Assessing Large Language Models' Proficiency in 3D Positional Modelling

Arun Patro
New York University
akp7833@nyu.edu

Vashu Raghav
New York University
vr2326@nyu.edu

Soowhan Park
New York University
sp6682@nyu.edu

Henry Ying
New York University
hy1672@nyu.edu

Abstract

In this paper, we investigate the positional awareness of LLMs and their capability to encode spatial relationships amongst objects. We conduct various experiments to gauge this ability and determine how LLMs understand concepts of 3D sense and spatial relationships amongst objects. We report that the LLM-generated programmatic solutions perform better than the LLM-generated general responses, which suggests that programming inherently entails a structured problem-solving process that is easier for the LLMs to logically reason. We show that GPT models do generate valid 3D world to some prompts.

1 Introduction

Large Language Models have seen a phenomenal interest ever since the ChatGPT moment. These models are transforming the field of NLP by providing unprecedented accuracy and efficiency in language understanding and generation. The vast applications of these language models range from language translation to text summarization, question-answering, and even creative writing. One such application that has received relatively little attention so far is 3D modeling. With the increasing demand for 3D modeling in various fields such as architecture, engineering, and entertainment, language models could be leveraged to make the 3D modeling process more efficient and accessible. This paper explores the potential of language models in 3D modeling and their possible applications. To do this, we subject the LLMs to understand prompts about spatial positioning concepts.

2 Literature Survey

Our literature survey highlights the capabilities and limitations of existing LLMs in solving mathematical problems, performing reasoning tasks, and 3D modeling using natural language. While Ernest Davis (Davis, 2023) pointed out the need for further research to improve the mathematical reasoning capabilities of LLMs, Swaroop Mishra (Mishra et al., 2023) proposed a unified benchmark called LILA for evaluating the mathematical reasoning abilities of LLMs. They fine-tuned GPT-Neo which outperformed other SoTA models, indicating the potential of LLMs for mathematical reasoning tasks. Furthermore, Yan Ding (Ding et al., 2023) proposed a task and motion planning framework using LLMs for object rearrangement in a simulated environment, while Jacob Austin (Austin et al., 2021) suggested a program synthesis method allowing the model to generate code from natural language input, which could be applied to 3D modeling tasks. Although the literature suggests that LLMs show promise in these areas, further research is required to improve their capabilities and explore their potential for real-world applications.

3 Approach

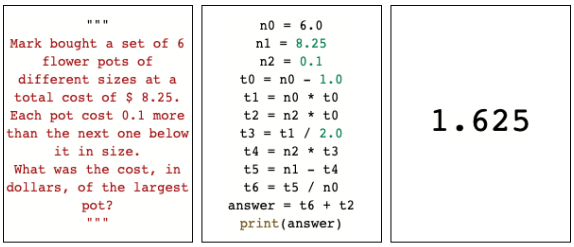
To investigate the capacity of LLMs to accomplish positional modeling in a 3D environment, we conducted three experiments.

Experiment 1 (Common Maths): We prompt LLMs to solve elementary mathematical problems provided from the LILA benchmark dataset across 5 categories.

Experiment 2 (Positional Understanding): We prompt LLMs to simulate games requiring positional comprehension of the game world, like Tic Tac Toe, Sudoku, and Chess.

Experiment 3 (Spatial Understanding): Lastly, drawing upon the findings from the previous experiments, we test the LLMs' ability to generate 3D python blender (3D modeling application) code to create a 3D scene satisfying the prompt.

We reason that for an LLM to understand the positional sense, it must first be able to solve simpler and more primitive mathematical problems and understand numbers. After that, it must also be able to understand trivial 2-D positional relationships in



(a) Input (b) Output Program (c) Output Answer

Figure 1: This is an example from the mathqa part of the LILA Dataset, which has 3 components.

games like Tic Tac Toe, Sudoku, and Chess. With some positive results, we finally test the models' capability on generating blender-python code for prompts to model the 3D world.

3.1 Datasets

For the **Common Maths** test, we sample points from every mathqa dataset in the LILA Benchmark (Mishra et al., 2023). These are word problems, for which we have a definite answer and a Python program to solve them, see Fig 1. For the **Spatial Understanding** test we download 11 free open source .obj files from the internet. These are common objects found in a home, see Fig 3a.

3.2 Models and API

We use many LLMs - GPT-3.5-Turbo and GPT-4 (from OpenAI[1]), Command-Nightly (from Cohere[2]), Bard (from Google), Bing Chat (from Microsoft), HuggingChat. Bing and Bard do not have API access, and we only tested them manually. We do not show comparisons with HuggingChat because of very poor results like token limitations, abrupt terminations, and often unparseable output with regex filters.

