MANGO: A Benchmark for Evaluating Mapping and Navigation Abilities of Large Language Models

Peng Ding[1]🥭 **Jiading Fang**[2]🥭 **Peng Li**[2]🥭 **Kangrui Wang**[2]🥭 **Xiaochen Zhou**[3,5]🥭
Mo Yu[4] **Jing Li**[5] **Matthew R. Walter**[2] **Hongyuan Mei**[2]
[1]University of Chicago [2]Toyota Technological Institute at Chicago [3]Syracuse University
[4]WeChat AI [5]New Jersey Institute of Technology
`{kangrui,mwalter,hongyuan}@ttic.edu`

Abstract

Large language models such as ChatGPT and GPT-4 have recently achieved astonishing performance on a variety of natural language processing tasks. In this paper, we propose MANGO, a benchmark to evaluate their ability to perform text-based mapping and navigation. Our benchmark includes 53 mazes taken from a suite of textgames: each maze is paired with a walkthrough that visits every location but does *not* cover all possible paths. The task is question-answering: for each maze, a large language model reads the walkthrough and answers hundreds of mapping and navigation questions such as "How should you go to `Attic` from `West of House`?" and "Where are we if we go `north` and `east` from `Cellar`?". Although these questions are easy for humans, it turns out that even GPT-4, the best-to-date language model, performs poorly when answering them. Further, our experiments suggest that a strong mapping and navigation ability would benefit the performance of large language models on relevant downstream tasks, such as playing textgames. Our MANGO benchmark will facilitate future research on methods that improve the mapping and navigation capabilities of LLMs. We host our leaderboard, data, code, and evaluation program at `https://mango.ttic.edu` and `https://github.com/oaklight/mango/`.

1 Introduction

Mapping and navigation are fundamental abilities of human intelligence (Spiers & Maguire, 2006; Epstein et al., 2017). Humans are able to construct maps—in their minds (Epstein et al., 2017) or on physical media like paper—as they explore unknown environments. Following these maps, humans can navigate through complex environments (Spiers & Maguire, 2006; Spiers & Gilbert, 2015; Javadi et al., 2017), making informed decisions, and interact with their surroundings. Such abilities empower humans to explore, adapt, and thrive in diverse environments. An example is remote (e.g., deep-sea) exploration for which humans have drawn upon their intuition to develop algorithms that enable robots to autonomously navigate and map their surroundings based only on onboard sensing.

Do large language models (LLMs) possess such abilities? In this paper, we investigate this research question by creating a benchmark and evaluating several widely used LLMs. Our MANGO benchmark is the *first* to measure the mapping and navigation abilities of LLMs. It includes 53 complex mazes, such as the one visualized in Figure 1. It pairs each maze with hundreds of destination-finding questions (e.g., "Where will you be if you go `north`, `north`, and then `up` from `Altar`?") and route-finding questions (e.g., "How do you reach `Dome Room` from `Altar`?"). For each maze, the language model has to answer these questions after reading a walkthrough of the maze. Many questions involve possible routes that are not traced during the walkthrough, making the benchmark challenging. In our experiments, GPT-4 only correctly answered half of the route-finding questions, performing disastrously

🥭Equal contribution. Work done while visiting TTIC.

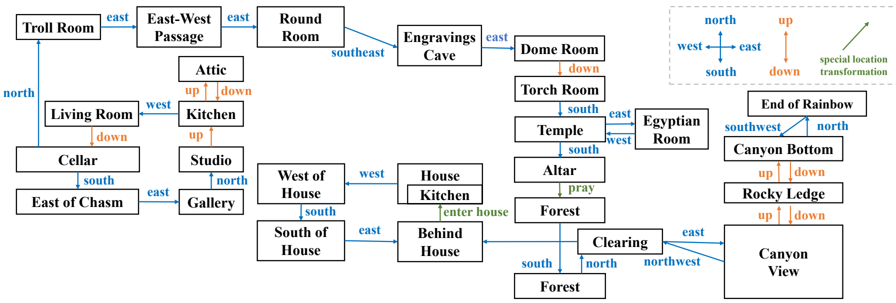


Figure 1: Map of Zork-I. Arrows denote the direction of travel during the walkthrough, while the reverse direction is unseen but may be possible. Note that it is a 3D map projected onto a 2D plane so `up` may not point upward in the 2D visualization (e.g., `Rocky Ledge` to `Canyon View`).

on the difficult questions (e.g., those involving long and unseen routes). MANGO will facilitate future research in improving the mapping and navigation abilities of LLMs.

Another contribution of MANGO is to draw a novel connection between natural language processing and robotics. There has been significant interest in employing LLMs to endow intelligent agents (including robots) with complex reasoning (Yang et al., 2023). Aligning with this interest, MANGO enables the investigation of the LLMs' capabilities in simultaneous localization and mapping (SLAM) within text-based worlds. Focusing on this aspect, our work stands out and complements previous SLAM-related research, which predominantly relies on richer sensory inputs (e.g., vision and LiDAR).

2 MANGO: A Benchmark for Text-Based Mapping and Navigation

Our MANGO benchmark measures the mapping and navigation capabilities of LLMs. It leverages a suite of text-based adventure games that offer expert-designed complex environments but only require simple actions. Figure 1 is an example: it was drawn according to the first 70 steps of the walkthrough of Zork-I, which can be found in Example 1. This map is imperfect: the annotator had to draw the only `Kitchen` twice to avoid a cluttered visualization; the `Living Room` was incorrectly placed outside the `House`. However, equipped with this map, one could correctly answer questions about any route in the maze such as "How do you reach `Dome Room` from `Altar`?" and "Where will you be if you go `north`, `north`, and `up` from `Altar`?". The walkthrough has not traced a route from `Altar` to `Dome Room`, but humans possess the remarkable capacity to plan a route by identifying the three individual steps—which the walkthrough has covered—from `Dome Room` to `Altar` and retracing those steps. MANGO tests whether a large language model can perform the same kind of reasoning. Particularly, when evaluating a language model, we first let it read a walkthrough like Example 1 and then ask it questions like those in Examples 2 and 3. A question like Example 2 is a *destination-finding* (DF) question, and a question like Example 3 is a *route-finding* (RF) question. Users of MANGO have the flexibility to phrase the DF and RF questions in their own ways: as shown in Examples 4 and 5, we provide the *skeletons* of these questions, which users can plug into their own templates.

2.1 Maze Collection: From Game Walkthroughs to Mazes

Our mazes are taken from the textgames in the Jericho game suite (Hausknecht et al., 2020). The main release of Jericho includes 57 popular textgames as well as a program that can generate walkthroughs for 56 of them. The original walkthrough of a game is a list of actions (such as `east`, `north`, and `open door`) that one could execute to efficiently complete the game. We enhanced each walkthrough by executing the sequence of actions and augmenting each step with the new observation (i.e., the text feedback that the game engine provides after the action is executed). Unless explicitly specified, the word "walkthrough" refers to the enhanced, but not original, walkthroughs (such as Example 1) throughout the paper. More details about walkthroughs can be found in Appendix A.1.

In a walkthrough, not every action triggers a location change: it may update the inventory (such as `take lamp` and `drop pen`) or time (such as `wait`). For each game, we read the walkthrough, labeled the actions (such as `east` and `up`) that change the locations, and made

```
STEP NUM: 0
ACT: Init
OBSERVATION: West of House
You are standing in an open field west
of a white house, with a boarded front
door. There is a small mailbox here.

STEP NUM: 1
ACT: south
OBSERVATION: South of House
You are facing the south side of a
white house.

STEP NUM: 2
ACT: east
OBSERVATION: Behind House
You are behind the white house. A path
leads into the forest to the east. In
one corner of the house there is a
small window which is slightly ajar.
⋮

STEP NUM: 70
ACT: east
OBSERVATION: Gallery
This is an art gallery. Most of the
paintings have been stolen by vandals
with exceptional taste. The vandals
left through either the north or west
exits. Fortunately, there is still one
chance for you to be a vandal, for on
the far wall is a painting of
unparalleled beauty.
```

Example 1: An example of Zork-I walkthrough.

```
Starting from Altar, perform actions [
north, north, up], where are you now?
```

Example 2: A destination-finding question.

```
How can you go from Altar to Dome Room?
```

Example 3: A route-finding question.

```
S: Altar              # starting location
A: north, north, up   # list of actions
```

Example 4: Skeleton of DF question in Example 2.

```
S: Altar              # starting location
D: Dome Roomm               # destination
```

Example 5: Skeleton of RF question in Example 3.

```
S: Altar
A: north
D: Temple

S: Temple
A: north
D: Torch Room

S: Torch Room
A: up
D: Dome Room
```

Example 6: Full route of Examples 2 and 4.

note of the names of the locations (such as `Temple` and `Altar`). This annotation is nontrivial and can not be automated. We had to pay extra attention to appropriately handle the tricky cases including: ① the name of a location may be mentioned in a rich, but distracting context (e.g., the context may have ten paragraphs and hundreds of words with the name briefly mentioned in the middle); ② a location may be visited multiple times, so we need to assign the same name to all its mentions; ③ different locations may be referred to with the same name in the textual feedback, so we need to rename them in a sensible way.

The location name resolution (see Appendix A.2 for a full procedure) results in a maze for each game. Three of the games have no location change, and so we left them out, resulting in 53 mazes. We store each maze as a directed graph: each node is a named location (e.g., `Altar`); each directed edge is a movement (e.g., `north`); and each node-edge-node combination is a location-changing step that was followed in the walkthrough. Note that a graph may be cyclic since the walkthrough may trace back-and-forth between locations (e.g., `Temple` and `Egyptian Room` in Figure 1).

2.2 Generation of Question Skeletons: Traversing Mazes and Imputing Edges

To generate DF and RF skeletons for a maze, a naive approach is to perform brute-force traversal. First, we collect all the possible S-P-D tuples, where S and D are locations and P is a simple path from S to D. A simple path is a directed path that does not visit any location more than once. This "simple" restriction ensures that we will have a finite number of S-P-D tuples. Example 6 is a simple path of 3 S-A-D edges from `Altar` to `Dome Room`. Each unique S-P-D tuple gives a unique DF skeleton: e.g., Example 4 is obtained from Example 6. Each unique S-P-D tuple gives an RF skeleton as well, such as Example 5 obtained from Example 6. However, the same RF skeleton may be obtained from other tuples since there may be multiple possible simple paths between the same pair of locations `S` and `D`. As a consequence, we may end up with fewer RF questions than DF questions for a given maze.

The particular DF and RF questions in Examples 2 and 3 are challenging to large language models, since they involve actions—such as going `north` from `Altar` to `Temple`—that are not covered in the walkthrough. Answering such hard questions requires a deeper understanding of the spatial relationships between locations. However, also because these steps are not in the walkthrough, the skeletons in Examples 4 and 5 can not be obtained through a naive traversal of the directed graph in Figure 1. That is, we have to traverse an extended graph that includes *imputed* edges. An imputed edge denotes a valid step that is not explicitly mentioned in the walkthrough, such as going `north` from `Altar` to `Temple` (i.e., `Altar-north-Temple`). Most mentioned edges involve directional moves (e.g., `up`, `east`), so reversing them is a straightforward way to impute new edges. We manually examined other edges: for some of them, we proposed intuitive reverses (such as `exit` for `enter`); for the others (e.g., `pray`), no reverse could be found. We then examined the imputed edges through real game play and discarded those failing to cause the expected location changes. Appendix A.3 documents the full procedure of edge imputation and examination.

After extending all the mazes in our benchmark, we collected 21046 DF skeletons and 14698 RF skeletons by traversing the extended graphs. Being evaluated on a maze, the LLM may not be able to consume the entire walkthrough in its context window. That is, we may only feed it an appropriate prefix of the walkthrough (e.g., the first 70 steps for Zork-I as shown in Example 1), leaving some of the DF and RF skeletons *unanswerable* given that prefix. Therefore, our benchmark provides the ANSWERABLE label (an integer) for each skeleton such that this skeleton is only answerable if the maximum STEP NUM in that prefix (e.g., 70 in Example 1) is greater than or equal to its ANSWERABLE label. Furthermore, given a walkthrough prefix, an answerable skeleton may be easy or hard, depending on whether it involves edges that are not covered in the prefix. Precisely, a DF skeleton is considered to be easy if all the `S-A-D` edges in its corresponding simple path are covered in the walkthrough prefix; an RF skeleton is easy if the shortest simple path from its starting location to its destination only involves the `S-A-D` steps covered in the prefix. When a longer walkthrough prefix is used, more answerable questions tend to become easy. Our benchmark provides the EASY label (also an integer) for each skeleton: a skeleton is easy if the maximum STEP NUM in the walkthrough prefix is no smaller than its EASY label; otherwise, it is a hard skeleton. Table 3 in Appendix A documents the statistics of the full dataset, such as the number of locations and the number of skeletons. Tables 5–8 in Appendix B shows the information about the data on which each LLM was evaluated in our experiments.

