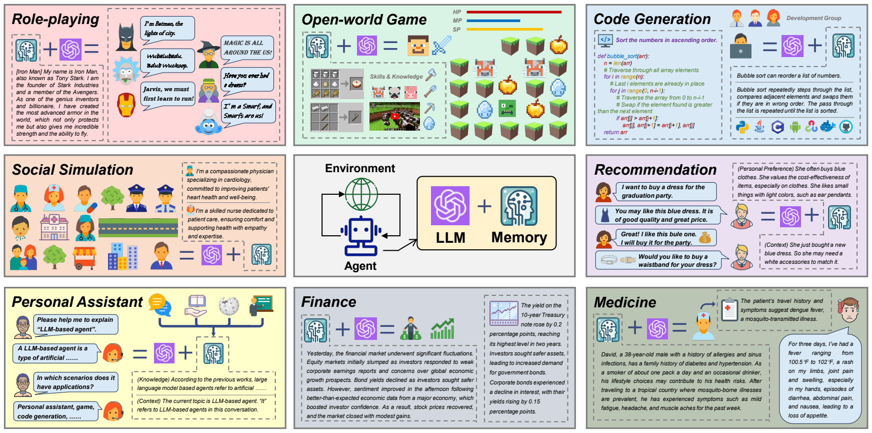
A Survey on the Memory Mechanism of Large Language Model based Agents

Zeyu Zhang[1], **Xiaohe Bo**[1], **Chen Ma**[1], **Rui Li**[1], **Xu Chen**[1], **Quanyu Dai**[2], **Jieming Zhu**[2], **Zhenhua Dong**[2], **Ji-Rong Wen**[1]
[1]Gaoling School of Artificial Intelligence, Renmin University of China, Beijing, China
[2]Huawei Noah's Ark Lab, China
zeyuzhang@ruc.edu.cn, xu.chen@ruc.edu.cn

Abstract

Large language model (LLM) based agents have recently attracted much attention from the research and industry communities. Compared with original LLMs, LLM-based agents are featured in their self-evolving capability, which is the basis for solving real-world problems that need long-term and complex agent-environment interactions. The key component to support agent-environment interactions is the memory of the agents. While previous studies have proposed many promising memory mechanisms, they are scattered in different papers, and there lacks a systematical review to summarize and compare these works from a holistic perspective, failing to abstract common and effective designing patterns for inspiring future studies. To bridge this gap, in this paper, we propose a comprehensive survey on the memory mechanism of LLM-based agents. In specific, we first discuss "what is" and "why do we need" the memory in LLM-based agents. Then, we systematically review previous studies on how to design and evaluate the memory module. In addition, we also present many agent applications, where the memory module plays an important role. At last, we analyze the limitations of existing work and show important future directions. To keep up with the latest advances in this field, we create a repository at `https://github.com/nuster1128/LLM_Agent_Memory_Survey`.


Role-playing
I'm Batman, the lights of city.
Wubalubadu. Bdub Wuckoop.
Jarvis, we must first learn to run!
MAGIC IS ALL AROUND THE US!
Have you ever had a dream?
I'm a Smurf, and Smurfs are us!
Open-world Game
HP
MP
SP
Skills & Knowledge
Code Generation
Development Group
Sort the numbers in ascending order.
Social Simulation
Environment
Agent
LLM
Memory
Recommendation
I want to buy a dress for the graduation party.
You may like this blue dress. It is of good quality and great price.
Great! I like this bule one. I will buy it for the party.
Would you like to buy a waistband for your dress?
Personal Assistant
Please help me to explain "LLM-based agent".
A LLM-based agent is a type of artificial ……
In which scenarios does it have applications?
Personal assistant, game, code generation, ……
Finance
Medicine
For three days, I've had a fever ranging from 100.5℉ to 102℉, a rash on my limbs, joint pain and swelling, especially in my hands, episodes of diarrhea, abdominal pain, and nausea, leading to a loss of appetite.

