AgentSims: An Open-Source Sandbox for Large Language Model Evaluation

Jiaju Lin[1,2], **Haoran Zhao**[1,3 *], **Aochi Zhang**[1], **Yiting Wu**[1,4],
Huqiuyue Ping[1,5], **Qin Chen**[6]
[1]PTA Studio
[2] Pennsylvania State University, [3] Beihang University,
[4] Sun Yat-sen University, [5]Zhejiang University, [6]East China Normal University
[3]zhaohaoran@buaa.edu.cn
[2]jjlin.unfake@gmail.com and [6]qchen@cs.ecnu.edu.cn

Abstract

With ChatGPT-like large language models (LLM) prevailing in the community, how to evaluate the ability of LLMs is an open question. Existing evaluation methods suffer from following shortcomings: (1) constrained evaluation abilities, (2) vulnerable benchmarks, (3) unobjective metrics. We suggest that task-based evaluation, where LLM agents complete tasks in a simulated environment, is a one-for-all solution to solve above problems. We present AgentSims, an easy-to-use infrastructure for researchers from all disciplines to test the specific capacities they are interested in. Researchers can build their evaluation tasks by adding agents and buildings on an interactive GUI or deploy and test new support mechanisms, i.e. memory, planning and tool-use systems, by a few lines of codes. Our demo is available at `https://agentsims.com` .

1 Introduction

LLMs have revolutionized Natural Language Processing (NLP) and beyond. They demonstrate great potential in few-shot learning(Brown et al., 2020), code generation(Nijkamp et al., 2023), reasoning(Yao et al., 2023) and other tasks. Furthermore, LLM powered autonomous agents(Weng, 2023) are widely applied in solving complex problems, like multimodal generation(Shen et al., 2023), software developing(Qian et al., 2023) and social simulating (Park et al., 2023).

Although LLMs have reformed the paradigm of NLP, the problem of evaluation keeps haunting this field. Old benchmarks become out-of-date. Since LLMs achieve human-level Natural Language Understanding (NLU) and Natural Language Generation (NLG) abilities(OpenAI, 2023). To address the pressing need for novel benchmarks, the NLP community has introduced an array of fresh evaluation tasks and datasets, encompassing a diverse spectrum of abilities, including close-book question-answering (QA) based knowledge testing(Hendrycks et al., 2020; Huang et al., 2023), human-centric standardized exams(Zhong et al., 2023), multi-turn dialogue(Lin and Chen, 2023), reasoning(Liu et al., 2023a; bench authors, 2023) and safety assessment(Sun et al., 2023).

However, there are still many problems with these new benchmarks. 1) Evaluated abilities are limited by the task formats. Since a majority of these tasks adopt a single-turn QA format, they are insufficient to comprehensively evaluate various aspects of LLMs' capabilities. For instance, they fail to assess the models' proficiency in adhering to instructions in dialogue or mimicking human-like social interactions. 2) Benchmarks can be easily hacked. Avoiding the leakage of test set is of paramount importance when evaluate a model's ability. Nonetheless, considering the amount of pretrained knowledge of LLM, it has become more and more inevitable to inadvertently mix test cases into the training set.(Gunasekar et al., 2023). 3) For open-ended QA, existing metrics are not objective. Previous metrics for open-ended QA involve automatic metrics, and human-rating as subjective metrics(Zhou et al., 2023). In the LLM era, text segment matching based metrics become out-of-date. To mitigate the high-costly issue of human-rating, today's researchers employ well-aligned LLMs like GPT4 as automatic raters. Nevertheless, the most significant problem of this approach is that it can not evaluate super GPT4-level models, and LLMs are biased toward specific features (Wang et al., 2023b).

Based on these observations, we suggest task-based evaluation for LLM benchmarks. Specifically, given an artificial social-economic environment, LLM-driven agents should achieve the predefined task goals to prove their abilities, just like humans accomplishing goals in real world or games to show their capacities. Task-based evaluation is

[*] Corresponding author.

a one-for-all solution for current issues: 1) Task-based evaluation can test an LLM's overall ability. The complexity of social simulation and adaptation far exceeds simple QA and can formulate more challenging tasks for LLMs. LLM agents need to be equipped with the ability from NLU to Theory of Mind (ToM) (Premack and Woodruff, 1978). 2) Task solving processes are less likely to be hacked. Different from unchanged test datasets whose formats can be easily mimicked and added to training data. Task settings are diversified and the emergent social behaviors and groups are less likely to be described and included in training corpus. 3) Task passing rate is an objective metric. Compared with popular rating methods by ChatGPT, the passing rate does not rely on any black-box rating process, i.e. deep neural networks or human brains, thus it is an objective and fair metric for the comparison between LLMs.

