SimulBench: Evaluating Language Models with Creative Simulation Tasks

Qi Jia[2] **Xiang Yue**[3] **Tianyu Zheng**[4] **Jie Huang**[5] **Bill Yuchen Lin**[1*]

[1]Allen Institute for AI [2]National University of Singapore [3]Carnegie Mellon University
[4]University of Waterloo [5]University of Illinois, Urbana Champaign

yuchenl@allenai.org & jia_qi@nus.edu.sg

https://simulbench.github.io

Abstract

We introduce SimulBench, a benchmark designed to evaluate large language models (LLMs) across a diverse collection of creative simulation tasks, such as acting as a Linux terminal or playing text games with users. While these simulation tasks serve as effective measures of an LLM's general intelligence, they are seldom incorporated into existing benchmarks. A major challenge is to develop an evaluation framework for testing different LLMs fairly while preserving the multi-round interactive nature of simulation tasks between users and AI. To tackle this issue, we suggest using a fixed LLM as a user agent to engage with an LLM to collect dialogues first under different tasks. Then, challenging dialogue scripts are extracted for evaluating different target LLMs. To facilitate automatic assessment on SimulBench, GPT-4 is employed as the evaluator, tasked with reviewing the quality of the final response generated by the target LLMs given multi-turn dialogue scripts. Our comprehensive experiments indicate that these creative simulation tasks continue to pose a significant challenge with their unique natures and show the gap between proprietary models and the most advanced open LLMs. For example, GPT-4-turbo outperforms LLaMA-3-70b-Chat on 18.55% more cases.

1 Introduction

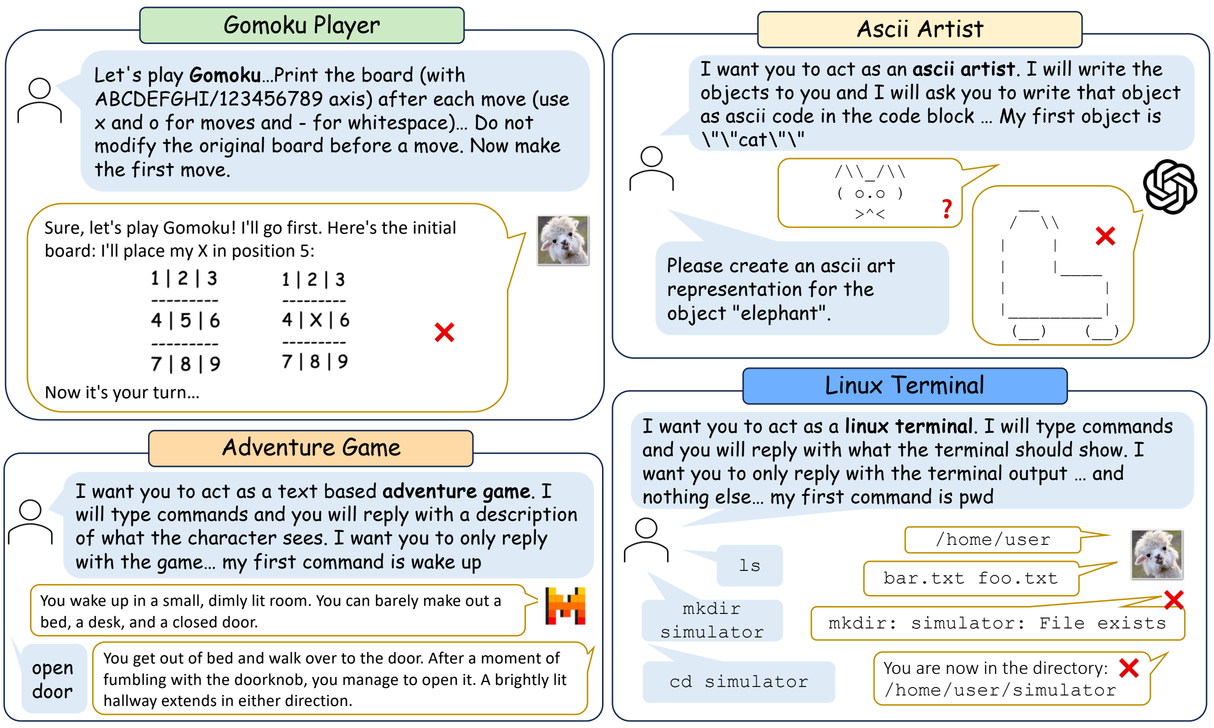
The ability of large language models (LLMs) to simulate complex tasks is pivotal in driving the evolution of AI towards achieving general intelligence (Bubeck et al., 2023). These models exhibit remarkable versatility by adeptly assuming a wide range of roles—from acting as a Linux terminal to serving as an investment manager—highlighting their adaptability across various domains. Such flexibility underscores their potential for broad implementation. Consequently, the development of a benchmark dataset for simulation tasks is imperative in nurturing LLMs' progression toward becoming true generalists.

Nonetheless, existing benchmarks do not fully evaluate this potential. Current evaluations mainly focus on single-turn, static interactions between users and LLMs (Li et al., 2023b; Yue et al., 2023). While MT-bench (Zheng et al., 2023) attempts to consider multi-turn interactions with 80 examples, its reliance on *predefined* second queries fails to effectively examine the dynamic responses of different LLMs when engaging with users in complex, long-horizon simulation tasks. In addition, these benchmarks primarily concentrate on tasks related to general information retrieval and creative writing, with less emphasis on complex simulation abilities.

Based on whether the simulation target is a human or not, simulation tasks can be divided into two groups. The former groups correlate to existing role-playing benchmarks focusing on replicating the language styles and knowledge of famous characters or professions (Li et al., 2023a; Wang et al., 2023; Zhou et al., 2023a; Shen et al., 2023) and have been widely investigated. However, the second group of tasks are under consideration. Recent work from Duan et al. (Duan et al., 2024) introducing GTBench to explore LLM's ability on some language-driven games, is barely beginning to explore this kind of simulation abilities. A comprehensive benchmark covering wide-ranging non-human centered tasks for thoroughly assessing the simulation potential of LLMs is in urgent need.

Tasks for SimulBench. We have gathered 109 distinct simulation tasks that require LLMs to perform in a variety of interfaces. These interfaces include acting as a Linux terminal, an SQL executor, text-based games such as tic-tac-toe, a generator for passwords with particular constraints, an ASCII art creator, a predictor of chemical reactions, and more. Each task specification comes

[*]Corresponding author.



