Front. Comput. Sci., 2024, 18(6): 186345
https://doi.org/10.1007/s11704-024-40231-1

REVIEW ARTICLE

A survey on large language model based autonomous agents

Lei WANG, Chen MA*, Xueyang FENG*, Zeyu ZHANG, Hao YANG, Jingsen ZHANG, Zhiyuan CHEN, Jiakai TANG, Xu CHEN (✉), Yankai LIN (✉), Wayne Xin ZHAO, Zhewei WEI, Jirong WEN

Gaoling School of Artificial Intelligence, Renmin University of China, Beijing 100872, China

Abstract Autonomous agents have long been a research focus in academic and industry communities. Previous research often focuses on training agents with limited knowledge within isolated environments, which diverges significantly from human learning processes, and makes the agents hard to achieve human-like decisions. Recently, through the acquisition of vast amounts of Web knowledge, large language models (LLMs) have shown potential in human-level intelligence, leading to a surge in research on LLM-based autonomous agents. In this paper, we present a comprehensive survey of these studies, delivering a systematic review of LLM-based autonomous agents from a holistic perspective. We first discuss the construction of LLM-based autonomous agents, proposing a unified framework that encompasses much of previous work. Then, we present a overview of the diverse applications of LLM-based autonomous agents in social science, natural science, and engineering. Finally, we delve into the evaluation strategies commonly used for LLM-based autonomous agents. Based on the previous studies, we also present several challenges and future directions in this field.

Keywords autonomous agent, large language model, human-level intelligence

1 Introduction

“*An autonomous agent is a system situated within and a part of an environment that senses that environment and acts on it, over time, in pursuit of its own agenda and so as to effect what it senses in the future.*”

Franklin and Graesser (1997)

Autonomous agents have long been recognized as a promising approach to achieving artificial general intelligence (AGI), which is expected to accomplish tasks through self-directed planning and actions. In previous studies, the agents are assumed to act based on simple and heuristic policy functions, and learned in isolated and restricted environments [1–6]. Such assumptions significantly differs from the human learning process, since the human mind is highly complex, and individuals can learn from a much wider variety of environments. Because of these gaps, the agents obtained from the previous studies are usually far from replicating human-level decision processes, especially in unconstrained, open-domain settings.

In recent years, large language models (LLMs) have achieved notable successes, demonstrating significant potential in attaining human-like intelligence [5–10]. This capability arises from leveraging comprehensive training datasets alongside a substantial number of model parameters. Building upon this capability, there has been a growing research area that employs LLMs as central controllers to construct autonomous agents to obtain human-like decision-making capabilities [11–17].

Comparing with reinforcement learning, LLM-based agents have more comprehensive internal world knowledge, which facilitates more informed agent actions even without training on specific domain data. Additionally, LLM-based agents can provide natural language interfaces to interact with humans, which is more flexible and explainable.

Along this direction, researchers have developed numerous promising models (see Fig. 1 for an overview of this field), where the key idea is to equip LLMs with crucial human capabilities like memory and planning to make them behave like humans and complete various tasks effectively. Previously, these models were proposed independently, with limited efforts made to summarize and compare them holistically. However, we believe a systematic summary on this rapidly developing field is of great significance to comprehensively understand it and benefit to inspire future research.

In this paper, we conduct a comprehensive survey of the field of LLM-based autonomous agents. Specifically, we organize our survey based on three aspects including the construction, application, and evaluation of LLM-based autonomous agents. For the agent construction, we focus on two problems, that is, (1) how to design the agent architecture to better leverage LLMs, and (2) how to inspire and enhance the agent capability to complete different tasks. Intuitively, the first problem aims to build the hardware fundamentals for the agent, while the second problem focus on providing the agent

Received March 5, 2024; accepted March 10, 2024

E-mail: xu.chen@ruc.edu.cn; yankailin@ruc.edu.cn

* These authors contributed equally to this work.

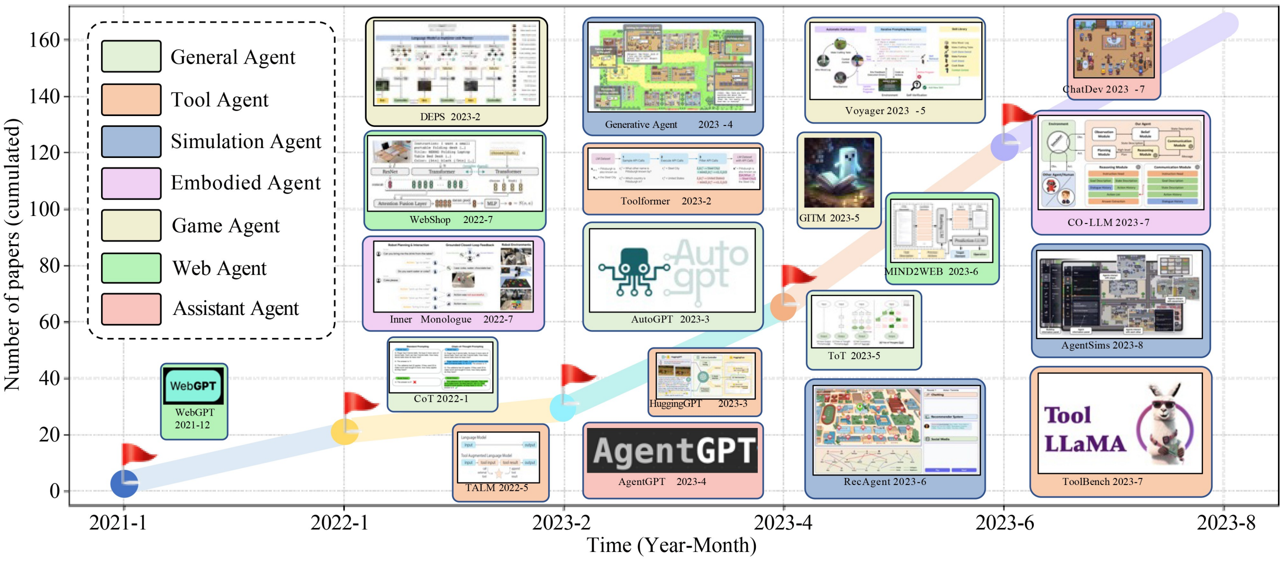


Fig. 1 Illustration of the growth trend in the field of LLM-based autonomous agents. We present the cumulative number of papers published from January 2021 to August 2023. We assign different colors to represent various agent categories. For example, a game agent aims to simulate a game-player, while a tool agent mainly focuses on tool using. For each time period, we provide a curated list of studies with diverse agent categories

with software resources. For the first problem, we present a unified agent framework, which can encompass most of the previous studies. For the second problem, we provide a summary on the commonly-used strategies for agents' capability acquisition. In addition to discussing agent construction, we also provide an systematic overview of the applications of LLM-based autonomous agents in social science, natural science, and engineering. Finally, we delve into the strategies for evaluating LLM-based autonomous agents, focusing on both subjective and objective strategies.

In summary, this survey conducts a systematic review and establishes comprehensive taxonomies for existing studies in the burgeoning field of LLM-based autonomous agents. Our focus encompasses three primary areas: construction of agents, their applications, and methods of evaluation. Drawing from a wealth of previous studies, we identify various challenges in this field and discuss potential future directions. We expect that our survey can provide newcomers of LLM-based autonomous agents with a comprehensive background knowledge, and also encourage further groundbreaking studies.

2 LLM-based autonomous agent construction

LLM-based autonomous agents are expected to effectively perform diverse tasks by leveraging the human-like capabilities of LLMs. In order to achieve this goal, there are two significant aspects, that is, (1) which architecture should be designed to better use LLMs and (2) give the designed architecture, how to enable the agent to acquire capabilities for accomplishing specific tasks. Within the context of architecture design, we contribute a systematic synthesis of existing research, culminating in a comprehensive unified framework. As for the second aspect, we summarize the strategies for agent capability acquisition based on whether they fine-tune the LLMs. When comparing LLM-based autonomous agents to traditional machine learning, designing the agent architecture is analogous to determining the network structure, while the agent capability acquisition is similar to learning the network parameters. In the following, we introduce these two aspects more in detail.

2.1 Agent architecture design

Recent advancements in LLMs have demonstrated their great potential to accomplish a wide range of tasks in the form of question-answering (QA). However, building autonomous agents is far from QA, since they need to fulfill specific roles and autonomously perceive and learn from the environment to evolve themselves like humans. To bridge the gap between traditional LLMs and autonomous agents, a crucial aspect is to design rational agent architectures to assist LLMs in maximizing their capabilities. Along this direction, previous work has developed a number of modules to enhance LLMs. In this section, we propose a unified framework to summarize these modules. In specific, the overall structure of our framework is illustrated Fig. 2, which is composed of a profiling module, a memory module, a planning module, and an action module. The purpose of the profiling module is to identify the role of the agent. The memory and planning modules place the agent into a dynamic environment, enabling it to recall past behaviors and plan future actions. The action module is responsible for translating the agent's decisions into specific outputs. Within these modules, the profiling module impacts the memory and planning modules, and collectively, these three modules influence the action module. In the following, we detail these modules.

2.1.1 Profiling module

Autonomous agents typically perform tasks by assuming specific roles, such as coders, teachers and domain experts [18,19]. The profiling module aims to indicate the profiles of the agent roles, which are usually written into the prompt to

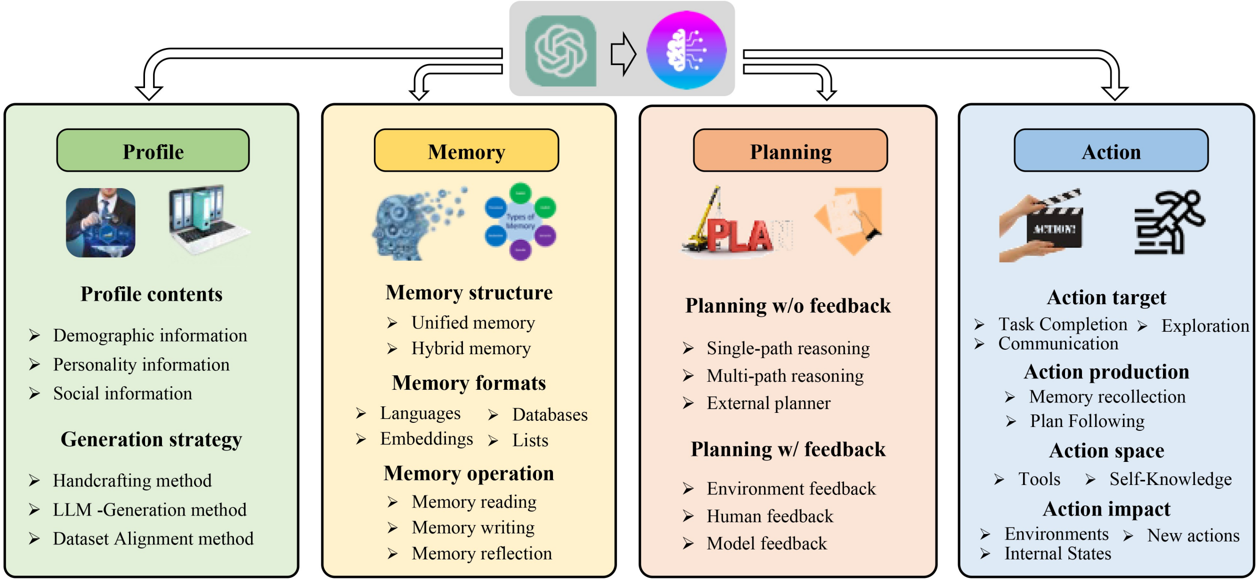


Fig. 2 A unified framework for the architecture design of LLM-based autonomous agent

influence the LLM behaviors. Agent profiles typically encompass basic information such as age, gender, and career [20], as well as psychology information, reflecting the personalities of the agent, and social information, detailing the relationships between agents [21]. The choice of information to profile the agent is largely determined by the specific application scenarios. For instance, if the application aims to study human cognitive process, then the psychology information becomes pivotal. After identifying the types of profile information, the next important problem is to create specific profiles for the agents. Existing literature commonly employs the following three strategies.

Handcrafting method: in this method, agent profiles are manually specified. For instance, if one would like to design agents with different personalities, he can use "you are an outgoing person" or "you are an introverted person" to profile the agent. The handcrafting method has been leveraged in a lot of previous work to indicate the agent profiles. For example, Generative Agent [22] describes the agent by the information like name, objectives, and relationships with other agents. MetaGPT [23], ChatDev [18], and Self-collaboration [24] predefine various roles and their corresponding responsibilities in software development, manually assigning distinct profiles to each agent to facilitate collaboration. PTLLM [25] aims to explore and quantify personality traits displayed in texts generated by LLMs. This method guides LLMs in generating diverse responses by manfully defining various agent characters through the use of personality assessment tools such as IPIP-NEO [26] and BFI [27]. [28] studies the toxicity of the LLM output by manually prompting LLMs with different roles, such as politicians, journalists and businesspersons. In general, the handcrafting method is very flexible, since one can assign any profile information to the agents. However, it can be also labor-intensive, particularly when dealing with a large number of agents.

LLM-generation method: in this method, agent profiles are automatically generated based on LLMs. Typically, it begins by indicating the profile generation rules, elucidating the composition and attributes of the agent profiles within the target population. Then, one can optionally specify several seed agent profiles to serve as few-shot examples. At last, LLMs are leveraged to generate all the agent profiles. For example, RecAgent [21] first creates seed profiles for a few number of agents by manually crafting their backgrounds like age, gender, personal traits, and movie preferences. Then, it leverages ChatGPT to generate more agent profiles based on the seed information. The LLM-generation method can save significant time when the number of agents is large, but it may lack precise control over the generated profiles.

Dataset alignment method: in this method, the agent profiles are obtained from real-world datasets. Typically, one can first organize the information about real humans in the datasets into natural language prompts, and then leverage it to profile the agents. For instance, in [29], the authors assign roles to GPT-3 based on the demographic backgrounds (such as race/ethnicity, gender, age, and state of residence) of participants in the American National Election Studies (ANES). They subsequently investigate whether GPT-3 can produce similar results to those of real humans. The dataset alignment method accurately captures the attributes of the real population, thereby making the agent behaviors more meaningful and reflective of real-world scenarios.

Remark. While most of the previous work leverage the above profile generation strategies independently, we argue that combining them may yield additional benefits. For example, in order to predict social developments via agent simulation, one can leverage real-world datasets to profile a subset of the agents, thereby accurately reflecting the current social status. Subsequently, roles that do not exist in the real world but may emerge in the future can be manually assigned to the other agents, enabling the prediction of future social development. Beyond this example, one can also flexibly combine the other strategies. The profile module serves as the foundation for agent design, exerting significant influence on the agent memorization, planning, and action procedures.

2.1.2 Memory module

The memory module plays a very important role in the agent architecture design. It stores information perceived from the environment and leverages the recorded memories to facilitate future actions. The memory module can help the agent to accumulate experiences, self-evolve, and behave in a more consistent, reasonable, and effective manner. This section provides a comprehensive overview of the memory module, focusing on its structures, formats, and operations.

Memory structures: LLM-based autonomous agents usually incorporate principles and mechanisms derived from cognitive science research on human memory processes. Human memory follows a general progression from sensory memory that registers perceptual inputs, to short-term memory that maintains information transiently, to long-term memory that consolidates information over extended periods. When designing the agent memory structures, researchers take inspiration from these aspects of human memory. In specific, short-term memory is analogous to the input information within the context window constrained by the transformer architecture. Long-term memory resembles the external vector storage that agents can rapidly query and retrieve from as needed. In the following, we introduce two commonly used memory structures based on the short- and long-term memories.

- Unified memory. This structure only simulates the human shot-term memory, which is usually realized by in-context learning, and the memory information is directly written into the prompts. For example, RLP [30] is a conversation agent, which maintains internal states for the speaker and listener. During each round of conversation, these states serve as LLM prompts, functioning as the agent's short-term memory. SayPlan [31] is an embodied agent specifically designed for task planning. In this agent, the scene graphs and environment feedback serve as the agent's short-term memory, guiding its actions. CALYPSO [32] is an agent designed for the game Dungeons & Dragons, which can assist Dungeon Masters in the creation and narration of stories. Its short-term memory is built upon scene descriptions, monster information, and previous summaries. DEPS [33] is also a game agent, but it is developed for Minecraft. The agent initially generates task plans and then utilizes them to prompt LLMs, which in turn produce actions to complete the task. These plans can be deemed as the agent's short-term memory. In practice, implementing short-term memory is straightforward and can enhance an agent's ability to perceive recent or contextually sensitive behaviors and observations. However, due to the limitation of context window of LLMs, it's hard to put all memories into prompt, which may degrade the performance of agents. This method has high requirements on the window length of LLMs and the ability to handle long contexts. Therefore, many researchers resort to hybrid memory to alleviate this question. However, the limited context window of LLMs restricts incorporating comprehensive memories into prompts, which can impair agent performance. This challenge necessitates LLMs with larger context windows and the ability to handle extended contexts. Consequently, numerous researchers turn to hybrid memory systems to mitigate this issue.