To all-around estimate LLMs' capacities, we hope researchers from all fields take part in the development of evaluation tasks. However, a key obstacle to fostering a collaborative research community is the absence of a standard paradigm, an easy-to-use and extensible research platform. Previous works pursue the most efficient way to implement a sandbox while ignoring the need of non-specialist users. Besides, the poor readability further results in poor extensiblity and user churn. Moreover, the agents' performance varies with different support systems, i.e. memory, planning and tool-use system. We need a standard implementation to ensure the reproducibility of experimental results.

To this end, we introduce AgentSims, an interactive, visualized, and program-based infrastructure for curating evaluation tasks for LLMs. It creates an artificial town with various buildings and residents. The core objective of AgentSims is to streamline the task design process, eliminating hurdles that researchers from various backgrounds and programming proficiencies might encounter.

- For researchers focusing on LLM, AgentSims is **extendable and combinable** to allow users to combine different plan, memory and learning systems to study the impacts and effectiveness of various system design.

- For experts from other fields like behavioral economics or social psychology, AgentSims provides **an interactive UI** for map design and agent creation and lower the entry threshold. Such a user-friendly architecture further facilitates the cooperation between different fields and the future prosperity of the LLM community.

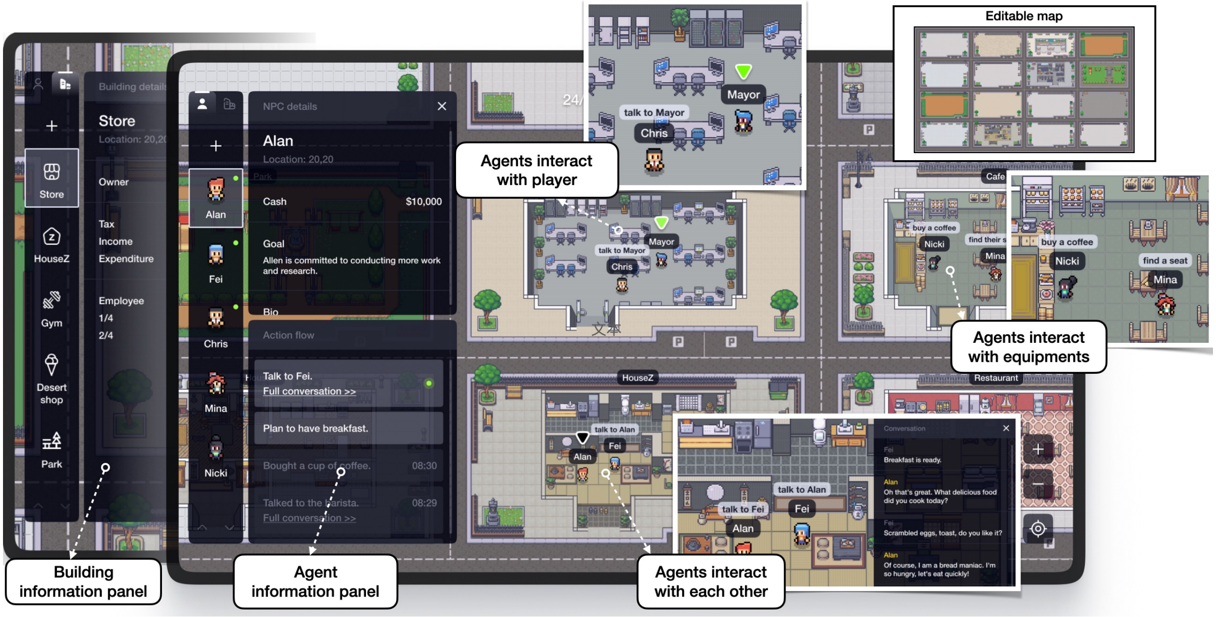
2 Related Work

2.1 Benchmarks for Large Language Models

The emergency of ChatGPT and other LLMs requires new benchmarks for effective evaluation. bench authors (2023) is the most accepted benchmark to evaluate LLM's general abilities. It contains more than 200 tasks, covering from childhood development, to social bias. Zhong et al. (2023) collect test tasks from human-centric standardized exams like GRE and SAT. (Hendrycks et al., 2020; Huang et al., 2023) are benchmarks focusing on measuring knowledge acquired in pretraining. They covers subjects across STEM, the humanities, the social sciences. Lin and Chen (2023) build a benchmark for LLMs' multiturn dialogue abilities. Every dialogue is limited to two turns for simplicity. Sun et al. (2023) focus on measure the safety of LLMs. They curate a adversarial attack dataset containing insulting instructions and test whether LLMs can be jailbroke. However, as mentioned above, existing datasets have issues that can not fully demonstrate abilities of LLMs. AgentSims overcomes these difficulties and renders a chance for overall evaluation of LLMs.

2.2 Multi Agent Cooperation