[Iron Man] My name is Iron Man, also known as Tony Stark. I am the founder of Stark Industries and a member of the Avengers. As one of the genius inventors and billionaire, I have created the most advanced armor in the world, which not only protects me but also gives me incredible strength and the ability to fly.

```
def bubble_sort(arr):
    n = len(arr)
    # Traverse through all array elements
    for i in range(n):
        # Last i elements are already in place
        for j in range(0, n-i-1):
            # Traverse the array from 0 to n-i-1
            # Swap if the element found is greater
than the next element
            if arr[j] > arr[j+1]:
                arr[j], arr[j+1] = arr[j+1], arr[j]
    return arr
```

Bubble sort can reorder a list of numbers.

Bubble sort repeatedly steps through the list, compares adjacent elements and swaps them if they are in wrong order. The pass through the list is repeated until the list is sorted.

I'm a compassionate physician specializing in cardiology, committed to improving patients' heart health and well-being.

I'm a skilled nurse dedicated to patient care, ensuring comfort and supporting health with empathy and expertise.

(Personal Preference) She often buys blue clothes. She values the cost-effectiveness of items, especially on clothes. She likes small things with light colors, such as ear pendants.

(Context) She just bought a new blue dress. So she may need a white accessories to match it.

(Knowledge) According to the previous works, large language model based agents refer to artificial ……

(Context) The current topic is LLM-based agent. "It" refers to LLM-based agents in this conversation.

Yesterday, the financial market underwent significant fluctuations. Equity markets initially slumped as investors responded to weak corporate earnings reports and concerns over global economic growth prospects. Bond yields declined as investors sought safer assets. However, sentiment improved in the afternoon following better-than-expected economic data from a major economy, which boosted investor confidence. As a result, stock prices recovered, and the market closed with modest gains.

The yield on the 10-year Treasury note rose by 0.2 percentage points, reaching its highest level in two years. Investors sought safer assets, leading to increased demand for government bonds. Corporate bonds experienced a decline in interest, with their yields rising by 0.15 percentage points.

The patient's travel history and symptoms suggest dengue fever, a mosquito-transmitted illness.

David, a 38-year-old male with a history of allergies and sinus infections, has a family history of diabetes and hypertension. As a smoker of about one pack a day and an occasional drinker, his lifestyle choices may contribute to his health risks. After traveling to a tropical country where mosquito-borne illnesses are prevalent, he has experienced symptoms such as mild fatigue, headache, and muscle aches for the past week.

Figure 1: The importance of the memory module in LLM-based agents.

Preprint. Under review.

Contents

1 Introduction

"Without memory, there is no culture. Without memory, there would be no civilization, no society, no future."

Elie Wiesel, 1928-2016

Recently, large language models (LLMs) have achieved remarkable success in a large number of domains, ranging from artificial intelligence and software engineering to education and social science [1–3]. Original LLMs usually accomplish different tasks without interacting with environments. However, to achieve the final goal of artificial general intelligence (AGI), intelligent machines should be able to improve themselves by autonomously exploring and learning from the real world. For example, if a trip-planning agent intends to book a ticket, it should send an order request to the ticket website, and observe the response before taking the next action. A personal assistant agent should adjust its behaviors according to the user's feedback, providing personalized responses to improve user's satisfaction. To further push the boundary of LLMs towards AGI, recent years have witnessed a large number of studies on LLM-based agents [3, 4], where the key is to equip LLMs with additional modules to enhance their self-evolving capability in real-world environments.

Among all the added modules, memory is a key component that differentiates the agents from original LLMs, making an agent truly an agent (see **Figure 1**). It plays an extremely important role in determining how the agent accumulates knowledge, processes historical experience, retrieves informative knowledge to support its actions, and so on. Around the memory module, people have devoted much effort to designing its information sources, storage forms, and operation mechanisms. For example, Shinn et al. [5] incorporate both in-trial and cross-trial information to build the memory module for enhancing the agent's reasoning capability. Zhong et al. [6] store memory information in the form of natural languages, which is explainable and friendly to the users. Modarressi et al. [7] design both memory reading and writing operations to interact with environments for task solving.