- Hybrid memory. This structure explicitly models the human short-term and long-term memories. The short-term memory temporarily buffers recent perceptions, while long-term memory consolidates important information over time. For instance, Generative Agent [20] employs a hybrid memory structure to facilitate agent behaviors. The short-term memory contains the context information about the agent current situations, while the long-term memory stores the agent past behaviors and thoughts, which can be retrieved according to the current events. AgentSims [34] also implements a hybrid memory architecture. The information provided in the prompt can be considered as short-term memory. In order to enhance the storage capacity of memory, the authors propose a long-term memory system that utilizes a vector database, facilitating efficient storage and retrieval. Specifically, the agent's daily memories are encoded as embeddings and stored in the vector database. If the agent needs to recall its previous memories, the long-term memory system retrieves relevant information using embedding similarities. This process can improve the consistency of the agent's behavior. In GITM [16], the short-term memory stores the current trajectory, and the long-term memory saves reference plans summarized from successful prior trajectories. Long-term memory provides stable knowledge, while short-term memory allows flexible planning. Reflexion [12] utilizes a short-term sliding window to capture recent feedback and incorporates persistent long-term storage to retain condensed insights. This combination allows for the utilization of both detailed immediate experiences and high-level abstractions. SCM [35] selectively activates the most relevant long-term knowledge to combine with short-term memory, enabling reasoning over complex contextual dialogues. SimplyRetrieve [36] utilizes user queries as short-term memory and stores long-term memory using external knowledge bases. This design enhances the model accuracy while guaranteeing user privacy. MemorySandbox [37] implements long-term and short-term memory by utilizing a 2D canvas to store memory objects, which can then be accessed throughout various conversations. Users can create multiple conversations with different agents on the same canvas, facilitating the sharing of memory objects through a simple drag-and-drop interface. In practice, integrating both short-term and long-term memories can enhance an agent's ability for long-range reasoning and accumulation of valuable experiences, which are crucial for accomplishing tasks in complex environments.

Remark. Careful readers may find that there may also exist another type of memory structure, that is, only based on the long-term memory. However, we find that such type of memory is rarely documented in the literature. Our speculation is that the agents are always situated in continuous and dynamic environments, with consecutive actions displaying a high correlation. Therefore, the capture of short-term memory is very important and usually cannot be disregarded.

Memory formats: In addition to the memory structure, another perspective to analyze the memory module is based on the formats of the memory storage medium, for example, natural language memory or embedding memory. Different memory formats possess distinct strengths and are suitable for various applications. In the following, we introduce several representative memory formats.

- Natural languages. In this format, memory information such as the agent behaviors and observations are directly described using raw natural language. This format possesses several strengths. Firstly, the memory information can be expressed in a flexible and understandable manner. Moreover, it retains rich semantic information that can provide comprehensive signals to guide agent behaviors. In the previous work, Reflexion [12] stores experiential feedback in natural language within a sliding window. Voyager [38] employs natural language descriptions to represent skills within the Minecraft game, which are directly stored in memory.
- Embeddings. In this format, memory information is encoded into embedding vectors, which can enhance the memory retrieval and reading efficiency. For instance, MemoryBank [39] encodes each memory segment into an embedding vector, which creates an indexed corpus for retrieval. [16] represents reference plans as embeddings to facilitate matching and reuse. Furthermore, ChatDev [18] encodes dialogue history into vectors for retrieval.
- Databases. In this format, memory information is stored in databases, allowing the agent to manipulate memories efficiently and comprehensively. For example, ChatDB [40] uses a database as a symbolic memory module. The agent can utilize SQL statements to precisely add, delete, and revise the memory information. In DB-GPT [41], the memory module is constructed based on a database. To more intuitively operate the memory information, the agents are fine-tuned to understand and execute SQL queries, enabling them to interact with databases using natural language directly.
- Structured lists. In this format, memory information is organized into lists, and the semantic of memory can be conveyed in an efficient and concise manner. For instance, GITM [16] stores action lists for sub-goals in a hierarchical tree structure. The hierarchical structure explicitly captures the relationships between goals and corresponding plans. RET-LLM [42] initially converts natural language sentences into triplet phrases, and subsequently stores them in memory.

Remark. Here we only show several representative memory formats, but it is important to note that there are many uncovered ones, such as the programming code used by [38]. Moreover, it should be emphasized that these formats are not mutually exclusive; many models incorporate multiple formats to concurrently harness their respective benefits. A notable example is the memory module of GITM [16], which utilizes a key-value list structure. In this structure, the keys are represented by embedding vectors, while the values consist of raw natural languages. The use of embedding vectors allows for efficient retrieval of memory records. By utilizing natural languages, the memory contents become highly comprehensive, enabling more informed agent actions.

Above, we mainly discuss the internal designs of the memory module. In the following, we turn our focus to memory operations, which are used to interact with external environments.

Memory operations: The memory module plays a critical role in allowing the agent to acquire, accumulate, and utilize significant knowledge by interacting with the environment. The interaction between the agent and the environment is accomplished through three crucial memory operations: memory reading, memory writing, and memory reflection. In the following, we introduce these operations more in detail.

- Memory reading. The objective of memory reading is to extract meaningful information from memory to enhance the agent's actions. For example, using the previously successful actions to achieve similar goals [16]. The key of memory reading lies in how to extract valuable information from history actions. Usually, there three commonly used criteria for information extraction, that is, the recency, relevance, and importance [20]. Memories that are more recent, relevant, and important are more likely to be extracted. Formally, we conclude the following equation from existing literature for memory information extraction:

$$m^* = \arg\min_{m \in M} \alpha s^{rec}(q,m) + \beta s^{rel}(q,m) + \gamma s^{imp}(m), \qquad (1)$$

where q is the query, for example, the task that the agent should address or the context in which the agent is situated. M is the set of all memories. $s^{rec}(\cdot)$, $s^{rel}(\cdot)$, and $s^{imp}(\cdot)$ are the scoring functions for measuring the recency, relevance, and importance of the memory m. These scoring functions can be implemented using various methods, for example, $s^{rel}(q,m)$ can be realized based on LSH, ANNOY, HNSW, FAISS, and so on. It should be noted that s^{imp} only reflects the characters of the memory itself, thus it is unrelated to the query q. α, β, and γ are balancing parameters. By assigning them with different values, one can obtain various memory reading strategies. For example, by setting $\alpha = \gamma = 0$, many studies [16,30,38,42] only consider the relevance score s^{rel} for memory reading. By assigning $\alpha = \beta = \gamma = 1.0$, [20] equally weights all the above three metrics to extract information from memory.

- Memory writing. The purpose of memory writing is to store information about the perceived environment in memory. Storing valuable information in memory provides a foundation for retrieving informative memories in the future, enabling the agent to act more efficiently and rationally. During the memory writing process, there are two potential problems that should be carefully addressed. On one hand, it is crucial to address how to store information that is similar to existing memories (i.e., memory duplicated). On the other hand, it is important to consider how to remove information when the memory reaches its storage limit (i.e., memory overflow). In the following, we discuss these problems more in detail. (1) Memory duplicated. To incorporate similar information, people have developed various methods for integrating new and previous records. For instance, in [7], the successful action sequences related to the same sub-goal are stored in a list. Once the size of the list reaches N(=5), all the sequences in it are condensed into a unified plan solution using LLMs. The original sequences in the memory are replaced with the newly generated one. Augmented LLM [43] aggregates duplicate information via count accumulation, avoiding redundant storage. (2) Memory overflow. In order to write information into the memory when it is full, people design different methods to delete existing information to continue the memorizing process. For example, in ChatDB [40], memories can be explicitly deleted based on user commands. RET-LLM [42] uses a fixed-size buffer for memory, overwriting the oldest entries in a first-in-first-out (FIFO) manner.

- Memory reflection. Memory reflection emulates humans' ability to witness and evaluate their own cognitive, emotional, and behavioral processes. When adapted to agents, the objective is to provide agents with the capability to independently summarize and infer more abstract, complex and high-level information. More specifically, in Generative Agent [20], the agent has the capability to summarize its past experiences stored in memory into broader and more abstract insights. To begin with, the agent generates three key questions based on its recent memories. Then, these questions are used to query the memory to obtain relevant information. Building upon the acquired information, the agent generates five insights, which reflect the agent high-level ideas. For example, the low-level memories "Klaus Mueller is writing a research paper", "Klaus Mueller is engaging with a librarian to further his research", and "Klaus Mueller is conversing with Ayesha Khan about his research" can induce the high-level insight "Klaus Mueller is dedicated to his research". In addition, the reflection process can occur hierarchically, meaning that the insights can be generated based on existing insights. In GITM [16], the actions that successfully accomplish the sub-goals are stored in a list. When the list contains more than five elements, the agent summarizes them into a common and abstract pattern and replaces all the elements. In ExpeL [44], two approaches are introduced for the agent to acquire reflection. Firstly, the agent compares successful or failed trajectories within the same task. Secondly, the agent learns from a collection of successful trajectories to gain experiences.

A significant distinction between traditional LLMs and the agents is that the latter must possess the capability to learn and complete tasks in dynamic environments. If we consider the memory module as responsible for managing the agents' past behaviors, it becomes essential to have another significant module that can assist the agents in planning their future actions. In the following, we present an overview of how researchers design the planning module.

2.1.3 Planning module

When faced with a complex task, humans tend to deconstruct it into simpler subtasks and solve them individually. The planning module aims to empower the agents with such human capability, which is expected to make the agent behave more reasonably, powerfully, and reliably. In specific, we summarize existing studies based on whether the agent can receive feedback in the planing process, which are detailed as follows:

Planning without feedback: In this method, the agents do not receive feedback that can influence its future behaviors after taking actions. In the following, we present several representative strategies.

- Single-path reasoning. In this strategy, the final task is decomposed into several intermediate steps. These steps are connected in a cascading manner, with each step leading to only one subsequent step. LLMs follow these steps to achieve the final goal. Specifically, Chain of Thought (CoT) [45] proposes inputting reasoning steps for solving complex problems into the prompt. These steps serve as examples to inspire LLMs to plan and act in a step-by-step manner. In this method, the plans are created based on the inspiration from the examples in the prompts. Zero-shot-CoT [46] enables LLMs to generate task reasoning processes by prompting them with trigger sentences like "think step by step". Unlike CoT, this method does not incorporate reasoning steps as examples in the prompts. Re-Prompting [47] involves checking whether each step meets the necessary prerequisites before generating a plan. If a step fails to meet the prerequisites, it introduces a prerequisite error message and prompts the LLM to regenerate the plan. ReWOO [48] introduces a paradigm of separating plans from external observations, where the agents first generate plans and obtain observations independently, and then combine them together to derive the final results. HuggingGPT [13] first decomposes the task into many sub-goals, and then solves each of them based on Huggingface. Different from CoT and Zero-shot-CoT, which outcome all the reasoning steps in a one-shot manner, ReWOO and HuggingGPT produce the results by accessing LLMs multiply times.

- Multi-path reasoning. In this strategy, the reasoning steps for generating the final plans are organized into a tree-like structure. Each intermediate step may have multiple subsequent steps. This approach is analogous to human thinking, as individuals may have multiple choices at each reasoning step. In specific, Self-consistent CoT (CoT-SC) [49] believes that each complex problem has multiple ways of thinking to deduce the final answer. Thus, it starts by employing CoT to generate various reasoning paths and corresponding answers. Subsequently, the answer with the highest frequency is chosen as the final output. Tree of Thoughts (ToT) [50] is designed to generate plans using a tree-like reasoning structure. In this approach, each node in the tree represents a "thought," which corresponds to an intermediate reasoning step. The selection of these intermediate steps is based on the evaluation of LLMs. The final plan is generated using either the breadth-first search (BFS) or depth-first search (DFS) strategy. Comparing with CoT-SC, which generates all the planed steps together, ToT needs to query LLMs for each reasoning step. In RecMind [51], the authors designed a self-inspiring mechanism, where the discarded historical information in the planning process is also leveraged to derive new reasoning steps. In GoT [52], the authors expand the tree-like reasoning structure in ToT to graph structures, resulting in more powerful prompting strategies. In AoT [53], the authors design a novel method to enhance the reasoning processes of LLMs by incorporating algorithmic examples into the prompts. Remarkably, this method only needs to query LLMs for only one or a few times. In [54], the LLMs are leveraged as zero-shot planners.

At each planning step, they first generate multiple possible next steps, and then determine the final one based on their distances to admissible actions. [55] further improves [54] by incorporating examples that are similar to the queries in the prompts. RAP [56] builds a world model to simulate the potential benefits of different plans based on Monte Carlo Tree Search (MCTS), and then, the final plan is generated by aggregating multiple MCTS iterations. To enhance comprehension, we provide an illustration comparing the strategies of single-path and multi-path reasoning in Fig. 3.

● External planner. Despite the demonstrated power of LLMs in zero-shot planning, effectively generating plans for domain-specific problems remains highly challenging. To address this challenge, researchers turn to external planners. These tools are well-developed and employ efficient search algorithms to rapidly identify correct, or even optimal, plans. In specific, LLM+P [57] first transforms the task descriptions into formal Planning Domain Definition Languages (PDDL), and then it uses an external planner to deal with the PDDL. Finally, the generated results are transformed back into natural language by LLMs. Similarly, LLM-DP [58] utilizes LLMs to convert the observations, the current world state, and the target objectives into PDDL. Subsequently, this transformed data is passed to an external planner, which efficiently determines the final action sequence. CO-LLM [22] demonstrates that LLMs is good at generating high-level plans, but struggle with low-level control. To address this limitation, a heuristically designed external low-level planner is employed to effectively execute actions based on high-level plans.

Planning with feedback: In many real-world scenarios, the agents need to make long-horizon planning to solve complex tasks. When facing these tasks, the above planning modules without feedback can be less effective due to the following reasons: firstly, generating a flawless plan directly from the beginning is extremely difficult as it needs to consider various complex preconditions. As a result, simply following the initial plan often leads to failure. Moreover, the execution of the plan may be hindered by unpredictable transition dynamics, rendering the initial plan non-executable. Simultaneously, when examining how humans tackle complex tasks, we find that individuals may iteratively make and revise their plans based on external feedback. To simulate such human capability, researchers have designed many planning modules, where the agent can receive feedback after taking actions. The feedback can be obtained from environments, humans, and models, which are detailed in the following.

● Environmental feedback. This feedback is obtained from the objective world or virtual environment. For instance, it could be the game's task completion signals or the observations made after the agent takes an action. In specific, ReAct [59] proposes constructing prompts using thought-act-observation triplets. The thought component aims to facilitate high-level reasoning and planning for guiding agent behaviors. The act represents a specific action taken by the agent. The observation corresponds to the outcome of the action, acquired through external feedback, such as search engine results. The next thought is influenced by the previous observations, which makes the generated plans more adaptive to the environment. Voyager [38] makes plans by incorporating three types of environment feedback including the intermediate progress of program execution, the execution error and self-verification results. These signals can help the agent to make better plans for the next action. Similar to Voyager, Ghost [16] also incorporates feedback into the reasoning and action taking processes. This feedback encompasses the environment states as well as the success and failure information for each executed action. SayPlan [31] leverages environmental feedback derived from a scene graph simulator to validate and refine its strategic formulations. This simulator is adept at discerning the outcomes and state transitions subsequent to agent actions, facilitating SayPlan's iterative recalibration of its strategies until a viable plan is ascertained. In DEPS [33], the authors argue that solely providing information about the completion of a task is often inadequate for correcting planning errors. Therefore, they propose informing the agent about the detail reasons for task failure, allowing them to more effectively revise their plans. LLM-Planner [60] introduces a grounded re-planning algorithm that dynamically updates plans generated by LLMs when encountering object mismatches and unattainable plans during task completion. Inner Monologue [61] provides three types of feedback to the agent after it takes actions: (1) whether the task is successfully

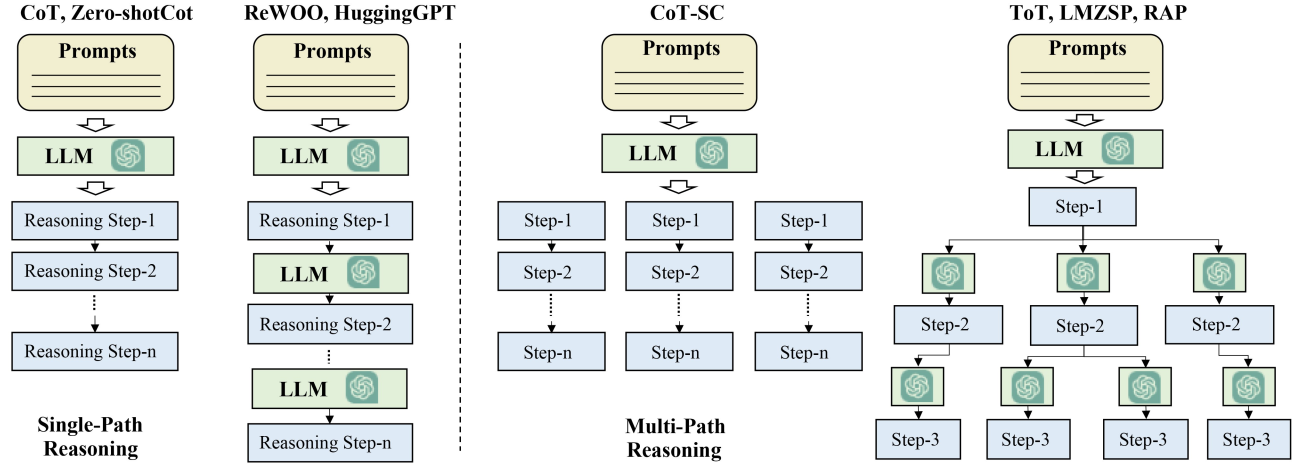


Fig. 3 Comparison between the strategies of single-path and multi-path reasoning. LMZSP is the model proposed in [54]

completed, (2) passive scene descriptions, and (3) active scene descriptions. The former two are generated from the environments, which makes the agent actions more reasonable.