With LLMs demonstrate their overwhelming abilities, researchers find that multi LLM agents can generate better results than a single one. Nair et al. (2023) is one of the earliest attempts of multi-agent cooperation. It builds a forum for agents to communicate feedback and iteratively improve their healthcare suggestions. Li et al. (2023) expand the application field of agent cooperation method by role-playing. From programming to domain-specific QA, it surpass single agent baselines. Qian et al. (2023) build a software development company, by meticulously dividing the development process into four distinct stages, leading to efficient resolution of specific subtasks. Liu et al. (2023b) first apply multi-agent simulated society for alignment, where agents in a sandbox learn from social interaction to understand moral rules. (Park et al., 2023) is the most sophisticated application of multi agent sandbox. Authors build support mechanisms to enable agents to produce believable individual and emergent social behaviors. However, none existing methods provide a user-friendly interface



Building details

Store

Location: 20,20

Owner

Tax

Income

Expenditure

Employee

1/4

2/4

NPC details

Alan

Location: 20,20

Cash $10,000

Goal

Allen is committed to conducting more work and research.

Bio

Action flow

Talk to Fei.

Full conversation >>

Plan to have breakfast.

Bought a cup of coffee. 08:30

Talked to the barista. 08:29

Full conversation >>

Conversation

Fei

Breakfast is ready.

Alan

Oh that's great. What delicious food did you cook today?

Fei

Scrambled eggs, toast, do you like it?

Alan

Of course, I am a bread maniac. I'm so hungry, let's eat quickly!

Figure 1: Front end of AgentSims, showing in a pixel game style. Users can create agents and buildings in the left-side panel and observe agents behaviors in the main screen. Besides setting-then-observing, users can also play as the mayor and talk with agents to intervene the experiment.

for unprofessional researchers or build a standard paradigm for agent support system. Nonetheless, current multi-agent systems are task-oriented rather than evaluation-oriented. AgentSims works as a platform for easy benchmark construction.

3 Key Components

As shown in Figure 2, key components of AgentSims can be divided into two parts: 1) generative agents driven by LLM support mechanisms. 2) buidlings and equipment that consist the sandbox environment.

3.1 Generative Agents

If prompted properly, LLMs can generate believable behaviors(Park et al., 2022). However, to achieve human-like memory performance and long-term coherence, LLM is not enough. We need auxiliary systems to enable agents to perform more naturally. Referring to recent work(Park et al., 2023; Wang et al., 2023a), we abstract these supportive mechanisms into three parts: Planning System, Memory System, and Tool-Use System.

Planning System LLMs have shown some planning and reasoning capacities. However, faced with complex tasks, vanilla LLMs always fail for lacking long-term arrangement abilities. Hence, we introduce a Planning System to ensure agents' behaviors are coherent and believable. The Planning System reorganizes a goal by decomposing the target, summarizing current condition and generating subtasks. Specifically, it is assembled by a series of pluggable prompt modules, which assess current achievement of ultimate goals by checking the memory system and making decisions for next steps. Once a new step is completed, it would be recorded in the memory system.