2.3 Evaluation Program

The evaluation program in our benchmark implements a range of evaluation and analysis methods. Reading the model-generated answers, it can return a set of evaluation scores together with rich analysis. In this section, we introduce the most important scores used in our main experiments. Other scores are discussed in Appendix A.6, with their related experiments presented in Appendix C.

For DF questions, the most straightforward evaluation is the success rate: i.e., the fraction of questions that the language model answers correctly. What answers will be considered to be correct? A strict criteria is that the model answer is correct if and only if it exactly matches the ground-truth location name. However, due to the variability of natural language, a correct answer may not exactly match the ground-truth. For example, the model may yield `The House` or `That House` when the ground-truth location name is just `House`. To account for such cases, we generalize the success rate to allow partial matches. Given a model answer $\hat{A}$ and the ground-truth answer A, we compute their (character-level) edit-distance d and define a correctness score $c \overset{\text{def}}{=} 1 - d/\ell$ where ℓ is the length of the longer answer. The score is $\in [0, 1]$: when the answer exactly matches the ground-truth, we have $c = 1$; if they have no character overlap at all, then $c = 0$. We then define the success rate to be the sum of the correctness scores over all the questions, divided by the number of questions.

For RF questions, the main metric is still the success rate, but the definition of "success" is different from that for DF questions. Note that an answer to an RF question is a sequence of

```
The allowed actions are: ...
The list of places are: ...
Starting from S, perform a list of
actions [A], where are you now?
Describe the trajectory in a Python
list of Python dictionaries with keys '
prev_node', 'node' and 'action'.
Start your response with '['.
```

Example 7: Our DF template.

```
The allowed actions are: ...
The list of places are: ...
How can you go from S to D?

Describe the trajectory in a Python
list of Python dictionaries with keys '
prev_node', 'node' and 'action'.
Start your response with '['.
```

Example 8: Our RF template.

moves. We consider an answer to be correct if and only if it can reach the destination after our evaluation program executes it in the maze. A correct answer to an RF question may not be a good path: it doesn't have to be the shortest; it doesn't even have to be a simple path. It is possible that an LLM-generated move is meaningful but doesn't exactly match any valid move in the graph: e.g., the LLM may give `walk south`, which means the same as `south`. Therefore, when executing a model-generated move, our evaluation program will select the closest (i.e., smallest edit-distance) valid move.

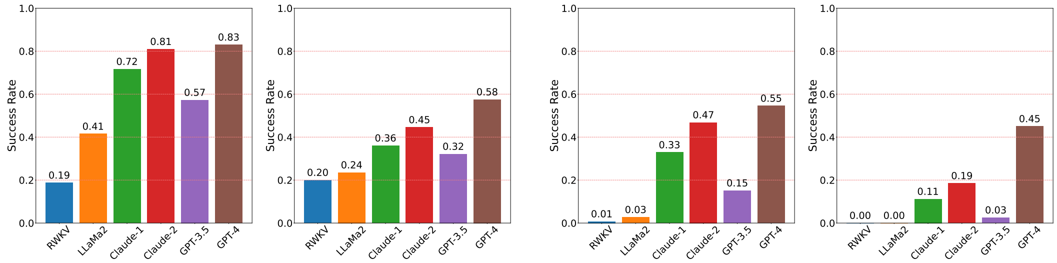
3 Experiments

In this section, we present the results our evaluation of several widely used LLMs.

3.1 Experiment Setup

The evaluated models are: GPT-3.5-turbo (Brown et al., 2020; Stiennon et al., 2020; Gao et al., 2022), GPT-4 (OpenAI, 2023), Claude-instant-1 (Anthopic, 2023a), Claude-2 (Anthopic, 2023b), Llama-2 with 13B parameters (Touvron et al., 2023b), and RWKV with 14B parameters (Peng et al., 2023). For GPTs and Claudes, we used the prompt templates in Examples 7 and 8, converting the DF and RF skeletons like Examples 4 and 5 into LLM-friendly questions like Examples 2 and 3. The templates were carefully designed and examined through pilot experiments, in order to ensure that we do not underestimate the models on our benchmark. In our templates, each question starts with a list of legal actions, followed by a list of reachable locations; these lists help mitigate the hallucination of language models. The templates ask the model to spell out the entire trajectory including all the intermediate locations. This design is inspired by Chain-of-Thought prompting (Wei et al., 2022): eliciting an LLM to give its entire reasoning process tends to improve its overall performance on downstream tasks. In addition, it allows us to conduct a deeper evaluation and analysis, such as the reasoning accuracies of the models (see Appendices A.6 and C). Note that our templates request the model to form its answer as a list of Python dictionaries with specific key names. We found that this restriction encourages the model to generate structured answers—which are easy to parse and analyze—as well as improves its performance. For Llama-2 and RWKV, we made moderate revisions to the prompts in order to generate well-structured answers as well as optimize for their performance.

For GPT-3.5, we experimented with the 4K version, which can consume 4096 tokens in its context window. This context limit restricts the length of the walkthrough that it can read, and the number of DF and RF questions that it can answer. Table 5 shows the statistics about the walkthrough prefix and questions that GPT-3.5 used for each maze. For GPT-4, Claude-1 and Claude-2, we used the same walkthrough prefixes and questions as GPT-3.5 for a fair comparison. we used the same walkthrough prefixes and questions as GPT-3.5 for a fair comparison. Llama-2 has a 4096 context window as well. But its tokenizer is different from GPTs' so we evaluated it on a slightly different set of questions. RWKV is capable of handling infinite context. For each maze, we experimented it with the 70-step prefix of the walkthrough so that its set of answerable questions includes all the questions answered by all the other models. We also evaluated Llama-2 and RWKV in a simplified setting, where the observation at each step of the walkthrough only includes the location name but nothing else. For example, at STEP 1 of the simplified Example 1, OBSERVATION only has `South of House` and everything else (i.e., `Your are...`) is omitted. We refer to Llama-2 and RWKV with the simplified walkthroughs as Llama-2-S and RWKV-S, respectively. More details about the experiment setup are in Appendix B.



(a) On easy (left) and hard (right) DF questions. (b) On easy (left) and hard (right) RF questions.

Figure 2: Success rates of the examined models on (a) DF and (b) RF questions, averaged over all 53 mazes. Appendix C provides similar graphs (e.g., Figure 6) for other evaluation metrics.

METHOD	RWKV	LLAMA-2	CLAUDE-1	CLAUDE-2	GPT-3.5	GPT-4	$\overline{\text{HARD}}$|
RWKV	*	0.20 ∣ 0.24	0.19 ∣ 0.41	0.19 ∣ 0.51	0.19 ∣ 0.33	0.19 ∣ 0.62	*
LLAMA-2	0.43 ∣ 0.20	*	0.24 ∣ 0.41	0.24 ∣ 0.45	0.24 ∣ 0.31	0.24 ∣ 0.66	*
CLAUDE-1	0.74 ∣ 0.19	0.78 ∣ 0.41	*	0.36 ∣ 0.44	0.38 ∣ 0.32	0.36 ∣ 0.57	*
CLAUDE-2	0.82 ∣ 0.19	0.85 ∣ 0.41	0.81 ∣ 0.72	*	0.44 ∣ 0.32	0.44 ∣ 0.58	*
GPT-3.5	0.59 ∣ 0.19	0.61 ∣ 0.42	0.57 ∣ 0.74	0.57 ∣ 0.83	*	0.32 ∣ 0.59	*
GPT-4	0.86 ∣ 0.19	0.90 ∣ 0.42	0.84 ∣ 0.72	0.83 ∣ 0.81	0.86 ∣ 0.57	*	*
|$\underline{\text{EASY}}$	*	*	*	*	*	*	*

(a) Pairwise comparison on easy (lower left) and hard (higher right) DF questions.

METHOD	RWKV	LLAMA-2	CLAUDE-1	CLAUDE-2	GPT-3.5	GPT-4	$\overline{\text{HARD}}$|
RWKV	*	0.00 ∣ 0.00	0.00 ∣ 0.13	0.00 ∣ 0.20	0.00 ∣ 0.03	0.00 ∣ 0.54	*
LLAMA-2	0.02 ∣ 0.02	*	0.00 ∣ 0.16	0.00 ∣ 0.21	0.00 ∣ 0.05	0.00 ∣ 0.46	*
CLAUDE-1	0.36 ∣ 0.01	0.34 ∣ 0.03	*	0.11 ∣ 0.19	0.13 ∣ 0.03	0.11 ∣ 0.45	*
CLAUDE-2	0.49 ∣ 0.01	0.46 ∣ 0.03	0.47 ∣ 0.33	*	0.20 ∣ 0.03	0.19 ∣ 0.46	*
GPT-3.5	0.16 ∣ 0.01	0.17 ∣ 0.03	0.15 ∣ 0.36	0.15 ∣ 0.50	*	0.03 ∣ 0.48	*
GPT-4	0.57 ∣ 0.01	0.56 ∣ 0.03	0.55 ∣ 0.33	0.55 ∣ 0.47	0.58 ∣ 0.15	*	*
|$\underline{\text{EASY}}$	*	*	*	*	*	*	*

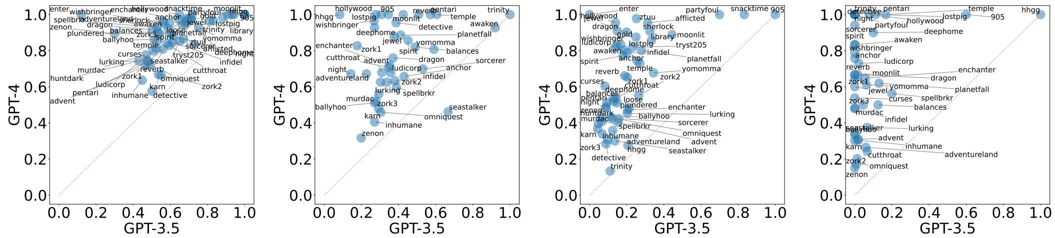
(b) Pairwise comparison on easy (lower left) and hard (higher right) RF questions.

Table 1: Success rates on DF and RF questions broken down into pairwise comparisons. In each table, the cell of row-A and col-B contains the success rates of the models—in the format of A ∣ B—on the intersection of the questions that A and B answered individually. The lower left triangle displays the results on easy questions, while the upper right triangle shows the results on hard questions.

3.2 Main Results

Figure 2 presents the success rates of all models. For each kind of question (i.e., DF or RF), we show the results on easy and hard questions separately. As we can see, GPT-4 significantly outperforms all the other models on all kinds of questions. However, it only correctly answers half of the RF questions, far worse than what a human could do: in our experiments, humans perfectly answered a randomly sampled set of questions. Note that each model was evaluated on its specific set of questions determined by the length and format of the walkthrough it read. To be fair, we also compared each pair of models on the intersection of the questions that they answered. The results are presented in Table 1: as we can see, GPT-4 and GPT-3.5 consistently outperform the other models and GPT-4 significantly outperforms GPT-3.5.

More results are in Appendix C, including results on other evaluation metrics (e.g., weighted success rates) and comparison between Llama-2 with Llama-1 and Llama-2-chat. We also explored an alternative approach that first maps a walkthrough to a symbolic graph and then uses a search algorithm to answer the given (DF or RF) question. Results and analysis of this approach can be found in Appendix C.4. Overall, we found it to be very challenging to translate natural language walkthroughs into searchable symbolic graphs in the first place, making this approach not promising.



(a) Easy DF. (b) Hard DF. (c) Easy RF. (d) Hard RF.

Figure 3: Success rates of GPT-3.5 and GPT-4 broken down into individual games. Figure 7 in Appendix B provides a similar visualization of reasoning accuracy.

3.3 Analysis of GPTs

Now, we focus our analysis on the best model, namely GPT-4. Particularly, we would like to understand the improvements of GPT-4 over GPT-3.5 as well as its current bottlenecks, shedding light on opportunities for future improvements.

By analyzing the errors of GPT-3.5 and GPT-4, we discovered that these models occasionally hallucinate nonexistent locations or edges. Once they made such a mistake at any step of reasoning, they would be misled and deviate from the correct path towards the correct answer. Furthermore, we found that the mazes are not equally difficult for the models. Figure 3 displays the success rates of the GPT models broken down into their per-game results. In Figure 3, each dot is a maze: the x-axis coefficient is the performance of GPT-3.5 on this maze while the y-axis is that of GPT-4. As we can see, the success rates of the models vary across different mazes as well as across different kinds of questions. GPT-4 consistently outperforms GPT-3.5 across nearly all the mazes. The only exception is Seastalker: there are too few hard DF questions for this maze, and thus it is a noisy outlier. Apparently, both GPTs tend to work better on easy questions than on hard questions. However, some mazes seem to be particularly challenging to GPT-4, such as Zenon and OMNIQuest.

