HIGH SCHOOL EDITION
Journal of Student Research
Volume 13 Issue 2 (2024)

LLMs and Spatial Reasoning: Assessing Roadblocks and Providing Pathways for Improvement

William Peng[1] and Sam Powers[#]

[1]Erindale Secondary School
[#]Advisor

ABSTRACT

In this paper, we show how relative location prompting can improve how Large Language Models (LLMs) such as ChatGPT interact with Spatial Reasoning (SR) tasks. LLMs likely have difficulty with SR tasks due to being designed for language based tasks, whilst SR tasks are more visual (Lee, 2023). By reading papers on Self-Ask (Press et al., 2023) and Chain-of-thought (Jason Zhanshun Wei et al., 2022) and recognizing their success, we hypothesized that prompting techniques similar to the ones mentioned would be effective in increasing the success-rate of the LLM agent in our Spatial Reasoning task. Taking these two factors into account, the solution we came up with was to turn the multi-step interaction-based SR tasks into more simple tasks by prompting the AI agent with its relative location to the target location after each step taken. We set up a 2D 5x5 grid world environment to test the LLM agent against, and by setting up a separate environment which includes relative location prompting, as well as a random environment, we saw the difference between the three success-rates. We collected and analysed data of 300 trials total (100 trials on each of the 3 environments) to conclude that relative location prompting does improve the success rate of LLMs when tackling SR tasks. This showed that by converting SR tasks into text, and by breaking down large tasks into smaller tasks, AI can solve SR problems better. Future studies should investigate other types of SR tasks, such as folding scenarios, and test out different prompting methods to determine the best one.

Introduction

Large Language Models (LLMs) are deep learning Artificial Intelligence (AI) models, which are particularly strong at analysing text prompts and generating human-like answers. LLMs are now widely used across the internet, a highly popularized version being GPT-3 by OpenAI. Even search engines such as Bing and Google have evolved to introduce AI Chatbots to their platforms (e.g., Bing AI, Google Bard). Despite such widespread use, AI is nowhere near perfect. In particular, AI struggles with Spatial Reasoning (SR) tasks. SR tasks are situations which require models to understand, interpret, and manipulate objects within a 2D or 3D plane. This would mean that the agent must understand the task, plan a course of action, and take the action towards the goal. An example of a SR task would be navigating a 2d plane with obstacles to avoid and a goal to reach.

Why is it important for AI models to have strong spatial reasoning capability? In the age where people are becoming more and more dependent on Artificial Intelligence algorithms to make important decisions, seen in Tesla's Autopiloting Cars, it becomes increasingly important that AI models are able to navigate and manipulate spaces in the real world and virtual world properly.

While there have been attempts to improve Spatial Reasoning for AI, such as Google's CAPTCHA (Completely Automated Public Turing test to tell Computers and Humans Apart) program, which uses millions of users' input to create training datasets for AI, success at scale has been limited. This can be seen when using GPT-3.5, where it is unable to solve a simple SR task without cheating (See Appendix. A for full transcript).

A possible explanation for this bottlenecked process could be that LLMs are designed to specialize in textual understanding and generation (Lee, 2023). Hence, they were undertrained for achieving optimal performance in SR tasks, which are visually oriented.

In this work, we use this explanation as the basis of our solution. By translating the SR tasks from visual into text-based prompts and breaking down the task into step-by-step processes, the AI may have a more optimal performance in SR tasks. We hypothesize that by using self-prompting strategies similar to Self-Ask or Chain-of-thought, the AI may perform better with the SR tasks we give them.

Our hypothesis is inspired by how the Self-Ask and Chain-of-thought prompting techniques have worked for improving other aspects of LLM's problem solving abilities. However, the prompting we use is different from how Self-Ask and Chain-of-thought operate. It is similar in that all of the techniques are prompting themselves with known information, which lets the LLM process the information differently. However, each of the techniques prompt themselves differently. Chain-of-thought works by restating information given, then answering the question, which leads into the next part of the problem, and then it repeats the cycle (Jason Zhanshun Wei et al., 2022). Self-Ask, on the other hand, works by determining if it needs follow-up questions, then asks itself questions in small chunks that lead into the answer (Press et al., 2023).