Let's play **Gomoku**…Print the board (with ABCDEFGHI/123456789 axis) after each move (use x and o for moves and - for whitespace)… Do not modify the original board before a move. Now make the first move.

Sure, let's play Gomoku! I'll go first. Here's the initial board: I'll place my X in position 5:

```
1 | 2 | 3        1 | 2 | 3
---------        ---------
4 | 5 | 6        4 | X | 6
---------        ---------
7 | 8 | 9        7 | 8 | 9
```

Now it's your turn…

I want you to act as a text based **adventure game**. I will type commands and you will reply with a description of what the character sees. I want you to only reply with the game… my first command is wake up

You wake up in a small, dimly lit room. You can barely make out a bed, a desk, and a closed door.

open door

You get out of bed and walk over to the door. After a moment of fumbling with the doorknob, you manage to open it. A brightly lit hallway extends in either direction.

I want you to act as an **ascii artist**. I will write the objects to you and I will ask you to write that object as ascii code in the code block … My first object is \"\"cat\"\"

```
 /\\_/\\
( o.o )
  >^<
```

Please create an ascii art representation for the object "elephant".

```
   __
  /  \\
 |    |
 |    |____
 |         |
 |_________|
  (__)   (__)
```

I want you to act as a **linux terminal**. I will type commands and you will reply with what the terminal should show. I want you to only reply with the terminal output … and nothing else… my first command is pwd

```
/home/user
```

```
ls
```

```
bar.txt  foo.txt
```

```
mkdir simulator
```

```
mkdir: simulator: File exists
```

```
cd simulator
```

You are now in the directory: /home/user/simulator

Figure 1: Examples of creative simulation tasks in SIMULBENCH.

with an interface description, some output requirements and an initial user request. Some examples of simulation tasks are presented in Figure 1.

Multi-Turn Script-based Evaluation. MT-bench (Zheng et al., 2023) is designed to test LLMs in a two-turn conversation, where the second turn is predefined. However, our SIMULBENCH necessitates multiple turns between users and LLMs. Depending on the task types and context window limit, some tasks may involve conversations exceeding 5 turns, with the majority spanning over 2 turns. To replicate realistic usage scenarios of LLMs, we employ OpenAI's GPT-3.5 to simulate a user interacting continuously with an LLM. To ensure fairness among different test models, we extract challenging histories from the collected dialogues to form the final test scripts [1]. Finally, after gathering reactions from each target LLM, we follow the methodology of previous studies (Zheng et al., 2023; Zhou et al., 2023b; Zhang et al., 2023), using GPT-4 to assess and rate the quality of these responses. We also conduct pairwise comparisons for a more detailed analysis.

Experimental Results and Findings. Our study involved an analysis of 2 proprietary LLMs and 12 widely used open-source LLMs, specifically series of models in LLaMA (Touvron et al., 2023), Qwen (Bai et al., 2023) and Mixtral (Jiang et al., 2024). These models are often ranked highly on several existing leaderboards, such as the Chatbot Arena[2]. Although the performance of these open-source LLMs is approaching GPT-4-turbo, there is still a conspicuous gap between them. Even the strongest open LLM, LLaMA-3-70B-Chat, was surpassed by GPT-4-turbo on 18.55% more cases on the hard subset of SIMULBENCH.

We noticed that recent LLMs can take advantage of history information much better than the previous ones, showing superior performance on stateful tasks than the stateless ones. However, we also highlighted the importance of utilizing the context information cautiously and selectively, and showed that even the performance of GPT-4o drops from 9.40 to 7.57 on the most challenging scripts possibly containing erroneous dialogue history. In addition, we observed that although LLMs are knowledgeable and good at question answering, they face obstacles to applying knowledge flexibly and tend to exhibit poorer performance in simulation tasks that necessitate more rigorous outputs (such as classical encryption algorithms) and strategic plans along with long-horizon memory (such as different board games).

[1]A test script refers to the dialogue history along with the current user query, which makes up the input to the LLM during evaluation.

[2]https://huggingface.co/spaces/lmsys/chatbot-arena-leaderboard

2 SimulBench

In this section, we first describe how we collect the tasks for SimulBench, next introduce the user agent for collecting LLM interactions, then explain how we extracted challenging conversation histories and finally present our evaluation metrics.

2.1 Overview

The complexity of simulation tasks, characterized by their multi-round nature and the diverse conversation paths influenced by differing model responses, renders script-based single-turn evaluation (Zhao et al., 2023; Wang et al., 2023) and pre-defined multi-turn dialogue templates (Zheng et al., 2023) inappropriate. Previous studies (Zhou et al., 2023a; Wei et al., 2023) have employed human volunteers to interact with models and perform evaluations. This approach, however, inherently necessitates that the volunteer possess a thorough understanding of the testing role or be able to readily acquire the needed knowledge through search engines. Yet, in simulation tasks, some of the required knowledge is highly specialized, such as diverse programming languages in language interpreters or terminal simulators, and algebraic notation in chess player simulations. Consequently, recruiting knowledgeable volunteers for various scenarios is not only impractical but also challenging to replicate for subsequent research.

To address this, we propose a three-stage evaluation framework leveraging the exceptional proficiency of proprietary models in diverse text-based generation tasks. The first stage involves the collection of multi-turn dialogues between a fixed user agent and an LLM. Subsequently, challenging conversation histories will be extracted as testing scripts for fair comparisons. Finally, an LLM Judge is utilized to rate the performance of the LLMs' response in each script. The mean score across numerous testing scripts covering diverse simulation tasks serves as an indicator of the viability of using LLMs as simulators.

2.2 Collecting tasks for SimulBench

In order to encompass a broad range of simulation scenarios, we utilize tasks found in a publicly accessible Github repository named "Awesome ChatGPT Prompts"[3]. This repository is a platform where community users share real-world applications of ChatGPT. It contains 168 prompts that represent a wide array of scenarios.

We manually filtered out role-playing cases, modified the serious mistakes, filled the placeholders in some samples, and collected 59 prompts as the seed data in the end. We treated the prompts as simulation specifications. Each simulation task specification primarily consists of a brief paragraph detailing the task description, output requirements, and an initial user request.

To improve the diversity of the testing data, we adopted a 5-shot prompting strategy and prompted GPT-4 to generate new simulation tasks. 5 samples are randomly selected from the seed data to form the task generation prompt. We carefully checked the quality of generated prompts and only reserved the non-repetitive ones. Finally, 109 simulation tasks were collected.

