs learn the syntax, structure and pattern of code, and the correspondence between code and natural language, so that they can generate code according to the nature language input. LLMs learn from code data is similar as learning natural language, the difference is that the language of code has a stricter syntax, so the error probity increased.

Reasons for limitation of LLM on dealing with CS problems:

1. Quality and Quantity of the pre-trained data.

   The ability of LLMs is closely related to the quality of the training data. If there is wrong code in the pre-training data, LLMs may learned wrong code. Even though the LLMs trained the code data from GitHub, there are still some codes with poor quality. So, it is necessary to rigorously inspect and screen the pre-training data. Besides, to generate good code, the model requires a large amount of code data as a data set. Some models are not trained enough on code data and do not have better code generation ability. In fact, even if the model is trained on a hundred million lines of code data, its ability to solve code problems is just close to less capable than an introductory computer science student's.
2. Inaccurate understanding of the question requirement and lack of deep understanding of semantics and logic of code.

   When LLMs generate code according to the input prompts or instructions, they need to understand the meaning and logic of text. If models got an inaccurate understanding of the input, then it is possible to generate incorrect code. Sometimes LLMs may simply fill in the code based on the rules they learned from training data, without a deep understanding of the semantics and logic of the code.
3. Need to understand more contextual and semantic information

   Large language models are trained by predicting the next token, which means they are better at generating code that is like their training data. When solving simple and common CS questions, it is easy to find the same question on the training dataset. So almost all LLMs can solve common code problems quickly and accurately. However, for complex code problems, the model may need to understand more contextual and semantic information, which may be beyond the ability to predict the next token. For example, in our previous test example "Shuttle Bus Problem", there are some particular information like UIC conference, program ending time ..., which does not exist in the question of training dataset so it is not familiar for LLMs. LLMs need to understand the meaning of new items and then find the mathematical and logical relationship between the unfamiliar items.

4. The length of input question requirement.
   When the question becomes longer, the ability of LLM to generating code decreases. In 27 OpenAI's Codex [4], they found that when the length of docstring increased, the code generating performance decrease a lot (See Figure 22). The component of docstring is equivalent as the word in the sentence of question requirement, and the pass rate is equivalent as the learning performance of LLMs.

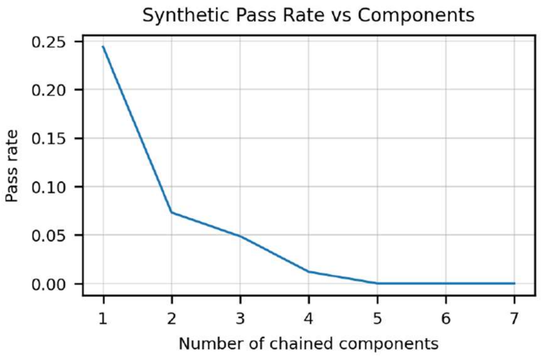


Fig. 19. Pass rates of Codex-12B samples against the number of chained components in the synthetically generated docstring. With each additional component, pass rate drops by roughly a factor of 2 to 3 [4]

5. The way LLMs deal with numbers.
   In our test, when LLMs deal with the data analysis in R, they always miss some input data, especially the repeated digits. We found that LLMs do not treat numbers as normal text or process them like MATLAB, they use method to identify and split number (the way as mentioned in the part of mathematic problems), so they may recognize wrong numbers.

Advice for teachers to modify CS questions:

- Give more information and practical context in question. For simple code questions, it is easy for LLMs to solve, so teachers can design an actual background for simple CS questions. Although this background information has no effect on the meaning of the questions, it can reduce the completion rate of LLMs to some extent.
- Ask students to work on questions with every detailed steps. In many cases, the model will not refine to the output results of each step when dealing with the problem. If the students cannot give the specific solution on each step, but have the correct answer, then they are highly likely to use LLMs.
- Add continuously repeated data to the data of question given to students. LLMs has a poor ability to deal with repeated data and is often ignored directly, so that it can be judged whether students have completed it independently.

C. Writing Problems

LLMs' writing capability limitations result from many aspects, spanning the steps of building large language models. In this section, we discuss the causes of writing limitations in LLMs, basically categorized into two key origins: **Data** and **Text Generation Theory**.