As for past research, although several research papers have investigated Computer Vision, it is less common to find previous studies about AI and Spatial Reasoning. The few studies on this topic explored Spatial Reasoning and AI within specific fields, such as Neural Networks and Models (Kim et al., 2021) or Self-driving Cars (Kikot, 2023). Even these papers specifically called for more research and testing on the reasoning side of AI (Kim et al., 2021). Other than a few of these papers, most of the papers that are somewhat related have to do with Computer Vision (Dilek et al., 2023), which is similar to Spatial Reasoning, though with a key distinction: Computer Vision is identifying objects in a 3D space, whilst Spatial Reasoning is what we can do with that information. This underscores the scarcity of knowledge at this intersection.

Unfortunately, creating a general solution to solve all Spatial Reasoning problems is out of scope for this work. Nevertheless, this work aims to assess roadblocks and provide pathways to improve AI's SR task problem-solving abilities. To test this, multi-step interaction-based tasks were used to challenge the AI's abilities. In these instances, the model is assigned a SR task, has to understand the assigned task, and is then required to pre-plan and implement multiple steps to successfully complete the task. Multi-step interaction-based tasks were used because they require the AI to be consistent, and methodical when taking step by step action to complete a task. In doing so, we aim to provide a benchmark against which future SR task performance can be compared as well as identify a framework for possible ways to expand its problem-solving abilities.

Methodology

An AI model that is able to understand its environment and make decisions towards its goal will be referred to as an 'agent', or interchangeably, 'LLM agent'. An example of an intelligent agent in AI would be Apple's Siri. Our prompting technique works by prompting the agent with its relative location to the target location, then the agent will use this information before proceeding. By prompting the agent with this information, it makes the agent have to decide on a path before taking it, ensures that the agent thinks about the problem at hand more thoroughly, and breaks the task down to more simple tasks.

First, we assessed the feasibility of addressing different issues in this LLM/SR pipeline and thereby identified those which we could sufficiently address. We ultimately selected SR because of how big of an issue it remains, as well as being easily accessible and replicable. Then, we identified the multi-step-interaction-based tasks, defined as tasks that the LLM could complete quickly, but with complex elements, such as obstacles to push the agent to have to reason more. After considering all of these synthesized SR tasks, the one we ultimately decided on to work with was navigating to the target location while avoiding obstacles in a 2D 5x5 grid world. This test balanced feasibility with challenge. Not only was it possible for us to test and complete this simple task within the temporal constraints of our experiment, but it also included obstacles, defined as elements that could increase the difficulty of the task for the base agent. We then picked OpenAI's GPT-3.5 to use as our agent's LLM due to its widespread usage and suitability for generalization.

Proposed Benchmark

In this work, we propose a lightweight grid-world environment for efficiently analyzing the effectiveness of new Spatial Reasoning agents; we refer to this new environment as Spatial MiniGrid (SMG). Additionally, SMG is very fast to run, so algorithm designers may iterate on it more quickly.

Proceeding with the idea, we created a scenario with a 5x5 grid, which allowed the agent to go from its initial location to its target location (Figure 1). Subsequently, we created grid_world_env, a 5x5 grid world with an agent, 3 obstacles, and 1 target location, all randomized locations on the grid, using Python scripts. We created a move_tool that let the LLM pick which direction to move in by coordinating 0, 1, 2, 3 to right, down, left, and up respectively. To reduce errors, we added "*np.clip*" to prevent the agent from leaving the grid world, and made sure that the generated target locations and obstacles were not colliding with the agent's starting point. Furthermore, we modified the environment by changing the prompt to be more descriptive of the task and adding max_steps, which set an upper bound and therefore prevented the test from running indefinitely. The max_steps set for our experiments was 10. We chose this number because it is the least number of steps needed to get from one corner of the grid to the other corner, while also reserving enough steps for error. This also allowed us to give the agent a reasonable amount of time to complete the task.