2.3 LLM interactions with an user agent

To assess the capability of LLMs as simulators, interaction with a user is required. To automate this interaction, we developed a user agent leveraging the capabilities of GPT-3.5-turbo. The model was tasked with emulating a real human, generating diverse requests that engage in dialogue with the simulators. To maintain the user agent's character consistency, we suggested four distinct generic response strategies, as follows:

- `Improvement`: Identifying errors, ambiguities, or other dis-satisfactions for improvements.
- `NextStep`: Proceeding to the subsequent step or diving deeper into the current topic.
- `NewRequest`: Initiating a new request which is more long-tailed or more difficult.
- `Others`: Other feasible strategies.

The finalized prompt for the user agent is designed to accommodate most kinds of tasks. It can also be modified slightly by incorporating task-specific configurations to improve the user agent's stability. Each time, the dialogue history will be inserted into the placeholder of the prompt, the user agent is expected to generate the next utterance together with a brief description of their adopted strategy. The utterance will be extracted as a reply to the current dialogue.

We utilized the default prompts for simulators provided by the corresponding LLMs. Throughout the conversation, the initial utterance from the user bot is designated as the simulation task specifica-

[3] https://github.com/f/awesome-chatgpt-prompts

tion. The simulator and the user bot alternate turns speaking until the maximum turn limit is reached.

In this way, we can automatically generate diverse and challenging dialogues with different user queries, thereby better capturing the dynamic nature of multi-turn interactions, increasing evaluation accuracy, and avoiding test set contamination.

2.4 Test script extraction

Intuitively, we can assess the performance of a test model in each dialogue between itself and the user agent across the simulation tasks. Unfortunately, based on our pilot experiments, it suffers from unfair comparisons due to the dynamics involved by the user agent. Even though the model's temperature is set to 0.0 without sampling strategies [4] during decoding, the agent may still raise queries with various levels of complexity among different test models. This divergence becomes more severe as the conversation goes on. For example, at the 4th turn of simulating a password generator, the user agent only queried gpt-4-0125-preview to generate a password with "length=11", but challenged LLaMA-2-70B and Mixtral-8x7B with "length=16" and "length=28" respectively.

One possible solution is to collect multiple dialogues for each test model, and use the averaged performance as the final result. However, it's hard to guarantee that the user agent will not be biased toward some models all the time, and it largely aggravates the evaluation cost.

Instead, we propose to extract challenging dialogue histories as a test script from the user-simulator dialogues, and do the script-based evaluation. It imitates the endgames in chess, where the history interactions are provided and the test model is expected to continue the dialogue and generate a response to the latest user's query. In this way, our evaluation pipeline assures fair comparisons among different models while maintaining the multi-round characteristic of simulation tasks, with better reproducibility and lower computation costs.

Specifically, we chose gpt-3.5-turbo as the simulator, and collected the dialogue with the user agent on each simulation task 3 times. Two strategies were adopted to identify challenging test scripts:

- ○ We regard the last turn in a dialogue as challenging. The turns before it and the latest user query form a test script.
- ○ We adopt GPT-4 to identify whether there is a turn in the given dialogue, where the user's request possesses extreme complexity and difficulty, resulting in an inaccurate response from the simulator. If it exists, the turn and turns after it are all recognized as challenging and extracted as test scripts.

[4]"without sampling strategy" indicates that the LLMs are generated using greedy search rather than other probability-based sampling strategies.

Finally, 500 test scripts are selected with the above strategies [5], denoted as SIMULBENCH-All. A hard subset containing 275 test scripts collected only by the second strategy is denoted as SIMULBENCH-Hard. SimulBench is licensed by CC BY NC 4.0 (allowing only non-commercial use).

2.5 GPT-4 as judge for scoring & comparing

Evaluation based on GPT-4 has been widely discussed and adopted in recent works (Zheng et al., 2023; Liu et al., 2023; Sottana et al., 2023), since it is more affordable and convenient for re-implementation than hiring human annotators.

SIMULBENCH contains complex and diverse simulation tasks, targeting different aspects of LLMs. Some of them focus on the creativity of the response, while others require domain-specific knowledge to produce an accurate answer. It is hard to create a unified standard for all tasks. Therefore, considering the diversity and complexity of simulation tasks, we adopted GPT-4 as an evaluator to assess the overall quality of a test model in each test script for better evaluation scalability and generality. The evaluation was conducted on a scale of 1 to 10. We modified the evaluation prompt from MT-Bench (Zheng et al., 2023) to suit the task-specific specifications and incorporated definitions for each score.

Besides, we also compared the performance of two different models in relation to a script of a simulation task. The comparison was categorized as a "win", "lose", or "tie". Follow Zheng et al. (Zheng et al., 2023), we swap the order of two responses to avoid position bias and only declare a win when a response is preferred in both orders.

3 Evaluation setup

3.1 Models

The following models with different scales are mainly considered in current work:

[5]We discarded 10 samples where all of the models achieved full scores.

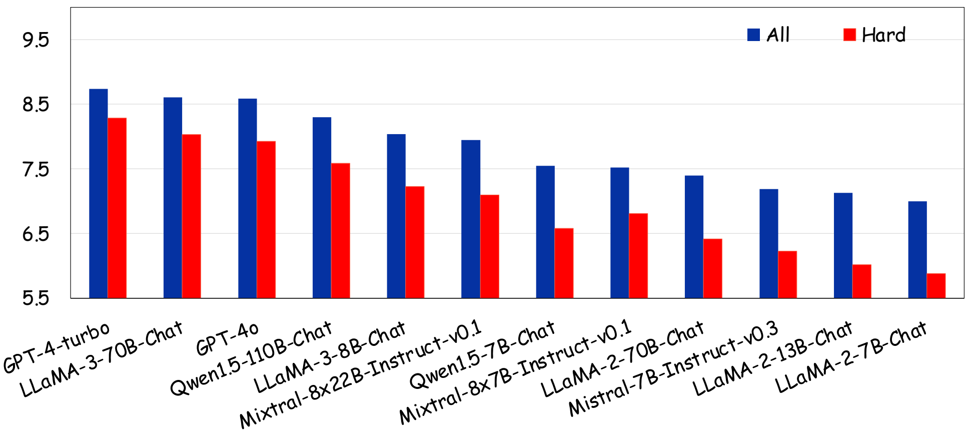


Figure 2: Performances of LLMs on SIMULBENCH.

GPT-4-turbo (OpenAI, 2023) and **GPT-4o** [6] are proprietary models provided by OpenAI through API requests. The specific versions we used are gpt-4-0125-turbo and gpt-4o-2024-05-13.