- Human feedback. In addition to obtaining feedback from the environment, directly interacting with humans is also a very intuitive strategy to enhance the agent planning capability. The human feedback is a subjective signal. It can effectively make the agent align with the human values and preferences, and also help to alleviate the hallucination problem. In Inner Monologue [61], the agent aims to perform high-level natural language instructions in a 3D visual environment. It is given the capability to actively solicit feedback from humans regarding scene descriptions. Then, the agent incorporates the human feedback into its prompts, enabling more informed planning and reasoning. In the above cases, we can see, different types of feedback can be combined to enhance the agent planning capability. For example, Inner Monologue [61] collects both environment and human feedback to facilitate the agent plans.
- Model feedback. Apart from the aforementioned environmental and human feedback, which are external signals, researchers have also investigated the utilization of internal feedback from the agents themselves. This type of feedback is usually generated based on pre-trained models. In specific, [62] proposes a self-refine mechanism. This mechanism consists of three crucial components: output, feedback, and refinement. Firstly, the agent generates an output. Then, it utilizes LLMs to provide feedback on the output and offer guidance on how to refine it. At last, the output is improved by the feedback and refinement. This output-feedback-refinement process iterates until reaching some desired conditions. SelfCheck [63] allows agents to examine and evaluate their reasoning steps generated at various stages. They can then correct any errors by comparing the outcomes. InterAct [64] uses different language models (such as ChatGPT and InstructGPT) as auxiliary roles, such as checkers and sorters, to help the main language model avoid erroneous and inefficient actions. ChatCoT [65] utilizes model feedback to improve the quality of its reasoning process. The model feedback is generated by an evaluation module that monitors the agent reasoning steps. Reflexion [12] is developed to enhance the agent's planning capability through detailed verbal feedback. In this model, the agent first produces an action based on its memory, and then, the evaluator generates feedback by taking the agent trajectory as input. In contrast to previous studies, where the feedback is given as a scalar value, this model leverages LLMs to provide more detailed verbal feedback, which can provide more comprehensive supports for the agent plans.

Remark. In conclusion, the implementation of planning module without feedback is relatively straightforward. However, it is primarily suitable for simple tasks that only require a small number of reasoning steps. Conversely, the strategy of planning with feedback needs more careful designs to handle the feedback. Nevertheless, it is considerably more powerful and capable of effectively addressing complex tasks that involve long-range reasoning.

2.1.4 Action module

The action module is responsible for translating the agent's decisions into specific outcomes. This module is located at the most downstream position and directly interacts with the environment. It is influenced by the profile, memory, and planning modules. This section introduces the action module from four perspectives: (1) Action goal: what are the intended outcomes of the actions? (2) Action production: how are the actions generated? (3) Action space: what are the available actions? (4) Action impact: what are the consequences of the actions? Among these perspectives, the first two focus on the aspects preceding the action ("before-action" aspects), the third focuses on the action itself ("in-action" aspect), and the fourth emphasizes the impact of the actions ("after-action" aspect).

Action goal: The agent can perform actions with various objectives. Here, we present several representative examples: (1) *Task Completion*. In this scenario, the agent's actions are aimed at accomplishing specific tasks, such as crafting an iron pickaxe in Minecraft [38] or completing a function in software development [18]. These actions usually have well-defined objectives, and each action contributes to the completion of the final task. Actions aimed at this type of goal are very common in existing literature. (2) *Communication*. In this case, the actions are taken to communicate with the other agents or real humans for sharing information or collaboration. For example, the agents in ChatDev [18] may communicate with each other to collectively accomplish software development tasks. In Inner Monologue [61], the agent actively engages in communication with humans and adjusts its action strategies based on human feedback. (3) *Environment Exploration*. In this example, the agent aims to explore unfamiliar environments to expand its perception and strike a balance between exploring and exploiting. For instance, the agent in Voyager [38] may explore unknown skills in their task completion process, and continually refine the skill execution code based on environment feedback through trial and error.

Action production: Different from ordinary LLMs, where the model input and output are directly associated, the agent may take actions via different strategies and sources. In the following, we introduce two types of commonly used action production strategies. (1) Action via memory recollection. In this strategy, the action is generated by extracting information from the agent memory according to the current task. The task and the extracted memories are used as prompts to trigger the agent actions. For example, in Generative Agents [20], the agent maintains a memory stream, and before taking each action, it retrieves recent, relevant and important information from the memory steam to guide the agent actions. In GITM [16], in order to achieve a low-level sub-goal, the agent queries its memory to determine if there are any successful experiences related to the task. If similar tasks have been completed previously, the agent invokes the previously successful actions to handle the current task directly. In collaborative agents such as ChatDev [18] and MetaGPT [23], different agents may communicate with each other. In this process, the conversation history in a dialog is remembered in the agent memories. Each utterance generated by the agent is influenced by its memory. (2) Action via plan following. In this strategy, the agent takes actions following its pre-generated plans. For instance, in DEPS [33], for a given task, the agent first makes action plans. If there are no signals indicating plan failure, the agent will strictly adhere to these plans. In GITM [16], the agent makes high-level plans by decomposing the task into many sub-goals. Based on these plans, the agent takes actions to solve each sub-goal sequentially to complete the final task.

Action space: Action space refers to the set of possible actions that can be performed by the agent. In general, we can roughly divide these actions into two classes: (1) external tools and (2) internal knowledge of the LLMs. In the following, we introduce these actions more in detail.

● External tools. While LLMs have been demonstrated to be effective in accomplishing a large amount of tasks, they may not work well for the domains which need comprehensive expert knowledge. In addition, LLMs may also encounter hallucination problems, which are hard to be resolved by themselves. To alleviate the above problems, the agents are empowered with the capability to call external tools for executing action. In the following, we present several representative tools which have been exploited in the literature.

(1) APIs. Leveraging external APIs to complement and expand action space is a popular paradigm in recent years. For example, HuggingGPT [13] leverages the models on HuggingFace to accomplish complex user tasks. [66,67] propose to automatically generate queries to extract relevant content from external Web pages when responding to user request. TPTU [67] interfaces with both Python interpreters and LaTeX compilers to execute sophisticated computations such as square roots, factorials and matrix operations. Another type of APIs is the ones that can be directly invoked by LLMs based on natural language or code inputs. For instance, Gorilla [68] is a fine-tuned LLM designed to generate accurate input arguments for API calls and mitigate the issue of hallucination during external API invocations. ToolFormer [15] is an LLM-based tool transformation system that can automatically convert a given tool into another one with different functionalities or formats based on natural language instructions. API-Bank [69] is an LLM-based API recommendation agent that can automatically search and generate appropriate API calls for various programming languages and domains. API-Bank also provides an interactive interface for users to easily modify and execute the generated API calls. ToolBench [14] is an LLM-based tool generation system that can automatically design and implement various practical tools based on natural language requirements. The tools generated by ToolBench include calculators, unit converters, calendars, maps, charts, etc. RestGPT [70] connects LLMs with RESTful APIs, which follow widely accepted standards for Web services development, making the resulting program more compatible with real-world applications. TaskMatrix.AI [71] connects LLMs with millions of APIs to support task execution. At its core lies a multimodal conversational foundational model that interacts with users, understands their goals and context, and then produces executable code for particular tasks. All these agents utilize external APIs as their tools, and provide interactive interfaces for users to easily modify and execute the generated or transformed tools.

(2) Databases & Knowledge Bases. Integrating external database or knowledge base enables agents to obtain specific domain information for generating more realistic actions. For example, ChatDB [40] employs SQL statements to query databases, facilitating actions by the agents in a logical manner. MRKL [72] and OpenAGI [73] incorporate various expert systems such as knowledge bases and planners to access domain-specific information.

(3) External models. Previous studies often utilize external models to expand the range of possible actions. In comparison to APIs, external models typically handle more complex tasks. Each external model may correspond to multiple APIs. For example, to enhance the text retrieval capability, MemoryBank [39] incorporates two language models: one is designed to encode the input text, while the other is responsible for matching the query statements. ViperGPT [74] firstly uses Codex, which is implemented based on language model, to generate Python code from text descriptions, and then executes the code to complete the given tasks. TPTU [67] incorporates various LLMs to accomplish a wide range of language generation tasks such as generating code, producing lyrics, and more. ChemCrow [75] is an LLM-based chemical agent designed to perform tasks in organic synthesis, drug discovery, and material design. It utilizes seventeen expert-designed models to assist its operations. MM-REACT [76] integrates various external models, such as VideoBERT for video summarization, X-decoder for image generation, and SpeechBERT for audio processing, enhancing its capability in diverse multimodal scenarios.

● Internal knowledge. In addition to utilizing external tools, many agents rely solely on the internal knowledge of LLMs to guide their actions. We now present several crucial capabilities of LLMs that can support the agent to behave reasonably and effectively. (1) Planning capability. Previous work has demonstrated that LLMs can be used as decent planers to decompose complex task into simpler ones [45]. Such capability of LLMs can be even triggered without incorporating examples in the prompts [46]. Based on the planning capability of LLMs, DEPS [33] develops a Minecraft agent, which can solve complex task via sub-goal decomposition. Similar agents like GITM [16] and Voyager [38] also heavily rely on the planning capability of LLMs to successfully complete different tasks. (2) Conversation capability. LLMs can usually generate high-quality conversations. This capability enables the agent to behave more like humans. In the previous work, many agents take actions based on the strong conversation capability of LLMs. For example, in ChatDev [18], different agents can discuss the software developing process, and even can make reflections on their own behaviors. In RLP [30], the agent can communicate with the listeners based on their potential feedback on the agent's utterance. (3) Common sense understanding capability. Another important capability of

LLMs is that they can well comprehend human common sense. Based on this capability, many agents can simulate human daily life and make human-like decisions. For example, in Generative Agent, the agent can accurately understand its current state, the surrounding environment, and summarize high-level ideas based on basic observations. Without the common sense understanding capability of LLMs, these behaviors cannot be reliably simulated. Similar conclusions may also apply to RecAgent [21] and S^3 [77], where the agents aim to simulate user recommendation and social behaviors.

Action impact: Action impact refers to the consequences of the action. In fact, the action impact can encompass numerous instances, but for brevity, we only provide a few examples. (1) Changing environments. Agents can directly alter environment states by actions, such as moving their positions, collecting items, and constructing buildings. For instance, in GITM [16] and Voyager [38], the environments are changed by the actions of the agents in their task completion process. For example, if the agent mines three woods, then they may disappear in the environments. (2) Altering internal states. Actions taken by the agent can also change the agent itself, including updating memories, forming new plans, acquiring novel knowledge, and more. For example, in Generative Agents [20], memory streams are updated after performing actions within the system. SayCan [78] enables agents to take actions to update understandings of the environment. (3) Triggering new actions. In the task completion process, one agent action can be triggered by another one. For example, Voyager [38] constructs buildings once it has gathered all the necessary resources.

2.2 Agent capability acquisition

In the above sections, we mainly focus on how to design the agent architecture to better inspire the capability of LLMs to make it qualified for accomplishing tasks like humans. The architecture functions as the “hardware” of the agent. However, relying solely on the hardware is insufficient for achieving effective task performance. This is because the agent may lack the necessary task-specific capabilities, skills and experiences, which can be regarded as “software” resources. In order to equip the agent with these resources, various strategies have been devised. Generally, we categorize these strategies into two classes based on whether they require fine-tuning of the LLMs. In the following, we introduce each of them more in detail.

Capability acquisition with fine-tuning: A straightforward method to enhance the agent capability for task completion is fine-tuning the agent based on task-dependent datasets. Generally, the datasets can be constructed based on human annotation, LLM generation or collected from real-world applications. In the following, we introduce these methods more in detail.

● Fine-tuning with human annotated datasets. To fine-tune the agent, utilizing human annotated datasets is a versatile approach that can be employed in various application scenarios. In this approach, researchers first design annotation tasks and then recruit workers to complete them. For example, in CoH [79], the authors aim to align LLMs with human values and preferences. Different from the other models, where the human feedback is leveraged in a simple and symbolic manner, this method converts the human feedback into detailed comparison information in the form of natural languages. The LLMs are directly fine-tuned based on these natural language datasets. In RET-LLM [42], in order to better convert natural languages into structured memory information, the authors fine-tune LLMs based on a human constructed dataset, where each sample is a “triplet-natural language” pair. In WebShop [80], the authors collect 1.18 million real-world products form amazon.com, and put them onto a simulated e-commerce website, which contains several carefully designed human shopping scenarios. Based on this website, the authors recruit 13 workers to collect a real-human behavior dataset. At last, three methods based on heuristic rules, imitation learning and reinforcement learning are trained based on this dataset. Although the authors do not fine-tune LLM-based agents, we believe that the dataset proposed in this paper holds immense potential to enhance the capabilities of agents in the field of Web shopping. In EduChat [81], the authors aim to enhance the educational functions of LLMs, such as open-domain question answering, essay assessment, Socratic teaching, and emotional support. They fine-tune LLMs based on human annotated datasets that cover various educational scenarios and tasks. These datasets are manually evaluated and curated by psychology experts and frontline teachers. SWIFTSAGE [82] is an agent influenced by the dual-process theory of human cognition [83], which is effective for solving complex interactive reasoning tasks. In this agent, the SWIFT module constitutes a compact encoder-decoder language model, which is fine-tuned using human-annotated datasets.

● Fine-tuning with LLM generated datasets. Building human annotated dataset needs to recruit people, which can be costly, especially when one needs to annotate a large amount of samples. Considering that LLMs can achieve human-like capabilities in a wide range of tasks, a natural idea is using LLMs to accomplish the annotation task. While the datasets produced from this method can be not as perfect as the human annotated ones, it is much cheaper, and can be leveraged to generate more samples. For example, in ToolBench [14], to enhance the tool-using capability of open-source LLMs, the authors collect 16,464 real-world APIs spanning 49 categories from the RapidAPI Hub. They used these APIs to prompt ChatGPT to generate diverse instructions, covering both single-tool and multi-tool scenarios. Based on the obtained dataset, the authors fine-tune LLaMA [9], and obtain significant performance improvement in terms of tool using. In [84], to empower the agent with social capability, the authors design a sandbox, and deploy multiple agents to interact with each other. Given a social question, the central agent first generates initial responses. Then, it shares the responses to its nearby agents for collecting their feedback. Based on the feedback as well as its detailed explanations, the central agent revise its initial responses to make them more consistent with social norms. In this process, the authors collect a large amount of agent social interaction data, which is then leveraged to fine-tune the LLMs.

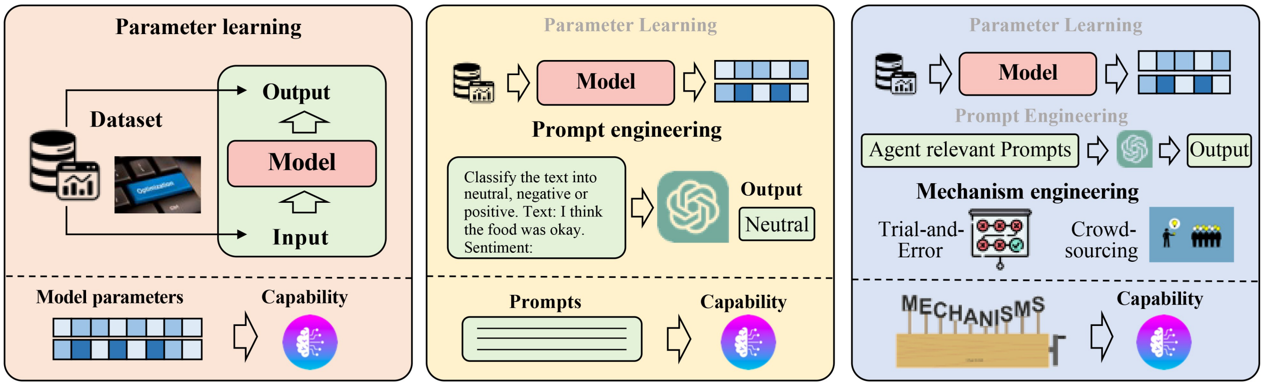
- Fine-tuning with real-world datasets. In addition to building datasets based on human or LLM annotation, directly using real-world datasets to fine-tune the agent is also a common strategy. For example, in MIND2WEB [85], the authors collect a large amount of real-world datasets to enhance the agent capability in the Web domain. In contrast to prior studies, the dataset presented in this paper encompasses diverse tasks, real-world scenarios, and comprehensive user interaction patterns. Specifically, the authors collect over 2,000 open-ended tasks from 137 real-world websites spanning 31 domains. Using this dataset, the authors fine-tune LLMs to enhance their performance on Web-related tasks, including movie discovery and ticket booking, among others. In SQL-PALM [86], researchers fine-tune PaLM-2 based on a cross-domain large-scale text-to-SQL dataset called Spider. The obtained model can achieve significant performance improvement on text-to-SQL tasks.

Capability acquisition without fine-tuning: In the era of tradition machine learning, the model capability is mainly acquired by learning from datasets, where the knowledge is encoded into the model parameters. In the era of LLMs, the model capability can be acquired either by training/fine-tuning the model parameters or designing delicate prompts (i.e., prompt engineer). In prompt engineer, one needs to write valuable information into the prompts to enhance the model capability or unleash existing LLM capabilities. In the era of agents, the model capability can be acquired based on three strategies: (1) model fine-tuning, (2) prompt engineer, and (3) designing proper agent evolution mechanisms (we called it as *mechanism engineering*). Mechanism engineering is a broad concept that involves developing specialized modules, introducing novel working rules, and other strategies to enhance agent capabilities. For clearly understanding such transitions on the strategy of model capability acquisition, we illustrate them in Fig. 4. In the following, we introduce prompting engineering and mechanism engineering for agent capability acquisition.