What makes those mazes challenging? We collected some important statistics about the mazes and analyzed their correlation with the success rates of the models. To understand the success rates on the easy questions, it is interesting to investigate:

- number of locations (# locations) and number of explicit edges (# exp edges). They directly measure the size of a maze, which may be a key indicator of its difficulty.
- number of potentially confusingly named locations (# conf locations). Recall from section 2.1 that different locations may have similar or related names, which may confuse a language model. To quantify the number of confusingly named locations, we compute a confusion score for each location, and then sum the scores across all the locations. For a location name A, the confusion score is defined to be the maximum word-level edit distance between A and any other location name in the maze, divided by the maximum word-level length of the pair of location names being compared. Technically, it is $\max_B (\text{edit-distance}(A, B) / \max(\text{len}(A), \text{len}(B)))$, and it is $\in [0, 1]$.
- average length of the easy simple paths (avg len easy), i.e., the simple paths that do not include any imputed edges. A longer path may tend to be more difficult for models.
- average number of words in the scene descriptions (avg len scene). The walkthroughs exhibit very diverse styles: for some of them, the text description for each scene is very concise and the name of each location is appropriately highlighted; for others, each description may be verbose (e.g., ten paragraphs and hundreds of words) and the location names are often not obvious from the contexts. It is useful to analyze whether a long scene description poses a challenge for the models.

In order to understand the models' performance on hard questions, we analyze the effects of the variables above (except avg len easy) as well as the following:

- number of imputed edges (# imp edges);
- average length of hard—i.e., involving imputed edges—simple routes (avg len hard);
- average number of imputed edges in the hard simple routes (avg # imp in hard).

We use regression analysis to understand the effects of these variables on model performance. In particular, for each model on each type of question (DF or RF, easy or hard), we ran single-variable linear regression to understand how the success rate varies with each variable. Detailed results (e.g., coefficients and p-values) are Appendix C.3. Overall,

- on easy questions, GPTs are significantly influenced by the size of the maze, the confusion level of location name, and the path length. The p-values are extremely small.
- on easy questions, the average length of the scene descriptions does not have a significant effect on the performance of GPT-3.5, but interestingly has a significant positive effect on GPT-4's performance. It is perhaps because GPT-4 possesses a strong capability to understand texts and can leverage the rich contexts in each description. This allows it to better distinguish confusingly named locations and establish a better internal representation of the map. However, this richness seems to confuse GPT-3.5 and impede its ability to create a good internal representation of the maze, possibly due to GPT-3.5's weaker overall language understanding capabilities.
- on hard questions, the variables do not significantly affect the performance of GPT-3.5. Note that GPT-3.5 yields very low success rates when answering the hard DF and RF questions. GPT-3.5 seems to struggle when it has to reason about a path with any number of imputed edges, making the effect of other factors less important to its performance.
- on hard questions, GPT-4 exhibits a stronger ability to handle paths with imputed edges, compared to GPT-3.5. However, it will experience difficulties when the challenge of inferring imputed edges is amplified by other factors such as the size of the maze or the length of the path. As a result, nearly all the variables have significant effects on GPT-4.

The results of our regression analysis are consistent with the plots in Figure 3. For example, both Zenon and OMNIQuest stay at the lower-left corners of the hard-question plots in Figure 3 since their mazes are particularly challenging to both GPT-3.5 and GPT-4: they both have substantially larger numbers of imputed edges than the other mazes; OMNIQuest also has more locations. Wishbringer and Lost Pig have several imputed edges, but their paths are short, so they fall in the upper-left corners of the hard-question plots in Figure 3.

3.4 Human Performance

We measured human performance on a subset of our data. This subset includes 30 DF questions (21 easy and 9 hard) and 31 RF questions (20 easy, 11 hard). The student authors participated in two rounds of evaluation: in the first round, each author answered a random split of the questions, and all the questions were answered; in the second round, we randomly sampled 10 DF and 10 RF questions and let each of them be re-evaluated by a different author. The second round allows us to analyze human agreement; this analysis can be found in Appendix C.6.

Task	Difficulty	Success Rate	Reasoning Accuracy
Route Finding	All (31)	0.8211	0.6129
	Easy (20)	0.7727	0.5500
	Hard (11)	0.9091	0.7273
Destination Finding	All (30)	1.0000	0.5667
	Easy (21)	1.0000	0.6667
	Hard (9)	1.0000	0.3333

Table 2: Human Evaluation Results

As shown in Table 2, humans generally achieve high success rates on both RF and DF questions, and exhibit small to no difference across easy and hard questions. This is interesting since LLMs tend to struggle on hard questions. Reasoning accuracies are lower than success rates; this is because human answers exhibit a relatively high variability in writing (e.g., location names with special symbols).

Despite the high scores, answering the questions is actually not trivial for human raters. Answering each DF question takes an average of 15 minutes, while answering each RF question takes an average of 30 minutes. All the human raters needed to take notes (e.g., drawing maps) while completing these tasks.

```
...            # previous actions and observations
Small local map info: if you want to go to North
of House, you should go south; if you want to go
to Up a Tree, you should go up; if you want to go
 to Altar, you should go west.
Consider what you should do next, and choose one
appropriate action from the valid actions list: [
up, take on egg, put down egg, go around forest,
throw egg at tree, open egg with all, north,
south, west, east]
Please just tell me the selected action without
any extra words.
```

Example 9: A prompt of the playing game experiments

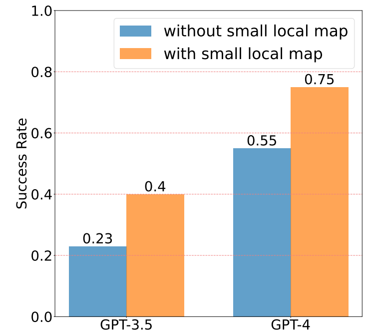


Figure 4: Playing minigames.

3.5 Does Mapping and Navigation Ability Matter in Downstream Tasks?

Now we present a case study showing that a strong mapping and navigation ability of an LLM would benefit it in downstream tasks. In particular, we selected 284 *minigames* in the Jericho game suite, and investigated how the map knowledge may improve the performance of an LLM in playing these minigames. Each minigame is a selected prefix of a walkthrough from one of 53 textgames; the selection criterion is that the best action to take at this step is a movement. In other words, each minigame is a scenario where the LLM has to figure out the best action to take given its previous actions and observations (i.e., the prefix of walkthrough up to the current step). This task is different and more challenging than answering the DF and RF questions: the LLM is not explicitly given a route (as in DF questions) or a destination (as in RF questions), but has to spontaneously figure out which action may contribute to its long-term goal.

For this task, we evaluated GPT-3.5 and GPT-4. For each model, we tried two settings: the first is to condition the LLM on the walkthrough like Example 1; the second is to include in the prompt the information about the nearby locations, and an example of the full prompt is given in Example 9. The information about nearby locations is the ground-truth information that the LLM, in principle, should have learned from the walkthrough prefix. If the LLM had a perfect mapping and navigation ability, it would be able to perfectly spell it out and use that information to guide its decision making. Figure 4 presents the results of this experiment. GPT-4 significantly outperforms GPT-3.5 in playing these minigames, consistent with their relative performance when answering the DF and RF questions of our MANGO benchmark. For each of the GPT models, having access to nearby location information significantly improves its performance, demonstrating that a strong mapping and navigation ability is essential to succeeding at relevant downstream tasks.

4 Related Work

Over the past few years, the field of natural language processing has experienced remarkable advancements with the emergence of large language models. This progress has spurred a multitude of research endeavors that propose benchmarks challenging the limits of these models. Those benchmarks assess the capacities of LLMs in linguistics (Wang et al., 2018; 2019), reading comprehension (Richardson et al., 2013; Lai et al., 2017), commonsense reasoning (Zellers et al., 2019; Bisk et al., 2020; Huang et al., 2019; Talmor et al., 2019), arithmetic reasoning (Miao et al., 2020; Cobbe et al., 2021; Patel et al., 2021), and knowledge memorization and understanding (Clark et al., 2018; Mihaylov et al., 2018; Khot et al., 2020; Clark et al., 2020; Hendrycks et al., 2021; Srivastava et al., 2022). Recent models have achieved remarkable performance not only on these benchmarks, but also across a diversity of human-oriented academic and professional exams (OpenAI, 2023) as well as general tasks (Bubeck et al., 2023). Our benchmark presents a unique challenge to large language models, evaluating their capacity to acquire spatial knowledge about new environments and answering complex navigation questions; it is a dimension orthogonal to the aforementioned reasoning abilities.

The advances of LLMs have sparked a recent wave of endeavors that integrate these models into embodied agents (Huang et al., 2022c; Yang et al., 2023; Vemprala et al., 2023; Wang et al., 2023a). Generally, they utilize language models as a means to understand human instructions and plan executable actions (Driess et al., 2023; Liang et al., 2022; Huang et al., 2022b; Ichter et al., 2023). This includes instructions related to object manipulation and tool operation (Wang et al., 2023b; Ren et al., 2023) as well as localization and navigation (Majumdar et al., 2020; Gadre et al., 2023; Shah et al., 2023; Huang et al., 2022a). Our MANGO benchmark aligns with the growing trend to deploy LLMs in embodied agents and provides a comprehensive investigation of their capacities in mapping and navigation. Our benchmark operates in text-based environments, distinguishing itself from previous benchmarks (Puig et al., 2018; Shridhar et al., 2020; Fan et al., 2022) that allow agents to utilize visual signals. This "text-only" design enables us to conduct controlled experiments that investigate the capacity of language models to acquire knowledge about environments solely from textual inputs and answer navigation questions based on that knowledge. It complements the existing benchmark and methodological research in vision-language navigation (Duvallet et al., 2014; Mei et al., 2016; Anderson et al., 2017; Fried et al., 2018; Zhu et al., 2020; Min et al., 2021). Our work is related to recent studies that demonstrate the emergence of maps with learned neural representations as a consequence of navigation (Huynh et al., 2020; Wijmans et al., 2023) with the key distinction that our agents are provided with textual descriptions of their environments.

Given our focus on mapping and navigation, it is worth noting the work on simultaneous localization and mapping (SLAM), a classic problem in which a mobile agent (e.g., a robot or hand-held camera) is tasked with mapping an a priori unknown environment while concurrently using its estimated map to localize itself in the environment (Mur-Artal et al., 2015; Cadena et al., 2016). Particularly relevant are the methods that maintain spatial-semantic maps of the environments based on natural language descriptions (Walter et al., 2013; Hemachandra & Walter, 2015), however they rely on non-linguistic observations (e.g., vision) to ground these descriptions.

5 Conclusion

We present MANGO, a benchmark that evaluates the mapping and navigation abilities of large language models. Our benchmark covers a diversity of mazes as well as a variety of evaluation and analysis programs, offering a comprehensive testbed in a great breadth and depth. In our experiments, the current best model still performs poorly on the benchmark, with a sharp degradation on the more difficult questions. We release our benchmark—along with the source code for data generation and evaluation—to track the advances of the mapping and navigation capabilities of future LLMs as well as to facilitate future research in related areas. Several interesting future directions are discussed in Appendix D.

Acknowledgments

This work was supported by a research gift to the last author by Adobe Research. It was also supported by the TRI University 2.0 Project, the U.S. National Science Foundation (Grant Number CNS-1948457 and CNS-2340171), and Office of Advanced Scientific Computing Research at Department of Energy (Award Number DE-SC0024424). The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof. We thank the anonymous COLM reviewers and meta-reviewer for their constructive feedback. We also thank our colleagues at TTIC and UChicago for helpful discussion.

References

Anderson, P., Wu, Q., Teney, D., Bruce, J., Johnson, M., Sünderhauf, N., Reid, I. D., Gould, S., and van den Hengel, A. Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3674–3683, 2017.

Anderson, P., Chang, A., Chaplot, D. S., Dosovitskiy, A., Gupta, S., Koltun, V., Kosecka, J., Malik, J., Mottaghi, R., Savva, M., et al. On evaluation of embodied navigation agents. *arXiv preprint arXiv:1807.06757*, 2018.

Anthopic. Introducing Claude. https://www.anthropic.com/news/introducing-claude, 2023a. Accessed: March 1, 2024.

Anthopic. Model card and evaluations for Claude models. https://www-cdn.anthropic.com/bd2a28d2535bfb0494cc8e2a3bf135d2e7523226/Model-Card-Claude-2.pdf, 2023b. Accessed: March 1, 2024.

Bisk, Y., Zellers, R., Gao, J., and Choi, Y. PIQA: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 2020.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., and Amodei, D. Language models are few-shot learners. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

Bubeck, S., Chandrasekaran, V., Eldan, R., Gehrke, J., Horvitz, E., Kamar, E., Lee, P., Lee, Y. T., Li, Y., Lundberg, S., et al. Sparks of artificial general intelligence: Early experiments with GPT-4. *arXiv preprint arXiv:2303.12712*, 2023.

Cadena, C., Carlone, L., Carrillo, H., Latif, Y., Scaramuzza, D., Neira, J., Reid, I., and Leonard, J. J. Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age. *IEEE Transactions on Robotics*, 2016.

Clark, P., Cowhey, I., Etzioni, O., Khot, T., Sabharwal, A., Schoenick, C., and Tafjord, O. Think you have solved question answering? Try ARC, the AI2 Reasoning Challenge. *arXiv preprint arXiv:1803.05457*, 2018.