4 Common Math Problems

4.1 Experiment set up

In the first experiment, we sample 500 samples across five subcategories: geometry, physics, probability, etc.

The structure of the prompt captures a role of a mathematician, followed by the specific output format and the question. Two response types were generated for each question: one for general solutions and another for programming-based solutions. For the latter, the instruction was: "You are a mathematician. Your answer should be a Python program to solve the question. Please don't explain anything, I just want the runnable Python program with no explanation".

The generated responses and code were compared programmatically[3] to the ground truth solutions corresponding to the respective problems. Finally, the solution rate was computed to assess the mathematical ability of the models.

4.2 Results

See Figure 2 for the results and Table 1 for the average. The Command-Nightly model showed the weakest performance, followed by GPT 3.5 Turbo, with GPT-4 demonstrating the most robust results. Furthermore, the average accuracy of programmatic solutions generated by the LLMs surpassed that of the general LLM-generated responses. The Command-Nightly model's performance was particularly suffered for the program-generated solutions, as the model frequently failed to produce functional Python code for many data points.

4.3 Observation

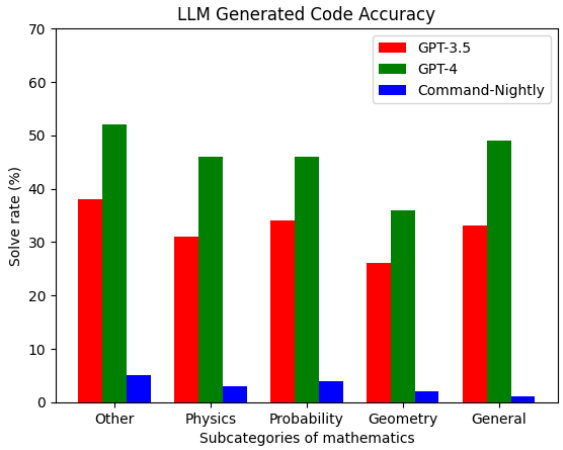
Programmatic solutions generated by the LLMs demonstrated better performance compared to the LLM-generated general responses. Programming inherently entails a structured problem-solving process underpinned by logical reasoning. Consequently, the language model is more inclined to deconstruct verbose questions into smaller, more manageable components, thereby enhancing the probability of obtaining a correct solution. Additionally, the model's efforts to conform to the syntactic rules and conventions of the programming language impose specific constraints, which may reinforce the likelihood of producing precise answers.

Conversely, the general responses are likely generated by a less structured approach and mostly entail natural language processing tasks, governed by the conditional probability(Davis, 2023) of producing w_n following $w_n \cdots w_{n-1}, P(\omega_n \mid w_1 \cdots w_{n-1})$. As a result, verbose questions tend to produce greater ambiguity or foster multiple interpretations, which subsequently hinders the models' capacity to make precise solutions.

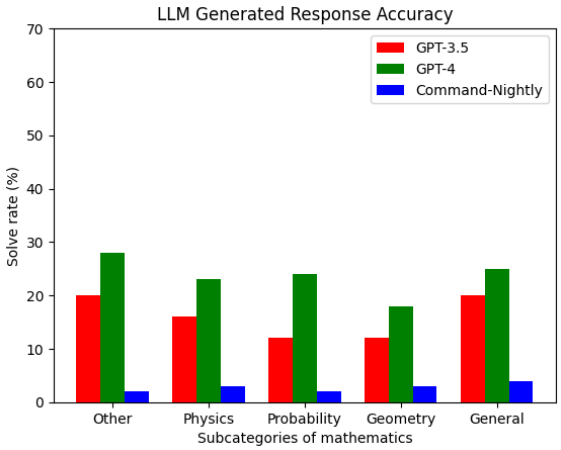
[1] https://platform.openai.com/docs/introduction/overview

[2] https://docs.cohere.com/docs/command-beta

[3] https://github.com/arunpatro/nlp_final_proj



(a) Accuracy when LLM Generates code to solve the problem



(b) Accuracy when LLM directly generates the answer

Figure 2: Graphs for a solve rate percentage of LLMs getting correct answers for 100 questions on each topic. We clearly observe that expressing answers as code significantly improves mathematical reasoning ability.

Language Model	Avg ACC. response	Avg ACC. code
GPT-3.5 Turbo	16%	32.4%
GPT-4	23.6%	45.8%
Command-Nightly	2.8%	3.0%

Table 1: The average accuracy of each language model's generated response and code

5 Positional Game Analysis

5.1 Experiment set up

For this experiment, we had four instruction-tuned LLMs play three games (Tic-Tac-Toe, Sudoku, and Chess) that require simple 2-D positional understanding. This would be key to solving Positional Composition in 3D. We test Bing Chat, ChatGPT 3.5, ChatGPT 4, and HuggingChat.