- Prompting engineering. Due to the strong language comprehension capabilities, people can directly interact with LLMs using natural languages. This introduces a novel strategy for enhancing agent capabilities, that is, one can describe the desired capability using natural language and then use it as prompts to influence LLM actions. For example, in CoT [45], in order to empower the agent with the capability for complex task reasoning, the authors present the intermediate reasoning steps as few-shot examples in the prompt. Similar techniques are also used in CoT-SC [49] and ToT [50]. In SocialAGI [30], in order to enhance the agent self-awareness capability in conversation, the authors prompt LLMs with the agent beliefs about the mental states of the listeners and itself, which makes the generated utterance more engaging and adaptive. In addition, the authors also incorporate the target mental states of the listeners, which enables the agents to make more strategic plans. Retroformer [87] presents a retrospective model that enables the agent to generate reflections on its past failures. The reflections are integrated into the prompt of LLMs to guide the agent's future actions. Additionally, this model utilizes reinforcement learning to iteratively improve the retrospective model, thereby refining the LLM prompt.

- Mechanism engineering. Unlike model fine-tuning and prompt engineering, mechanism engineering is a unique strategy to enhance agent capability. In the following, we present several representative methods of mechanism engineering.

(1) Trial-and-error. In this method, the agent first performs an action, and subsequently, a pre-defined critic is invoked to judge the action. If the action is deemed unsatisfactory, then the agent reacts by incorporating the critic's feedback. In RAH [88], the agent serves as a user assistant in recommender systems. One of the agent's crucial roles is to simulate human behavior and generate responses on behalf of the user. To fulfill this objective, the agent first generates a predicted response and then compares it with the real human feedback. If the predicted response and the real human feedback differ, the critic generates failure information, which is subsequently incorporated into the agent's next action. In DEPS [33], the agent first designs a plan to accomplish a given task. In the plan execution process, if an action fails, the explainer generates specific details explaining the cause of the failure. This information is then incorporated by the agent to redesign the plan. In RoCo [89], the agent first proposes a sub-task plan and a path of 3D waypoints for each robot in a multi-robot collaboration task. The plan and waypoints are then validated



Classify the text into neutral, negative or positive. Text: I think the food was okay. Sentiment:

The era of machine learning   The era of large language model   The era of agent

Fig. 4   Illustration of transitions in strategies for acquiring model capabilities

by a set of environment checks, such as collision detection and inverse kinematics. If any of the checks fail, the feedback is appended to each agent's prompt and another round of dialog begins. The agents use LLMs to discuss and improve their plan and waypoints until they pass all validations. In PREFER [90], the agent first evaluates its performance on a subset of data. If it fails to solve certain examples, LLMs are leveraged to generate feedback information reflecting on the reasons of the failure. Based on this feedback, the agent improves itself by iteratively refining its actions.

(2) Crowd-sourcing. In [91], the authors design a debating mechanism that leverages the wisdom of crowds to enhance agent capabilities. To begin with, different agents provide separate responses to a given question. If their responses are not consistent, they will be prompted to incorporate the solutions from other agents and provide an updated response. This iterative process continues until reaching a final consensus answer. In this method, the capability of each agent is enhanced by understanding and incorporating the other agents' opinions.

(3) Experience accumulation. In GITM [16], the agent does not know how to solve a task in the beginning. Then, it makes explorations, and once it has successfully accomplished a task, the actions used in this task are stored into the agent memory. In the future, if the agent encounters a similar task, then the relevant memories are extracted to complete the current task. In this process, the improved agent capability comes from the specially designed memory accumulation and utilization mechanisms. In Voyager [38], the authors equip the agent with a skill library, and each skill in the library is represented by executable codes. In the agent-environment interaction process, the codes for each skill will be iteratively refined according to the environment feedback and the agent self-verification results. After a period of execution, the agent can successfully complete different tasks efficiently by accessing the skill library. In AppAgent [92], the agent is designed to interact with apps in a manner akin to human users, learning through both autonomous exploration and observation of human demonstrations. Throughout this process, it constructs a knowledge base, which serves as a reference for performing intricate tasks across various applications on a mobile phone. In MemPrompt [93], the users are requested to provide feedback in natural language regarding the problem-solving intentions of the agent, and this feedback is stored in memory. When the agent encounters similar tasks, it attempts to retrieve related memories to generate more suitable responses.

(4) Self-driven evolution. In LMA3 [94], the agent can autonomously set goals for itself, and gradually improve its capability by exploring the environment and receiving feedback from a reward function. Following this mechanism, the agent can acquire knowledge and develop capabilities according to its own preferences. In SALLM-MS [95], by integrating advanced large language models like GPT-4 into a multi-agent system, agents can adapt and perform complex tasks, showcasing advanced communication capabilities, thereby realizing self-driven evolution in their interactions with the environment. In CLMTWA [96], by using a large language model as a teacher and a weaker language model as a student, the teacher can generate and communicate natural language explanations to improve the student's reasoning skills via theory of mind. The teacher can also personalize its explanations for the student and intervene only when necessary, based on the expected utility of intervention. In NLSOM [97], different agents communicate and collaborate through natural language to solve tasks that a single agent cannot solve. This can be seen as a form of self-driven learning, utilizing the exchange of information and knowledge between multiple agents. However, unlike other models such as LMA3, SALLM-MS, and CLMTWA, NLSOM allows for dynamic adjustment of agent roles, tasks, and relationships based on the task requirements and feedback from other agents or the environment.

Remark. Upon comparing the aforementioned strategies for agent capability acquisition, we can find that the fine-tuning method improves the agent capability by adjusting model parameters, which can incorporate a large amount of task-specific knowledge, but is only suitable for open-source LLMs. The method without fine-tuning usually enhances the agent capability based on delicate prompting strategies or mechanism engineering. They can be used for both open- and closed-source LLMs. However, due to the limitation of the input context window of LLMs, they cannot incorporate too much task information. In addition, the designing spaces of the prompts and mechanisms are extremely large, which makes it not easy to find optimal solutions.

In the above sections, we have detailed the construction of LLM-based agents, where we focus on two aspects including the architecture design and capability acquisition. We present the correspondence between existing work and the above taxonomy in Table 1. It should be noted that, for the sake of integrity, we have also incorporated several studies, which do not explicitly mention LLM-based agents but are highly related to this area.

3 LLM-based autonomous agent application

Owing to the strong language comprehension, complex task reasoning, and common sense understanding capabilities, LLM-based autonomous agents have shown significant potential to influence multiple domains. This section provides a succinct summary of previous studies, categorizing them according to their applications in three distinct areas: social science, natural science, and engineering (see the left part of Fig. 5 for a global overview).

3.1 Social science

Social science is one of the branches of science, devoted to the study of societies and the relationships among individuals within those societies. LLM-based autonomous agents can promote this domain by leveraging their impressive human-like understanding, thinking and task solving capabilities. In the following, we discuss several key areas that can be affected by LLM-based autonomous agents.

Psychology: For the domain of psychology, LLM-based agents can be leveraged for conducting simulation experiments, providing mental health support and so on

Table 1 For the profile module, we use ①, ②, and ③ to represent the handcrafting method, LLM-generation method, and dataset alignment method, respectively. For the memory module, we focus on the implementation strategies for memory operation and memory structure. For memory operation, we use ① and ② to indicate that the model only has read/write operations and has read/write/reflection operations, respectively. For memory structure, we use ① and ② to represent unified and hybrid memories, respectively. For the planning module, we use ① and ② to represent planning w/o feedback and w/ feedback, respectively. For the action module, we use ① and ② to represent that the model does not use tools and use tools, respectively. For the agent capability acquisition (CA) strategy, we use ① and ② to represent the methods with and without fine-tuning, respectively. "–" indicates that the corresponding content is not explicitly discussed in the paper

Model	Profile	Memory		Planning	Action	CA	Time
		Operation	Structure				
WebGPT [66]	–	–	–	–	②	①	12/2021
SayCan [78]	–	–	–	①	①	②	04/2022
MRKL [72]	–	–	–	①	②	–	05/2022
Inner Monologue [61]	–	–	–	②	①	②	07/2022
Social Simulacra [98]	②	–	–	–	①	–	08/2022
ReAct [59]	–	–	–	②	②	①	10/2022
MALLM [43]	–	①	②	–	①	–	01/2023
DEPS [33]	–	–	–	②	①	②	02/2023
Toolformer [15]	–	–	–	①	②	①	02/2023
Reflexion [12]	–	②	②	②	①	②	03/2023
CAMEL [99]	① ②	–	–	②	①	–	03/2023
API-Bank [69]	–	–	–	②	②	②	04/2023
ViperGPT [74]	–	–	–	–	②	–	03/2023
HuggingGPT [13]	–	–	①	①	②	–	03/2023
Generative Agents [20]	①	②	②	②	①	–	04/2023
LLM+P [57]	–	–	–	①	①	–	04/2023
ChemCrow [75]	–	–	–	②	②	–	04/2023
OpenAGI [73]	–	–	–	②	②	①	04/2023
AutoGPT [100]	–	①	②	②	②	②	04/2023
SCM [35]	–	②	②	–	①	–	04/2023
Socially Alignment [84]	–	①	②	–	①	①	05/2023
GITM [16]	–	②	②	②	①	②	05/2023
Voyager [38]	–	②	②	②	①	②	05/2023
Introspective Tips [101]	–	–	–	②	①	②	05/2023
RET-LLM [42]	–	①	②	–	①	①	05/2023
ChatDB [40]	–	①	②	②	②	–	06/2023
S^3 [77]	③	②	②	–	①	–	07/2023
ChatDev [18]	①	②	②	②	①	②	07/2023
ToolLLM [14]	–	–	–	②	②	①	07/2023
MemoryBank [39]	–	②	②	–	①	–	07/2023
MetaGPT [23]	①	②	②	②	②	–	08/2023

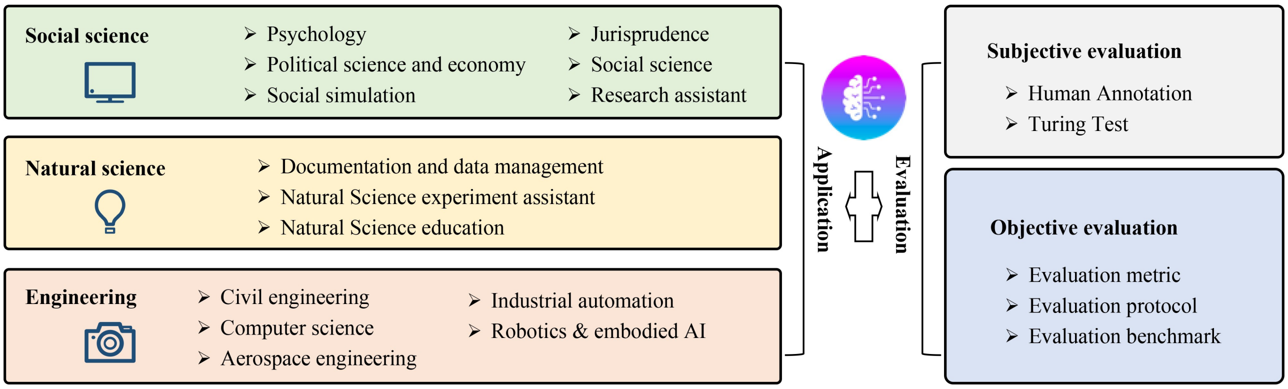


Fig. 5 The applications (left) and evaluation strategies (right) of LLM-based agents

[102–105]. For example, in [102], the authors assign LLMs with different profiles, and let them complete psychology experiments. From the results, the authors find that LLMs are capable of generating results that align with those from studies involving human participants. Additionally, it was observed that larger models tend to deliver more accurate simulation results compared to their smaller counterparts. An interesting discovery is that, in many experiments, models like ChatGPT and GPT-4 can provide too perfect estimates (called "hyper-accuracy distortion"), which may influence the downstream applications. In [104], the authors systematically analyze the effectiveness of LLM-based conversation agents for mental well-being support. They collect 120 posts from Reddit, and find that such agents can help users cope with anxieties, social isolation and depression on demand. At the same time, they also find that the agents may produce harmful contents sometimes.

Political science and economy: LLM-based agents can also be utilized to study political science and economy [29,105,106]. In [29], LLM-based agents are utilized for ideology detection and predicting voting patterns. In [105], the authors focuses on understanding the discourse structure and persuasive elements of political speech through the assistance of LLM-based agents. In [106], LLM-based agents are provided with specific traits such as talents, preferences, and personalities to explore human economic behaviors in simulated scenarios.

Social simulation: Previously, conducting experiments with human societies is often expensive, unethical, or even infeasible. With the ever prospering of LLMs, many people explore to build virtual environment with LLM-based agents to simulate social phenomena, such as the propagation of harmful information, and so on [20,34,77,79,107–110]. For example, Social Simulacra [79] simulates an online social community and explores the potential of utilizing agent-based simulations to aid decision-makers to improve community regulations. [107,108] investigates the potential impacts of different behavioral characteristics of LLM-based agents in social networks. Generative Agents [20] and AgentSims [34] construct multiple agents in a virtual town to simulate the human daily life. SocialAI School [109] employs LLM-based agents to simulate and investigate the fundamental social cognitive skills during the course of child development. S^3 [77] builds a social network simulator, focusing on the propagation of information, emotion and attitude. CGMI [111] is a framework for multi-agent simulation. CGMI maintains the personality of the agents through a tree structure and constructs a cognitive model. The authors simulated a classroom scenario using CGMI.

Jurisprudence: LLM-based agents can serve as aids in legal decision-making processes, facilitating more informed judgements [112,113]. Blind Judgement [113] employs several language models to simulate the decision-making processes of multiple judges. It gathers diverse opinions and consolidates the outcomes through a voting mechanism. ChatLaw [112] is a prominent Chinese legal model based on LLM. It adeptly supports both database and keyword search strategies, specifically designed to mitigate the hallucination issue prevalent in such models. In addition, this model also employs self-attention mechanism to enhance the LLM's capability via mitigating the impact of reference inaccuracies.

Research assistant: Beyond their application in specialized domains, LLM-based agents are increasingly adopted as versatile assistants in the broad field of social science research [105,114]. In [105], LLM-based agents offer multifaceted assistance, ranging from generating concise article abstracts and extracting pivotal keywords to crafting detailed scripts for studies, showcasing their ability to enrich and streamline the research process. Meanwhile, in [114], LLM-based agents serve as a writing assistant, demonstrating their capability to identify novel research inquiries for social scientists, thereby opening new avenues for exploration and innovation in the field. These examples highlight the potential of LLM-based agents in enhancing the efficiency, creativity, and breadth of social science research.

3.2 Natural science

Natural science is one of the branches of science concerned with the description, understanding and prediction of natural phenomena, based on empirical evidence from observation and experimentation. With the ever prospering of LLMs, the application of LLM-based agents in natural sciences becomes more and more popular. In the following, we present many representative areas, where LLM-based agents can play important roles.

Documentation and data management: Natural scientific research often involves the collection, organization, and synthesis of substantial amounts of literature, which requires a significant dedication of time and human resources. LLM-based agents have shown strong capabilities on language understanding and employing tools such as the internet and databases for text processing. These capabilities empower the agent to excel in tasks related to documentation and data management [75,115,116]. In [115], the agent can efficiently query and utilize internet information to complete tasks such as question answering and experiment planning. ChatMOF [116] utilizes LLMs to extract important information from text descriptions written by humans. It then formulates a plan to apply relevant tools for predicting the properties and structures of metal-organic frameworks. ChemCrow [75] utilizes chemistry-related databases to both validate the precision of compound representations and identify potentially dangerous substances. This functionality enhances the reliability and comprehensiveness of scientific inquiries by ensuring the accuracy of the data involved.

Experiment assistant: LLM-based agents have the ability to independently conduct experiments, making them valuable tools for supporting scientists in their research projects [75,115]. For instance, [115] introduces an innovative agent system that utilizes LLMs for automating the design, planning, and execution of scientific experiments. This system, when provided with the experimental objectives as input, accesses the Internet and retrieves relevant documents to gather the necessary information. It subsequently utilizes Python code to conduct essential calculations and carry out the following experiments. ChemCrow [75] incorporates 17 carefully developed tools that are specifically designed to assist researchers in their chemical research. Once the input objectives are received, ChemCrow provides valuable recommendations for experimental procedures, while also emphasizing any potential safety risks associated with the proposed experiments.

Natural science education: LLM-based agents can communicate with humans fluently, often being utilized to develop agent-based educational tools [115,117–119]. For example, [115] develops agent-based education systems to facilitate students learning of experimental design, methodologies, and analysis. The objective of these systems is to enhance students' critical thinking and problem-solving skills, while also fostering a deeper comprehension of scientific principles. Math Agents [117] can assist researchers in exploring, discovering, solving and proving mathematical problems. Additionally, it can communicate with humans and aids them in understanding and using mathematics. [118] utilize the capabilities of CodeX [119] to automatically solve and explain university-level mathematical problems, which can be used as education tools to teach students and researchers. CodeHelp [120] is an education agent for programming. It offers many useful features, such as setting course-specific keywords, monitoring student queries, and providing feedback to the system. EduChat [81] is an LLM-based agent designed specifically for the education domain. It provides personalized, equitable, and empathetic educational support to teachers, students, and parents through dialogue. FreeText [121] is an agent that utilizes LLMs to automatically assess students' responses to open-ended questions and offer feedback.