Clark, P., Etzioni, O., Khot, T., Khashabi, D., Mishra, B., Richardson, K., Sabharwal, A., Schoenick, C., Tafjord, O., Tandon, N., et al. From F to A on the NY Regents Science Exams: An overview of the Aristo project. *AI Magazine*, 2020.

Cobbe, K., Kosaraju, V., Bavarian, M., Chen, M., Jun, H., Kaiser, L., Plappert, M., Tworek, J., Hilton, J., Nakano, R., et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Driess, D., Xia, F., Sajjadi, M. S. M., Lynch, C., Chowdhery, A., Ichter, B., Wahid, A., Tompson, J., Vuong, Q., Yu, T., Huang, W., Chebotar, Y., Sermanet, P., Duckworth, D., Levine, S., Vanhoucke, V., Hausman, K., Toussaint, M., Greff, K., Zeng, A., Mordatch, I., and Florence, P. PaLM-E: An embodied multimodal language model. *arXiv preprint arXiv:2303.03378*, 2023.

Duvallet, F., Walter, M. R., Howard, T., Hemachandra, S., Oh, J., Teller, S., Roy, N., and Stentz, A. Inferring maps and behaviors from natural language instructions. In *Proceedings of the International Symposium on Experimental Robotics (ISER)*, 2014.

Epstein, R. A., Patai, E. Z., Julian, J. B., and Spiers, H. J. The cognitive map in humans: spatial navigation and beyond. *Nature Neuroscience*, (11):1504–1513, 2017.

Fan, L., Wang, G., Jiang, Y., Mandlekar, A., Yang, Y., Zhu, H., Tang, A., Huang, D.-A., Zhu, Y., and Anandkumar, A. MineDojo: Building open-ended embodied agents with internet-scale knowledge. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

Fried, D., Hu, R., Cirik, V., Rohrbach, A., Andreas, J., Morency, L.-P., Berg-Kirkpatrick, T., Saenko, K., Klein, D., and Darrell, T. Speaker-follower models for vision-and-language navigation. In *Advances in Neural Information Processing Systems (NeurIPS)*, December 2018.

Gadre, S. Y., Wortsman, M., Ilharco, G., Schmidt, L., and Song, S. Cows on pasture: Baselines and benchmarks for language-driven zero-shot object navigation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.

Gao, L., Schulman, J., and Hilton, J. Scaling laws for reward model overoptimization. *arXiv preprint arXiv:2210.10760*, 2022.

Hausknecht, M., Ammanabrolu, P., Côté, M.-A., and Yuan, X. Interactive fiction games: A colossal adventure. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 2020.

Hemachandra, S. and Walter, M. R. Information-theoretic dialog to improve spatial-semantic representations. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2015.

Hendrycks, D., Burns, C., Basart, S., Zou, A., Mazeika, M., Song, D., and Steinhardt, J. Measuring massive multitask language understanding. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2021.

Huang, C., Mees, O., Zeng, A., and Burgard, W. Visual language maps for robot navigation. *arXiv preprint arXiv:2210.05714*, 2022a.

Huang, L., Bras, R. L., Bhagavatula, C., and Choi, Y. Cosmos QA: Machine reading comprehension with contextual commonsense reasoning. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2019.

Huang, W., Abbeel, P., Pathak, D., and Mordatch, I. Language models as zero-shot planners: Extracting actionable knowledge for embodied agents. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2022b.

Huang, W., Xia, F., Xiao, T., Chan, H., Liang, J., Florence, P., Zeng, A., Tompson, J., Mordatch, I., Chebotar, Y., et al. Inner monologue: Embodied reasoning through planning with language models. *arXiv preprint arXiv:2207.05608*, 2022c.

Huynh, T., Maire, M., and Walter, M. R. Multigrid neural memory. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2020.

Ichter, b., Brohan, A., Chebotar, Y., Finn, C., Hausman, K., Herzog, A., Ho, D., Ibarz, J., Irpan, A., Jang, E., Julian, R., Kalashnikov, D., Levine, S., Lu, Y., Parada, C., Rao, K., Sermanet, P., Toshev, A. T., Vanhoucke, V., Xia, F., Xiao, T., Xu, P., Yan, M., Brown, N., Ahn, M., Cortes, O., Sievers, N., Tan, C., Xu, S., Reyes, D., Rettinghouse, J., Quiambao, J., Pastor, P., Luu, L., Lee, K.-H., Kuang, Y., Jesmonth, S., Joshi, N. J., Jeffrey, K., Ruano, R. J., Hsu, J., Gopalakrishnan, K., David, B., Zeng, A., and Fu, C. K. Do as I can, not as I say: Grounding language in robotic affordances. In *Proceedings of the Conference on Robot Learning (CoRL)*, 2023.

Javadi, A.-H., Emo, B., Howard, L. R., Zisch, F. E., Yu, Y., Knight, R., Pinelo Silva, J., and Spiers, H. J. Hippocampal and prefrontal processing of network topology to simulate the future. *Nature Communications*, 2017.

Khot, T., Clark, P., Guerquin, M., Jansen, P., and Sabharwal, A. Qasc: A dataset for question answering via sentence composition. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 2020.

Lai, G., Xie, Q., Liu, H., Yang, Y., and Hovy, E. Race: Large-scale reading comprehension dataset from examinations. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2017.

Li, K., Hopkins, A. K., Bau, D., Viégas, F., Pfister, H., and Wattenberg, M. Emergent world representations: Exploring a sequence model trained on a synthetic task. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2023.

Liang, J., Huang, W., Xia, F., Xu, P., Hausman, K., Ichter, B., Florence, P., and Zeng, A. Code as policies: Language model programs for embodied control. *arXiv preprint arXiv:2209.07753*, 2022.

Majumdar, A., Shrivastava, A., Lee, S., Anderson, P., Parikh, D., and Batra, D. Improving vision-and-language navigation with image-text pairs from the Web. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2020.

Mei, H., Bansal, M., and Walter, M. Listen, attend, and walk: Neural mapping of navigational instructions to action sequences. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 2016.

Miao, S.-Y., Liang, C.-C., and Su, K.-Y. A diverse corpus for evaluating and developing english math word problem solvers. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, 2020.

Mihaylov, T., Clark, P., Khot, T., and Sabharwal, A. Can a suit of armor conduct electricity? a new dataset for open book question answering. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2018.

Min, S. Y., Chaplot, D. S., Ravikumar, P., Bisk, Y., and Salakhutdinov, R. FILM: Following instructions in language with modular methods. *arXiv preprint arXiv:2110.07342*, 2021.

Mur-Artal, R., Montiel, J. M. M., and Tardós, J. D. ORB-SLAM: A versatile and accurate monocular SLAM system. *IEEE Transactions on Robotics*, 31:1147–1163, 2015.

OpenAI. GPT-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

Patel, A., Bhattamishra, S., and Goyal, N. Are nlp models really able to solve simple math word problems? In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*, 2021.

Peng, B., Alcaide, E., Anthony, Q., Albalak, A., Arcadinho, S., Cao, H., Cheng, X., Chung, M., Grella, M., GV, K. K., He, X., Hou, H., Kazienko, P., Kocon, J., Kong, J., Koptyra, B., Lau, H., Mantri, K. S. I., Mom, F., Saito, A., Tang, X., Wang, B., Wind, J. S., Wozniak, S., Zhang, R., Zhang, Z., Zhao, Q., Zhou, P., Zhu, J., and Zhu, R.-J. RWKV: Reinventing RNNs for the transformer era. *arXiv preprint arXiv:2305.13048*, 2023.

Puig, X., Ra, K., Boben, M., Li, J., Wang, T., Fidler, S., and Torralba, A. Virtualhome: Simulating household activities via programs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

Ren, A. Z., Govil, B., Yang, T.-Y., Narasimhan, K. R., and Majumdar, A. Leveraging language for accelerated learning of tool manipulation. In *Proceedings of the Conference on Robot Learning (CoRL)*, 2023.

Richardson, M., Burges, C. J., and Renshaw, E. MCTest: A challenge dataset for the open-domain machine comprehension of text. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2013.

Rozière, B., Gehring, J., Gloeckle, F., Sootla, S., Gat, I., Tan, X. E., Adi, Y., Liu, J., Sauvestre, R., Remez, T., Rapin, J., Kozhevnikov, A., Evtimov, I., Bitton, J., Bhatt, M., Ferrer, C. C., Grattafiori, A., Xiong, W., Défossez, A., Copet, J., Azhar, F., Touvron, H., Martin, L., Usunier, N., Scialom, T., and Synnaeve, G. Code llama: Open foundation models for code. *arXiv preprint arXiv:2308.12950*, 2024.

Shah, D., Osiński, B., Levine, S., et al. LM-Nav: Robotic navigation with large pre-trained models of language, vision, and action. In *Proceedings of the Conference on Robot Learning (CoRL)*, 2023.

Shridhar, M., Thomason, J., Gordon, D., Bisk, Y., Han, W., Mottaghi, R., Zettlemoyer, L., and Fox, D. Alfred: A benchmark for interpreting grounded instructions for everyday tasks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

Spiers, H. J. and Gilbert, S. J. Solving the detour problem in navigation: A model of prefrontal and hippocampal interactions. *Frontiers in Human Neuroscience*, 2015.

Spiers, H. J. and Maguire, E. A. Thoughts, behaviour, and brain dynamics during navigation in the real world. *Neuroimage*, 31(4):1826–1840, 2006.

Srivastava, A., Rastogi, A., Rao, A., Shoeb, A. A. M., Abid, A., Fisch, A., Brown, A. R., Santoro, A., Gupta, A., Garriga-Alonso, A., et al. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *Transactions of Machine Learning Research*, 2022.

Stiennon, N., Ouyang, L., Wu, J., Ziegler, D., Lowe, R., Voss, C., Radford, A., Amodei, D., and Christiano, P. F. Learning to summarize with human feedback. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

Talmor, A., Herzig, J., Lourie, N., and Berant, J. CommonsenseQA: A question answering challenge targeting commonsense knowledge. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*, 2019.

Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.-A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., Rodriguez, A., Joulin, A., Grave, E., and Lample, G. LLaMA: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023a.

Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., Bashlykov, N., Batra, S., Bhargava, P., Bhosale, S., et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023b.

Vemprala, S., Bonatti, R., Bucker, A., and Kapoor, A. ChatGPT for robotics: Design principles and model abilities. Technical Report MSR-TR-2023-8, Microsoft, February 2023.

Walter, M. R., Hemachandra, S., Homberg, B., Tellex, S., and Teller, S. J. Learning semantic maps from natural language descriptions. *The International Journal of Robotics Research*, 2013.

Wang, A., Singh, A., Michael, J., Hill, F., Levy, O., and Bowman, S. R. GLUE: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*, 2018.

Wang, A., Pruksachatkun, Y., Nangia, N., Singh, A., Michael, J., Hill, F., Levy, O., and Bowman, S. R. SuperGLUE: A stickier benchmark for general-purpose language understanding systems. *arXiv preprint 1905.00537*, 2019.

Wang, G., Xie, Y., Jiang, Y., Mandlekar, A., Xiao, C., Zhu, Y., Fan, L., and Anandkumar, A. Voyager: An open-ended embodied agent with large language models. *arXiv preprint arXiv:2305.16291*, 2023a.

Wang, R., Mao, J., Hsu, J., Zhao, H., Wu, J., and Gao, Y. Programmatically grounded, compositionally generalizable robotic manipulation. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2023b.

Wei, J., Wang, X., Schuurmans, D., Bosma, M., Chi, E., Le, Q., and Zhou, D. Chain of thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

Wijmans, E., Savva, M., Essa, I., Lee, S., Morcos, A. S., and Batra, D. Emergence of maps in the memories of blind navigation agents. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2023.

Yang, S., Nachum, O., Du, Y., Wei, J., Abbeel, P., and Schuurmans, D. Foundation models for decision making: Problems, methods, and opportunities. *arXiv preprint arXiv:2303.04129*, 2023.

Zellers, R., Holtzman, A., Bisk, Y., Farhadi, A., and Choi, Y. HellaSwag: Can a machine really finish your sentence? In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, 2019.

Zhu, F., Zhu, Y., Chang, X., and Liang, X. Vision-language navigation with self-supervised auxiliary reasoning tasks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

A Benchmark Details

Our data and program is released at `https://mango.ttic.edu` and `https://github.com/oaklight/mango/`. In the `data` folder, each game has a folder that contains multiple JSON files. The most important files are the DF and RF skeletons (where `X` is the game name such as `zork` or `detective`):

- `X.df`: it contains the DF skeletons like Example 4.
- `X.rf`: it contains the RF skeletons like Example 5.