5.2 Results

See Table 2 for results. Tic-Tac-Toe was measured by the percentage of valid moves played. Sudoku was measured by the percentage of cells filled in that was valid. Chess was measured by the number of valid moves played in a row before failing. Figure 2 shows the best-performing LLM at each game.

Even though these games are not a perfect representation of positional understanding, it is still a good sign if they are able to be played properly. Whichever method the LLM uses to achieve positional understanding in the games, they can use the same for Positional Composition in 3D. In other words, the logic is that achieving positional understanding in games is a very good sign that it can do the same in other contexts.

5.3 Observation

ChatGPT 4 is the most promising out of the four in achieving positional understanding with the best performance in two games and the second-best in the other game. On the other hand, HuggingChat struggled to even get one valid test case.

We also found through this experiment that LLMs perform better across the board if it generates a program to make the decisions for them. This phenomenon is consistent with the first experiment of Common Math Problems, and we will take advantage of it in the next experiment.

6 Positional Composition in 3D

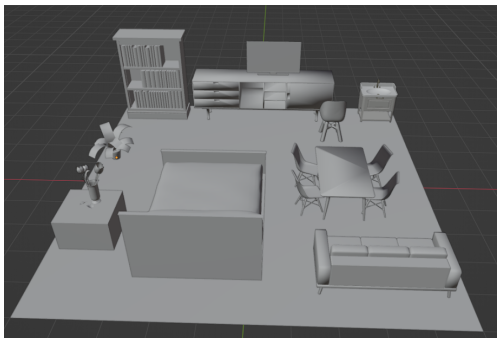
6.1 Experiment set up

To test the model's 3D understanding, we evaluated the LLMs qualitatively based on how true the scene is based on the prompt. We ignore minor errors and offsets in positions. Our initial scene and library consist of 11 objects of a house. We place them in a room of dimensions 10m x 10m. We test 3 things to understand that the LLM indeed understands absolute and relative concepts like position and scale. We qualitatively assess the output produced by LLM-generated Python Blender Code.

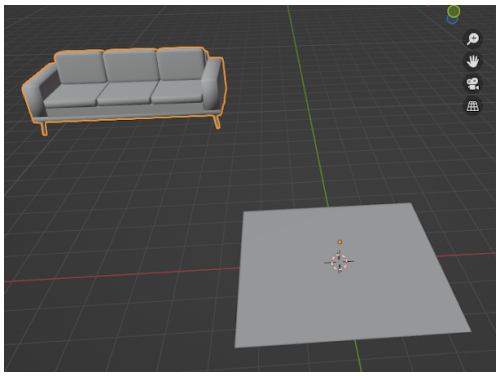
First, we check if the LLMs are capable of im-

Language Model	Tic-Tac-Toe	Sudoku	Chess
Bing Chat	90%	N/A	N/A
GPT-3.5	30%	77%	3-4
GPT-4	60%	83%	9-10
Hugging Chat	0%	11%	0

Table 2: The average accuracy of each language model's ability to play the games.

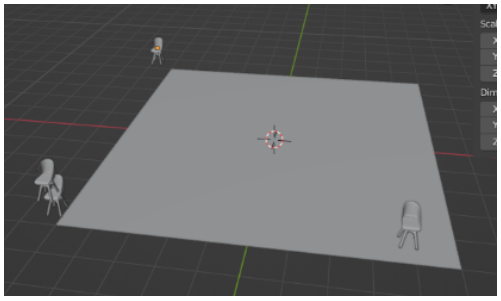


(a) We created a library of only 11 objects to be manipulated. By default all the objects are to scale, and approximately rightly placed.

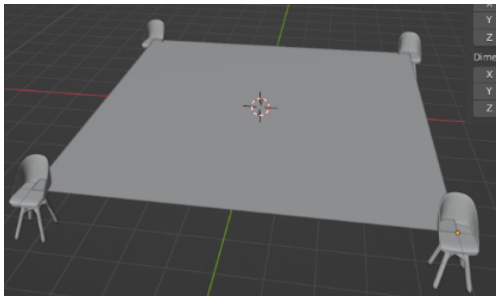


(b) For task 1, GPT-3.5 doesn't get the z-coordinate correct and sometimes objects are out of bounds.

Figure 3: Position Testing: For the prompt "Place chairs at each corner of the room", we see that GPT-4 correctly places the objects quite exactly but GPT-3.5 fails.