3.3 Engineering

LLM-based autonomous agents have shown great potential in assisting and enhancing engineering research and applications. In this section, we review and summarize the applications of LLM-based agents in several major engineering domains.

Civil engineering: In civil engineering, LLM-based agents can be used to design and optimize complex structures such as buildings, bridges, dams, roads. [122] proposes an interactive framework where human architects and agents collaborate to construct structures in a 3D simulation environment. The interactive agent can understand natural language instructions, place blocks, detect confusion, seek clarification, and incorporate human feedback, showing the potential for human-AI collaboration in engineering design.

Computer science & software engineering: In the field of computer science and software engineering, LLM-based agents offer potential for automating coding, testing, debugging, and documentation generation [14,18,23,24,123–125]. ChatDev [18] proposes an end-to-end framework, where multiple agent roles communicate and collaborate through natural language conversations to complete the software development life cycle. This framework demonstrates efficient and cost-effective generation of executable software systems. ToolBench [14] can be used for tasks such as code auto-completion and code recommendation. MetaGPT [23] abstracts multiple roles, such as product managers, architects, project managers, and engineers, to supervise code generation process and enhance the quality of the final output code. This enables low-cost software development. [24] presents a self-collaboration framework for code generation using LLMs. In this framework, multiple LLMs are assumed to be distinct "experts" for specific sub-tasks. They collaborate and interact according to specified instructions, forming a virtual team that facilitates each other's work. Ultimately, the virtual team collaboratively addresses code generation tasks without requiring human intervention. LLIFT [126] employs LLMs to assist in conducting static analysis, specifically for identifying potential code vulnerabilities. This approach effectively manages the trade-off between accuracy and scalability. ChatEDA [127] is an agent developed for electronic design automation (EDA) to streamline the design process by integrating task planning, script generation, and execution. CodeHelp [120] is an agent designed to assist students and developers in debugging and testing their code. Its features include providing detailed explanations of error messages, suggesting potential fixes, and ensuring the accuracy of the code. PENTESTGPT [128] is a penetration testing tool based on LLMs, which can effectively identify common vulnerabilities, and interpret source code to develop exploits. DB-GPT [41] utilizes the capabilities of LLMs to systematically assess potential root causes of anomalies in databases. Through the implementation of a tree of thought approach, DB-GPT enables LLMs to backtrack to previous steps in case the current step proves unsuccessful, thus enhancing the accuracy of the diagnosis process.

Industrial automation: In the field of industrial automation, LLM-based agents can be used to achieve intelligent planning and control of production processes. [129] proposes a novel framework that integrates large language models (LLMs) with digital twin systems to accommodate flexible production needs. The framework leverages prompt engineering techniques to create LLM agents that can adapt to specific tasks based on the information provided by digital twins. These agents can coordinate a series of atomic functionalities and skills to complete production tasks at different levels within the automation pyramid. This research demonstrates the potential of integrating LLMs into industrial automation systems, providing innovative solutions for more agile, flexible and adaptive production processes. IELLM [130] showcases a case study on LLMs' role in the oil and gas industry, covering applications like rock physics, acoustic reflectometry, and coiled tubing control.

Robotics & embodied artificial intelligence: Recent works have developed more efficient reinforcement learning agents for robotics and embodied artificial intelligence [16,38,78,131–138]. The focus is on enhancing autonomous agents' abilities for planning, reasoning, and collaboration in embodied environments. In specific, [135] proposes a unified agent system for embodied reasoning and task planning. In this system, the authors design high-level commands to enable improved planning while propose low-level controllers to translate commands into actions. Additionally, one can leverage dialogues to gather information [136] to accelerate the optimization process. [137,138] employ autonomous agents for embodied decision-making and exploration. To overcome the physical constraints, the agents can generate executable plans and accomplish long-term tasks by leveraging multiple skills. In terms of control policies, SayCan [78] focuses on investigating a wide range of manipulation and navigation skills utilizing a mobile manipulator robot. Taking inspiration from typical tasks encountered in a kitchen environment, it presents a comprehensive set of 551 skills that cover seven skill families and 17 objects. These skills encompass various actions such as picking, placing, pouring, grasping, and manipulating objects, among others. TidyBot [139] is an embodied agent designed to personalize household cleanup tasks. It can learn users' preferences on object placement and manipulation methods through textual examples.

To promote the application of LLM-based autonomous agents, researchers have also introduced many open-source libraries, based on which the developers can quickly implement and evaluate agents according to their customized requirements [19,108,124,140–153]. For example, LangChain [145] is an open-source framework that automates coding, testing, debugging, and documentation generation tasks. By integrating language models with data sources and facilitating interaction with the environment, LangChain enables efficient and cost-effective software development through natural language communication and collaboration among multiple agent roles. Based on LangChain, XLang [143] comes with a comprehensive set of tools, a complete user interface, and support three different agent scenarios, namely data processing, plugin usage, and Web agent. AutoGPT [100] is an agent that is fully automated. It sets one or more goals, breaks them down into corresponding tasks, and cycles through the tasks until the goal is achieved. WorkGPT [146] is an agent framework similar to AutoGPT and LangChain. By providing it with an instruction and a set of APIs, it engages in back-and-forth conversations with AI until the instruction is completed. GPT-Engineer [125], SmolModels [123] and DemoGPT [124] are open-source projects that focus on automating code generation through prompts to complete development tasks. AGiXT [142] is a dynamic AI automation platform designed to orchestrate efficient AI command management and task execution across many providers. AgentVerse [19] is a versatile framework that facilitates researchers in creating customized LLM-based agent simulations efficiently. GPT Researcher [148] is an experimental application that leverages large language models to efficiently develop research questions, trigger Web crawls to gather information, summarize sources, and aggregate summaries. BMTools [149] is an open-source repository that extends LLMs with tools and provides a platform for community-driven tool building and sharing. It supports various types of tools, enables simultaneous task execution using multiple tools, and offers a simple interface for loading plugins via URLs, fostering easy development and contribution to the BMTools ecosystem.

Remark. Utilization of LLM-based agents in supporting above applications may also entail risks and challenges. On one hand, LLMs themselves may be susceptible to illusions and other issues, occasionally providing erroneous answers, leading to incorrect conclusions, experimental failures, or even posing risks to human safety in hazardous experiments. Therefore, during experimentation, users must possess the necessary expertise and knowledge to exercise appropriate caution. On the other hand, LLM-based agents could potentially be exploited for malicious purposes, such as the development of chemical weapons, necessitating the implementation of security measures, such as human alignment, to ensure responsible and ethical use.

In summary, in the above sections, we introduce the typical applications of LLM-based autonomous agents in three important domains. To facilitate a clearer understanding, we have summarized the relationship between previous studies and their respective applications in Table 2.

4 LLM-based autonomous agent evaluation

Similar to LLMs themselves, evaluating the effectiveness of LLM-based autonomous agents is a challenging task. This section outlines two prevalent approaches to evaluation: subjective and objective methods. For a comprehensive overview, please refer to the right portion of Fig. 5.

4.1 Subjective evaluation

Subjective evaluation measures the agent capabilities based on human judgements [20,22,29,79,158]. It is suitable for the scenarios where there are no evaluation datasets or it is very

Table 2 Representative applications of LLM-based autonomous agents

Domain		Work
Social Science	Psychology	TE [102], Akata et al. [103], Ziems et al. [105], Ma et al. [104]
	Political Science and Economy	Out of One [29], Horton [106], Ziems et al. [105]
	Social Simulation	Social Simulacra [79], Generative Agents [20], SocialAI School [109], AgentSims [34], S^3 [77], Williams et al. [110], Li et al. [107], Chao et al. [108]
	Jurisprudence	ChatLaw [112], Blind Judgement [113]
	Research Assistant	Ziems et al. [105], Bail et al. [114]
Natural Science	Documentation and Data Management	ChemCrow [75], Boiko et al. [115]
	Experiment Assistant	ChemCrow [75], Boiko et al. [115], Grossmann et al. [154]
	Natural Science Education	ChemCrow [75], CodeHelp [120], Boiko et al. [115], MathAgent [117], Drori et al. [118]
Engineering	CS & SE	RestGPT [70], Self-collaboration [24], SQL-PALM [86], RAH [88], DB-GPT [41], RecMind [51], ChatEDA [127], InteRecAgent [155], PentestGPT [128], CodeHelp [120], SmolModels [123], DemoGPT [124], GPTEngineer [125]
	Industrial Automation	GPT4IA [129], IELLM [130], TaskMatrix.AI [71]
	Robotics & Embodied AI	ProAgent [156], LLM4RL [131], PET [132], REMEMBERER [133], DEPS [33], Unified Agent [134], SayCan [78], LMMWM [157], TidyBot [139], RoCo [89], SayPlan [31]

hard to design quantitative metrics, for example, evaluating the agent's intelligence or user-friendliness. In the following, we present two commonly used strategies for subjective evaluation.

Human annotation: This evaluation method involves human evaluators directly scoring or ranking the outputs generated by various agents [22,29,102]. For example, in [20], the authors employ many annotators, and ask them to provide feedback on five key questions that directly associated with the agent capability. Similarly, [159] assess model effectiveness by having human participants rate the models on harmlessness, honesty, helpfulness, engagement, and unbiasedness, subsequently comparing these scores across different models. In [79], annotators are asked to determine whether the specifically designed models can significantly enhance the development of rules within online communities.

Turing test: This evaluation strategy necessitates that human evaluators differentiate between outputs produced by agents and those created by humans. If, in a given task, the evaluators cannot separate the agent and human results, it demonstrates that the agent can achieve human-like performance on this task. For instance, researchers in [29] conduct experiments on free-form Partisan text, and the human evaluators are asked to guess whether the responses are from human or LLM-based agent. In [20], the human evaluators are required to identify whether the behaviors are generated from the agents or real-humans. In EmotionBench [160], human annotations are collected to compare the emotional states expressed by LLM software and human participants across various scenarios. This comparison serves as a benchmark for evaluating the emotional intelligence of the LLM software, illustrating a nuanced approach to understanding agent capabilities in mimicking human-like performance and emotional expression.

Remark. LLM-based agents are usually designed to serve humans. Thus, subjective agent evaluation plays a critical role, since it reflects human criterion. However, this strategy also faces issues such as high costs, inefficiency, and population bias. To address these issues, a growing number of researchers are investigating the use of LLMs themselves as intermediaries for carrying out these subjective assessments. For example, in ChemCrow [75], researchers assess the experimental results using GPT. They consider both the completion of tasks and the accuracy of the underlying processes. Similarly, ChatEval [161] introduces a novel approach by employing multiple agents to critique and assess the results generated by various candidate models in a structured debate format. This innovative use of LLMs for evaluation purposes holds promise for enhancing both the credibility and applicability of subjective assessments in the future. As LLM technology continues to evolve, it is anticipated that these methods will become increasingly reliable and find broader applications, thereby overcoming the current limitations of direct human evaluation.

4.2 Objective evaluation

Objective evaluation refers to assessing the capabilities of LLM-based autonomous agents using quantitative metrics that can be computed, compared and tracked over time. In contrast to subjective evaluation, objective metrics aim to provide concrete, measurable insights into the agent performance. For conducting objective evaluation, there are three important aspects, that is, the evaluation metrics, protocols and benchmarks. In the following, we introduce these aspects more in detail.

Metrics: In order to objectively evaluate the effectiveness of the agents, designing proper metrics is significant, which may influence the evaluation accuracy and comprehensiveness. Ideal evaluation metrics should precisely reflect the quality of the agents, and align with the human feelings when using them in real-world scenarios. In existing work, we can conclude the following representative evaluation metrics. (1) Task *success metrics*: These metrics measure how well an agent can complete tasks and achieve goals. Common metrics include success rate [12,22,57,59], reward/score [22,59,122], coverage [16], and accuracy [18,40,102]. Higher values indicate greater task completion ability. (2) *Human similarity metrics*: These metrics quantify the degree to which the agent behaviors closely resembles that of humans. Typical examples include trajectory/location accuracy [38,162], dialogue similarities [79,102], and mimicry of human responses [29,102]. Higher similarity suggests better human simulation performance. (3) *Efficiency metrics*: In contrast to the aforementioned metrics used to evaluate the agent effectiveness, these metrics aim to assess the efficiency of agent. Commonly considered metrics encompass the length of planning [57], the cost associated with development [18], the speed of inference [16,38], and number of clarification dialogues [122].

Protocols: In addition to the evaluation metrics, another important aspect for objective evaluation is how to leverage these metrics. In the previous work, we can identify the following commonly used evaluation protocols: (1) *Real-world simulation*: In this method, the agents are evaluated within immersive environments like games and interactive simulators. The agents are required to perform tasks autonomously, and then metrics like task success rate and human similarity are leveraged to evaluate the capability of the agents based on their trajectories and completed objectives [16,22,33,38,59,80,122,162,163,164]. This method is expected to evaluate the agents' practical capabilities in real-world scenarios. (2) *Social evaluation*: This method utilizes metrics to assess social intelligence based on the agent interactions in simulated societies. Various approaches have been adopted, such as collaborative tasks to evaluate teamwork skills, debates to analyze argumentative reasoning, and human studies to measure social aptitude [34,98,102,165,166]. These approaches analyze qualities such as coherence, theory of mind, and social IQ to assess agents' capabilities in areas including cooperation, communication, empathy, and mimicking human social behavior. By subjecting agents to complex interactive settings, social evaluation provides valuable insights into agents' higher-level social cognition. (3) *Multi-task evaluation*: In this method, people use a set of diverse tasks from different domains to evaluate the agent, which can effectively measure the agent generalization capability in open-domain environments [29,80,153,163,165,166,167]. (4) *Software testing*: In this method, researchers evaluate the agents by letting them conduct tasks such as software testing tasks, such as generating test cases, reproducing bugs, debugging code, and interacting with developers and external tools [166,168,169,170]. Then, one can use metrics like test coverage and bug detection rate to measure the effectiveness of LLM-based agents.

Benchmarks: Given the metrics and protocols, a crucial remaining aspect is the selection of an appropriate benchmark for conducting the evaluation. In the past, people have used various benchmarks in their experiments. For example, many researchers use simulation environments like ALFWorld [59], IGLU [122], and Minecraft [16,33,38] as benchmarks to evaluate the agent capabilities. Tachikuma [164] is a benchmark that leverages TRPG game logs to evaluate LLMs' ability to understand and infer complex interactions with multiple characters and novel objects. AgentBench [167] provides a comprehensive framework for evaluating LLMs as autonomous agents across diverse environments. It represents the first systematic assessment of LLMs as agents on real-world challenges across diverse domains. SocKET [165] is a comprehensive benchmark for evaluating the social capabilities of LLMs across 58 tasks covering five categories of social information such as humor and sarcasm, emotions and feelings, and credibility. AgentSims [34] is a versatile framework for evaluating LLM-based agents, where one can flexibly design the agent planning, memory and action strategies, and measure the effectiveness of different agent modules in interactive environments. ToolBench [149] is an open-source project that aims to support the development of powerful LLMs with general tool-use capability. It provides an open platform for training, serving, and evaluating LLMs based on tool learning. WebShop [80] develops a benchmark for evaluating LLM-based agents in terms of their capabilities for product search and retrieval. The benchmark is constructed using a collection of 1.18 million real-world items. Mobile-Env [163] is an extendable interactive platform which can be used to evaluate the multi-step interaction capabilities of LLM-based agents. WebArena [171] offers a comprehensive website environment that spans multiple domains. Its purpose is to evaluate agents in an end-to-end fashion and determine the accuracy of their completed tasks. GentBench [172] is a benchmark designed to evaluate the agent capabilities, including their reasoning, safety, and efficiency, when utilizing tools to complete complex tasks. RocoBench [89] is a benchmark with six tasks evaluating multi-agent collaboration across diverse scenarios, emphasizing communication and coordination strategies to assess adaptability and generalization in cooperative robotics. EmotionBench [160] evaluates the emotion appraisal ability of LLMs, i.e., how their feelings change when presented with specific situations. It collects over 400 situations that elicit eight negative emotions and measures the emotional states of LLMs and human subjects using self-report scales. PEB [128] is a benchmark tailored for assessing LLM-based agents in penetration testing scenarios, comprising 13 diverse targets from leading platforms. It offers a structured evaluation across varying difficulty levels, reflecting real-world challenges for agents. ClemBench [173] contains five Dialogue Games to assess LLMs' ability as a player. E2E [174] is an end-to-end benchmark for testing the accuracy and usefulness of chatbots.

Remark. Objective evaluation facilitates the quantitative analysis of capabilities in LLM-based agents through a variety of metrics. While current techniques can not perfectly measure all types of agent capabilities, objective evaluation provides essential insights that complement subjective assessment. Continued advancements in benchmarks and methodologies for objective evaluation will enhance the development and understanding of LLM-based autonomous agents further.

In the above sections, we introduce both subjective and objective strategies for LLM-based autonomous agents evaluation. The evaluation of the agents play significant roles in this domain. However, both subjective and objective evaluation have their own strengths and weakness. Maybe, in practice, they should be combined to comprehensively evaluate the agents. We summarize the correspondence between the previous work and these evaluation strategies in Table 3.