We first tested the agent on our base grid_world_env. We will refer to this as "Base Environment" (Table 1). This environment takes place in a 5x5 grid, where the target is on a random location, and the obstacles and target location are also randomized. A trial is considered a success if the agent can reach the target location within 10 steps (Figure 2). The alternative is that the agent gets stuck on an obstacle (Figure 3). Since we found it difficult to run the tests quickly, we added a loop to run 10 trials per test. Additionally, we added simple statistical calculations such as: trial number, success or fail, and mean of the trials succession in the code to make it even easier to gather data. Next, we added relative location prompts, which marks the second version of the grid_world_env. We will refer to this as "Relative Location Environment" (Table 1). We added relative location prompts to our observations to improve how the agent interacts with the grid_world_env. These prompts were simple observations such as "target is to your [left or right], [up or down]". It should be noted that in the case that the target is at the same x or y coordinate that the corresponding directive will not be included (e.g. agent is at (0,0), target is at (4,0). The prompt will be "target is to your right"). Finally, we added a random version of the agent, where the agent moves in a random direction for each step. This tested against both prior versions of the environment and agent. We will refer to this as "Random Environment" (Table 1).

If our hypothesis that using relative location prompts will improve the success rate of the LLM is true, then we should expect to see it doing better than both the Random Environment and Base Environment. During the testing phase, we ran each of the baselines 100 times to reduce random errors, which could result from small samples.

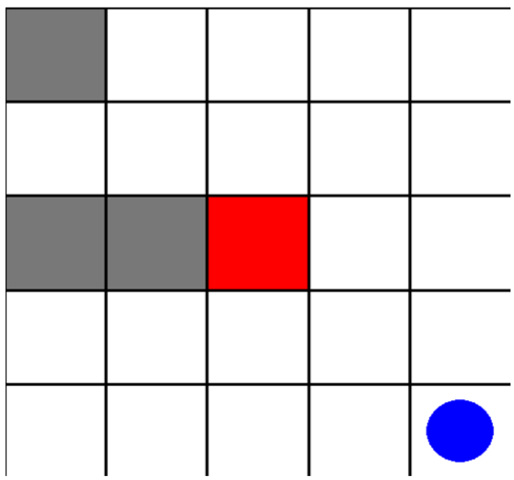

Figure 1. A screenshot of the PyGame visual of the test.

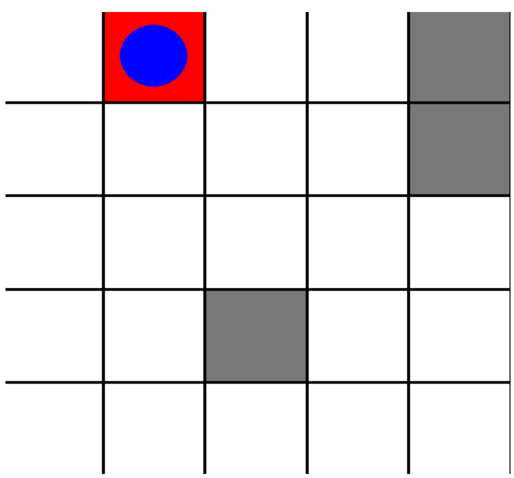

Figure 2. A screenshot of the success conditions being met.

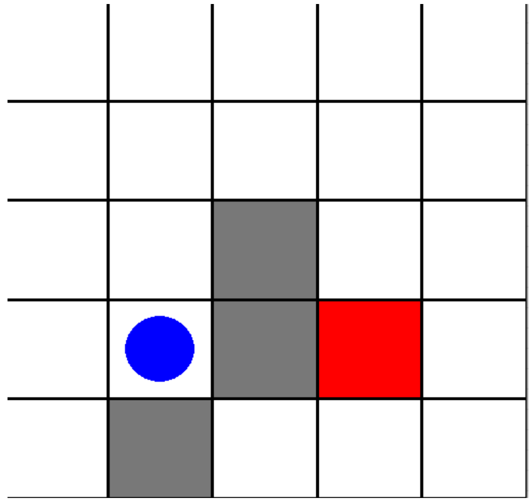

Figure 3. A screenshot showcasing the agent stuck on an obstacle.