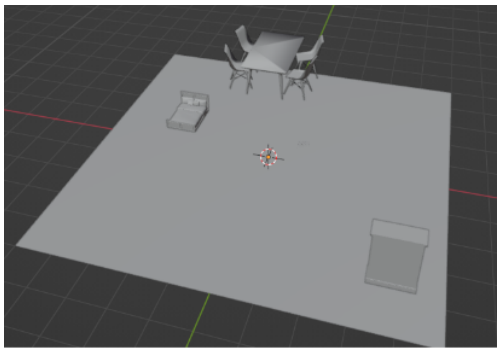


(a) GPT-3.5 places two chairs together, and the chairs are not as correctly placed as GPT-4

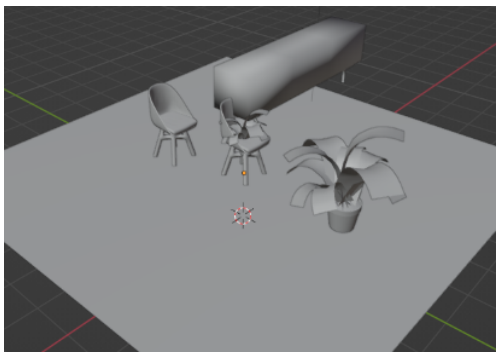


(b) GPT-4 understands the spatial relationship of the chairs here

Figure 4: Position Testing: For the prompt "Place chairs at each corner of the room", we see that GPT-4 correctly places the objects quite exactly but GPT-3.5 fails.



(a) Prompt: "A bed, a bookshelf and a dining table. Bookshelf must be half the size of the dining table. Bed should be half the size of the bookshelf."



(b) Prompt: "A table, two chair and two plants. The plants should be twice as large as the chairs."

Figure 5: Scale Testing: We see that for Fig. 5a, objects have not been scaled well, but for Fig.5b plants are indeed larger though not twice as larger.

LLM	Create	Position	Scale
GPT 3.5 Turbo	8/10	2/10	2/10
GPT-4	10/10	8/10	5/10
Bard	4/10	2/10	0/10

Table 3: Zero Shot ability of the LLMs

LLM	Create	Position	Scale
GPT 3.5 Turbo	10/10	5/10	2/10
GPT-4	10/10	7/10	6/10
Bard	6/10	4/10	3/10

Table 4: One Shot ability of the LLMs

porting model files and placing them correctly in the scene. We play with prompts involving multiple objects, like "a bed and a chair". Second, we test if the models understand absolute positioning through prompts like "A bed and two chairs on either side". Third, we test if the model understands the scale of objects so that it can relate objects and place them appropriately without violating geometry. We use prompts like "A dining table that is as large as the room".

In general, since these models may not be trained on Python Blender code and our task, we try to prompt-engineer a system prompt that finally gives decent results on top of which we build our Inputs. It is to be noted that, we experiment with various guardrails to constrain the generation.

6.2 Results

We see the results in Table 3 and Table 4. We also show some results qualitatively for all 3 kinds of tests in Figure 3b.

6.3 Observation

3D and Positional understanding using Python Blender Code is an elegant way to assess the spatial relationship of objects. The generated code assigns the location, scale, and orientation of the objects. Because of inconsistencies like default object origins, sometimes the LLMs are not able to place the objects on the floor exactly and sometimes out of bounds of the room, see Fig 3b.

For the second task, GPT-4 performs much better than GPT-3.5 on zero-shot setting and understands positions and numbers better, see Fig. 4

For the third task, models do not really understand scale but we observe that they are not fully correct, and sometimes invert scale relationships, see Fig 5.

We also clearly observe that 1-shot prompting with an example helps boost all three models. This is because it gets a sense of blender code syntax. We believe with few-shot prompting, the model can further improve on more nuanced descriptions.

7 Conclusion

In this paper, the development of large language models (LLMs) has shown great potential in various applications, including mathematical problem-solving, natural language processing, and 3D modeling. However, despite their remarkable performance in some areas, LLMs have certain limitations in others, which can hinder their effective use in real-world scenarios. As discussed, the limited positional understanding and mathematical reasoning capabilities of LLMs can impact their ability to accurately represent and manipulate 3D models. Therefore, further research is needed to address these limitations and improve the overall functionality of LLMs in 3D modeling applications.

References

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, and Charles Sutton. 2021. Program synthesis with large language models. arXiv:2108.07732.

Ernest Davis. 2023. Mathematics, word problems, common sense, and artificial intelligence. arXiv:2301.09723. Version 1.

Yan Ding, Xiaohan Zhang, Chris Paxton, and Shiqi Zhang. 2023. Task and motion planning with large language models for object rearrangement. arXiv:2303.06247. Version 2.

Swaroop Mishra, Matthew Finlayson, Pan Lu, Leonard Tang, Sean Welleck, Chitta Baral, Tanmay Rajpurohit, Oyvind Tafjord, Ashish Sabharwal, Peter Clark, and Ashwin Kalyan. 2023. Lila: A unified benchmark for mathematical reasoning. arXiv:2210.17517. Version 2.