5 Related surveys

With the vigorous development of large language models, a variety of comprehensive surveys have emerged, providing detailed insights into various aspects. [176] extensively introduces the background, main findings, and mainstream technologies of LLMs, encompassing a vast array of existing works. On the other hand, [177] primarily focus on the applications of LLMs in various downstream tasks and the challenges associated with their deployment. Aligning LLMs with human intelligence is an active area of research to address concerns such as biases and illusions. [178] have compiled existing techniques for human alignment, including data collection and model training methodologies. Reasoning is a crucial aspect of intelligence, influencing decision-making, problem-solving, and other cognitive abilities. [179] presents the current state of research on LLMs' reasoning abilities, exploring approaches to improve and evaluate their reasoning skills. [180] propose that language models can be enhanced with reasoning capabilities and the ability to utilize tools, termed Augmented Language Models (ALMs). They conduct a comprehensive review of the latest advancements in ALMs. As the utilization of large-scale models becomes more prevalent, evaluating their performance is increasingly critical. [181] shed light on evaluating LLMs, addressing what to evaluate, where to evaluate, and how to assess their performance in downstream tasks and societal impact. [182] also discusses the capabilities and limitations of LLMs in various downstream tasks. The aforementioned research encompasses various aspects of large models, including training, application, and evaluation. However, prior to this paper, no work has specifically focused on the rapidly emerging and highly promising field of LLM-based Agents. In this study, we have compiled 100 relevant works on LLM-based Agents, covering their construction, applications, and evaluation processes.

Table 3 For subjective evaluation, we use ① and ② to represent human annotation and the Turing test, respectively. For objective evaluation, we use ①, ②, ③, and ④ to represent environment simulation, social evaluation, multi-task evaluation, and software testing, respectively. "✓" indicates that the evaluations are based on benchmarks

Model	Subjective	Objective	Benchmark	Time
WebShop [80]	–	① ③	✓	07/2022
Social Simulacra [98]	①	②	–	08/2022
TE [102]	–	②	–	08/2022
LIBRO [168]	–	④	–	09/2022
ReAct [59]	–	①	✓	10/2022
Out of One, Many [29]	②	② ③	–	02/2023
DEPS [33]	–	①	✓	02/2023
Jalil et al. [169]	–	④	–	02/2023
Reflexion [12]	–	① ③	–	03/2023
IGLU [122]	–	①	✓	04/2023
Generative Agents [20]	① ②	–	–	04/2023
ToolBench [149]	–	③	✓	04/2023
GITM [16]	–	①	✓	05/2023
Two-Failures [162]	–	③	–	05/2023
Voyager [38]	–	①	✓	05/2023
SocKET [165]	–	② ③	✓	05/2023
MobileEnv [163]	–	① ③	✓	05/2023
Clembench [173]	–	① ③	✓	05/2023
Dialop [175]	–	②	✓	06/2023
Feldt et al. [170]	–	④	–	06/2023
CO-LLM [22]	①	①	–	07/2023
Tachikuma [164]	①	①	✓	07/2023
WebArena [171]	–	①	✓	07/2023
RocoBench [89]	–	① ② ③	–	07/2023
AgentSims [34]	–	②	–	08/2023
AgentBench [167]	–	③	✓	08/2023
BOLAA [166]	–	① ③ ④	✓	08/2023
Gentopia [172]	–	③	✓	08/2023
EmotionBench [160]	①	–	✓	08/2023
PTB [128]	–	④	–	08/2023

6 Challenges

While previous work on LLM-based autonomous agent has obtained many remarkable successes, this field is still at its initial stage, and there are several significant challenges that need to be addressed in its development. In the following, we present many representative challenges.

6.1 Role-playing capability

Different from traditional LLMs, autonomous agent usually has to play as specific roles (e.g., program coder, researcher, and chemist) for accomplishing different tasks. Thus, the capability of the agent for role-playing is very important. Although LLMs can effectively simulate many common roles such as movie reviewers, there are still various roles and aspects that they struggle to capture accurately. To begin with, LLMs are usually trained based on web-corpus, thus for the roles which are seldom discussed on the Web or the newly emerging roles, LLMs may not simulate them well. In addition, previous research [30] has shown that existing LLMs may not well model the human cognitive psychology characters, leading to the lack of self-awareness in conversation scenarios. Potential solution to these problems may include fine-tuning LLMs or carefully designing the agent prompts/architectures [183]. For example, one can firstly collect real-human data for uncommon roles or psychology characters, and then leverage it to fine-tune LLMs. However, how to ensure that fine-tuned model still perform well for the common roles may pose further challenges. Beyond fine-tuning, one can also design tailored agent prompts/architectures to enhance the capability of LLM on role-playing. However, finding the optimal prompts/architectures is not easy, since their designing spaces are too large.

6.2 Generalized human alignment

Human alignment has been discussed a lot for traditional LLMs. In the field of LLM-based autonomous agent, especially when the agents are leveraged for simulation, we believe this concept should be discussed more in depth. In order to better serve human-beings, traditional LLMs are usually fine-tuned to be aligned with correct human values, for example, the agent should not plan to make a bomb for avenging society. However, when the agents are leveraged for real-world simulation, an ideal simulator should be able to honestly depict diverse human traits, including the ones with incorrect values. Actually, simulating the human negative aspects can be even more important, since an important goal of simulation is to discover and solve problems, and without negative aspects means no problem to be solved. For example, to simulate the real-world society, we may have to allow the agent to plan for making a bomb, and observe how it will act to implement the plan as well as the influence of its behaviors. Based on these observations, people can make better actions to stop similar behaviors in real-world society. Inspired by the above case, maybe an important problem for agent-based simulation is how to conduct generalized human alignment, that is, for different purposes and applications, the agent should be able to align with diverse human values. However, existing powerful LLMs including ChatGPT and GPT-4 are mostly aligned with unified human values. Thus, an interesting direction is how to "realign" these models by designing proper prompting strategies.

6.3 Prompt robustness

To ensure rational behavior in agents, it's a common practice for designers to embed supplementary modules, such as memory and planning modules, into LLMs. However, the inclusion of these modules necessitates the development of more complex prompts in order to facilitate consistent operation and effective communication. Previous research [184,185] has highlighted the lack of robustness in prompts for LLMs, as even minor alterations can yield substantially different outcomes. This issue becomes more pronounced when constructing autonomous agents, as they encompass not a single prompt but a prompt framework that considers all modules, wherein the prompt for one module has the potential to influence others. Moreover, the prompt frameworks can vary significantly across different LLMs. The development of a unified and resilient prompt framework applicable across diverse LLMs remains a critical and unresolved challenge. There are two potential solutions to the aforementioned problems: (1) manually crafting the essential prompt elements through trial and error, or (2) automatically generating prompts using GPT.

6.4 Hallucination

Hallucination poses a fundamental challenge for LLMs, characterized by the models' tendency to produce false information with a high level of confidence. This challenge is not limited to LLMs alone but is also a significant concern in the domain of autonomous agents. For instance, in [186], it was observed that when confronted with simplistic instructions during code generation tasks, the agent may exhibit hallucinatory behavior. Hallucination can lead to serious consequences such as incorrect or misleading code, security risks, and ethical issues [186]. To mitigate this issue, incorporating human correction feedback directly into the iterative process of human-agent interaction presents a viable approach [23]. More discussions on the hallucination problem can be seen in [176].

6.5 Knowledge boundary

A pivotal application of LLM-based autonomous agents lies in simulating diverse real-world human behaviors [20]. The study of human simulation has a long history, and the recent surge in interest can be attributed to the remarkable advancements made by LLMs, which have demonstrated significant capabilities in simulating human behavior. However, it is important to recognize that the power of LLMs may not always be advantageous. Specifically, an ideal simulation should accurately replicate human knowledge. In this context, LLMs may display overwhelming capabilities, being trained on a vast corpus of Web knowledge that far exceeds what an average individual might know. The immense capabilities of LLMs can significantly impact the effectiveness of simulations. For instance, when attempting to simulate user selection behaviors for various movies, it is crucial to ensure that LLMs assume a position of having no prior knowledge of these movies. However, there is a possibility that LLMs have already acquired information about these movies. Without implementing appropriate strategies, LLMs may make decisions based on their extensive knowledge, even though real-world users would not have access to the contents of these movies beforehand. Based on the above example, we may conclude that for building believable agent simulation environment, an important problem is how to constrain the utilization of user-unknown knowledge of LLM.

6.6 Efficiency

Due to their autoregressive architecture, LLMs typically have slow inference speeds. However, the agent may need to query LLMs for each action multiple times, such as extracting information from memory, make plans before taking actions and so on. Consequently, the efficiency of agent actions is greatly affected by the speed of LLM inference.

7 Conclusion

In this survey, we systematically summarize existing research in the field of LLM-based autonomous agents. We present and review these studies from three aspects including the construction, application, and evaluation of the agents. For each of these aspects, we provide a detailed taxonomy to draw connections among the existing research, summarizing the major techniques and their development histories. In addition to reviewing the previous work, we also propose several challenges in this field, which are expected to guide potential future directions.

Acknowledgements This work was supported in part by the National Natural Science Foundation of China (Grant No. 62102420), the Beijing Outstanding Young Scientist Program (No. BJJWZYJH012019100020098), Intelligent Social Governance Platform, Major Innovation & Planning Interdisciplinary Platform for the "Double-First Class" Initiative, Renmin University of China, Public Computing Cloud, Renmin University of China, fund for building world-class universities (disciplines) of Renmin University of China, Intelligent Social Governance Platform.

Competing interests The authors declare that they have no competing interests or financial conflicts to disclose.

References

1. Mnih V, Kavukcuoglu K, Silver D, Rusu A A, Veness J, Bellemare M G, Graves A, Riedmiller M, Fidjeland A K, Ostrovski G, Petersen S, Beattie C, Sadik A, Antonoglou I, King H, Kumaran D, Wierstra D, Legg S, Hassabis D. Human-level control through deep reinforcement learning. Nature, 2015, 518(7540): 529–533
2. Lillicrap T P, Hunt J J, Pritzel A, Heess N, Erez T, Tassa Y, Silver D, Wierstra D. Continuous control with deep reinforcement learning. 2019, arXiv preprint arXiv: 1509.02971
3. Schulman J, Wolski F, Dhariwal P, Radford A, Klimov O. Proximal policy optimization algorithms. 2017, arXiv preprint arXiv: 1707.06347
4. Haarnoja T, Zhou A, Abbeel P, Levine S. Soft actor-critic: off-policy maximum entropy deep reinforcement learning with a stochastic actor. In: Proceedings of the 35th International Conference on Machine Learning. 2018, 1861−1870
5. Brown T B, 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 M, 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, Amodei D. Language models are few-shot learners. In: Proceedings of the 34th Conference on Neural Information Processing Systems. 2020, 1877−1901
6. Radford A, Wu J, Child R, Luan D, Amodei D, Sutskever I. Language models are unsupervised multitask learners. OpenAI Blog, 2019, 1(8): 9
7. OpenAI. GPT-4 technical report. 2024, arXiv preprint arXiv: 2303.08774
8. Anthropic. Model card and evaluations for Claude models. See Files.anthropic.com/production/images/Model-Card-Claude-2, 2023
9. 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, Lample G. LLaMA: open and efficient foundation language models. 2023, arXiv preprint arXiv: 2302.13971
10. Touvron H, Martin L, Stone K, Albert P, Almahairi A, et al. Llama 2: open foundation and fine-tuned chat models. 2023, arXiv preprint arXiv: 2307.09288
11. Chen X, Li S, Li H, Jiang S, Qi Y, Song L. Generative adversarial user model for reinforcement learning based recommendation system. In: Proceedings of the 36th International Conference on Machine Learning. 2019, 1052−1061
12. Shinn N, Cassano F, Gopinath A, rasimhan K, Yao S. Reflexion: language agents with verbal reinforcement learning. NaIn: Proceedings of the 37th Conference on Neural Information Processing Systems. 2023, 36
13. Shen Y, Song K, Tan X, Li D, Lu W, Zhuang Y. HuggingGPT: solving AI tasks with chatGPT and its friends in hugging face. In: Proceedings of the 37th Conference on Neural Information Processing Systems. 2023, 36
14. Qin Y, Liang S, Ye Y, Zhu K, Yan L, Lu Y, Lin Y, Cong X, Tang X, Qian B, Zhao S, Hong L, Tian R, Xie R, Zhou J, Gerstein M, Li D, Liu Z, Sun M. ToolLLM: facilitating large language models to master 16000+ real-world APIs. 2023, arXiv preprint arXiv: 2307.16789
15. Schick T, Dwivedi-Yu J, Dessì R, Raileanu R, Lomeli M, Hambro E, Zettlemoyer L, Cancedda N, Scialom T. Toolformer: language models can teach themselves to use tools. In: Proceedings of the 37th Conference on Neural Information Processing Systems. 2023, 36
16. Zhu X, Chen Y, Tian H, Tao C, Su W, Yang C, Huang G, Li B, Lu L, Wang X, Qiao Y, Zhang Z, Dai J. Ghost in the minecraft: generally capable agents for open-world environments via large language models with text-based knowledge and memory. 2023, arXiv preprint arXiv: 2305.17144
17. Sclar M, Kumar S, West P, Suhr A, Choi Y, Tsvetkov Y. Minding language models' (lack of) theory of mind: a plug-and-play multi-character belief tracker. In: Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics. 2023, 13960–13980
18. Qian C, Cong X, Liu W, Yang C, Chen W, Su Y, Dang Y, Li J, Xu J, Li S, Liu Z, Sun M. Communicative agents for software development. 2023, arXiv preprint arXiv: 2307.07924
19. Chen W, Su Y, Zuo J, Yang C, Yuan C, Chan C, Yu H, Lu Y, Hung Y, Qian C, Qin Y, Cong X, Xie R, Liu Z, Sun M, Zhou, J. Agentverse: Facilitating multi-agent collaboration and exploring emergent behaviors in agents. arXiv preprint arXiv:2308.10848 .
20. Park J S, O'Brien J, Cai C J, Morris M R, Liang P, Bernstein M S. Generative agents: interactive simulacra of human behavior. In: Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology. 2023, 2
21. Zhang H, Du W, Shan J, Zhou Q, Du Y, Tenenbaum J B, Shu T, Gan C. Building cooperative embodied agents modularly with large language models. 2024, arXiv preprint arXiv: 2307.02485
22. Hong S, Zhuge M, Chen J, Zheng X, Cheng Y, Zhang C, Wang J, Wang Z, Yau S K S, Lin Z, Zhou L, Ran C, Xiao L, Wu C, Schmidhuber J. MetaGPT: meta programming for a multi-agent collaborative framework. 2023, arXiv preprint arXiv: 2308.00352
23. Dong Y, Jiang X, Jin Z, Li G. Self-collaboration code generation via chatGPT. 2023, arXiv preprint arXiv: 2304.07590
24. Serapio-García G, Safdari M, Crepy C, Sun L, Fitz S, Romero P, Abdulhai M, Faust A, Matarić M. Personality traits in large language models. 2023, arXiv preprint arXiv: 2307.00184
25. Johnson J A. Measuring thirty facets of the five factor model with a 120-item public domain inventory: development of the IPIP-NEO-120. Journal of Research in Personality, 2014, 51: 78–89
26. John O P, Donahue E M, Kentle R L. Big five inventory. Journal of personality and social psychology, 1991.
27. Deshpande A, Murahari V, Rajpurohit T, Kalyan A, Narasimhan K. Toxicity in chatGPT: analyzing persona-assigned language models. In: Proceedings of Findings of the Association for Computational Linguistics. 2023, 1236–1270
28. Wang L, Zhang J, Yang H, Chen Z, Tang J, Zhang Z, Chen X, Lin Y, Song R, Zhao W X, Xu J, Dou Z, Wang J, Wen J R. User behavior simulation with large language model based agents. 2024, arXiv preprint arXiv: 2306.02552
29. Argyle L P, Busby E C, Fulda N, Gubler J R, Rytting C, Wingate D. Out of one, many: using language models to simulate human samples. Political Analysis, 2023, 31(3): 337–351
30. Fischer K A. Reflective linguistic programming (RLP): a stepping stone in socially-aware AGI (socialAGI). 2023, arXiv preprint arXiv: 2305.12647
31. Rana K, Haviland J, Garg S, Abou-Chakra J, Reid I, Suenderhauf N. SayPlan: grounding large language models using 3D scene graphs for scalable robot task planning. In: Proceedings of the 7th Conference on Robot Learning. 2023, 23−72
32. Zhu A, Martin L, Head A, Callison-Burch C. CALYPSO: LLMs as dungeon master's assistants. In: Proceedings of the 19th AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment. 2023, 380−390
33. Wang Z, Cai S, Chen G, Liu A, Ma X, Liang Y. Describe, explain, plan and select: interactive planning with large language models enables open-world multi-task agents. 2023, arXiv preprint arXiv: 2302.01560
34. Lin J, Zhao H, Zhang A, Wu Y, Ping H, Chen Q. AgentSims: an open-source sandbox for large language model evaluation. 2023, arXiv