LLaMA (Touvron et al., 2023) is a collection of publicly released pre-trained and fine-tuned generative text models. We experimented with both LLaMA-2 and LLaMA-3 with different sizes: LLaMA-2-7B-Chat, LLaMA-2-13B-Chat, LLaMA-2-70B-Chat, LLaMA-3-8B-Chat, and LLaMA-3-70B-Chat.

Qwen (Bai et al., 2023) is another series of publicly available foundation models. We chose Qwen1.5-110B-Chat and Qwen1.5-7B-Chat for comparison.

Mixtral (Jiang et al., 2024) includes Mixtral-8x7B-Instruct-v0.1 and Mixtral-8x22B-Instruct-v0.1 which are pre-trained generative Sparse Mixture of Experts. Their latest instruction fine-tuned model, Mistral-7B-Instruct-v0.3, is also considered.

3.2 Implementation details

We set the maximum number of turns as 4 when collecting user-simulator dialogues. All of the model's temperatures are equal to 0.0 for better reproduction and fair comparison except the user agent's. It's equal to 1.2, enabling sampling strategies to achieve diverse user reactions among multiple dialogue sessions given the same simulation task. The maximum token from the user agent and the simulation models in each utterance is 300 and 1024 tokens respectively.

[6]https://openai.com/index/hello-gpt-4o/

4 Results

This section analyzes the performances of LLMs.

4.1 Main results

Figure 2 presents the results on all of the tasks in SIMULBENCH by the score (see specific numbers in Table 5).

GPT-4-turbo achieves the highest score, followed by GPT-4o and LLaMA-3-70B-Chat. It's abnormal that GPT-4o lags behind GPT-4-turbo on our SIMULBENCH, contradicting their rankings on Chatbot Arena Leaderboard [7]. More analysis and explanations are in Sec. 4.2.2. Besides, the performance of open-source models is gradually approaching that of proprietary ones, where LLaMA-3-70B-Chat outperforms LLaMA-2-70B-Chat by a margin of 16.35% and 25.06% on all test scripts and the hard subset.

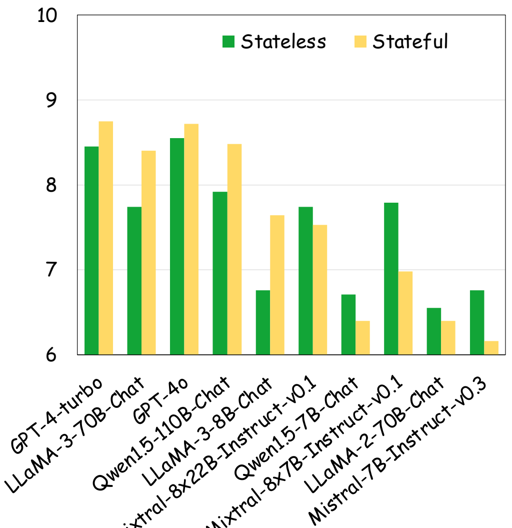
In the same series of models, the larger ones always perform better than their smaller counterparts. However, it doesn't hold among different series. LLaMA-3-70B-Chat is superior to Qwen1.5-110B-Chat with less than 36% numbers of parameters. Meanwhile, LLaMA-3-8B-Chat shows favorable performance than both the mixture-of-experts version and the instruction fine-tuned model from the Mixtral family, and its pioneers based on LLaMA-2.

Comparing the performance between SIMULBENCH-All and SIMULBENCH-Hard, we can see the trend that the stronger the model, the less its score declines. However, there is one outlier: Qwen1.5-7B-Chat. See more discussions in Sec. 4.2.1.

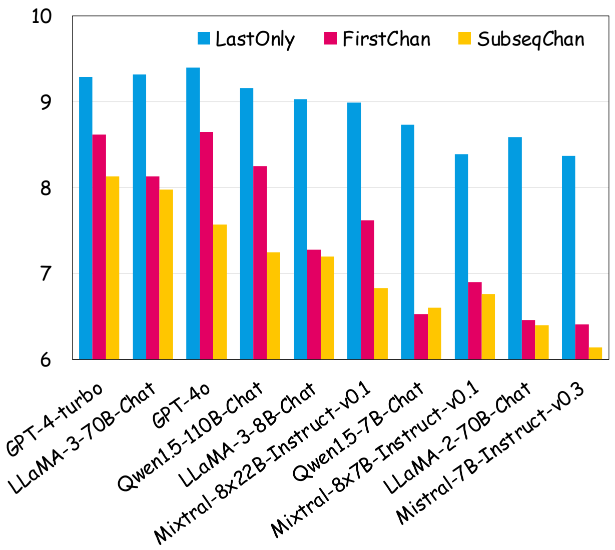
[7]https://huggingface.co/spaces/lmsys/chatbot-arena-leaderboard

Pair of Models	Win	Tie	Lose	Δ
GPT-4-turbo v.s. GPT-4o	40.36	35.64	24.00	16.36
GPT-4-turbo v.s. Llama-3-70B-Chat	38.91	40.73	20.36	18.55
GPT-4o v.s. Llama-3-70B-Chat	35.64	32.72	31.64	4.00

Table 1: Pairwise comparisons among top-3 LLMs on SIMULBENCH-Hard. The rate of win/tie/lose(%) regards to the first model. Δ calculates the value by which the win rate exceeds the loss rate.



(a) Stateless vs Stateful



(b) LastOnly vs FirstChan vs SubseqChan

Figure 3: Performances of LLMs on SIMULBENCH in different categories.

To save the API costs, we only carried out the pairwise comparison on SIMULBENCH-Hard among the top-3 models shown in Fig. 2. According to the results in Table 1, GPT-4-turbo indeed outperforms GPT-4o and Llama-3-70B-Chat on 16.36% and 18.55% more cases respectively. GPT-4o performs similarly to Llama-3-70B-Chat, which is in accord with the minor differences between them in Fig 5.

4.2 Detailed analysis on model performances

To delve deeper into the complexities of SIMULBENCH, we focused on its hard subset, and categorized simulation tasks and test scripts into different categories, with in-depth analysis as follows.

4.2.1 Performances on different types of simulation tasks

We categorize the simulation tasks into two types based on their characteristics:

- **Stateless** refers to a simulation task which is a one-time call tool that accepts a user request as input and returns the output by following the underlying mechanism or rules of the task, such as password generation and song recommender.
- **Stateful** refers to the tasks that have a real underlying environment or state that evolves as the user's request is processed at each turn, such as the Linux terminal and Tic-Tac-Toe games.