While previous studies have designed many promising memory modules, there still lacks a systemic study to view the memory modules from a holistic perspective. To bridge this gap, in this paper, we comprehensively review previous studies to present clear taxonomies and key principles for designing and evaluating the memory module. In specific, we discuss three key problems including: (1) what is the memory of LLM-based agents? (2) why do we need the memory in LLM-based agents? and (3) how to implement and evaluate the memory in LLM-based agents? To begin with, we detail the concepts of memory in LLM-based agents, providing both narrow and broad definitions. Then, we analyze the necessity of memory in LLM-based agents, showing its importance from three perspectives including cognitive psychology, self-evolution, and agent applications. Based on the problems of "what" and "why", we present commonly used strategies to design and evaluate the memory modules. For the memory design, we discuss previous works from three dimensions, that is, memory sources, memory forms, and memory operations. For the memory evaluation, we introduce two widely used approaches including direct evaluation and indirect evaluation via specific agent tasks. Next, we discuss agent applications including role-playing, social simulation, personal assistant, open-world games, code generation, recommendation, and expert systems, in order to show the importance of the memory module in practical scenarios. At last, we analyze the limitations of existing work and highlight significant future directions.

The main contributions of this paper can be summarized as follows: (1) We formally define the memory module and comprehensively analyze its necessity for LLM-based agents. (2) We systematically summarize existing studies on designing and evaluating the memory module in LLM-based agents, providing clear taxonomies and intuitive insights. (3) We present typical agent applications to show the importance of the memory module in different scenarios. (4) We analyze the key limitations of existing memory modules and show potential solutions for inspiring future studies. To our knowledge, this is the first survey on the memory mechanism of LLM-based agents.

The rest of this survey is organized as follows. First, we provide a systematical meta-survey for the fields of LLMs and LLM-based agents in **Section 2**, categorizing different surveys and summarizing their key contributions. Then, we discuss the problems of "what is", "why do we need" and "how to implement and evaluate" the memory module in LLM-based agents in **Section 3 to 6**. Next, we show the applications of memory-enhanced agents in **Section 7**. The discussions of the limitations of existing work and future directions come at last in **Section 8** and **Section 9**.

2 Related Surveys

In the past two years, LLMs have attracted much attention from the academic and industry communities. To systemically summarize the studies in this field, researchers have written a lot of survey papers. In this section, we briefly review these surveys (see **Figure 2** for an overview), highlighting their major focuses and contributions to better position our study.

2.1 Surveys on Large Language Models

In the field of LLMs, Zhao et al. [70] present the first comprehensive survey to summarize the background, evolution paths, model architectures, training methodologies, and evaluation strategies of LLMs. Hadi et al. [71] and Min et al. [72] also conduct LLM surveys from the holistic view, which, however, provide different taxonomies and understandings on LLMs. Following these surveys, people dive into specific aspects of LLMs and review the corresponding milestone studies and key technologies. These aspects can be classified into four categories including the fundamental problems, evaluation, applications, and challenges of LLMs.