Baselines/Environments

To evaluate the efficacy of our benchmark, we utilized 3 different baselines which are articulated in Table 1.

Table 1. A table showcasing and describing each baseline/environment used.

Baseline/Environment	Description
Base	This baseline is the LLM agent (GPT-3.5) at its most standard, unmanipulated form.
Random	This baseline is a non LLM agent that will move at random.
Simple Relative Location	This baseline is an LLM agent (GPT-3.5) that prompts itself with simple relative location information, such as "The target is to your right, down."

It should be noted that each of these baselines were run on the same 5x5 grid and were assigned the same tasks. These environments served as our independent variables, of which we counted the number of successes and failures up to 100.

Experimental Results

The Base and Random baselines had comparatively similar success rates, at 15/100 and 19/100 respectively, whereas the Relative Location baseline had a much higher success rate at 68/100 (Figure 4).

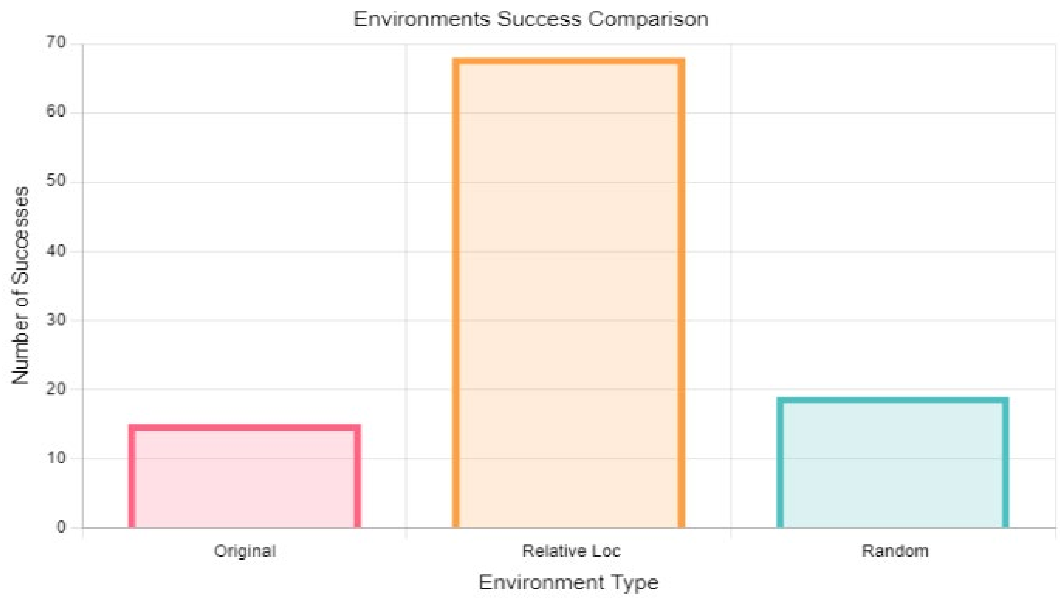

Figure 4. A Bar Graph showcasing the Environment Success Comparison.

Discussion

After looking at the results, it is shown that the Base Environment without relative location prompting failed in reaching the target location consistently, with a 15% success rate. In contrast, the Relative Location Environment, using relative location to help guide the agent, was shown to reach the target location more consistently, with a 68% success rate. Surprisingly, using the Random Environment, the success rate was higher than the Base Environment, with a 19% success rate. The minimal difference in success rate between the random and agent baselines illustrates the low performance of LLMs in handling simplistic Spatial Reasoning tasks.