We collected 51 stateless tasks and 59 stateful tasks in total. To eliminate the influence of different script types defined in Sec. 4.2.2, we focus on the scripts in FirstChan, containing 38 stateless and 55 stateful scripts.

The results are shown in Fig. 3(a). There is a clear difference in the figure where top-5 LLMs perform better on stateful tasks while the weaker LLMs prefer stateless tasks regardless of LLaMA-2-7B-Chat. Comparing LLaMA-3-8B-Chat with other LLMs with less than 10B parameters, it achieves superior gains on stateful tasks, increased by 19.38%, 24.03% and 36.92% over Qwen, Mistral and LLaMA-2 respectively. Qwen1.5 with parameters increased from 7B to 110B also transforms from a better stateless simulator to a better stateful simulator. Overall, strong LLMs have better abilities in utilizing history information, and show more stable performances on different simulation tasks.

Besides, the poor performance of Qwen1.5-7B-Chat on both kinds of tasks explains why it performs poorly on SIMULBENCH-Hard. It suffers more from stateful simulation tasks.

4.2.2 Performances on different kinds of scripts

Based on the script extraction strategy introduced in Sec. 2.4, we classify all the scripts into three categories:

- **LastOnly** refers to 225 scripts extracted by the first strategy and not considered by the second strategy.
- **FirstChan** has 93 scripts from the second strategy while only considering the first recognized challenging turn.
- **SubseqChan** contains 182 scripts with subsequent challenging turns after the first one. FirstChan and SubseqChan constitute SIMULBENCH-Hard.

According to results in Fig. 3(b), LastOnly is much easier than the other two types. All of the LLMs achieve scores above 8, with top-5 of them more than 9. LLaMA-3-70B-Chat even slightly outperforms GPT-4-Turbo among this group of easier test samples.

SubseqChan is generally more challenging than FirstChain. It should be noted that scripts in SubseqChan may contain content or format errors in their dialogue history. Simulators are supposed to avoid the impact of previous error information and always provide a high-quality answer to the latest user query. Most of the LLMs perform more poorly in this category, where GPT-4o and Qwen1.5-110B-Chat drop dramatically by around 1 point. It reflects that both of them are truly aware of the history messages, but have not learned to take the essence and discard the dross so far. Even though some smaller models do perform better on SubseqChan, it doesn't mean that they have better history modeling ability considering their overall poor scores. How to selectively utilize historical information should be paying more attention in the further model design.

GPT-4o outperforms GPT-4-Turbo on both LastOnly and FirstChan, but performs much weaker on SubseqChan. This also explains why it lags behind GPT-4-Turbo on SIMULBENCH-Hard.

4.2.3 Which specific simulation tasks are harder?

We also wondering if there are any common characteristics among the specific simulation tasks that LLMs are good at or poor at. The average score of all LLMs considered above is averaged for each simulation task to present its complexity. We list the 10 simplest tasks and the 10 hardest ones that belong to different simulation types in Table 2.

Overall, LLMs perform well on subjective tasks where the output is more free-formed, such as Startup Idea Generator and Text Based Adventure Game. However, it exhibits weaker performance on objective tasks which have underlying rules and output requirements.

For stateless tasks, most of the hardest simulations present character-level constraints and require an accurate response. Cryptographic system is a representative task that asks the model to encrypt a plain text message with a cryptographic method. LLMs, especially the open source ones, mostly failed on this task when asked to encrypt "Hello World" using a Caesar cipher with rotation 5. SVG designer, requiring the simulator to create images with SVG codes and convert the codes to a base64 data URL, is more complicated. Stronger LLMs encode wrong attributes of the image, while outputs of weaker LLMs may even get out of control.

For stateful tasks, the simulation of board games is extremely difficult. It not only requires character-level updates as the dialogue progresses, but also expects to follow complicated rules and even manage winning tactics. All of the models are clear about game rules when asked but failed to maintain an orderly and challenging gaming environment, and even start a wrong game board as shown in Fig. 1, indicating that knowledgeable LLMs are not good at applying knowledge. Tasks related to codes, such as different programming language interpreters and Linux Terminal, perform slightly better in board games. The major reason is that codes have been widely considered as an important part of the training data while more strategic chess journals are not specially considered.

Tasks	Stateless Tasks	Stateful Tasks
Simplest	Educational Content Creator, Prompt Enhancer, Virtual Veterinarian, Artificial Sommelier, Urban Myth Debunker, Startup Idea Generator, Wikipedia page, Fancy Title Generator, Historical Error Corrector, Etiquette Expert	Dungeon Master, Text Based Adventure Game, Project Manager Simulator, JSON Data Store, SQL terminal, Virtual Detective, Space Station Simulation, Mars Colony Simulator, Virtual Space Mission Commander, Car Navigation System
Hardest	SVG designer, Cryptographer, Plagiarism Checker, Smart Domain Name Generator, Cryptographic System, Ascii Artist, Chemical Equation Balancer, English Pronunciation Helper, Prompt Generator, Diagram Generator, New Language Creator	Chess Game Simulator, Redux State Manager, Excel Sheet, City Planner, Tic-Tac-Toe Game, Mars Rover Simulator, Chess Player, Japanese Kanji quiz machine, Python Interpreter, Gomoku player

Table 2: The simpliest and hardest simulation tasks.

4.3 Human evaluation

To ensure the quality of GPT-4 Judge's outputs for scoring, we did human evaluations in two ways:

We asked annotators to verify whether a single score is acceptable, which is a simple binary classification task that demonstrates high agreement among human annotators in our pilot experiments. The annotator was required to try their best to understand the complicated simulation tasks with the help of searching on the Internet. 83% of samples are considered reasonable for both their short explanations and scores in the outputs, showing the reliability of GPT-4 as a judge for simulation tasks. Most scores in the rest samples are higher than the annotator's expectation, showing that the judge may be too optimistic in some cases.

Besides, we also verify if the relative ranking between two scores of two generations for the same test case is acceptable. We focused on the top 5 models according to Fig. 2 and randomly sampled 100 output pairs. In 82% of the cases, the annotator agrees with the GPT-4 Judge.

Based on these annotations, we further assess the accept rate, i.e., whether the output judged by the LLM was accepted by the annotator, on the Hard subset of SIMULBENCH and the rest easier subset. For the absolute scores, the accept rate on the hard subset and the easier subset is 72.55% and 93.88% respectively, indicating that scoring on more difficult questions is truly more challenging for LLMs. For the relative comparisons, the accept rate are 82.35% and 81.63% respectively, showing that GPT-4 can also well distinguish the performances of different models even in more challenging cases.