Fundamental problems. The surveys in this category aim to summarize techniques that can be leveraged to tackle fundamental problems of LLMs. Specifically, Zhang et al. [8] provide a comprehensive survey on the methods of supervised fine-tuning, which is a key technique for better training LLMs. Shen et al. [9], Wang et al. [10] and Liu et al. [11] present surveys on the alignment of LLMs, which is a key requirement for LLMs to produce outputs consistent with human values. Gao et al. [12] propose a survey on the retrieval-augmented generation (RAG) capability of LLMs, which is key to providing LLMs with factual and up-to-date knowledge and removing hallucinations. Qin et al. [18] summarize the state-of-the-art methods on enabling LLMs to leverage external tools, which is fundamental for LLMs to expand their capability in domains that require specialized knowledge. Wang et al. [13], Yao et al. [14], Wang et al. [15], Feng et al. [16] and Zhang et al. [17] present surveys on the direction of LLM knowledge editing, which is important for customizing LLMs to satisfy specific requirements. Huang et al. [19], Wang et al. [20] and Pawar et al. [21] focus on long-context capabilities of LLMs, which is critical for LLMs to process more information at each time and enhance their application scenarios. Wu et al. [22], Song et al. [23], Caffagni et al. [24] and Yin et al. [25] summarize multi-modal LLMs, which expands the capability of LLMs from text to visual and other modalities. The above surveys mainly focus on the effectiveness of LLMs. Another important aspect of LLMs is their training and inference efficiency. To summarize studies on this aspect, Zhu et al. [30], Xu and McAuley [31], Wang et al. [32] and Park et al. [33] systematically review the techniques of model compression. Ding et al. [81] and Xu et al. [29] analyze and conclude the studies on parameter efficient fine-tuning. Bai et al. [26], Wan et al. [27], Miao et al. [28] and Ding et al. [81] put more focuses on the efficiency of resource utilization in a general sense.

Evaluation. The surveys in this category focus on how to evaluate the capability of LLMs. Specifically, Chang et al. [34] comprehensively summarize the evaluation methods from an overall perspective. It encompasses different evaluation tasks, methods, and benchmarks, which serve as critical parts in assessing LLM performances. Guo et al. [35] care more about the evaluation targets and describe how to evaluate the knowledge, alignment, and safety control capabilities of LLMs, which supplement evaluation metrics beyond performance.

Applications. The surveys in this category aim to summarize models that leverage LLMs to improve different applications. More concretely, Zhu et al. [37] focus on the field of information retrieval (IR) and summarize studies on LLM-based query processes. Xu et al. [38] pay more attention to information extraction (IE) and provide comprehensive taxonomies for LLM-based models in this field. Li et al. [50], Lin et al. [51] and Wang et al. [52] discuss the applications of LLMs in the field of recommender system, where they utilize agents to generate data and provide recommendations. Fan et al. [39], Wang et al. [40], and Zheng et al. [41] concentrate on how LLMs can benefit software engineering (SE) in terms of software design, development, and testing. Zeng et al. [42] summarize LLM-based methods in the field of robotics. Cui et al. [43] and Yang et al. [44] focus on the application of autonomous driving and summarize models in this domain based on LLMs from different perspectives. Beyond the above domains in artificial intelligence, LLMs have also been used in natural and social science. He et al. [45], Zhou et al. [46] and Wang et al. [47] summarize the applications of LLMs in medicine. Li et al. [48] focus on the applications of LLMs in finance. He et al. [49] review the models on leveraging LLMs to improve the development of psychology.