Seeing how our Relative Location Environment has a success rate of more than 4x that of the Base Environment, and more than 3x that of the Random Environment, our hypothesis that using relative location prompts would improve the success rate of the agent reaching the target location within 10 steps is true (Figure 4). Multiple contributors could explain the lack of improvement from random to agent. Rigorous testing was applied to both environments to minimize random errors (errors which are uncontrollable).

During tests on the Relative Location Environment, a systematic error (errors that are the result of poor experimental setup) occurred, where the agent would often get stuck on an obstacle if it was directly between the agent and target location (Figure 3). To mitigate this in future studies, we could add units into the observation (e.g. "The target location is 3 units right, 2 units down"), or even the agent's relative location to obstacles (e.g. "There is an obstacle 1 unit right"). A similar issue was that the target or agent location could be trapped within 3 obstacles, thus resulting in a guaranteed failure no matter how good the agent and thereby obscuring any differentials in performance metrics. To resolve this issue, a parameter must be placed where there must be a possible solution to the problem for the task to be created.

Conclusion

Despite the complexity and usefulness of Large Language Models, there are still many things which it struggles with. Spatial Reasoning is only one of the many. Through the research and experimentation done, we have proven our hypothesis that self-prompting techniques similar to those of Self-Ask and Chain-of-thought are effective in improving LLM's ability to solve Spatial Reasoning problems. We used relative location prompting to achieve over 4x more efficiency in the LLM agent's problem solving ability. Unfortunately, computing resources limited this study to 100 trials, though future studies could run additional trials to enhance accuracy. While outside of the scope of this study, future researchers could focus on exploring other prompting methods, other environments, and test different AI models as agents as well. Our study serves as an entrance point to much more that could be done in this field of study. It is clear that there is a big issue in the effectiveness of LLMs when tackling Spatial Reasoning tasks. We have introduced a benchmark for future testing and research. Future researchers should consider using similar or different prompting methods for the AI Language Model when testing, as it could possibly yield greater results.

Acknowledgments

I would like to thank my advisor for the valuable insight provided to me on this topic.

References

Dilek, E., & Dener, M. (2023). Computer Vision Applications in Intelligent Transportation Systems: A Survey. *Sensors (Basel, Switzerland)*, *23*(6), 2938. https://doi.org/10.3390/s23062938

Jason Zhanshun Wei, Wang, X., Schuurmans, D., Bosma, M., Chi, E. H., Le, Q. V., & Zhou, D. (2022). *Chain-of-Thought Prompting Elicits Reasoning in Large Language Models*. https://doi.org/10.48550/arxiv.2201.11903

Kikot, S. (2023). Spatial Intelligence of a Self-driving Car and Rule-Based Decision Making. *ArXiv (Cornell University)*. https://doi.org/10.48550/arxiv.2308.01085

Kim, H., Koh, Y., Baek, J., & Kang, J. (2021). Exploring the spatial reasoning ability of neural models in human IQ tests. *Neural Networks*. https://doi.org/10.1016/j.neunet.2021.02.018

Lee, A. (2023, January 26). *What Are Large Language Models and Why Are They Important?* NVIDIA Blog. https://blogs.nvidia.com/blog/what-are-large-language-models-used for/#:~:text=A%20large%20language%20model%2C%20or

OpenAI. (2024). *GPT-3.5 [Large Language Model]*. Chat.openai.com; OpenAI. https://chat.openai.com/chat

Press, O., Zhang, M., Min, S., Schmidt, L., Smith, N. A., & Lewis, M. (2023, May 22). *Measuring and Narrowing the Compositionality Gap in Language Models*. ArXiv.org. https://doi.org/10.48550/arXiv.2210.03350

reCAPTCHA: Easy on Humans, Hard on Bots. (n.d.). Www.google.com. Retrieved February 28, 2024, from https://www.google.com/recaptcha/intro/?zbcode=inc5000#:~:text=reCAPTCHA%20makes%20positive%20use%20of

Shridhar, M., Manuelli, L., & Fox, D. (2022, November 11). *Perceiver-Actor: A Multi-Task Transformer for Robotic Manipulation*. ArXiv.org. https://doi.org/10.48550/arXiv.2209.05451