Reasons for limitation of LLM on dealing with writing problems:

Data Theory

LLMs are pre-trained models, so they do not contain up-to-date data. Also, since they need large computation resources to train on all data of human knowledge, the developing teams will select to train on the representative data. That means, LLMs do not contain every local area, event, or academic detail that can be found in human's written text, even if it can be accessed freely on the internet. Therefore, LLMs are good at generating general opinions, but they are weak at providing specific content.

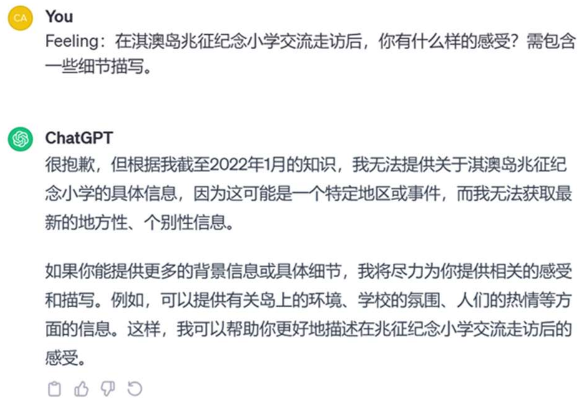


Fig. 20. GPT does not contain specific information about the Qi'Ao Island Siu Ching 29 Memorial Primary School, based on its knowledge as of January 2022.

Text Generation Theory

As introduced in the introduction, the output of LLMs is based on the probability of tokens. The probability distribution in the transformer architecture is calculated by the softmax function. On timestep i, the probability p is computed by applying the softmax function to a vector of scores v:

$$p(x_i) = \frac{e^{x_i}}{\sum_{j=1}^{v} e^{x_j}}$$

To control the randomness in the softmax's output, it is common to add a Temperature parameter (T) to the softmax function. The softmax function with a Temperature parameter is as follows [8]:

$$p(x_i) = \frac{e^{\frac{x_i}{T}}}{\sum_{j=1}^{v} e^{\frac{x_j}{T}}}$$

If raise the temperature T, $p(x_i)$ becomes more uniform, and then the output is more diverse. If lower the temperature T, $p(x_i)$ becomes spikier, and then the output is more rigid. So, there is a balance between focusing on a topic and containing abundant details. If users want to generate an expanded essay with various details, the temperature should be high, and the essay will lose focus on a topic. If users want to output an essay that is focused, the temperature should be low, and the LLM will repeat words. Therefore, cutting the output short can avoid generating messy content in tails. This is the reason why LLMs cannot generate long essays.

What's more, since many output parameters influence the output essay features, and users cannot adjust these parameters in the user interface of most of the LLMs, it explains the randomness in LLMs' output writing style and quality.

Advice for writing problems:

- Ask students to give references and citations in every corresponding place.
- Ask students to write on news and personal experience.
- Write long detailed requirements, so that LLMs could be unable to follow all of them.
- Increase the required length of essays.

V. CONCLUSION

This thesis provided a testing and analysis into the limitation of LLMs' performance on college students' assignments. We have shown that LLMs are poor at mathematical reasoning and calculation, generating intricate coding answers and responding to long word length, specific or complex writing prompts. Further, we analyzed the reasons for this weakness based on LLMs' architectures and their corresponding training methods. We give suggestions to educators on how to modify their assignments.

This work also has limitations:

1. Due to company secret, many technical details of different brands' LLMs are unknown.

2. Internal details of neural network is unreadable, and the parameters inside may be meaningless and hard to explain in human language, just like our human brains. The reason analyzing is based on their training methods but not deep into the parameters inside the network.

3. Technology is developing quickly. Our findings of these LLMs are before Nov 2023 and these LLMs update quickly so that the performance may be changed. For Qwen and Spark, they do not give a specific version. Our testing of different LLMs is all based on the published user interface (except Palm2, we used Google Cloud).

APPENDIX

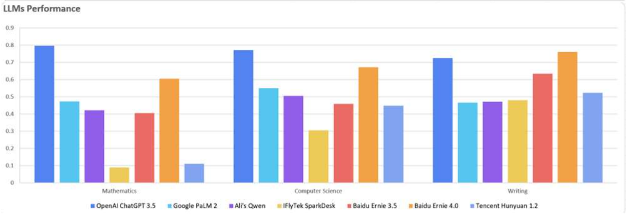
Appendix 1:
All the grading data, including problem descriptions to each question. Can be downloaded from:
https://github.com/jingbinQIAN/Ability-of-Large-Language-Models-on-Solving-Undergraduate-Assignments