Memory System. Agents capable of emulating human behavior necessitate comprehending a vast array of experiences, beyond what a prompt can contain. The complete memory stream is too expensive to be accommodated in the limited context window, and attempting to do so can overwhelm the model. Thus, we add a memory system for agents' experience retention and retrieval. The system is built upon a vector database for efficient storing and retrieving. Specifically, every agent's daily memory is encoded into embeddings and stored in the database. Every time when agents face some new situation that needs the previous memory, such as chatting with familiar people, the memory system can retrieve the information about their relationship to improve agent behaviour consistency.

Tool-Use System. Ideally, agents continuously explore the simulated world would learn from previous failures and successes, then acquire diverse skills. In our framework, to realize this feature, we present a tool-use system, which endows agents

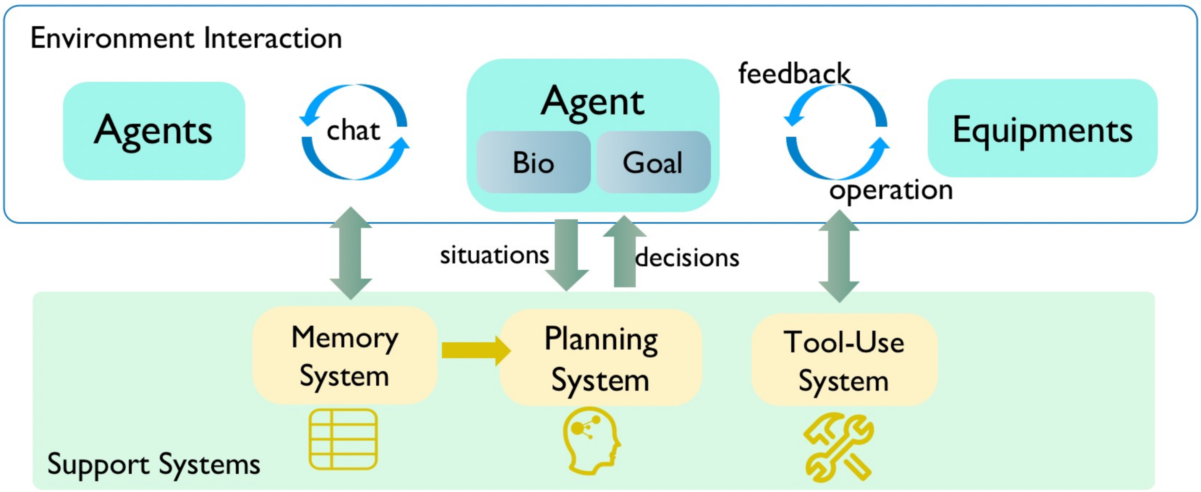


Figure 2: Overview of AgentSims architecture

with the ability to accomplish real-world tasks. Particularly, the tool use system stores equipment-operation pairs learning from feedback of using equipment. Once agents select equipment to interact with by planning and memory system, they need to infer an initial operation by the description of the equipment. And the equipment will return an operation result as feeedback. If the agent believes the result meets their operation purpose, a new skill would be stored in the Tool-Use System.

3.2 Buildings and Equipment

Interactive buildings and equipment are necessities for the diversity of an LLM sandbox. They compose the physical environments of the simulated world. In our framework, a building or location contains equipment like stoves or office desks. Thus, buildings are defined by the equipment they contain and equipment is the basic element composing the interactive environment. More specifically, the equipment can be defined by some definition texts describing its features and support function, which can be either hard-coded by the developer or a language model that supports self-adaptive agent-equipment interaction. When an agent interacts with equipment, as shown in Figure 2, its operation text will be sent to the background support model. The support function then returns the operation outcome based on the predefined rules or model-generated texts. For example, if an agent wants to get a cup of tea from a stove, the operation is 'Get a cup of tea' and the support function may return 'Meaningless operation' according to the hard code or 'You can not get tea from a stove' generated by the model. Then the agent would learn from the feedback and refine its operations.