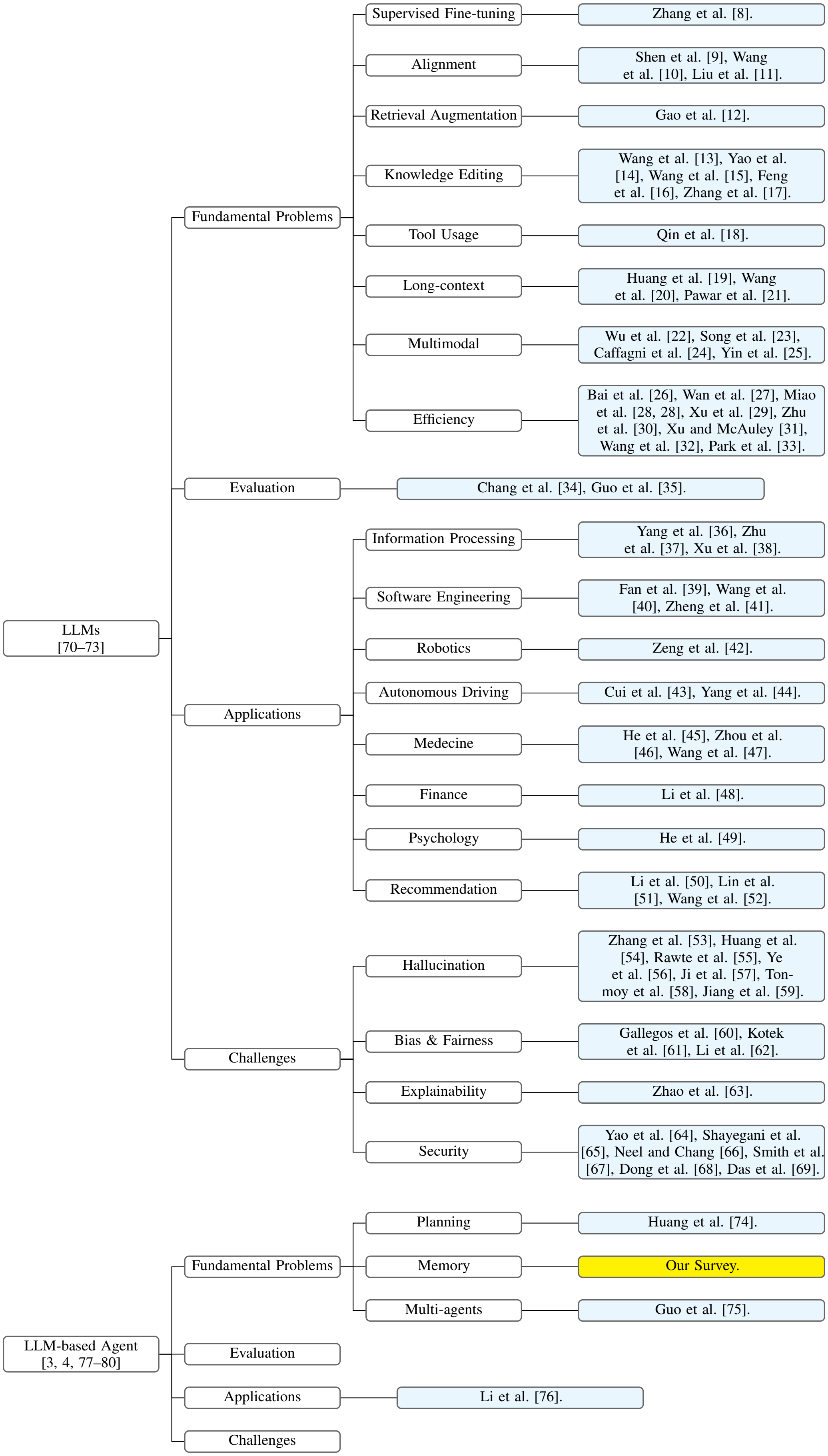

LLMs [70–73]
Fundamental Problems
Supervised Fine-tuning
Zhang et al. [8].
Alignment
Shen et al. [9], Wang et al. [10], Liu et al. [11].
Retrieval Augmentation
Gao et al. [12].
Knowledge Editing
Wang et al. [13], Yao et al. [14], Wang et al. [15], Feng et al. [16], Zhang et al. [17].
Tool Usage
Qin et al. [18].
Long-context
Huang et al. [19], Wang et al. [20], Pawar et al. [21].
Multimodal
Wu et al. [22], Song et al. [23], Caffagni et al. [24], Yin et al. [25].
Efficiency
Bai et al. [26], Wan et al. [27], Miao et al. [28, 28], Xu et al. [29], Zhu et al. [30], Xu and McAuley [31], Wang et al. [32], Park et al. [33].
Evaluation
Chang et al. [34], Guo et al. [35].
Applications
Information Processing
Yang et al. [36], Zhu et al. [37], Xu et al. [38].
Software Engineering
Fan et al. [39], Wang et al. [40], Zheng et al. [41].
Robotics
Zeng et al. [42].
Autonomous Driving
Cui et al. [43], Yang et al. [44].
Medecine
He et al. [45], Zhou et al. [46], Wang et al. [47].
Finance
Li et al. [48].
Psychology
He et al. [49].
Recommendation
Li et al. [50], Lin et al. [51], Wang et al. [52].
Challenges
Hallucination
Zhang et al. [53], Huang et al. [54], Rawte et al. [55], Ye et al. [56], Ji et al. [57], Tonmoy et al. [58], Jiang et al. [59].
Bias & Fairness
Gallegos et al. [60], Kotek et al. [61], Li et al. [62].
Explainability
Zhao et al. [63].
Security
Yao et al. [64], Shayegani et al. [65], Neel and Chang [66], Smith et al. [67], Dong et al. [68], Das et al. [69].
LLM-based Agent [3, 4, 77–80]
Fundamental Problems
Planning
Huang et al. [74].
Memory
Our Survey.
Multi-agents
Guo et al. [75].
Evaluation
Applications
Li et al. [76].
Challenges