Appendix 2: Scoring Table of all the LLMs answering:

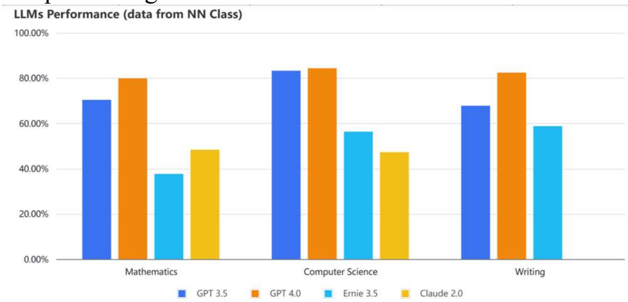
Question Type	No.	discribtion	Scores							
			Sample Answer	OpenAI-ChatGPT 3.5	Google-PaLM 2	Ali-Qwen	iFlyTek-SparkDesk	Baidu-Ernie 3.5	Baidu-Ernie 4.0	Tencent-Hunyuan 1.2
1.1 Algorithm and analysis	1.1.1	Algorithm and analysis	4	2	3	1	0	0	2	0
1.2 Database	1.2.1	DataBase relational schema	3	3	2	1.5	0.5	3	3	1
	1.2.2	DataBase relational algebra expression	3	3	1.5	1.5	0	0	3	2
1.3 programing and coding (C,java, python, html)	1.3.1	Check Words Palindrome (C)	3	3	2	2	2	2	3	2
	1.3.2	A simple website (html)	5	4.5	3	5	3.5	2.5	5	2
	1.3.3	Shuttle Bus Problem (C)	6	3.5	2.5	0.5	0	1	0.5	2
1.4 Machine Learning ((Un-)Supervised, NLP, Data Minning)	1.4.1	k-means clustering	12	1.5	0.5	0.5	0.5	1.5	1.5	1.5
	1.4.2	Determine Tree	5	5	4	5	5	5	5	5
1.5 data analysis (R)	1.5.1	one sample t-test (critical value & the p-value approach)	6	5	3	3	1	4	2	3
2.1 Calculus	2.1.1	local max min saddle	14	14	7	6	0	7	11	0
	2.1.2	mean value threom	5	5	5	2	1.5	3	4	2
2.2 Discrete Structure	2.2.1	recurrence relation	7	5	0	0	0	2	3	2
	2.2.2	pigeon hole	5	2.5	0	0	0	0	1	0
2.3 Statistic	2.3.1	hypothesis test	7	4	3	2	2	2.5	3	0.5
	2.3.2	possion distribution	5	5	4.5	5	0	1	4.5	0
2.4 Linear Algebra	2.4.1	vector basis	4	3	2.5	3	0.5	2.5	2.5	0.5
	2.4.2	linear combination	3	2.5	1	1.5	0	2	2	0
4.1 Chinese Essay	4.1.1	Material exposition	14	10	2	0	5	10	9	0
	4.1.2	Chinese history	6	2	4	0	0	2	5	3
	4.1.3	Chinese social field trip feeling	5	5	2	4	3	4	4	4
4.2 English Essay	4.2.1	Effect essay	45	26	19	25	20	19	28	23
	4.2.2	Effect essay with given writing plan	10	10	7	10	10	9	9	8

Appendix 3: Diagram of Performance of different LLMs:
Our result:

Performance	Total Sample Score	OpenAI ChatGPT 3.5	Google PaLM 2	Ali's Qwen	IFlyTek SparkDesk	Baidu Ernie 3.5	Baidu Ernie 4.0	Tencent Hunyuan 1.2
Mathematics	50	79.61%	47.34%	42.05%	8.88%	40.43%	60.43%	11.03%
Computer Science	43	77.13%	54.91%	50.46%	30.46%	45.83%	67.13%	44.72%
Writing	80	72.51%	46.63%	47.11%	48.03%	63.40%	75.97%	52.22%



Result from Beijing Normal University-Hong Kong Baptist University United International College, Neural Network and Deep Learning course classmates:



ACKNOWLEDGMENT

Jingbin Qian, Rongwei Peng and Zhiying He thank Professor Jefferson Fong who had given us invaluable advice to this paper and the students' data from his Neural Network and Deep Learning class.

REFERENCES

[1] Azaria, A., & Mitchell, T.M. (2023). The Internal State of an LLM Knows When its Lying.ArXiv, abs/2304.13734.