4 Interaction scenarios

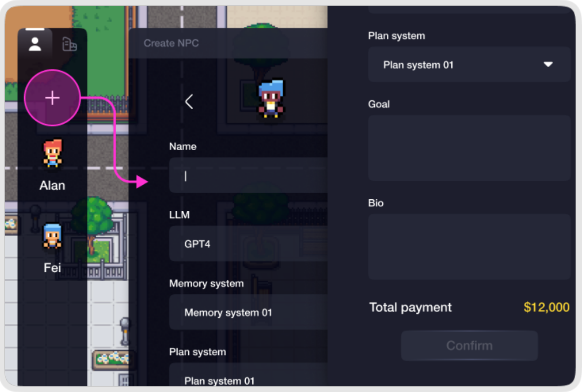
Regarding the researchers' backgrounds and purposes, we design two interaction modes: User Mode and Developer Mode. In the User Mode, researchers who consider little about background support systems are target users. For researchers chasing better LLMs performance, Developer Mode provides flexible protocols for their development of different support mechanisms.

4.1 User Mode

In the User Mode, AgentSims provides an interactive interface in a pixel game style, as shown in Figure 1. Researchers can create agents, construct buildings and equipment in a graphical interface, focusing on the rationality of experiment design, free from complex background driving mechanisms.

Agent Creation. Users can define agents within the system through an easy-to-use front end, as shown in the Figure 3. AgentSims provides various protocols for users to create functional agents. Not only basic information like goals and biography, but also options of Memory and Planning Systems. We pre-design a list of memory and planning systems and users can choose their preference from a drop-down menu.

Building Creation. Users can also customize the physical environment by constructing buildings. As shown in Figure 4, users define a building by choosing a pre-configured building with equipment inside. To be noticed, the equipment in buildings are predefined but can be modified in the Developer



Name

LLM: GPT4

Memory system: Memory system 01

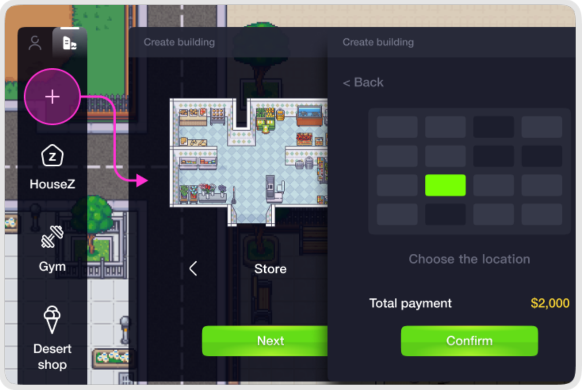
Plan system: Plan system 01

Goal

Bio

Total payment $12,000

Figure 3: Agent Creation



Total payment $2,000

Figure 4: Building Creation

Mode.

Experiment Intervene. Besides observing, users can play as the major agent to participate in the experiment. By talking with other agents, users can intervene the experiment naturally rather than modify agents' memory or goals roughly.

4.2 Developer Mode

Developer Mode is designed for professional developers who are familiar with the properties of LLMs and pursue better performance of LLMs on a well-defined complex task. The highly-modularized feature of AgentSims enables developers to add new functions within a few lines of code.

Agent Design. Developers have the flexibility to create agents tailored for various objectives and assemble diverse agents within a single sandbox for observation. To streamline the process of agent customization, we've abstracted the LLM backbone and distinct support systems into separate classes and function calls, as illustrated below. This empowers developers to personalize an agent by making adjustments to these abstract functions.

```
class LLMCaller:
    def __init__(self, model: str) -> None:
        self.model = get_model(model)

    def ask(self, prompt: str) :
        result = self.model.generate(prompt)
        return result

class Agent:
    def __init__(self, name, bio, goal, model,
         memorySystem, planSystem, buildings,
         cash):
        self.state = State()
        self.state.buildings = buildings
        self.state.cash = cash
        self.caller = Caller(model)

    def plan(self) -> None:
        self.state.plan_prompt = ...
        self.state.plan =
             self.caller.ask(self.state.pl_prompt)

    def memory_store(self) -> None:
        self.state.memory_prompt = ...
        self.state.memory =
        self.caller.ask(self.state.mem_prompt)

    def use(self, facility: str, operation: str,
         description: str) -> None:
        self.state.use_prompt = ...
        self.state.use =
             self.caller.ask(self.state.use_prompt)
```

Building and Equipment Design. To customize the physical environment, developers can design new buildings and equipment by configuring corresponding json files.