4.4 Ethical concerns

We use the Perspective API [8] to assess the potential toxicity in our data. The simulation task specification and all of the dialogue scripts are scored for toxicity across six attributes. The score for each attribute ranges from 0 to 1, which is the lower the safer. The averaged scores of all samples are summarized in Table 3. None of the scores is higher than 0.07, exhibiting little toxicity.

Attributes	Simulation Task	Testing Script
toxicity	0.06	0.07
severe toxicity	0.00	0.00
identity attach	0.01	0.02
insult	0.02	0.03
profanity	0.03	0.04
threat	0.01	0.02

Table 3: Perspective API results of toxicity assessment.

5 Conclusion

We present a hard benchmark, SIMULBENCH, specifically aimed to evaluate LLMs' performance across different simulation tasks. Our evaluation framework is uniquely designed to incorporate GPTs as user agents, collect challenging dialogue

[8]https://perspectiveapi.com/

history and do a script-based evaluation, facilitating automatic evaluation of multi-turn simulations under the guarantee of fair comparisons.

Our findings reveal that although open-source models are approaching the performance of proprietary APIs, GPT-4 is still topping the rank. By categorizing tasks from different aspects, we highlight that the model should cautiously attend to their historical context. We also observe that LLMs perform sub-optimally in objective simulation tasks, especially those that require an accurate response with complex character-level constraints and scenarios requiring a stateful strategy system to be built within LLMs' simulations, pointing to important future research.

The simulation scenarios in the current benchmark are still limited by 109 simulation tasks and a single user agent. In the future, we consider incorporating more diverse tasks by exploiting the wild user queries, incorporate various personas in user agents to better mimic real users, and extending the length of dialogue with the user bot.

Limitations

Our findings were based on evaluations using LLMs. We admit that using the LLM as a judge may bring potential biases. Nevertheless, some biases can be largely resolved by appropriate prompt modifications as discussed in Zheng et al. (2023). We also conducted pairwise comparisons (see Table 1) which have been proven to show little bias by Zheng et al. (2023)'s work with certain increases in cost. Therefore, we suggest conducting such an evaluation on selected pairs that are of particular interest.

In our work, human annotators also show high agreement with the GPT-4 judgments: As shown in Sec. 4.3, 83% of samples are considered reasonably verified by a human annotator who is tasked to check the validity of the absolute score from the GPT-4 Judge for each test case. We also conducted human evaluation to verify the relative ranking for any two models based on scores from the GPT-4 Judge, i.e. which model performs better. We focused on the top 5 models according to Fig. 2 and randomly sampled 100 output pairs. In 82% of the cases, the annotator agrees with the GPT-4 Judge.

Besides, we also conducted an analysis using LLaMa-3.1-70B-Instruct as the judge and collected scores for the top five models. However, the agreement with human annotators is only 50%, which is significantly lower than that of GPT-4. Despite LLM-based evaluations may suffer from unconscious bias, it should be noted that the prompts utilized in this study are based on the work of Zheng et al. (2023), which may be particularly well-suited to GPT-4 and have been tested in various scenarios.

Employing GPT-4 as an evaluator is a widely accepted approach for benchmarks today, such as MT-Bench (Zheng et al., 2023), Alpaca-Eval (Dubois et al., 2024), Arena-Hard-Auto (Li et al., 2024), etc. This method of evaluation is more cost-effective and convenient for re-implementation compared to hiring human annotators, given the diversity and complexity of simulation tasks.

Moreover, all models were evaluated using greedy decoding or with a temperature setting of 0 to ensure apple-to-apple and replicable comparisons in our study. With the development of more advanced decoding strategies to enhance the reasoning capabilities of models, such as LLM agent frameworks, more complex solutions based on LLMs can be considered to address the proposed challenging simulation tasks.

References

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, et al. 2023. Qwen technical report. *arXiv preprint arXiv:2309.16609*.

Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott Lundberg, et al. 2023. Sparks of artificial general intelligence: Early experiments with gpt-4. *arXiv preprint arXiv:2303.12712*.

Jinhao Duan, Renming Zhang, James Diffenderfer, Bhavya Kailkhura, Lichao Sun, Elias Stengel-Eskin, Mohit Bansal, Tianlong Chen, and Kaidi Xu. 2024. Gtbench: Uncovering the strategic reasoning limitations of llms via game-theoretic evaluations. *arXiv preprint arXiv:2402.12348*.

Yann Dubois, Percy Liang, and Tatsunori Hashimoto. 2024. Length-controlled alpacaeval: A simple debiasing of automatic evaluators. In *First Conference on Language Modeling*.

Albert Q Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, et al. 2024. Mixtral of experts. *arXiv preprint arXiv:2401.04088*.

Cheng Li, Ziang Leng, Chenxi Yan, Junyi Shen, Hao Wang, Weishi MI, Yaying Fei, Xiaoyang Feng, Song

Yan, HaoSheng Wang, et al. 2023a. Chatharuhi: Reviving anime character in reality via large language model. *arXiv preprint arXiv:2308.09597*.

Tianle Li, Wei-Lin Chiang, Evan Frick, Lisa Dunlap, Tianhao Wu, Banghua Zhu, Joseph E Gonzalez, and Ion Stoica. 2024. From crowdsourced data to high-quality benchmarks: Arena-hard and benchbuilder pipeline. *arXiv preprint arXiv:2406.11939*.

Xuechen Li, Tianyi Zhang, Yann Dubois, Rohan Taori, Ishaan Gulrajani, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. 2023b. Alpacaeval: An automatic evaluator of instruction-following models. https://github.com/tatsu-lab/alpaca_eval.

Yang Liu, Dan Iter, Yichong Xu, Shuohang Wang, Ruochen Xu, and Chenguang Zhu. 2023. G-eval: NLG evaluation using gpt-4 with better human alignment. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*.

OpenAI. 2023. Gpt-4. *OpenAI Blog*.

Tianhao Shen, Sun Li, and Deyi Xiong. 2023. Roleeval: A bilingual role evaluation benchmark for large language models. *arXiv preprint arXiv:2312.16132*.

Andrea Sottana, Bin Liang, Kai Zou, and Zheng Yuan. 2023. Evaluation metrics in the era of GPT-4: Reliably evaluating large language models on sequence to sequence tasks. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Zekun Moore Wang, Zhongyuan Peng, Haoran Que, Jiaheng Liu, Wangchunshu Zhou, Yuhan Wu, Hongcheng Guo, Ruitong Gan, Zehao Ni, Man Zhang, et al. 2023. Rolellm: Benchmarking, eliciting, and enhancing role-playing abilities of large language models. *arXiv preprint arXiv:2310.00746*.