As introduced in section 2.2, each skeleton is paired with an ANSWERABLE label and an EASY label: given a prefix of the walkthrough, the ANSWERABLE label indicates whether this skeleton is answerable, and the EASY label will decide whether this skeleton is easy or hard given this walkthrough prefix. Table 3 displays some important statistics of the full dataset broken down into each individual maze, including the number of DF and RF skeletons. In total, our full dataset has about 3M DF questions and 200K RF questions. On average, each maze has around 60K DF questions and 4K RF questions. Noticeably, the 53 games in our dataset are very diverse: they cover a range of topics and genres; they are situated in different eras; they cover a wide variety of maps (small vs. big houses, long vs. short halls, towns vs. forests, verbose vs. concise scene descriptions, etc). In addition, we also provide the following data files for easy reference:

- `X.walkthrough`: it contains the full walkthrough of the game. See Appendix A.1 for details.
- `X.locations`: it lists all the locations. Details about annotating these locations can be found in Appendix A.2.
- `X.moves`: it lists all the moves that may change the location.
- `X.all_pairs`: it contains all the pairs of distinct locations.
- `X.all2all`: it contains all the simple paths between any pair of distinct locations.

In the following subsections, we will explain the details about collecting this data.

A.1 Walkthrough Details

In this section, we document the technical details about walkthroughs.

The only game that doesn't have a walkthrough is Leather Goddesses of Phobos (LGoP).

There may be multiple correct ways to complete a game with the same level of efficiency. So the program has some randomness. In our experiments, we fixed the random seed for better reproducibility.

Each action in the program-generated walkthroughs is a highly abbreviated symbol such as E for East and NW for Northwest. For a better readability, we use the full words in our enhanced walkthrough such as Example 1.

A.2 Location Resolution Details

For each maze, our human annotator read the walkthrough and annotated all the locations. In most cases, the surrounding description given by the game engine includes the name of the location, and the annotator needs to manually extract it from the text; this process is difficult to automate because the text is often unstructured and an automatic extractor is hard to build. What makes this annotation tricky is

- a location may be visited more than once in the walkthrough but we should avoid assigning multiple names to it.
- distinct locations may be referred to in the same way by the game engine but we should distinguish them. The game of Night is an example: hallways on different floors are all referred to as `Hall`; we renamed each of them, e.g., with `Hall (1st floor, north end)`.

MAPS	# LOCS	# EDGES	AVG LEN PATH	# STEPS	DF		RF	
					EASY	HARD	EASY	HARD
905	5	7	1.88	22	11	5	11	5
ADVENT	70	137	19.38	277	24788	4220	4694	0
ADVENTURELAND	23	48	7.46	170	1117	208	484	22
AFFLICTED	13	24	3.18	99	156	0	156	0
ANCHOR	118	228	27.46	531	66133	23564	13573	0
AWAKEN	15	28	5.02	57	365	45	171	25
BALANCES	12	20	3.22	122	136	1	94	1
BALLYHOO	39	90	9.26	416	18148	6298	1444	0
CURSES	150	304	33.83	816	1951931	549433	22201	0
CUTTHROAT	52	108	10.76	336	5969	2688	1832	59
DEEPHOME	66	127	8.64	327	4161	235	4161	1
DETECTIVE	32	41	8.76	51	505	6	505	6
DRAGON	24	52	8.9	101	1100	2262	529	23
ENCHANTER	57	110	10.93	265	5118	2602	2773	99
ENTER	18	34	3.88	102	306	0	306	0
GOLD	22	44	4.99	345	682	0	462	0
HHGG	42	65	7.02	361	3794	3	1602	1
HOLLYWOOD	109	219	27.33	397	21641	15557	11451	213
HUNTDARK	12	13	4.24	67	66	2	66	2
INFIDEL	58	122	9.27	250	2786	1729	2158	453
INHUMANE	43	91	6.49	122	1623	3020	1474	332
JEWEL	43	74	8.66	223	1157	61	1157	61
KARN	56	124	15.36	362	24479	108983	3025	0
LIBRARY	7	12	2.48	52	42	0	42	0
LOOSE	12	21	4.18	50	94	27	94	27
LOSTPIG	13	26	4.5	146	492	105	114	0
LUDICORP	86	176	14.96	364	7099	7281	6002	1308
LURKING	60	116	13.04	294	5187	2316	2946	105
MOONLIT	6	9	2.2	59	18	7	18	7
MURDAC	84	157	11.05	304	6914	2016	5967	1005
NIGHT	20	41	6.93	90	633	59	380	0
OMNIQUEST	32	65	7.55	78	648	1431	642	90
PARTYFOUL	4	8	1.92	56	24	0	12	0
PENTARI	18	31	3.76	49	208	20	208	4
PLANETFALL	69	138	12.41	399	7100	2246	3887	247
PLUNDERED	45	87	13.46	189	3705	2394	1393	26
REVERB	17	31	5.26	74	321	20	256	16
SEASTALKER	20	43	7.55	204	1123	1143	260	3
SHERLOCK	71	140	12.39	339	16417	11758	4214	27
SNACKTIME	4	6	1.5	34	12	0	12	0
SORCERER	64	120	11.72	254	4042	2285	2503	264
SPELLBRKR	73	111	13.62	412	8713	1762	3462	1
SPIRIT	229	466	26.39	1264	111102	47410	51531	226
TEMPLE	24	46	6.07	181	563	106	486	21
THEATRE	20	38	5.33	296	329	130	329	51
TRINITY	94	189	20.39	610	65049	131324	8742	0
TRYST205	73	139	10.04	518	5089	392	3983	181
WISHBRINGER	46	94	13.32	184	4988	5646	2070	0
YOMOMMA	9	25	3.55	98	189	263	64	0
ZENON	17	32	4.85	83	144	128	144	128
ZORK1	84	166	20.79	396	20984	26857	6889	0
ZORK2	66	136	11.79	296	15174	13081	3798	178
ZORK3	55	105	15.37	273	6202	2754	2294	133
ZTUU	16	30	3.27	84	225	15	225	15

Table 3: Statistics of full data. Here, # LOCS represents the number of locations in each walkthrough; # EDGES represents the number of edges; AVG LEN PATH denotes the average length of all paths; # STEPS indicates the number of steps in each walkthrough; EASY and HARD of the DF and RF respectively represent the number of easy and hard skeletons of the DF and RF tasks. When counting the easy and hard skeletons, we assume that the full walkthrough will be used. For the statistics of the data used in our experiments, please see Tables 5–8.

We solved the problems above by hacking into the source code of the game engines and conducting multiple rounds of human verification. First, for each maze, we checked the source code of the game provided by the Jericho game suite (Hausknecht et al., 2020) and found the unique ID for each location. Matching IDs with human annotations allows us to perform the following post processing:

- when a location is given multiple names, our human annotators work together to select the most proper unique name that they all agree on. The selection principles are: it is descriptive; whenever this location is visited in the walkthrough, the name has an intuitive match with the surrounding description given by the game engine.
- when multiple locations share a name, our human annotators work together to distinguish them by adding descriptive marks. An example is the `Halls` mentioned above: we renamed them to be `Hall (1st floor, north end of north/south hall)`, `Hall (1st floor, middle of north/south hall)`, and `Hall (2nd floor, middle of north/south hall)`. There are rare cases in which all the annotators agreed that no marks could be added and the location names had to be kept fuzzy (i.e., a name corresponds to multiple different locations). The rationale is: if a human may confuse with these locations, then it is reasonable for a model to have the same confusion. Then allowing them to share the name is essentially to apply a looser evaluation to the models: e.g., if the name `Forest` is overloaded, then when the model answers "how to reach `Forest` from `House`", any path that ends at any of the `Forests` will be considered to be correct. This treatment is equivalent to merging the locations with the same human-annotated name.

In our repository, there is a `data-intermediate` folder that tracks such intermediate annotations. In the folder of each game, the JSON files `anno2code` and `code2anno` track the mapping between machine IDs and human-annotated location names.

Why don't we just use the unique IDs as the location names? Because the IDs are often not intuitive or descriptive and they are often just strings of digits. Such IDs may not match any content in the walkthrough so a human or model may be confused when asked about the path between "loc12" and "loc5" after reading the very descriptive walkthrough.

In addition, we also provide an alternative version of our data in which location names include machine IDs. Precisely, while resolving each location name, we

- add the machine ID (e.g., "(obj59)") to the location name as a mark (e.g., "in debris room" to "in debris room (obj59)");
- if the location name already has a mark (e.g., `1st floor, north end of north/south hall`, thanks to the name resolution process), we replace the mark with the machine ID (e.g., "hall (1st floor, middle of north/south hall)" to "hall (obj66)").

A.3 Move Resolution Details

During human annotation, we labeled all the moves that change the locations. Like in Appendix A.2, we used the source code of the game engine to verify the human annotations. This annotation led to a map for each game—like what's shown in Figure 1—but this map is incomplete since there exist implicit moves between locations. For example, `south` is a movement that can end up at `Altar` if we start from `Temple`; see Figure 1. It means that `north` is also a possible move from `Altar` and it leads to `Temple`; but this edge has never explicitly shown up in the walkthrough. We would like our questions to cover such implicit edges, so we examined every possible implicit edge and inserted the valid ones into our map (though Figure 1 only displays the explicit edges).

The examination was carried out through real game playing by our human annotators. For each directional move (e.g., `south`), we tested if its reverse directional move (e.g., `north` for `south`, `down` for `up`) could lead to the previous location. Not every move is directional: e.g., in Zork-I, you may `enter` and `exit` the `House`; `pray` moves the player from `Altar` to `Forest`. We had such edges examined by human annotators: for some (e.g., `enter`), we could find intuitive reverse moves and verify them; for the others (e.g., `pray`), we didn't propose any reverse. Usually, the list of moves we ended up for a game includes eight possible directional moves as well as a few special moves.

A.4 Path Details

Once we have figured out all the locations and moves for a game, we will end up with a map. We collected all the unique pairs of distinct locations in the map and stored them in the `all_pairs` file: for each pair of locations `A` and `B`, we could ask a route finding question that aims to reach `B` from `A`. Note that `A`,`B` and `B`,`A` are different pairs.

For each pair of the starting point `A` and destination `B`, we collected all the simple paths `P` that connects from `A` to `B`. Each (`A`,`B`,`P`) tuple defines a destination finding question about where one will be if they go through path `P` from `A`.

A.5 Program Details

Our graph and path operations (e.g., finding simple paths) are handled by the networkx package. Its documentation is at `https://networkx.org/`. Particularly, the program that finds all the simple paths for a pair of graph nodes is at `https://networkx.org/documentation/stable/reference/algorithms/generated/networkx.algorithms.simple_paths.all_simple_paths.html`.

A.6 Evaluation Details

Another important—yet more strict—evaluation is the reasoning accuracy. This evaluation requires the language model to spell out its planned trajectories of moves when answering questions (like requested in our Examples 7 and 8). For an RF question, the reasoning process is correct if and only if the model-generated trajectory is a valid path from the starting position `S` to the destination `D`: the first step starts from `S`; each step starts from where it ended up in the last step; the final step reaches `D`. For a DF question, the model-generated trajectory has to be a valid path from `S` to the model-generated destination `D`; in addition, the sequence of moves in the trajectory has to match the given list of actions `A`. Like we explained in section 2.3, the "match" here is not an exact match: if the closest valid move is the correct move, then it is counted as a "match".

B Experiment Details

In this section, we present our experiment details for reproducing the results.

B.1 Model Configuration Details

The specific GPTs used in our experiments are GPT-4-0314 and for GPT-3.5-turbo-0301. For Claude-1 and Claude-2, the specific model versions we are using are Claude-instant-1.2 and Claude-2.0. For 14B RWKV model, the specific model version we are using is "RWKV-4-Pile-14B-20230313-ctx8192-test1050". For Llama-2, we used the ckeckpoint officially released by Meta.

For all the models, we set the temperature of the LLMs to be 0 for reproducibility.

B.2 Prompt Details

For GPTs, our prompts are the concatenation of walkthrough like in Example 1 and the questions like in Examples 2 and 3; the questions are obtained by filling the templates in Examples 7 and 8. When calling OpenAI API, we set the "role" to be "user" and the "content" to be the prompt. After receiving the response from the API call, we processed the output string by fetching the content from the structured output (recall that we request the models to return Python lists of Python dictionaries).

What prompts are to LLMs are like what hyperparameters are to classifical deep neural nets. Our experiments show that the structure of LLM output is sensitive to the prompts, and thus we had to carefully tune the prompts such that LLMs could return easy-to-parse output. For example, we found it helpful to ask LLMs to format their answers as a Python list of Python dictionaries; we also found it helpful to end our prompt with the symbol '['—i.e., the start symbol of a Python list—to elicit the LLM to actually output in the desired format.

Fortunately, as long as the output follows the desired structure, its content is relatively robust to the prompts. In our pilot experiments, we found that different prompts (created by different authors) would generate the same or similar content (regardless of its format) and achieve the same level of success rates (after parsing).

To ensure that we could obtain the optimal results of RWKV and Llamas, we tuned the prompts—more precisely, experimented with variants of the prompts of GPTs—and ended up with a set of new prompts that are mostly the same but exhibit some prose differences. For example, at the beginning of the prompts, we added "Here is a walkthrough of a text game." It is worth noting that, even though we carefully tuned the prompts, the non-GPT models still suffer a high chance of failing to return well-structured answers. Table 4 shows the average number of answered questions as well as how many of them received ill-formatted answers. As we could see, GPT-3.5 and GPT-4 could generate well-structured answers for a large portion of the DF and RF questions, but the other models often gave ill-structured answers. As a result, we could only evaluate the non-GPT models on the questions to which the answers were well-structured and thus could be parsed. We also experimented with the function-calling interface of GPTs, but it didn't lead to an increased amount of well-structured answers compared to our prompt design.