A new equipment can be defined by its type, description and a support function.

```
[{"id": 1,
    "type": "counter",
    "function":...,
    "description": "This is the counter ...",}]
```

In some cases, agents can purchase commodities or earn salaries at the equipment. We use another configure file to annotate these economic features.

```
[{ "id": 1,
    "menu": {
        "chicken": 20,},
    "salary":0,}],
```

We define buildings by a type and the equipment it contains. Hence we use a two-dimensional array to mark the facility ids in the building blocks.

```
[{"assets": "store_v1.2_0719",
    "id": 1,
    "price": 2000,
    "type": "store",
    "blocks":[[1,0,0...1,1]],
    "equipment":[0,1,0..]]}]
```

5 Implementation

AgentSims is run using Python 3.9[1] and requires installing the requirements.txt file provided in the codebase using Python's package manager PyPI[2].

5.1 Backend

The web server is built using Tornado[3], a lightweight Python web framework. It also uses the websockets library for API calls and push notifications, and mysql-connector-python to interact with the MySQL[4] database.

5.2 Frontend

Frontend The web client is built with Unity[5]. The client built by WebGL[6] is embedded in the project code and can be accessed through a browser after proxying with nginx[7].

6 Example Application Tasks

6.1 Subject LLM as participants

When subject LLM agents are participants of an artificial scenario, researchers can evaluate LLM's social abilities, like ToM . In this case, the formulation of specific social scenes is realized by other baseline agents driven by stronger LLMs. For example, to study a new model's social adaptation abilities in a hostile environment, we can embed colleague agents driven by GPT4 with a strong desire of bullying newcomers. Then we place subject agents into this adversarial milieu and test whether the new model can understand other's emotion and improve how colleagues perceive it.

6.2 Subject LLM as mayor

To assess LLM's long-term planning and organization abilities, researchers can appoint the subject LLM as the mayor of a town or the president of a company, where residents or employees are driven by baseline agents like GPT4. To overcome the difficulties set ahead deliberately or emerging during the experiments, then achieve the final goal of the task, the subject LLM needs to recruit new residents to handle new problems, issue sound policies and modify the out-of-date ones, found new functional buildings to satisfy emerging requirements, and so on. By analyzing the success rate of LLM mayor under different difficulties, researchers can gain valuable insights into the diverse capabilities of the LLM.

6.3 Applications besides Evaluation

Besides evaluating LLMs, AgentSims can be used as a data generation platform. Due to the fantastic NLG abilities of LLMs, researchers have applied them in data annotation and augmentation. However, some data involving social judgement and participation necessitate a more intricate approach than a single prompt can provide. Thus, we can simulate a specific social background and let LLMs generate data more precisely. Liu et al. (2023b) have applied simulated society in alignment data generation. With AgentSims tailored for more intricate social simulations, its potential for enhancing data generation across various disciplines is undeniable.

Moreover, our program can also benefit social science researchers, by conducting more controllable preliminary experiments. Given that sota LLMs can understand human instructions and simulate human behaviours, social science researchers can design social environments as they wish for preliminary studies. Once researchers have a hypothesis, pilot experiments can be conducted in our virtual sandbox as a feasibility check.

7 Conclusion

In this paper, we present AgentSims, avisualized and program-based infrastructure for LLM test sandbox construction. AgentSims aims to facilitate researchers in effectively building LLM evaluation tasks. It not only intends to make all its code openly available but also commits to continuously updating its documentation with comprehensive tutorials.

Limitations

As a sandbox system, AgentSims' simulation ability is limited by the accuracy of LLMs and the diversity of buildings and equipment. It can never fully reflect real world cases. Besides, although task-based evaluation is a sound approach to measure the general ability of LLMs, it can hardly reflect fine-grained abilities like math reasoning. The pass rate of tasks can not provide insights on why LLMs success or fail.

[1] https://www.python.org/downloads/release/python-390

[2] https://pypi.org/

[3] https://www.tornadoweb.org/en/stable/

[4] https://www.mysql.com/

[5] https://unity3d.com

[6] https://get.webgl.org

[7] https://nginx.org/en/

References

BIG bench authors. 2023. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *Transactions on Machine Learning Research*.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners.

Suriya Gunasekar, Yi Zhang, Jyoti Aneja, Caio César Teodoro Mendes, Allie Del Giorno, Sivakanth Gopi, Mojan Javaheripi, Piero Kauffmann, Gustavo de Rosa, Olli Saarikivi, et al. 2023. Textbooks are all you need. *arXiv preprint arXiv:2306.11644*.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2020. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*.