preprint arXiv: 2308.04026

35. Wang B, Liang X, Yang J, Huang H, Wu S, Wu P, Lu L, Ma Z, Li Z. Enhancing large language model with self-controlled memory framework. 2024, arXiv preprint arXiv: 2304.13343
36. Ng Y, Miyashita D, Hoshi Y, Morioka Y, Torii O, Kodama T, Deguchi J. SimplyRetrieve: a private and lightweight retrieval-centric generative AI tool. 2023, arXiv preprint arXiv: 2308.03983
37. Huang Z, Gutierrez S, Kamana H, Macneil S. Memory sandbox: transparent and interactive memory management for conversational agents. In: Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology. 2023, 97
38. Wang G, Xie Y, Jiang Y, Mandlekar A, Xiao C, Zhu Y, Fan L, Anandkumar A. Voyager: an open-ended embodied agent with large language models. 2023, arXiv preprint arXiv: 2305.16291
39. Zhong W, Guo L, Gao Q, Ye H, Wang Y. MemoryBank: enhancing large language models with long-term memory. 2023, arXiv preprint arXiv: 2305.10250
40. Hu C, Fu J, Du C, Luo S, Zhao J, Zhao H. ChatDB: augmenting LLMs with databases as their symbolic memory. 2023, arXiv preprint arXiv: 2306.03901
41. Zhou X, Li G, Liu Z. LLM as DBA. 2023, arXiv preprint arXiv: 2308.05481
42. Modarressi A, Imani A, Fayyaz M, Schütze H. RET-LLM: towards a general read-write memory for large language models. 2023, arXiv preprint arXiv: 2305.14322
43. Schuurmans D. Memory augmented large language models are computationally universal. 2023, arXiv preprint arXiv: 2301.04589
44. Zhao A, Huang D, Xu Q, Lin M, Liu Y J, Huang G. Expel: LLM agents are experiential learners. 2023, arXiv preprint arXiv: 2308.10144
45. Wei J, Wang X, Schuurmans D, Bosma M, Ichter B, Xia F, Chi E H, Le Q V, Zhou D. Chain-of-thought prompting elicits reasoning in large language models. In: Proceedings of the 36th Conference on Neural Information Processing Systems. 2022, 24824−24837
46. Kojima T, Gu S S, Reid M, Matsuo Y, Iwasawa Y. Large language models are zero-shot reasoners. In: Proceedings of the 36th Conference on Neural Information Processing Systems. 2022, 22199−22213
47. Raman S S, Cohen V, Rosen E, Idrees I, Paulius D, Tellex S. Planning with large language models via corrective re-prompting. In: Proceedings of Foundation Models for Decision Making Workshop at Neural Information Processing Systems. 2022
48. Xu B, Peng Z, Lei B, Mukherjee S, Liu Y, Xu D. ReWOO: decoupling reasoning from observations for efficient augmented language models. 2023, arXiv preprint arXiv: 2305.18323
49. Wang X, Wei J, Schuurmans D, Le Q V, Chi E H, Narang S, Chowdhery A, Zhou D. Self-consistency improves chain of thought reasoning in language models. In: Proceedings of the 11th International Conference on Learning Representations. 2023
50. Yao S, Yu D, Zhao J, Shafran I, Griffiths T L, Cao Y, Narasimhan K. Tree of thoughts: deliberate problem solving with large language models. In: Proceedings of the 37th Conference on Neural Information Processing Systems. 2023, 36
51. Wang Y, Jiang Z, Chen Z, Yang F, Zhou Y, Cho E, Fan X, Huang X, Lu Y, Yang Y. RecMind: Large language model powered agent for recommendation. 2023, arXiv preprint arXiv: 2308.14296
52. Besta M, Blach N, Kubicek A, Gerstenberger R, Podstawski M, Gianinazzi L, Gajda J, Lehmann T, Niewiadomski H, Nyczyk P, Hoefler T. Graph of thoughts: solving elaborate problems with large language models. 2024, arXiv preprint arXiv: 2308.09687
53. Sel B, Al-Tawaha A, Khattar V, Jia R, Jin M. Algorithm of thoughts: enhancing exploration of ideas in large language models. 2023, arXiv preprint arXiv: 2308.10379
54. Huang W, Abbeel P, Pathak D, Mordatch I. Language models as zero-shot planners: extracting actionable knowledge for embodied agents. In: Proceedings of the 39th International Conference on Machine Learning. 2022, 9118−9147
55. Gramopadhye M, Szafir D. Generating executable action plans with environmentally-aware language models. In: Proceedings of 2023 IEEE/RSJ International Conference on Intelligent Robots and Systems. 2023, 3568−3575
56. Hao S, Gu Y, Ma H, Hong J, Wang Z, Wang D, Hu Z. Reasoning with language model is planning with world model. In: Proceedings of 2023 Conference on Empirical Methods in Natural Language Processing. 2023, 8154–8173
57. Liu B, Jiang Y, Zhang X, Liu Q, Zhang S, Biswas J, Stone P. LLM+P: empowering large language models with optimal planning proficiency. 2023, arXiv preprint arXiv: 2304.11477
58. Dagan G, Keller F, Lascarides A. Dynamic planning with a LLM. 2023, arXiv preprint arXiv: 2308.06391
59. Yao S, Zhao J, Yu D, Du N, Shafran I, Narasimhan K R, Cao Y. ReAct: synergizing reasoning and acting in language models. In: Proceedings of the 11th International Conference on Learning Representations. 2023
60. Song C H, Sadler B M, Wu J, Chao W L, Washington C, Su Y. LLM-planner: few-shot grounded planning for embodied agents with large language models. In: Proceedings of 2023 IEEE/CVF International Conference on Computer Vision. 2023, 2986−2997
61. Huang W, Xia F, Xiao T, Chan H, Liang J, Florence P, Zeng A, Tompson J, Mordatch I, Chebotar Y, Sermanet P, Jackson T, Brown N, Luu L, Levine S, Hausman K, Ichter B. Inner monologue: embodied reasoning through planning with language models. In: Proceedings of the 6th Conference on Robot Learning, 2023, 1769−1782
62. Madaan A, Tandon N, Gupta P, Hallinan S, Gao L, Wiegreffe S, Alon U, Dziri N, Prabhumoye S, Yang Y, Gupta S, Majumder B P, Hermann K, Welleck S, Yazdanbakhsh A, Clark P. Self-refine: iterative refinement with self-feedback. Advances in Neural Information Processing Systems, 2024, 36.
63. Miao N, Teh Y W, Rainforth T. SelfCheck: using LLMs to zero-shot check their own step-by-step reasoning. 2023, arXiv preprint arXiv: 2308.00436
64. Chen P L, Chang C S. InterAct: exploring the potentials of chatGPT as a cooperative agent. 2023, arXiv preprint arXiv: 2308.01552
65. Chen Z, Zhou K, Zhang B, Gong Z, Zhao X, Wen J R. ChatCoT: tool-augmented chain-of-thought reasoning on chat-based large language models. In: Proceedings of Findings of the Association for Computational Linguistics. 2023, 14777–14790
66. Nakano R, Hilton J, Balaji S, Wu J, Ouyang L, Kim C, Hesse C, Jain S, Kosaraju V, Saunders W, Jiang X, Cobbe K, Eloundou T, Krueger G, Button K, Knight M, Chess B, Schulman J. WebGPT: browser-assisted question-answering with human feedback. 2022, arXiv preprint arXiv: 2112.09332
67. Ruan J, Chen Y, Zhang B, Xu Z, Bao T, Du G, Shi S, Mao H, Li Z, Zeng X, Zhao R. TPTU: large language model-based AI agents for task planning and tool usage. 2023, arXiv preprint arXiv: 2308.03427
68. Patil S G, Zhang T, Wang X, Gonzalez J E. Gorilla: large language model connected with massive APIs. 2023, arXiv preprint arXiv: 2305.15334
69. Li M, Zhao Y, Yu B, Song F, Li H, Yu H, Li Z, Huang F, Li Y. API-bank: a comprehensive benchmark for tool-augmented LLMs. In: Proceedings of 2023 Conference on Empirical Methods in Natural Language Processing. 2023, 3102–3116
70. Song Y, Xiong W, Zhu D, Wu W, Qian H, Song M, Huang H, Li C, Wang K, Yao R, Tian Y, Li S. RestGPT: connecting large language models with real-world RESTful APIs. 2023, arXiv preprint arXiv: 2306.06624
71. Liang Y, Wu C, Song T, Wu W, Xia Y, Liu Y, Ou Y, Lu S, Ji L, Mao S, Wang Y, Shou L, Gong M, Duan N. TaskMatrix.AI: Completing

tasks by connecting foundation models with millions of APIs. 2023, arXiv preprint arXiv: 2303.16434

72. Karpas E, Abend O, Belinkov Y, Lenz B, Lieber O, Ratner N, Shoham Y, Bata H, Levine Y, Leyton-Brown K, Muhlgay D, Rozen N, Schwartz E, Shachaf G, Shalev-Shwartz S, Shashua A, Tenenholtz M. MRKL systems: a modular, neuro-symbolic architecture that combines large language models, external knowledge sources and discrete reasoning. 2022, arXiv preprint arXiv: 2205.00445
73. Ge Y, Hua W, Mei K, Tan J, Xu S, Li Z, Zhang Y. OpenAGI: When LLM meets domain experts. In: Proceedings of the 37th Conference on Neural Information Processing Systems, 2023, 36
74. Surís D, Menon S, Vondrick C. ViperGPT: visual inference via python execution for reasoning. 2023, arXiv preprint arXiv: 2303.08128
75. Bran A M, Cox S, Schilter O, Baldassari C, White A D, Schwaller P. ChemCrow: augmenting large-language models with chemistry tools. 2023, arXiv preprint arXiv: 2304.05376
76. Yang Z, Li L, Wang J, Lin K, Azarnasab E, Ahmed F, Liu Z, Liu C, Zeng M, Wang L. MM-REACT: Prompting chatGPT for multimodal reasoning and action. 2023, arXiv preprint arXiv: 2303.11381
77. Gao C, Lan X, Lu Z, Mao J, Piao J, Wang H, Jin D, Li Y. S3: social-network simulation system with large language model-empowered agents. 2023, arXiv preprint arXiv: 2307.14984
78. Ichter B, Brohan A, Chebotar Y, Finn C, Hausman K, et al. Do as I can, not as I say: grounding language in robotic affordances. In: Proceedings of the 6th Conference on Robot Learning. 2023, 287−318
79. Liu H, Sferrazza C, Abbeel P. Chain of hindsight aligns language models with feedback. arXiv preprint arXiv: 2302.02676
80. Yao S, Chen H, Yang J, Narasimhan K. WebShop: towards scalable real-world Web interaction with grounded language agents. In: Proceedings of the 36th Conference on Neural Information Processing Systems. 2022, 20744−20757
81. Dan Y, Lei Z, Gu Y, Li Y, Yin J, Lin J, Ye L, Tie Z, Zhou Y, Wang Y, Zhou A, Zhou Z, Chen Q, Zhou J, He L, Qiu X. EduChat: a large-scale language model-based chatbot system for intelligent education. 2023, arXiv preprint arXiv: 2308.02773
82. Lin B Y, Fu Y, Yang K, Brahman F, Huang S, Bhagavatula C, Ammanabrolu P, Choi Y, Ren X. SwiftSage: a generative agent with fast and slow thinking for complex interactive tasks. In: Proceedings of the 37th Conference on Neural Information Processing Systems. 2023, 36
83. Evans J S B T, Stanovich K E. Dual-process theories of higher cognition: advancing the debate. Perspectives on Psychological Science, 2013, 8(3): 223–241
84. Liu R, Yang R, Jia C, Zhang G, Zhou D, Dai A M, Yang D, Vosoughi S. Training socially aligned language models on simulated social interactions. 2023, arXiv preprint arXiv: 2305.16960
85. Weng X, Gu Y, Zheng B, Chen S, Stevens S, Wang B, Sun H, Su Y. Mind2Web: towards a generalist agent for the Web. In: Proceedings of the 37th Conference on Neural Information Processing Systems. 2023, 36
86. Sun R, Arik S O, Nakhost H, Dai H, Sinha R, Yin P, Pfister T. SQL-PaLm: improved large language model adaptation for text-to-SQL. 2023, arXiv preprint arXiv: 2306.00739
87. Yao W, Heinecke S, Niebles J C, Liu Z, Feng Y, Xue L, Murthy R, Chen Z, Zhang J, Arpit D, Xu R, Mui P, Wang H, Xiong C, Savarese S. Retroformer: retrospective large language agents with policy gradient optimization, 2023, arXiv preprint arXiv: 2308.02151
88. Shu Y, Zhang H, Gu H, Zhang P, Lu T, Li D, Gu N. RAH! RecSys-assistant-human: a human-centered recommendation framework with LLM agents. 2023, arXiv preprint arXiv: 2308.09904
89. Mandi Z, Jain S, Song S. RoCo: dialectic multi-robot collaboration with large language models. 2023, arXiv preprint arXiv: 2307.04738
90. Zhang C, Liu L, Wang J, Wang C, Sun X, Wang H, Cai M. PREFER: prompt ensemble learning via feedback-reflect-refine. 2023, arXiv preprint arXiv: 2308.12033
91. Du Y, Li S, Torralba A, Tenenbaum J B, Mordatch I. Improving factuality and reasoning in language models through multiagent debate. 2023, arXiv preprint arXiv: 2305.14325
92. Zhang C, Yang Z, Liu J, Han Y, Chen X, Huang Z, Fu B, Yu G. AppAgent: multimodal agents as smartphone users. 2023, arXiv preprint arXiv: 2312.13771
93. Madaan A, Tandon N, Clark P, Yang Y. Memory-assisted prompt editing to improve GPT-3 after deployment. In: Proceedings of 2022 Conference on Empirical Methods in Natural Language Processing. 2022, 2833–2861
94. Colas C, Teodorescu L, Oudeyer P Y, Yuan X, Côté M A. Augmenting autotelic agents with large language models. In: Proceedings of the 2nd Conference on Lifelong Learning Agents. 2023, 205–226
95. Nascimento N, Alencar P, Cowan D. Self-adaptive large language model (LLM)-based multiagent systems. In: Proceedings of 2023 IEEE International Conference on Autonomic Computing and Self-Organizing Systems Companion. 2023, 104−109
96. Saha S, Hase P, Bansal M. Can language models teach weaker agents? Teacher explanations improve students via personalization. 2023, arXiv preprint arXiv: 2306.09299
97. Zhuge M, Liu H, Faccio F, Ashley D R, Csordás R, Gopalakrishnan A, Hamdi A, Hammoud H A A K, Herrmann V, Irie K, Kirsch L, Li B, Li G, Liu S, Mai J, Piękos P, Ramesh A, Schlag I, Shi W, Stanić A, Wang W, Wang Y, Xu M, Fan D P, Ghanem B, Schmidhuber J. Mindstorms in natural language-based societies of mind. 2023, arXiv preprint arXiv: 2305.17066
98. Park J S, Popowski L, Cai C, Morris M R, Liang P, Bernstein M S. Social simulacra: creating populated prototypes for social computing systems. In: Proceedings of the 35th Annual ACM Symposium on User Interface Software and Technology. 2022, 74
99. Li G, Hammoud H A A K, Itani H, Khizbullin D, Ghanem B. CAMEL: communicative agents for "mind" exploration of large language model society. 2023, arXiv preprint arXiv: 2303.17760
100. AutoGPT. See Github.com/Significant-Gravitas/Auto, 2023
101. Chen L, Wang L, Dong H, Du Y, Yan J, Yang F, Li S, Zhao P, Qin S, Rajmohan S, Lin Q, Zhang D. Introspective tips: large language model for in-context decision making. 2023, arXiv preprint arXiv: 2305.11598
102. Aher G V, Arriaga R I, Kalai A T. Using large language models to simulate multiple humans and replicate human subject studies. In: Proceedings of the 40th International Conference on Machine Learning. 2023, 337−371
103. Akata E, Schulz L, Coda-Forno J, Oh S J, Bethge M, Schulz E. Playing repeated games with large language models. 2023, arXiv preprint arXiv: 2305.16867
104. Ma Z, Mei Y, Su Z. Understanding the benefits and challenges of using large language model-based conversational agents for mental well-being support. In: Proceedings of AMIA Symposium. 2023, 1105−1114
105. Ziems C, Held W, Shaikh O, Chen J, Zhang Z, Yang D. Can large language models transform computational social science? 2024, arXiv preprint arXiv: 2305.03514
106. Horton J J. Large language models as simulated economic agents: what can we learn from homo silicus? 2023, arXiv preprint arXiv: 2301.07543
107. Li S, Yang J, Zhao K. Are you in a masquerade? Exploring the behavior and impact of large language model driven social bots in online social networks. 2023, arXiv preprint arXiv: 2307.10337
108. Li C, Su X, Han H, Xue C, Zheng C, Fan C. Quantifying the impact of large language models on collective opinion dynamics. 2023, arXiv preprint arXiv: 2308.03313
109. Kovač G, Portelas R, Dominey P F, Oudeyer P Y. The SocialAI

school: insights from developmental psychology towards artificial socio-cultural agents. 2023, arXiv preprint arXiv: 2307.07871

110. Williams R, Hosseinichimeh N, Majumdar A, Ghaffarzadegan N. Epidemic modeling with generative agents. 2023, arXiv preprint arXiv: 2307.04986

111. Shi J, Zhao J, Wang Y, Wu X, Li J, He L. CGMI: configurable general multi-agent interaction framework. 2023, arXiv preprint arXiv: 2308.12503

112. Cui J, Li Z, Yan Y, Chen B, Yuan L. ChatLaw: open-source legal large language model with integrated external knowledge bases. 2023, arXiv preprint arXiv: 2306.16092

113. Hamilton S. Blind judgement: agent-based supreme court modelling with GPT. 2023, arXiv preprint arXiv: 2301.05327

114. Bail C A. Can generative AI improve social science? 2023

115. Boiko D A, MacKnight R, Gomes G. Emergent autonomous scientific research capabilities of large language models. 2023, arXiv preprint arXiv: 2304.05332

116. Kang Y, Kim J. ChatMOF: an autonomous AI system for predicting and generating metal-organic frameworks. 2023, arXiv preprint arXiv: 2308.01423

117. Swan M, Kido T, Roland E, Santos R P D. Math agents: computational infrastructure, mathematical embedding, and genomics. 2023, arXiv preprint arXiv: 2307.02502