MODEL	RF QUESTION		DF QUESTION	
	# ANSWERABLE	# ILL-STRUCTURED	# ANSWERABLE	# ILL-STRUCTURED
RWKV	277.32	240.04	397.08	325.98
LLAMA-2	138.40	112.25	157.98	98.0
CLAUDE-1	184.34	3.88	244.92	4.66
CLAUDE-2	184.34	6.28	244.92	1.86
GPT-3.5	184.34	30.81	244.92	65.60
GPT-4	184.34	2.49	244.92	5.74

Table 4: Average (per-maze) numbers of answerable questions and ill-structured answers for each model.

B.3 Walkthrough and Question Details

Due to the context window size of an LLM, it is often the case that we have to only use a prefix of the walkthrough when evaluating an LLM on a maze. Recall from section 2.2 that not every question is answerable given a walkthrough prefix. Therefore, each model was evaluated on a different set of questions. Tables 5–8 display the statistics about the data that each LLM was actually evaluated on in our experiments.

MAPS	# LOCS	# EDGES	AVG LEN PATH	# STEPS	DF		RF	
					EASY	HARD	EASY	HARD
905	5	7	1.88	21	11	5	11	5
ADVENT	31	57	7.79	70	692	100	532	100
ADVENTURELAND	18	35	6.13	70	579	80	260	46
AFFLICTED	10	16	2.78	40	67	0	67	0
ANCHOR	13	24	3.94	24	95	91	82	74
AWAKEN	14	24	4.8	44	262	12	157	12
BALANCES	11	18	3.09	67	96	8	76	8
BALLYHOO	14	28	4.8	56	225	99	156	13
CURSES	14	24	3.3	53	122	13	122	0
CUTTHROAT	22	40	5.81	62	303	158	202	107
DEEPHOME	17	28	4.35	49	175	10	175	10
DETECTIVE	26	34	7.17	43	334	4	334	4
DRAGON	14	25	3.59	29	105	64	111	58
ENCHANTER	21	38	5.69	53	216	165	216	165
ENTER	2	1	1.0	20	1	0	1	0
GOLD	11	17	2.82	47	83	0	83	0
HHGG	8	9	2.6	51	29	1	29	1
HOLLYWOOD	8	14	2.71	50	43	13	43	13
HUNTDARK	10	9	3.67	55	45	0	45	0
INFIDEL	13	26	3.6	55	114	138	88	68
INHUMANE	21	40	4.8	49	275	240	261	159
JEWEL	16	30	4.39	60	166	74	166	74
KARN	19	35	6.37	65	339	86	231	63
LIBRARY	7	12	2.48	49	42	0	42	0
LOOSE	8	14	3.0	39	56	0	56	0
LOSTPIG	6	9	1.96	56	16	9	16	9
LUDICORP	22	43	4.91	70	351	111	351	111
LURKING	10	16	2.89	56	66	16	65	16
MOONLIT	4	6	1.67	45	9	3	9	3
MURDAC	30	52	6.34	70	537	195	528	183
NIGHT	20	41	6.93	70	633	59	380	0
OMNIQUEST	29	59	7.75	70	536	1198	290	298
PARTYFOUL	4	6	1.67	24	12	0	11	1
PENTARI	18	30	3.72	48	208	4	208	4
PLANETFALL	21	37	5.27	68	246	50	246	50
PLUNDERED	10	11	3.23	32	47	0	47	0
REVERB	12	21	4.07	40	129	12	120	12
SEASTALKER	10	15	2.7	53	50	3	50	3
SHERLOCK	8	13	2.48	33	43	5	43	0
SNACKTIME	4	6	1.5	33	12	0	12	0
SORCERER	15	26	3.53	54	120	17	120	17
SPELLBRKR	15	23	3.6	52	120	17	120	16
SPIRIT	15	26	3.55	50	171	12	171	12
TEMPLE	10	15	3.11	46	48	18	48	18
TRINITY	10	10	3.61	45	45	1	45	1
TRYST205	9	14	1.93	65	57	0	57	0
WISHBRINGER	18	30	5.2	45	174	48	175	47
YOMOMMA	8	15	2.39	41	49	17	31	18
ZENON	13	24	4.05	63	83	73	83	73
ZORK1	19	34	7.16	70	351	46	279	45
ZORK2	22	33	5.65	50	242	18	146	103
ZORK3	23	44	6.84	61	600	167	394	68
ZTUU	11	18	2.79	43	91	0	91	0

Table 5: Map statistics for GPTs and Claudes. Here, # LOCS represents the number of locations in each walkthrough; # EDGES represents the number of edges; AVG LEN PATH denotes the average length of all paths; # STEPS indicates the number of steps in each walkthrough; EASY and HARD of the DF and RF respectively represent the number of easy and hard skeletons of the DF and RF tasks.

MAPS	# LOCS	# EDGES	AVG LEN PATH	# STEPS	DF		RF	
					EASY	HARD	EASY	HARD
905	5	7	1.88	21	11	5	11	5
ADVENT	31	57	7.79	70	692	100	532	100
ADVENTURELAND	18	35	6.13	70	579	80	260	46
AFFLICTED	11	20	2.95	70	100	10	100	10
ANCHOR	25	46	5.99	70	327	153	302	132
AWAKEN	15	28	5.02	56	365	45	171	25
BALANCES	11	18	3.09	70	96	8	76	8
BALLYHOO	17	35	4.83	70	302	188	213	59
CURSES	14	27	3.62	70	182	13	182	0
CUTTHROAT	25	49	6.66	70	471	362	360	216
DEEPHOME	27	49	4.83	70	429	19	429	19
DETECTIVE	32	40	8.79	50	505	4	505	4
DRAGON	21	44	7.11	70	533	990	272	148
ENCHANTER	23	43	6.35	70	265	219	265	219
ENTER	14	26	3.36	70	117	65	117	65
GOLD	15	25	3.45	70	143	0	143	0
HHGG	9	11	2.85	70	38	1	38	1
HOLLYWOOD	12	22	3.36	70	84	48	84	48
HUNTDARK	12	11	4.33	66	66	0	66	0
INFIDEL	24	48	7.53	70	312	446	264	288
INHUMANE	30	57	5.54	70	614	555	483	280
JEWEL	17	32	4.4	70	187	85	187	85
KARN	19	35	6.37	70	339	86	231	63
LIBRARY	7	12	2.48	51	42	0	42	0
LOOSE	12	21	4.18	49	94	27	94	27
LOSTPIG	7	11	2.28	70	22	14	22	14
LUDICORP	22	43	4.91	70	351	111	351	111
LURKING	16	29	4.29	70	144	97	143	97
MOONLIT	6	9	2.2	58	18	7	18	7
MURDAC	30	52	6.34	70	537	195	528	183
NIGHT	20	41	6.93	70	633	59	380	0
OMNIQUEST	29	59	7.75	70	536	1198	290	298
PARTYFOUL	4	9	1.97	55	24	6	11	1
PENTARI	18	30	3.72	48	208	4	208	4
PLANETFALL	22	39	5.48	70	267	63	267	63
PLUNDERED	22	37	6.02	70	450	52	289	26
REVERB	17	31	5.26	70	321	20	253	19
SEASTALKER	10	15	2.7	70	50	3	50	3
SHERLOCK	18	28	4.36	70	175	5	175	0
SNACKTIME	4	6	1.5	33	12	0	12	0
SORCERER	26	46	7.09	70	340	48	340	48
SPELLBRKR	20	31	4.84	70	295	23	276	21
SPIRIT	22	41	4.09	70	354	87	354	87
TEMPLE	19	33	4.72	70	178	69	178	69
TRINITY	17	17	5.96	70	136	1	136	1
TRYST205	9	15	1.94	70	64	0	64	0
WISHBRINGER	21	40	6.34	70	259	214	251	169
YOMOMMA	9	20	2.74	70	82	59	43	21
ZENON	14	26	4.27	70	96	86	96	86
ZORK1	19	34	7.16	70	351	46	279	45
ZORK2	22	45	7.01	70	536	754	239	130
ZORK3	23	45	6.93	70	627	174	414	70
ZTUU	15	26	3.15	70	183	0	183	0

Table 6: Map statistics for RWKV, RWKV-S, Llama-1-S, Llama-2-S, Code-Llama-S, Code-Llama-Instruct-S. Here, # LOCS represents the number of locations in each walkthrough; # EDGES represents the number of edges; AVG LEN PATH denotes the average length of all paths; # STEPS indicates the number of steps in each walkthrough; EASY and HARD of the DF and RF respectively represent the number of easy and hard skeletons of the DF and RF tasks.

MAPS	# LOCS	# EDGES	AVG LEN PATH	# STEPS	DF		RF	
					EASY	HARD	EASY	HARD
905	5	7	1.88	19	11	5	11	5
ADVENT	12	21	3.73	18	67	30	67	30
ADVENTURELAND	8	11	2.09	20	32	1	32	1
AFFLICTED	3	3	1.25	8	4	0	4	0
ANCHOR	5	5	1.91	5	10	1	7	4
AWAKEN	6	8	2.32	10	24	4	21	4
BALANCES	5	5	1.91	16	10	1	10	1
BALLYHOO	5	4	2.0	9	10	0	10	0
CURSES	6	8	1.95	12	17	2	17	0
CUTTHROAT	2	2	1.0	13	1	1	1	1
DEEPHOME	3	3	1.25	11	3	1	3	1
DETECTIVE	7	8	2.35	16	23	0	23	0
DRAGON	4	4	1.57	4	6	1	6	1
ENCHANTER	9	13	2.63	15	38	13	38	13
ENTER	0	0	0	3	0	0	0	0
GOLD	5	5	1.91	9	11	0	11	0
HHGG	4	4	1.57	15	6	1	6	1
HOLLYWOOD	4	5	1.56	10	6	3	6	3
HUNTDARK	3	2	1.33	10	3	0	3	0
INFIDEL	6	8	1.9	9	16	5	17	4
INHUMANE	6	8	1.95	14	19	2	19	2
JEWEL	6	9	2.08	17	21	4	21	4
KARN	3	4	1.33	20	6	0	6	0
LIBRARY	2	1	1.0	12	1	0	1	0
LOOSE	4	3	1.67	8	6	0	6	0
LOSTPIG	3	3	1.25	17	3	1	3	1
LUDICORP	8	11	2.62	24	32	0	32	0
LURKING	3	3	1.25	21	4	0	4	0
MOONLIT	3	2	1.33	14	3	0	3	0
MURDAC	11	19	3.75	25	110	8	102	8
NIGHT	11	12	3.62	17	58	0	58	0
OMNIQUEST	10	18	2.73	22	54	36	54	36
PARTYFOUL	0	0	0	4	0	0	0	0
PENTARI	8	10	2.78	7	28	4	28	4
PLANETFALL	2	2	1.0	20	1	1	1	1
PLUNDERED	2	1	1.0	9	1	0	1	0
REVERB	5	5	1.91	8	10	1	10	1
SEASTALKER	2	2	1.0	12	1	1	1	1
SHERLOCK	4	3	1.67	7	6	0	6	0
SNACKTIME	2	2	1.0	15	2	0	2	0
SORCERER	4	7	1.56	16	7	2	7	2
SPELLBRKR	3	3	1.25	10	3	1	3	1
SPIRIT	4	4	1.57	9	6	1	6	1
TEMPLE	3	3	1.25	11	3	1	3	1
TRINITY	4	4	1.57	14	6	1	6	1
TRYST205	3	2	1.33	9	3	0	3	0
WISHBRINGER	7	8	2.35	11	23	0	23	0
YOMOMMA	4	5	1.57	8	6	1	4	3
ZENON	4	4	1.57	21	6	1	6	1
ZORK1	8	12	2.27	22	34	3	34	3
ZORK2	8	10	2.65	11	29	2	29	2
ZORK3	7	14	2.59	17	40	24	30	12
ZTUU	5	4	2.0	5	10	0	10	0

Table 7: Map statistics for Llama-1. Here, # LOCS represents the number of locations in each walkthrough; # EDGES represents the number of edges; AVG LEN PATH denotes the average length of all paths; # STEPS indicates the number of steps in each walkthrough; EASY and HARD of the DF and RF respectively represent the number of easy and hard skeletons of the DF and RF tasks.