[2] Bai, J., Bai, S., Chu, Y., et al. (2023). Qwen Technical Report.ArXiv, abs/2309.16609.

[3] Brown, T.B., Mann, B., Ryder, N., et al. (2020). Language Models are Few-Shot Learners.ArXiv, abs/2005.14165.

[4] Chen, M., Tworek, J., Jun, H., et al. (2021). Evaluating Large Language Models Trained on Code.ArXiv, abs/2107.03374.

[5] Chowdhery, A., Narang, S., Devlin, J., et al. (2022). PaLM: Scaling Language Modeling with Pathways.J. Mach. Learn. Res., 24, 240:1-240:113.

[6] Cobbe, K., Kosaraju, V., Bavarian, M., et al. (2021). Training Verifiers to Solve Math Word Problems.ArXiv, abs/2110.14168.

[7] Dai, Z., Yang, Z., Yang, Y., Carbonell, J.G., Le, Q.V., & Salakhutdinov, R. (2019). Transformer-XL: Attentive Language Models beyond a Fixed-Length Context. ArXiv, abs/1901.02860.

[8] Holtzman, A., Buys, J., Du, L., Forbes, M., & Choi, Y. (2019). The Curious Case of Neural Text Degeneration. ArXiv, abs/1904.09751.

[9] Khlaaf, H., Mishkin, P., Achiam, J., Krueger, G., & Brundage, M. (2022). A Hazard Analysis Framework for Code Synthesis Large Language Models.ArXiv, abs/2207.14157.

[10]Li, Y., Choi, D.H., Chung, J., Kushman, N., et al. (2022). Competition-level code generation with AlphaCode.Science, 378, 1092 - 1097.

[11]Lin, J., Zhao, H., Zhang, A., Wu, Y., Ping, H., & Chen, Q. (2023). AgentSims: An Open-Source Sandbox for Large Language Model Evaluation.ArXiv, abs/2308.04026.

[12]Liu, C., & Wan, X. (2021). CodeQA: A Question Answering Dataset for Source Code Comprehension.ArXiv, abs/2109.08365.

[13]Lu, J., Li, J., Yan, Z., et al. (2018). Attribute-Based Synthetic Network (ABS-Net): Learning more from pseudo feature representations.Pattern Recognit., 80, 129-142.

[14]Neelakantan, A., Xu, T., Puri, R., et al. (2022). Text and Code Embeddings by Contrastive Pre-Training.ArXiv, abs/2201.10005.

[15]OpenAI (2023). GPT-4 Technical Report.ArXiv, abs/2303.08774.

[16]Ouyang, L., Wu, J., Jiang, X., et al. (2022). Training language models to follow instructions with human feedback.ArXiv, abs/2203.02155.

[17]Rae, J.W., Borgeaud, S., Cai, T., et al. (2021). Scaling Language Models: Methods, Analysis & Insights from Training Gopher.ArXiv, abs/2112.11446.

[18]Radford, A., & Narasimhan, K. (2018). Improving Language Understanding by Generative Pre-Training.

[19]Radford, A., Wu, J., Child, R., et al. (2019). Language Models are Unsupervised Multitask Learners.

[20]Rozière, B., Gehring, J., Gloeckle, et al. (2023). Code Llama: Open Foundation Models for Code.ArXiv, abs/2308.12950.

[21]Shazeer, N.M. (2020). GLU Variants Improve Transformer. ArXiv, abs/2002.05202.

[22]Su, J., Lu, Y., Pan, S., Wen, B., & Liu, Y. (2021). RoFormer: Enhanced Transformer with Rotary Position Embedding. ArXiv, abs/2104.09864.

[23]Sun, Y., Wang, S., Feng, S., et al. (2021). ERNIE 3.0: Large-scale Knowledge Enhanced Pre-training for Language Understanding and Generation.ArXiv, abs/2107.02137.

[24]Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., Rodriguez, A., Joulin, A., Grave, E., & Lample, G. (2023). LLaMA: Open and Efficient Foundation Language Models. ArXiv, abs/2302.13971.

[25]Vaswani, A., Shazeer, N.M., et al. (2017). Attention is All you Need. Neural Information Processing Systems.

[26]Wang, B. and Komatsuzaki, A. (2021). GPT-J-6B: A 6 Billion Parameter Autoregressive Language Model. Retrieved from https: //github.com/kingoflolz/mesh-transformer-jax.