118. Drori I, Zhang S, Shuttleworth R, Tang L, Lu A, Ke E, Liu K, Chen L, Tran S, Cheng N, Wang R, Singh N, Patti T L, Lynch J, Shporer A, Verma N, Wu E, Strang G. A neural network solves, explains, and generates university math problems by program synthesis and few-shot learning at human level. Proceedings of the National Academy of Sciences of the United States of America, 2022, 119(32): e2123433119

119. Chen M, Tworek J, Jun H, Yuan Q, de Oliveira Pinto H P, et al. Evaluating large language models trained on code. 2021, arXiv preprint arXiv: 2107.03374

120. Liffiton M, Sheese B E, Savelka J, Denny P. CodeHelp: using large language models with guardrails for scalable support in programming classes. In: Proceedings of the 23rd Koli Calling International Conference on Computing Education Research. 2023, 8

121. Matelsky J K, Parodi F, Liu T, Lange R D, Kording K P. A large language model-assisted education tool to provide feedback on open-ended responses. 2023, arXiv preprint arXiv: 2308.02439

122. Mehta N, Teruel M, Sanz P F, Deng X, Awadallah A H, Kiseleva J. Improving grounded language understanding in a collaborative environment by interacting with agents through help feedback. 2024, arXiv preprint arXiv: 2304.10750

123. SmolModels. See Githubcom/smol-ai/developer website, 2023

124. DemoGPT. See Github.com/melih-unsal/Demo website, 2023

125. GPT-engineer. See Github.com/AntonOsika/gpt website, 2023

126. Li H, Hao Y, Zhai Y, Qian Z. The hitchhiker’s guide to program analysis: a journey with large language models. 2023, arXiv preprint arXiv: 2308.00245

127. He Z, Wu H, Zhang X, Yao X, Zheng S, Zheng H, Yu B. ChatEDA: a large language model powered autonomous agent for EDA. In: Proceedings of the 5th ACM/IEEE Workshop on Machine Learning for CAD. 2023, 1−6

128. Deng G, Liu Y, Mayoral-Vilches V, Liu P, Li Y, Xu Y, Zhang T, Liu Y, Pinzger M, Rass S. PentestGPT: an LLM-empowered automatic penetration testing tool. 2023, arXiv preprint arXiv: 2308.06782

129. Xia Y, Shenoy M, Jazdi N, Weyrich M. Towards autonomous system: flexible modular production system enhanced with large language model agents. In: Proceedings of the 2023 IEEE 28th International Conference on Emerging Technologies and Factory Automation. 2023, 1−8

130. Ogundare O, Madasu S, Wiggins N. Industrial engineering with large language models: a case study of chatGPT’s performance on oil & gas problems. In: Proceedings of the 2023 11th International Conference on Control, Mechatronics and Automation. 2023, 458−461

131. Hu B, Zhao C, Zhang P, Zhou Z, Yang Y, Xu Z, Liu B. Enabling intelligent interactions between an agent and an LLM: a reinforcement learning approach. 2024, arXiv preprint arXiv: 2306.03604

132. Wu Y, Min S Y, Bisk Y, Salakhutdinov R, Azaria A, Li Y, Mitchell T, Prabhumoye S. Plan, eliminate, and track−language models are good teachers for embodied agents. 2023, arXiv preprint arXiv: 2305.02412

133. Zhang D, Chen L, Zhang S, Xu H, Zhao Z, Yu K. Large language models are semi-parametric reinforcement learning agents. In: Proceedings of the 37th Conference on Neural Information Processing Systems. 2023, 36

134. Di P N, Byravan A, Hasenclever L, Wulfmeier M, Heess N, Riedmiller M. Towards a unified agent with foundation models. 2023, arXiv preprint arXiv: 2307.09668

135. Dasgupta I, Kaeser-Chen C, Marino K, Ahuja A, Babayan S, Hill F, Fergus R. Collaborating with language models for embodied reasoning. 2023, arXiv preprint arXiv: 2302.00763

136. Zhou W, Peng X, Riedl M O. Dialogue shaping: empowering agents through NPC interaction. 2023, arXiv preprint arXiv: 2307.15833

137. Nottingham K, Ammanabrolu P, Suhr A, Choi Y, Hajishirzi H, Singh S, Fox R. Do embodied agents dream of pixelated sheep: embodied decision making using language guided world modelling. In: Proceedings of the 40th International Conference on Machine Learning. 2023, 26311–26325

138. Wu Z, Wang Z, Xu X, Lu J, Yan H. Embodied task planning with large language models. 2023, arXiv preprint arXiv: 2307.01848

139. Wu J, Antonova R, Kan A, Lepert M, Zeng A, Song S, Bohg J, Rusinkiewicz S, Funkhouser T. TidyBot: personalized robot assistance with large language models. In: Proceedings of 2023 IEEE/RSJ International Conference on Intelligent Robots and Systems. 2023, 3546−3553

140. AgentGPT. See Github.com/reworkd/Agent website, 2023

141. Ai-legion. See Github.com/eumemic/ai website, 2023

142. AGiXT. See Githubcom/Josh-XT/AGiXT website, 2023

143. Xlang. See Githubcom/xlang-ai/xlang website, 2023

144. Babyagi. See Githubcom/yoheinakajima website, 2023

145. LangChain. See Docs.langchaincom/docs/ website, 2023

146. WorkGPT. See Githubcom/team-openpm/workgpt website, 2023

147. LoopGPT. See Githubcom/farizrahman4u/loopgpt website, 2023

148. GPT-researcher. See Github.com/assafelovic/gpt website, 2023

149. Qin Y, Hu S, Lin Y, Chen W, Ding N, Cui G, Zeng Z, Huang Y, Xiao C, Han C, Fung Y R, Su Y, Wang H, Qian C, Tian R, Zhu K, Liang S, Shen X, Xu B, Zhang Z, Ye Y, Li B, Tang Z, Yi J, Zhu Y, Dai Z, Yan L, Cong X, Lu Y, Zhao W, Huang Y, Yan J, Han X, Sun X, Li D, Phang J, Yang X, Wu T, Ji H, Liu Z, Sun M. Tool learning with foundation models. 2023, arXiv preprint arXiv: 2304.08354

150. Transformers agent. See Huggingface.co/docs/transformers/transformers website, 2023

151. Mini-agi. See Github.com/muellerberndt/mini website, 2023

152. SuperAGI. See Github.com/TransformerOptimus/Super website, 2023

153. Wu Q, Bansal G, Zhang J, Wu Y, Li B, Zhu E, Jiang L, Zhang X, Zhang S, Liu J, Awadallah A H, White R W, Burger D, Wang C. AutoGen: enabling next-gen LLM applications via multi-agent conversation. 2023, arXiv preprint arXiv: 2308.08155

154. Grossmann I, Feinberg M, Parker D C, Christakis N A, Tetlock P E, Cunningham W A. AI and the transformation of social science research: careful bias management and data fidelity are key. Science, 2023, 380(6650): 1108–1109

155. Huang X, Lian J, Lei Y, Yao J, Lian D, Xie X. Recommender AI agent: integrating large language models for interactive recommendations. 2023, arXiv preprint arXiv: 2308.16505

156. Zhang C, Yang K, Hu S, Wang Z, Li G, Sun Y, Zhang C, Zhang Z, Liu A, Zhu S C, Chang X, Zhang J, Yin F, Liang Y, Yang Y. ProAgent: building proactive cooperative agents with large language models. 2024, arXiv preprint arXiv: 2308.11339

157. Xiang J, Tao T, Gu Y, Shu T, Wang Z, Yang Z, Hu Z. Language models meet world models: embodied experiences enhance language models. In: Proceedings of the 37th Conference on Neural Information Processing Systems. 2023, 36

158. Lee M, Srivastava M, Hardy A, Thickstun J, Durmus E, Paranjape A, Gerard-Ursin I, Li X L, Ladhak F, Rong F, Wang R E, Kwon M, Park J S, Cao H, Lee T, Bommasani R, Bernstein M, Liang P. Evaluating human-language model interaction. 2024, arXiv preprint arXiv: 2212.09746

159. Krishna R, Lee D, Fei-Fei L, Bernstein M S. Socially situated artificial intelligence enables learning from human interaction. Proceedings of the National Academy of Sciences of the United States of America, 2022, 119(39): e2115730119

160. Huang J T, Lam M H, Li E J, Ren S, Wang W, Jiao W, Tu Z, Lyu M R. Emotionally numb or empathetic? Evaluating how LLMs feel using emotionbench. 2024, arXiv preprint arXiv: 2308.03656

161. Chan C M, Chen W, Su Y, Yu J, Xue W, Zhang S, Fu J, Liu Z. ChatEval: towards better LLM-based evaluators through multi-agent debate. 2023, arXiv preprint arXiv: 2308.07201

162. Chen A, Phang J, Parrish A, Padmakumar V, Zhao C, Bowman S R, Cho K. Two failures of self-consistency in the multi-step reasoning of LLMs. 2024, arXiv preprint arXiv: 2305.14279

163. Zhang D, Xu H, Zhao Z, Chen L, Cao R, Yu K. Mobile-env: an evaluation platform and benchmark for LLM-GUI interaction. 2024, arXiv preprint arXiv: 2305.08144

164. Liang Y, Zhu L, Yang Y. Tachikuma: understading complex interactions with multi-character and novel objects by large language models. 2023, arXiv preprint arXiv: 2307.12573

165. Choi M, Pei J, Kumar S, Shu C, Jurgens D. Do LLMs understand social knowledge? Evaluating the sociability of large language models with socKET benchmark. In: Proceedings of 2023 Conference on Empirical Methods in Natural Language Processing. 2023, 11370–11403

166. Liu Z, Yao W, Zhang J, Xue L, Heinecke S, Murthy R, Feng Y, Chen Z, Niebles J C, Arpit D, Xu R, Mui P, Wang H, Xiong C, Savarese S. BOLAA: benchmarking and orchestrating LLM-augmented autonomous agents. 2023, arXiv preprint arXiv: 2308.05960

167. Liu X, Yu H, Zhang H, Xu Y, Lei X, Lai H, Gu Y, Ding H, Men K, Yang K, Zhang S, Deng X, Zeng A, Du Z, Zhang C, Shen S, Zhang T, Su Y, Sun H, Huang M, Dong Y, Tang J. AgentBench: evaluating LLMs as agents. 2023, arXiv preprint arXiv: 2308.03688

168. Kang S, Yoon J, Yoo S. Large language models are few-shot testers: exploring LLM-based general bug reproduction. In: Proceedings of the 45th IEEE/ACM International Conference on Software Engineering. 2023, 2312−2323

169. Jalil S, Rafi S, LaToza T D, Moran K, Lam W. ChatGPT and software testing education: Promises & perils. In: Proceedings of 2023 IEEE International Conference on Software Testing, Verification and Validation Workshops. 2023, 4130−4137

170. Feldt R, Kang S, Yoon J, Yoo S. Towards autonomous testing agents via conversational large language models. In: Proceedings of the 38th IEEE/ACM International Conference on Automated Software Engineering. 2023, 1688−1693

171. Zhou S, Xu F F, Zhu H, Zhou X, Lo R, Sridhar A, Cheng X, Ou T, Bisk Y, Fried D, Alon U, Neubig G. WebArena: a realistic Web environment for building autonomous agents. 2023, arXiv preprint arXiv: 2307.13854

172. Xu B, Liu X, Shen H, Han Z, Li Y, Yue M, Peng Z, Liu Y, Yao Z, Xu D. Gentopia.AI: a collaborative platform for tool-augmented LLMs. In: Proceedings of 2023 Conference on Empirical Methods in Natural Language Processing: System Demonstrations. 2023, 237−245

173. Chalamalasetti K, Götze J, Hakimov S, Madureira B, Sadler P, Schlangen D. clembench: Using game play to evaluate chat-optimized language models as conversational agents. In: Proceedings of 2023 Conference on Empirical Methods in Natural Language Processing. 2023, 11174–11219

174. Banerjee D, Singh P, Avadhanam A, Srivastava S. Benchmarking LLM powered chatbots: methods and metrics. 2023, arXiv preprint arXiv: 2308.04624

175. Lin J, Tomlin N, Andreas J, Eisner J. Decision-oriented dialogue for human-AI collaboration. 2023, arXiv preprint arXiv: 2305.20076

176. Zhao W X, Zhou K, Li J, Tang T, Wang X, Hou Y, Min Y, Zhang B, Zhang J, Dong Z, Du Y, Yang C, Chen Y, Chen Z, Jiang J, Ren R, Li Y, Tang X, Liu Z, Liu P, Nie J Y, Wen J R. A survey of large language models. 2023, arXiv preprint arXiv: 2303.18223

177. Yang J, Jin H, Tang R, Han X, Feng Q, Jiang H, Zhong S, Yin B, Hu X. Harnessing the power of LLMs in practice: a survey on chatGPT and beyond. ACM Transactions on Knowledge Discovery from Data, 2024, doi: 10.1145/3649506

178. Wang Y, Zhong W, Li L, Mi F, Zeng X, Huang W, Shang L, Jiang X, Liu Q. Aligning large language models with human: a survey. 2023, arXiv preprint arXiv: 2307.12966

179. Huang J, Chang K C C. Towards reasoning in large language models: a survey. In: Proceedings of Findings of the Association for Computational Linguistics: ACL 2023. 2023, 1049–1065

180. Mialon G, Dessì R, Lomeli M, Nalmpantis C, Pasunuru R, Raileanu R, Rozière B, Schick T, Dwivedi-Yu J, Celikyilmaz A, Grave E, LeCun Y, Scialom T. Augmented language models: a survey. 2023, arXiv preprint arXiv: 2302.07842

181. Chang Y, Wang X, Wang J, Wu Y, Yang L, Zhu K, Chen H, Yi X, Wang C, Wang Y, Ye W, Zhang Y, Chang Y, Yu P S. A survey on evaluation of large language models. ACM Transactions on Intelligent Systems and Technology, 2023, doi: 10.1145/3641289

182. Chang T A, Bergen B K. Language model behavior: a comprehensive survey. Computational Linguistics, 2024, doi: 10.1162/coli_a_00492

183. Li C, Wang J, Zhu K, Zhang Y, Hou W, Lian J, Xie X. Emotionprompt: Leveraging psychology for large language models enhancement via emotional stimulus. 2023, arXiv preprint arXiv: 2307.11760

184. Zhuo T Y, Li Z, Huang Y, Shiri F, Wang W, Haffari G, Li Y F. On robustness of prompt-based semantic parsing with large pre-trained language model: an empirical study on codex. In: Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics. 2023, 1090–1102

185. Gekhman Z, Oved N, Keller O, Szpektor I, Reichart R. On the robustness of dialogue history representation in conversational question answering: a comprehensive study and a new prompt-based method. Transactions of the Association for Computational Linguistics, 2023, 11(11): 351–366

186. Ji Z, Lee N, Frieske R, Yu T, Su D, Xu Y, Ishii E, Bang Y J, Madotto A, Fung P. Survey of hallucination in natural language generation. ACM Computing Surveys, 2023, 55(12): 248

Lei Wang is a PhD candidate at Renmin University of China, China. His research focuses on recommender systems and agent-based large language models.

Chen Ma is currently pursuing a Master's degree at Renmin University of China, China. His research interests include recommender system, agent based on large language model.

Xueyang Feng is currently studying for a PhD degree at Renmin University of China, China. His research interests include recommender system, agent based on large language model.

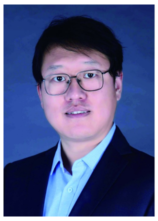
Zeyu Zhang is currently pursuing a Master's degree at Renmin University of China, China. His research interests include recommender system, causal inference, agent based on large language model.

Hao Yang is currently studying for a PhD degree at Renmin University of China, China. His research interests include recommender system, causal inference.

Jingsen Zhang is currently studying for a PhD degree at Renmin University of China, China. His research interests include recommender system.

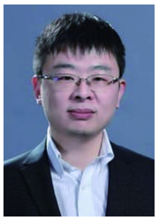
Zhiyuan Chen is pursuing his PhD in Gaoling school of Artificial Intelligence, Renmin University of China, China. His research mainly focuses on language model reasoning and agent based on large language model.

Jiakai Tang is currently pursuing a Master's degree at Renmin University of China, China. His research interests include recommender system.


Xu Chen obtained his PhD degree from Tsinghua University, China. Before joining Renmin University of China, he was a postdoc researcher at University College London, UK. In the period from March to September of 2017, he was studying at Georgia Institute of Technology, USA as a visiting scholar. His research mainly focuses on the recommender system, reinforcement learning, and causal inference.

Yankai Lin received his BE and PhD degrees from Tsinghua University, China in 2014 and 2019, respectively. After that, he worked as a senior researcher in Tencent WeChat, and joined Renmin University of China, China in 2022 as a tenure-track assistant professor. His main research interests are pretrained models and natural language processing.

Wayne Xin Zhao received his PhD degree in Computer Science from Peking University, China in 2014. His research interests include data mining, natural language processing and information retrieval in general. The main goal is to study how to organize, analyze and mine user generated data for improving the service of real-world applications.


Zhewei Wei received his PhD degree in Computer Science and Engineering from The Hong Kong University of Science and Technology, China. He did postdoctoral research in Aarhus University, Denmark from 2012 to 2014, and joined Renmin University of China, China in 2014.

Jirong Wen is a full professor, the executive dean of Gaoling School of Artificial Intelligence, and the dean of School of Information at Renmin University of China, China. He has been working in the big data and AI areas for many years, and publishing extensively on prestigious international conferences and journals.