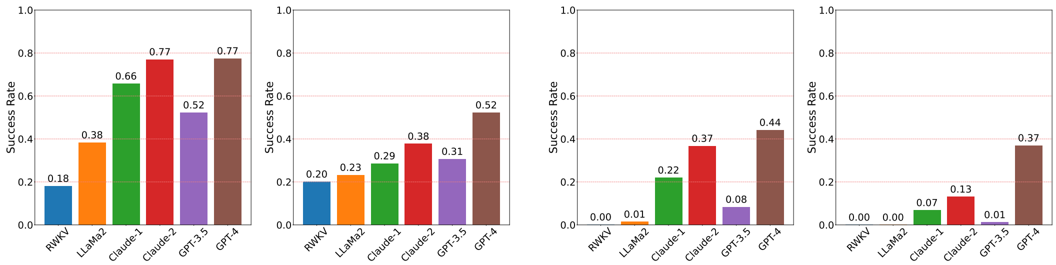
MAPS	# LOCS	# EDGES	AVG LEN PATH	# STEPS	DF		RF	
					EASY	HARD	EASY	HARD
905	5	7	1.88	21	11	5	11	5
ADVENT	25	45	7.52	54	473	76	355	76
ADVENTURELAND	18	33	5.91	56	413	42	260	46
AFFLICTED	9	14	2.43	32	46	0	46	0
ANCHOR	9	15	3.0	19	50	16	41	13
AWAKEN	13	18	3.98	34	114	4	90	4
BALANCES	8	12	2.51	49	38	1	38	1
BALLYHOO	12	21	3.98	43	111	35	101	11
CURSES	10	17	2.93	42	67	9	67	0
CUTTHROAT	19	28	4.75	42	180	9	136	53
DEEPHOME	12	21	3.43	40	102	10	102	10
DETECTIVE	25	32	6.99	40	307	4	307	4
DRAGON	11	20	3.16	22	68	42	70	40
ENCHANTER	17	31	4.61	41	142	114	142	114
ENTER	0	0	0	15	0	0	0	0
GOLD	8	14	2.5	40	56	0	56	0
HHGG	7	8	2.35	39	22	1	22	1
HOLLYWOOD	7	8	2.5	34	21	3	21	3
HUNTDARK	7	6	2.67	41	21	0	21	0
INFIDEL	12	24	3.34	42	102	116	78	54
INHUMANE	18	34	4.82	40	218	168	190	116
JEWEL	15	25	3.77	46	141	30	141	30
KARN	15	27	4.11	49	136	38	144	30
LIBRARY	7	11	2.22	40	32	0	32	0
LOOSE	8	13	2.86	28	49	0	49	0
LOSTPIG	5	7	1.69	47	11	5	11	5
LUDICORP	19	37	4.14	65	291	51	291	51
LURKING	6	9	2.42	40	30	1	29	1
MOONLIT	3	2	1.33	36	3	0	3	0
MURDAC	23	43	5.9	62	355	171	346	160
NIGHT	20	39	5.79	53	380	19	380	0
OMNIQUEST	26	49	6.28	58	348	180	222	192
PARTYFOUL	4	6	1.67	19	12	0	11	1
PENTARI	17	29	3.67	42	191	4	191	4
PLANETFALL	17	29	4.96	52	164	16	164	16
PLUNDERED	7	7	2.59	26	22	0	22	0
REVERB	12	15	3.54	28	68	2	68	2
SEASTALKER	10	15	2.7	43	50	3	50	3
SHERLOCK	6	9	2.08	21	25	0	25	0
SNACKTIME	4	6	1.5	33	12	0	12	0
SORCERER	10	18	2.3	44	60	13	60	13
SPELLBRKR	10	13	2.71	41	60	2	59	2
SPIRIT	13	21	3.22	38	99	6	99	6
TEMPLE	9	14	2.84	37	39	18	39	18
TRINITY	8	8	2.93	37	28	1	28	1
TRYST205	8	13	1.92	49	49	0	49	0
WISHBRINGER	18	30	5.2	38	174	48	175	47
YOMOMMA	7	13	2.18	31	36	14	26	10
ZENON	12	22	3.71	52	70	62	70	62
ZORK1	19	32	7.25	56	332	22	279	45
ZORK2	19	27	5.15	42	176	18	107	76
ZORK3	18	35	4.78	48	282	100	199	45
ZTUU	8	11	2.26	28	38	0	38	0

Table 8: Map statistics for Llama-2, Code-Llama, Code-Llama-Instruct. Here, # LOCS represents the number of locations in each walkthrough; # EDGES represents the number of edges; AVG LEN PATH denotes the average length of all paths; # STEPS indicates the number of steps in each walkthrough; EASY and HARD of the DF and RF respectively represent the number of easy and hard skeletons of the DF and RF tasks.

C More Results

C.1 Success Rate Weighted by Route Length

Following the evaluation standard proposed by Anderson et al. (2018), we also computed the success rates weighted by route length, which are shown in Appendix C.1. As we can see, the ranking of the models stays the same as in Figure 2.

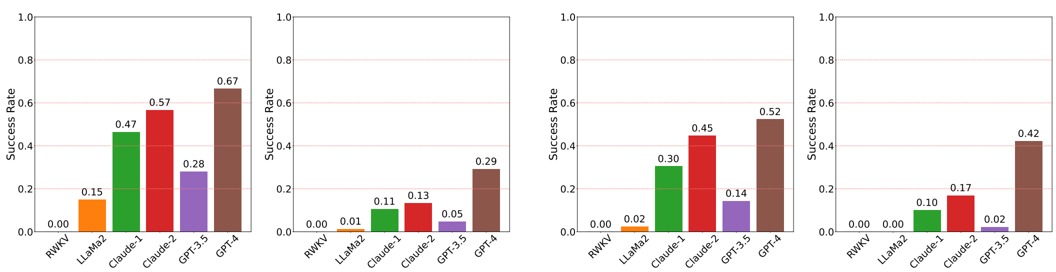


(a) On easy (left) and hard (right) DF questions. (b) On easy (left) and hard (right) RF questions.

Figure 5: Success rates weighted by route length on DF (a) and RF (b) questions, averaged over all 53 mazes.

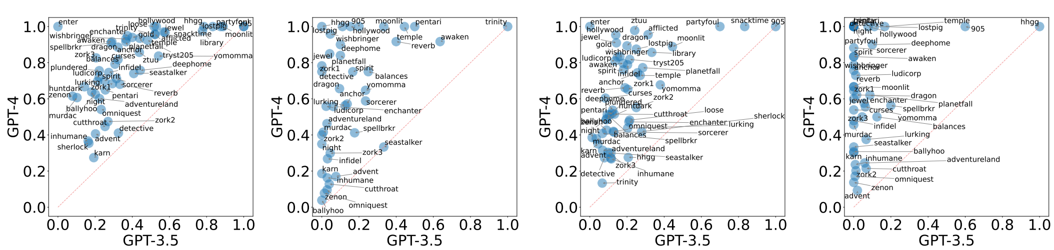
C.2 Reasoning Accuracy Results

In this section, we present the results of each LLM measured by reasoning accuracy, the metric introduced in Appendix A.6. Results are shown in Figure 6, with their pair-wise comparison shown in Table 9. As we can see, the trend measured by this metric is similar to what's shown in section 3: GPT-4 is the best among all the evaluated models but still suffers a low accuracy. Figure 7 shows the reasoning accuracies of GPT-3.5 vs. GPT-4 broken down into individual games, showing similar patterns with Figure 3.



(a) On easy (left) and hard (right) DF questions. (b) On easy (left) and hard (right) RF questions.

Figure 6: Reasoning accuracy of each model on DF (a) and RF (b) questions, averaged over all 53 mazes.



(a) GPT-3.5 vs. GPT-4 on DF questions. (b) GPT-3.5 vs. GPT-4 on RF questions.

Figure 7: Reasoning accuracies of GPT-3.5 and GPT-4 broken down into individual games. Similar to Figure 3, in each subfigure, the left scatterplot is for easy questions while the right is for hard questions.

METHOD	RWKV	LLAMA-2	CLAUDE-1	CLAUDE-2	GPT-3.5	GPT-4	$\overline{\text{HARD}}\vert$
RWKV	*	0.00 ∣ 0.02	0.00 ∣ 0.15	0.00 ∣ 0.18	0.00 ∣ 0.05	0.00 ∣ 0.41	*
LLAMA-2	0.15 ∣ 0.00	*	0.01 ∣ 0.20	0.01 ∣ 0.24	0.01 ∣ 0.07	0.01 ∣ 0.52	*
CLAUDE-1	0.51 ∣ 0.00	0.62 ∣ 0.15	*	0.11 ∣ 0.13	0.13 ∣ 0.05	0.11 ∣ 0.29	*
CLAUDE-2	0.60 ∣ 0.00	0.72 ∣ 0.15	0.58 ∣ 0.47	*	0.16 ∣ 0.05	0.13 ∣ 0.29	*
GPT-3.5	0.32 ∣ 0.00	0.38 ∣ 0.15	0.28 ∣ 0.52	0.28 ∣ 0.63	*	0.05 ∣ 0.35	*
GPT-4	0.71 ∣ 0.00	0.79 ∣ 0.15	0.67 ∣ 0.47	0.67 ∣ 0.58	0.73 ∣ 0.28	*	*
∣ $\underline{\text{EASY}}$	*	*	*	*	*	*	*

(a) Pairwise comparison on easy (lower left) and hard (higher right) DF questions.

METHOD	RWKV	LLAMA-2	CLAUDE-1	CLAUDE-2	GPT-3.5	GPT-4	$\overline{\text{HARD}}\vert$
RWKV	*	0.00 ∣ 0.00	0.00 ∣ 0.12	0.00 ∣ 0.20	0.00 ∣ 0.03	0.00 ∣ 0.51	*
LLAMA-2	0.02 ∣ 0.00	*	0.00 ∣ 0.14	0.00 ∣ 0.19	0.00 ∣ 0.04	0.00 ∣ 0.45	*
CLAUDE-1	0.33 ∣ 0.00	0.32 ∣ 0.02	*	0.10 ∣ 0.17	0.12 ∣ 0.02	0.10 ∣ 0.42	*
CLAUDE-2	0.46 ∣ 0.00	0.45 ∣ 0.03	0.45 ∣ 0.31	*	0.18 ∣ 0.02	0.17 ∣ 0.43	*
GPT-3.5	0.15 ∣ 0.00	0.16 ∣ 0.03	0.15 ∣ 0.34	0.14 ∣ 0.48	*	0.02 ∣ 0.46	*
GPT-4	0.56 ∣ 0.00	0.55 ∣ 0.02	0.53 ∣ 0.31	0.53 ∣ 0.45	0.56 ∣ 0.15	*	*
∣ $\underline{\text{EASY}}$	*	*	*	*	*	*	*

(b) Pairwise comparison on easy (lower left) and hard (higher right) RF questions.

Table 9: Reasoning accuracies on DF and RF questions broken down into pairwise comparison.

METRIC	GPT-3.5				GPT-4			
	DF		RF		DF		RF	
	β	p	β	p	β	p	β	p
# LOCATIONS	−0.076	0.001	−0.084	0.000	−0.073	0.000	−0.120	0.000
# EXP EDGES	−0.068	0.003	−0.081	0.001	−0.067	0.000	−0.096	0.003
# CONF LOCATIONS	−0.056	0.017	−0.067	0.006	−0.065	0.000	−0.088	0.006
AVG LEN EASY	−0.066	0.005	−0.081	0.001	−0.066	0.000	−0.131	0.000
AVG LEN SCENE	−0.037	0.119	−0.002	0.939	0.031	0.036	0.069	0.035

(a) Regression analysis results on easy questions.

METRIC	GPT-3.5				GPT-4			
	DF		RF		DF		RF	
	β	p	β	p	β	p	β	p
# LOCATIONS	−0.049	0.088	−0.055	0.043	−0.083	0.007	−0.115	0.006
# EXP EDGES	−0.052	0.070	−0.064	0.018	−0.080	0.010	−0.107	0.011
# IMP EDGES	−0.059	0.038	−0.032	0.252	−0.118	0.000	−0.152	0.000
# CONF LOCATIONS	−0.045	0.125	−0.055	0.043	−0.068	0.030	−0.081	0.061
AVG LEN HARD	−0.072	0.011	−0.055	0.046	−0.094	0.002	−0.122	0.004
AVG # IMP IN HARD	−0.057	0.046	−0.022	0.428	−0.081	0.009	−0.096	0.024
AVG LEN SCENE	0.032	0.268	0.050	0.069	0.084	0.007	0.105	0.013

(b) Regression analysis results on hard questions.

Table 10: Regression analysis results, where β is the regression coefficient and p denotes the p-value. When $p < 0.001$, we write 0.000 for presentation simplicity.

C.3 Analysis Details of GPTs

As discussed in section 3.3, we used regression analysis to understand the effects of the aforementioned variables on model performance. In particular, for each model (GPT-3.5 or GPT-4) on each type of question (DF or RF, easy or hard), we ran single-variable linear regression to understand how the success rate varies with each of the variables of interest. Table 10 displays the regression results.

In our pilot experiments, we ran a multivariate regression analysis that used the aforementioned variables jointly. However, the results of this regression are misleading: due to colinearity among the explanatory variables, the estimated coefficients are unreliable and the p-values are inflated. We also tried principled component regression, but the first principle component has nearly equal loadings across all the variables, making it inconvenient to interpret the results.