Yuzhen Huang, Yuzhuo Bai, Zhihao Zhu, Junlei Zhang, Jinghan Zhang, Tangjun Su, Junteng Liu, Chuancheng Lv, Yikai Zhang, Jiayi Lei, Yao Fu, Maosong Sun, and Junxian He. 2023. C-eval: A multi-level multi-discipline chinese evaluation suite for foundation models. *arXiv preprint arXiv:2305.08322*.

Guohao Li, Hasan Abed Al Kader Hammoud, Hani Itani, Dmitrii Khizbullin, and Bernard Ghanem. 2023. Camel: Communicative agents for "mind" exploration of large scale language model society.

Yen-Ting Lin and Yun-Nung Chen. 2023. Llm-eval: Unified multi-dimensional automatic evaluation for open-domain conversations with large language models.

Hanmeng Liu, Ruoxi Ning, Zhiyang Teng, Jian Liu, Qiji Zhou, and Yue Zhang. 2023a. Evaluating the logical reasoning ability of chatgpt and gpt-4.

Ruibo Liu, Ruixin Yang, Chenyan Jia, Ge Zhang, Denny Zhou, Andrew M. Dai, Diyi Yang, and Soroush Vosoughi. 2023b. Training socially aligned language models in simulated human society.

Varun Nair, Elliot Schumacher, Geoffrey Tso, and Anitha Kannan. 2023. Dera: Enhancing large language model completions with dialog-enabled resolving agents.

Erik Nijkamp, Bo Pang, Hiroaki Hayashi, Lifu Tu, Huan Wang, Yingbo Zhou, Silvio Savarese, and Caiming Xiong. 2023. Codegen: An open large language model for code with multi-turn program synthesis.

OpenAI. 2023. Gpt-4 technical report.

Joon Sung Park, Joseph C. O'Brien, Carrie J. Cai, Meredith Ringel Morris, Percy Liang, and Michael S. Bernstein. 2023. Generative agents: Interactive simulacra of human behavior.

Joon Sung Park, Lindsay Popowski, Carrie J. Cai, Meredith Ringel Morris, Percy Liang, and Michael S. Bernstein. 2022. Social simulacra: Creating populated prototypes for social computing systems.

David Premack and Guy Woodruff. 1978. Does the chimpanzee have a theory of mind? *Behavioral and brain sciences*, 1(4):515–526.

Chen Qian, Xin Cong, Cheng Yang, Weize Chen, Yusheng Su, Juyuan Xu, Zhiyuan Liu, and Maosong Sun. 2023. Communicative agents for software development.

Yongliang Shen, Kaitao Song, Xu Tan, Dongsheng Li, Weiming Lu, and Yueting Zhuang. 2023. Hugginggpt: Solving ai tasks with chatgpt and its friends in huggingface. *arXiv preprint arXiv:2303.17580*.

Hao Sun, Zhexin Zhang, Jiawen Deng, Jiale Cheng, and Minlie Huang. 2023. Safety assessment of chinese large language models.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. 2023a. Voyager: An open-ended embodied agent with large language models.

Jiaan Wang, Yunlong Liang, Fandong Meng, Zengkui Sun, Haoxiang Shi, Zhixu Li, Jinan Xu, Jianfeng Qu, and Jie Zhou. 2023b. Is chatgpt a good nlg evaluator? a preliminary study.

Lilian Weng. 2023. Llm-powered autonomous agents. *lilianweng.github.io*.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. 2023. Tree of thoughts: Deliberate problem solving with large language models.

Wanjun Zhong, Ruixiang Cui, Yiduo Guo, Yaobo Liang, Shuai Lu, Yanlin Wang, Amin Saied, Weizhu Chen, and Nan Duan. 2023. Agieval: A human-centric benchmark for evaluating foundation models.

Chunting Zhou, Pengfei Liu, Puxin Xu, Srini Iyer, Jiao Sun, Yuning Mao, Xuezhe Ma, Avia Efrat, Ping Yu, Lili Yu, Susan Zhang, Gargi Ghosh, Mike Lewis, Luke Zettlemoyer, and Omer Levy. 2023. Lima: Less is more for alignment.