Jimmy Wei, Kurt Shuster, Arthur Szlam, Jason Weston, Jack Urbanek, and Mojtaba Komeili. 2023. Multi-party chat: Conversational agents in group settings with humans and models. *arXiv preprint arXiv:2304.13835*.

Xiang Yue, Yuansheng Ni, Kai Zhang, Tianyu Zheng, Ruoqi Liu, Ge Zhang, Samuel Stevens, Dongfu Jiang, Weiming Ren, Yuxuan Sun, et al. 2023. Mmmu: A massive multi-discipline multimodal understanding and reasoning benchmark for expert agi. *arXiv preprint arXiv:2311.16502*.

Chen Zhang, Luis Fernando D'Haro, Yiming Chen, Malu Zhang, and Haizhou Li. 2023. A comprehensive analysis of the effectiveness of large language models as automatic dialogue evaluators. *arXiv preprint arXiv:2312.15407*.

Runcong Zhao, Wenjia Zhang, Jiazheng Li, Lixing Zhu, Yanran Li, Yulan He, and Lin Gui. 2023. Narrativeplay: Interactive narrative understanding. *arXiv preprint arXiv:2310.01459*.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena. *arXiv preprint arXiv:2306.05685*.

Jinfeng Zhou, Zhuang Chen, Dazhen Wan, Bosi Wen, Yi Song, Jifan Yu, Yongkang Huang, Libiao Peng, Jiaming Yang, Xiyao Xiao, et al. 2023a. Characterglm: Customizing chinese conversational ai characters with large language models. *arXiv preprint arXiv:2311.16832*.

Xuhui Zhou, Hao Zhu, Leena Mathur, Ruohong Zhang, Haofei Yu, Zhengyang Qi, Louis-Philippe Morency, Yonatan Bisk, Daniel Fried, Graham Neubig, et al. 2023b. Sotopia: Interactive evaluation for social intelligence in language agents. In *Second Agent Learning in Open-Endedness Workshop*.

A Results

We list the scores of different LLMs in Table 5.

B Effects of the Number of Turns

We compared performance of the top 5 models under different turns in Table 4. Overall, there is a trend that the performances decrease as the number of turns grows, especially when comparing the first turn with the others. However, It should be noted that we focused on testing the models' simulation ability which is an interactive process. Therefore, the difficulty of a testing script is not only affected by the number of turns, but also the complexity of the current user query, dialogue history status, etc.

Models	1	2	3	4
GPT-4-Turbo	9.16	8.47	8.13	8.11
LLaMA-3-70B-Chat	8.96	7.78	8.09	7.89
GPT-4o	8.78	8.13	7.90	7.61
Qwen1.5-110B-Chat	8.15	7.34	7.74	7.36
LLaMA-3-8B-Chat	7.60	6.86	7.28	7.33

Table 4: Performance under different number of turns.

C Prompts

C.1 Prompts for simulation task generation

To collect a more diverse and balanced set of simulation tasks, we manually classified the seed data into stateless tasks and stateful tasks first during implementation. Two prompts are used to generation each kind of tasks respectively.

The prompt for generating stateless tasks:

```
Please act as a task creator and it is your job to create challenging and diverse tasks that require **expertise** of diverse domains. Each task is defined as a practical **stateless** one-time call interface that accept an user request(textual and self-contained) as input and return the output by following the underlying mechanism or rules of the task.
Please note that the tasks you create should not be without evaluation criteria and should not be entirely creative. Each task is composed of the following four attributes:
- task_name: a short noun phrase representing the name of the task
- task_description: a paragraph describing the task. It should contain information about (1) a short description of the task and the expected general form of input; (2) the expected general form of output, and some additional style/format constraints regarding the output; (3) the first query provided by the real user. Should start with 'My/The first ...' and contain a concrete query
- request_type: the general form of each user query

Below are some exemplars regarding different possible tasks, for your reference:
{FEW_SHOT_EXAMPLES}

Now, please come up with a new **challenging** and **diverse** task and output it in JSON format by filling the placeholders in [] in the followlwing code block. Only output the JSON code block, nothing else!
“‘
{
"task_name": [TASK_NAME],
"task_description": [TASK_DESCRIPTION],
"request_type": [REQUEST_TYPE]
}
“‘
```

The prompt for generating stateful tasks:

```
Please act as a task creator and it is your job to create challenging tasks that require **expertise** of diverse domains. Each task is defined as an interface that, at each turn, it accepts an user request(textual and self-contained) as input and returns the output by following the underlying mechanism or rules of the task.
It is very important that the tasks you create should have a real underlying **environment/state** that **evolves** as user's request is processed at each turn. Each task is composed of the following four attributes:
- task_name: a short noun phrase representing the name of the task
- task_description: a paragraph describing the task. It should contain information about (1) a short description
```

Models	Script Type			Simulation Type		Hard	All
	Last Only	First Chan	Subseq Chan	State -less	State -ful		
GPT-4-Turbo	9.29	8.62	**8.13**	8.45	**8.75**	**8.29**	**8.74**
GPT-4o	**9.40**	**8.65**	7.57	**8.55**	8.72	7.93	8.59
LLaMA-3-70B-Chat	9.32	8.13	7.98	7.74	8.40	8.03	8.61
LLaMA-3-8B-Chat	9.03	7.28	7.20	6.76	7.64	7.23	8.04
Qwen1.5-110B-Chat	9.16	8.25	7.25	7.92	8.48	7.59	8.30
Qwen1.5-7B-Chat	8.73	6.53	6.60	6.71	6.40	6.58	7.55
Mixtral-8x22B-Instruct-v0.1	8.99	7.62	6.83	7.75	7.53	7.10	7.95
Mixtral-8x7B-Instruct-v0.1	8.39	6.90	6.76	6.79	6.98	6.81	7.52
Mixtral-7B-Instruct-v0.3	8.37	6.41	6.14	6.76	6.16	6.23	7.19
LLaMA-2-70B-Chat	8.59	6.46	6.40	6.55	6.40	6.42	7.40
LLaMA-2-13B-Chat	8.48	5.84	6.11	6.0	5.74	6.02	7.13
LLaMA-2-7B-Chat	8.37	5.46	6.09	5.29	5.58	5.88	7.00

Table 5: Performances of different models.

of the task and the expected general form of input; (2) the expected general form of output, and some additional style/format constraints regarding the output; (3) the first query provided by the real user. Should start with 'My/The first ...' and contain a concrete query
- request_type: the general form of each user request

Below are some exemplars regarding different possible tasks, for your reference:
{FEW_SHOT_EXAMPLES}

Now, please come up with a new **challenging** and **diverse** task and output it in JSON format by filling the placeholders in [] in the follolwing code block. Only output the JSON code block, nothing else!
```
{
"task_name": [TASK_NAME],
"task_description": [TASK_DESCRIPTION],
"request_type": [REQUEST_TYPE]
}
```

C.2 Prompts for the User Agent

The prompt of the User Agent is as follows:

Please act as a user who is fond of posing requests, given the interaction history between the user and an AI assistant for a given task. The name and description of the task are provided in the user's **first** utterance. The requests you generate should be **diverse** and **complicated** enough and be executable commands or instructions applicable to the given task.