C.4 Results of Search-Based Approaches

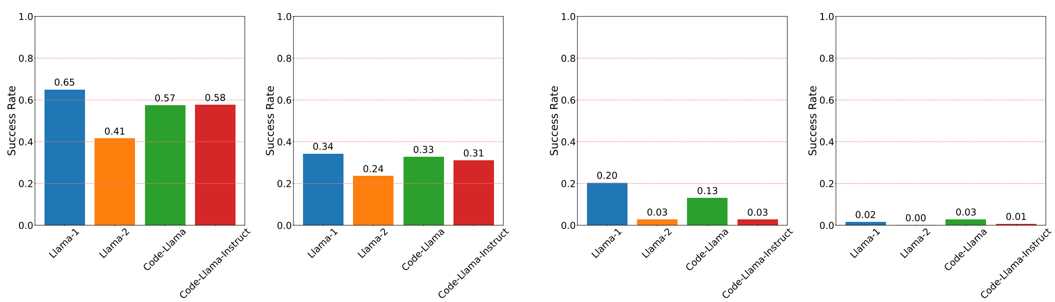
We also explored an approach that first maps a walkthrough to a symbolic graph and then uses a search algorithm to answer the given (DF or RF) question. However, we found it to be very challenging to translate natural language walkthrough into searchable symbolic graphs in the first place, making this approach not promising.

We tried two different settings. In the first setting, we did 5-shot prompting and each in-context example is a single S-A-D triplet; the examples are the first five steps in the walkthrough. In the second setting, we did 0-shot prompting and the prompt includes the entire walkthrough; the LLM has to generate a sequence of S-A-D triplets which then could be used to construct the graph. For each setting, we ran experiments on the following six mazes: Zork-I, Night, Partyfoul, Plundered, Spirit, and Temple.

In both settings, GPT-3.5 yields a low accuracy of the S-A-D triplet completion, indicating the difficulty of translating natural language into symbolic graphs. In the first setting, the average accuracy is 70.5 %. In the second setting, the average success rate is 6.3 %. The second setting is more challenging because any mistake would cause all subsequent steps to be incorrect. Some LLM mistakes and our error analysis can be found at `https://github.com/Oaklight/mango/tree/camera-ready/utils/supp_exp`.

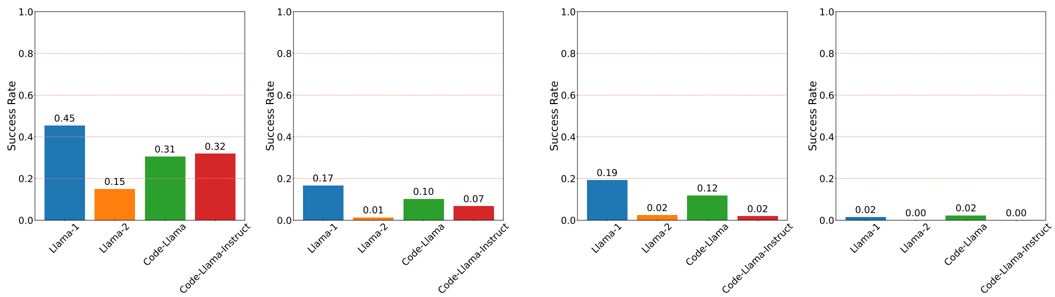
C.5 More Llama Results

The Llama-2 we used in the experiments of section 3 is the base model. We also experimented with the the 32.5B Llama-1 model released earlier (Touvron et al., 2023a), 13b CodeLlama and 13b CodeLlama-Instruct (Rozière et al., 2024). Llama-1 has a significantly smaller context window, so it had to read shorter walkthrough prefixes and answer fewer questions. The results of comparing different Llamas are in Figure 8, Figure 9, Table 11, and Table 12. As we can see, both CodeLlama and CodeLlama-Instruct outperform Llama-2, and Llama-1 performs the best in the group. In the main paper, we present the results of Llama-2-base because it is commonly considered to be the standard choice of the Llama series.



(a) On easy (left) and hard (right) DF questions. (b) On easy (left) and hard (right) RF questions.

Figure 8: Success rates of the Llama-1 and Llama-2 on DF (a) and RF (b) questions, averaged over all 53 mazes.



(a) On easy (left) and hard (right) DF questions. (b) On easy (left) and hard (right) RF questions.

Figure 9: Reasoning accuracy of the Llama-1 and Llama-2 on DF (a) and RF (b) questions, averaged over all 53 mazes.

METHOD	LLAMA-1	LLAMA-2	CODE-LLAMA	CODE-LLAMA-INSTRUCT	$\overline{\text{HARD}}$|
LLAMA-1	*	0.29 | 0.25	0.34 | 0.38	0.34 | 0.38	*
LLAMA-2	0.47 | 0.65	*	0.24 | 0.30	0.24 | 0.30	*
CODE-LLAMA	0.65 | 0.67	0.60 | 0.44	*	0.31 | 0.31	*
CODE-LLAMA-INSTRUCT	0.65 | 0.67	0.60 | 0.44	0.58 | 0.58	*	*
| $\underline{\text{EASY}}$	*	*	*	*	*

(a) Pairwise comparison on easy (lower left) and hard (higher right) DF questions.

METHOD	LLAMA-1	LLAMA-2	CODE-LLAMA	CODE-LLAMA-INSTRUCT	$\overline{\text{HARD}}$|
LLAMA-1	*	0.00 | 0.00	0.00 | 0.00	0.00 | 0.00	*
LLAMA-2	0.05 | 0.18	*	0.00 | 0.01	0.00 | 0.01	*
CODE-LLAMA	0.03 | 0.20	0.04 | 0.03	*	0.01 | 0.01	*
CODE-LLAMA-INSTRUCT	0.03 | 0.20	0.04 | 0.03	0.03 | 0.03	*	*
| $\underline{\text{EASY}}$	*	*	*	*	*

(b) Pairwise comparison on easy (lower left) and hard (higher right) RF questions.

Table 11: Llamas' success rates on DF and RF questions broken down into pairwise comparison.

METHOD	LLAMA-1	LLAMA-2	CODE-LLAMA	CODE-LLAMA-INSTRUCT	$\overline{\text{HARD}}$|
LLAMA-1	*	0.09 | 0.01	0.16 | 0.19	0.16 | 0.19	*
LLAMA-2	0.25 | 0.40	*	0.02 | 0.06	0.02 | 0.06	*
CODE-LLAMA	0.47 | 0.47	0.34 | 0.18	*	0.07 | 0.07	*
CODE-LLAMA-INSTRUCT	0.47 | 0.47	0.34 | 0.18	0.32 | 0.32	*	*
| $\underline{\text{EASY}}$	*	*	*	*	*

(a) Pairwise comparison on easy (lower left) and hard (higher right) DF questions.

METHOD	LLAMA-1	LLAMA-2	CODE-LLAMA	CODE-LLAMA-INSTRUCT	$\overline{\text{HARD}}$|
LLAMA-1	*	0.00 | 0.00	0.00 | 0.00	0.00 | 0.00	*
LLAMA-2	0.05 | 0.18	*	0.00 | 0.00	0.00 | 0.00	*
CODE-LLAMA	0.02 | 0.18	0.03 | 0.03	*	0.00 | 0.00	*
CODE-LLAMA-INSTRUCT	0.02 | 0.18	0.03 | 0.03	0.02 | 0.02	*	*
| $\underline{\text{EASY}}$	*	*	*	*	*

(b) Pairwise comparison on easy (lower left) and hard (higher right) RF questions.

Table 12: Llamas' reasoning accuracies on DF and RF questions broken down into pairwise comparison.

C.6 Human Performance Details

To measure human agreement, we computed the mean square error (MSE) for each metric across the questions that were answered by more than one human rater. As shown in Table 13, the human agreement turns out to be high.

	All		Easy		Hard	
Task Type	SR	RA	SR	RA	SR	RA
Route Finding	0.1442	0.3333	0.0425	0.1429	0.5000	1.0000
Destination Finding	0.0000	0.6000	0.0000	0.5714	0.0000	0.6667

Table 13: Human agreement measured by MSE. SR stands for success rate, while RA denotes reasoning accuracy.

C.7 More Results About Playing Minigames

In section 3.5, we have shown that a strong mapping and navigation ability can help an LLM achieve better performance in playing the minigames. Now we show the fine-grained results for that set of experiments. Precisely, Table 14 shows the spelled-out success rates of GPT-3.5 and GPT-4 on each maze, where M/N means that M of N minigames were successfully played by this model. Since GPT-4 by default has a larger context window size (8K) than GPT-3.5 (4K), in order to make a fair comparison, we restrict the context window size to 4K for both GPTs.

D Future Directions

Our MANGO benchmark sets up a leaderboard, measuring the mapping and navigation abilities of LLMs. An interesting future direction is to investigate how the internal representations of each LLM have—if any—captured the structures of the maps. For example, one may probe the LLM representations and examine whether these representations are predictive of certain characteristics of the maze, similar to the experiments of Li et al. (2023) in board games. This can potentially reveal critical insights into the underlying mechanism of an LLM to represent spatial relations, benefiting their application into real-world scenarios.

The second interesting direction is to investigate how low-cost-adaptation can improve the performance of an LLM. For example, can an LLM, trained on a limited set of mazes, generalize its knowledge and reasoning capability to unseen mazes? Findings in this direction could enhance their usefulness in dynamic environments.

Another direction is to upgrade the MANGO benchmark by enriching its spatial and structural configurations on top of the current maps. These configurations include:

- spatial notations (e.g., distance in meters, area in square meters) such that one would need complex movements to achieve a target (e.g., not "north" but "north 3 meters");
- notions of facing directions and rotations such that one would need to turn to switch facing directions.

Such upgrades will enhance its practical application in more complex scenarios. They will also enable investigation in interesting topics, including

- LLMs' generalization to significantly out-of-distribution navigation scenarios (e.g., different structures, styles, and configurations);
- LLMs' robustness to distracting context.

GAMES	GPT-3.5		GPT-4	
	W/O MAPS	W/ MAPS	W/O MAPS	W/ MAPS
905	0/0	0/0	0/0	0/0
ADVENT	3/9	3/9	4/9	5/9
ADVENTURELAND	1/7	1/7	3/7	5/7
AFFLICTED	0/12	1/12	4/12	4/12
ANCHOR	0/2	0/2	2/2	2/2
AWAKEN	0/7	0/7	3/7	7/7
BALANCES	0/3	1/3	2/3	2/3
BALLYHOO	1/6	1/6	3/6	2/6
CURSES	2/7	3/7	2/7	6/7
CUTTHROAT	4/5	5/5	4/5	4/5
DEEPHOME	3/10	6/10	8/10	10/10
DETECTIVE	4/6	5/6	6/6	6/6
DRAGON	0/6	1/6	0/6	2/6
ENCHANTER	0/1	1/1	1/1	1/1
ENTER	0/0	0/0	0/0	0/0
GOLD	1/7	2/7	3/7	4/7
HHGG	0/0	0/0	0/0	0/0
HOLLYWOOD	2/6	2/6	4/6	5/6
HUNTDARK	0/0	0/0	0/0	0/0
INFIDEL	1/4	2/4	2/4	2/4
INHUMANE	1/2	2/2	2/2	2/2
JEWEL	0/10	1/10	4/10	7/10
KARN	0/8	0/8	4/8	5/8
LIBRARY	1/5	2/5	5/5	5/5
LOOSE	3/4	3/4	3/4	3/4
LOSTPIG	0/1	0/1	0/1	1/1
LUDICORP	1/13	2/13	9/13	10/13
LURKING	1/4	0/4	0/4	2/4
MOONLIT	0/2	2/2	2/2	2/2
MURDAC	1/2	1/2	2/2	2/2
NIGHT	5/16	8/16	9/16	13/16
OMNIQUEST	1/11	1/11	1/11	7/11
PARTYFOUL	0/2	1/2	1/2	1/2
PENTARI	4/13	3/13	4/13	8/13
PLANETFALL	3/6	4/6	3/6	5/6
PLUNDERED	0/0	0/0	0/0	0/0
REVERB	0/2	0/2	1/2	1/2
SEASTALKER	0/2	0/2	0/2	2/2
SHERLOCK	3/3	3/3	3/3	3/3
SNACKTIME	0/3	0/3	3/3	3/3
SORCERER	4/5	4/5	3/5	4/5
SPELLBRKR	1/1	1/1	1/1	1/1
SPIRIT	4/14	5/14	8/14	10/14
TEMPLE	0/2	1/2	1/2	2/2
TRINITY	0/0	0/0	0/0	0/0
TRYST205	0/3	1/3	0/3	1/3
WISHBRINGER	0/1	1/1	0/1	0/1
YOMOMMA	0/4	0/4	0/4	1/4
ZENON	1/5	3/5	5/5	5/5
ZORK1	2/18	3/18	10/18	13/18
ZORK2	4/9	5/9	7/9	8/9
ZORK3	0/6	1/6	0/6	4/6
ZTUU	0/9	0/9	7/9	7/9

Table 14: The experiment results (# successful / # answerable minigames) of each model.