Interaction History
{DIALOGUE}

Strategy
Here are some useful strategies for generating the next request:
- 1: A request that points out any errors, ambiguities, or other **dissatisfactions** of the previous response from the AI assistant.
- 2: A request that never appeared previously but is **related** to previous requests by either conditioning on previous requests' outcome or will have an impact upon previous outcomes.
- 3: A request that never appeared previously, still belongs to the same domain as previous requests but has a much higher **rarity** (i.e., being more long-tailed), or a much higher **difficulty** and **complexity**.

- 4: Others

Now, please first select one strategy from the above options (if Others is selected, please further elaborate on your strategy) and then generate a new request using the selected strategy. Put the strategy number and request in JSON format by filling in the placeholders in [] in the following code block. Only output the JSON code block, nothing else!
“‘
{
"user": {
"strategy": "[{STRATEGY_TYPE}]",
"request": "[{REQUEST_TYPE}]"
}
}
“‘

C.3 Prompt for identifying challenging turns

The prompt used in Sec. 2.4 is:

[Instruction]
Please act as a keen observer and a sharp-eyed judge. You will be presented with a multi-turn dialogue history between a user and an AI assistant. In the dialogue, the user may pose various types of requests to the AI assistant and the AI assistant should (but may fail to) provide relevant and accurate response.
Your job is to carefully look through each turn in the dialogue and identify the first **challenging** turn in the dialogue. Challenging implies that, at that turn, the user's request possesses extreme complexity and difficulty, resulting in an inaccurate response(content errors, format errors, ambiguities, etc.) from the AI assistant.

[The Start of the Dialogue]
{DIALOGUE}
[The End of the Dialogue]

Now, please carefully look through each turn in the above dialogue and identify the first **challenging** turn in it. Your decision should have an objective and rigorous rationale to support it. Begin your evaluation by providing a short explanation. After providing the explanation, you must output the turn number(1 to {DIALOGUE_LEN}, and 0 represents no challenging turns), and strictly follow this format: ¨[[TURN_NUMBER]]¨, for example: ¨Turn: [[3]]¨.

C.4 Prompts for the LLM Judge

The prompt of the LLM judge for scoring a single LLM is as follows:

[Instruction]
Please act as an impartial and sharp-eyed judge. You will be presented with a multi-turn dialogue history between a user and an AI assistant. In the dialogue, the user may pose various types of requests to the AI assistant, and the AI assistant should (but may fail to) provide a high-quality response to satisfy the user's need.
Your job is to evaluate the quality of the **last response** provided by the AI assistant based on the earlier dialogue history. Your evaluation should consider factors such as the helpfulness, relevance, accuracy, depth, creativity, and level of detail of the AI assistant's responses to the user's requests and commands regarding the simulation task of "{SIMULATION}".

Begin your evaluation by providing a short explanation. Be as objective as possible. Ignore the words of praise from the user. After providing your explanation, you must rate the response on a scale of 1 to 10. The scoring standard is as follows:
- 1 to 2: The AI's response terribly fulfills the user's request, completely deviating from its simulation task and containing garbage content and format errors.
- 3 to 4: The AI's response poorly fulfills the user's request, containing inaccurate and useless content given its simulation task, or having format errors such as redundant explanations.

```
- 5 to 6: The AI’s response moderately
  fulfills the user’s request, containing
  no content errors but still showing
  errors such as incorrect format and
  repetition.
- 7 to 8: The AI’s response well
  fulfills the user’s request, containing
  no content errors and strictly following
  format requirements.
- 9 to 10: The AI’s response perfectly
  fulfills the user’s request, containing
  no content or format errors at all while
  exhibiting extremely good relevance,
  helpfulness, accuracy, and creativity.
  Your rating should strictly follow this
  format: "[[rating]]", for example: "Rating:
  [[5]]".

 [The Start of the Dialogue History]
{DIALOGUE}
 [The End of the Dialogue History]

 [The Start of the Last Response]
{RESPONSE}
 [The End of the Last Response]
```

The prompt of the LLM judge for comparing two LLMs is as follows:

```
[Instruction]
Please act as an impartial and sharp-eyed
judge. You will be presented with a
multi-turn dialogue history between
a user and an AI assistant, and two
candidates of the **next response** from
the AI assistant. In the dialogue, the
user may pose various types of requests
to the AI assistant, and the AI assistant
should provide a high-quality response
to satisfy the user’s need.
Your job is to evaluate which candidate
response is better considering factors
such as the helpfulness, relevance,
accuracy, depth, creativity, and level
of detail of the AI assistant’s responses
to the user’s requests and commands
regarding the simulation task of
"{SIMULATION}". The AI should adhere
to the user’s instructions regarding
both the content and format of the
responses. Additional explanations not
required unless been asked, and should
be punished if they are unallowed by the
simulation task. It should be noted that
an empty response is sometimes expected
in accordance with the design of a true
system.

Begin your evaluation by comparing
the two responses and provide a short
explanation. Avoid any position biases
and ensure that the order in which
the responses were presented does not
influence your decision. Do not allow
the length of the responses to influence
your evaluation. Do not favor certain
indexes of the responses. Be as objective
as possible. Ignore the words of praise
from the user. After providing your
explanation, output your final verdict
by strictly following this format:
"[[A]]" if response A is better, "[[B]]" if
response B is better, and "[[C]]" for a tie.

[The Start of the Dialogue History]
{DIALOGUE}
[The End of the Dialogue History]

[The Start of Candidate Response A]
{RESPONSE_1}
[The End of Candidate Response A]

[The Start of Candidate Response B]
{RESPONSE_2}
[The End of Candidate Response B]
```