Figure 2: The organization of related surveys on LLMs and LLM-based agents.

Challenges. The surveys in this category focus on trustworthiness in LLMs, such as hallucination, bias, unfairness, explainability, security, and privacy. Hallucination in LLMs refers to the problem that LLMs may generate misconceptions or fabrications, impacting their reliability for downstream applications. Zhang et al. [53], Huang et al. [54], Rawte et al. [55], Ye et al. [56], Ji et al. [57], Tonmoy et al. [58] and Jiang et al. [59] summarize the mainstream models for alleviating the hallucination problem in LLMs. The bias and unfairness problems refer to the phenomenon that LLMs may unequally treat different humans or objectives, which can lead to the propagation of societal stereotypes and discrimination. Gallegos et al. [60], Kotek et al. [61] and Li et al. [62] comprehensively discuss these challenges and summarize existing methods for alleviating them. The problem of explainability means that the internal working mechanisms of LLMs are still unclear. Zhao et al. [63] systematically discuss this problem and summarize previous efforts on improving the explainability of LLMs. Security and privacy are also challenging problems, which have been comprehensively surveyed in Yao et al. [64], Shayegani et al. [65], Neel and Chang [66], Smith et al. [67], Dong et al. [68] and Das et al. [69].

2.2 Surveys on Large Language Model-based Agents

Based on the capability of LLMs, people have conducted a lot of studies on building LLM-based agents, which can autonomously perceive environments, take actions, accumulate knowledge, and evolve themselves. In this field, Wang et al. [3] present the first survey paper to systematically summarize LLM-based agents from the perspectives of agent construction, agent application, and agent evaluation. Xi et al. [4], Zhao et al. [77], Cheng et al. [78] and Ge et al. [80] also summarize LLM-based agent studies from the overall perspective, but they have different focuses and taxonomies, delivering more diverse understandings on this field. In addition to these overall surveys, there have also emerged several papers reviewing specific aspects of LLM-based agents. For the fundamental problems, Durante et al. [79] summarize studies on multi-modal agents. Huang et al. [74] focus on the planning capability of LLM-based agents. Guo et al. [75] pay more attention to the scenarios of multi-agent interactions. For the applications, Li et al. [76] provide a summarization on LLM-based agents that are leveraged as personal assistants.

Position of this work. Our survey summarizes the studies on a fundamental problem of LLM-based agents, that is, the agent's memory mechanism. To our knowledge, this is the first survey in this direction. We hope it can not only inspire more advanced memory architectures in the future, but also provide newcomers with comprehensive starting materials.

3 What is the Memory of LLM-based Agent

Interacting and learning from environments is a basic requirement of LLM-based agents. In the agent-environment interaction process, there are three key phases, that is, (1) the agent perceives information from the environment, and stores it into the memory; (2) the agent processes the stored information to make it more usable; and (3) the agent takes the next action based on the processed memory information. In all these phases, memory plays an extremely important role. In the following, we first define the memory of the agent from both narrow and broad perspectives, and then, detail the execution processes of the above three phases based on the memory module.

3.1 Basic Knowledge

For clear presentations, we first introduce several important background knowledge as follows:

Definition 1 (Task)**.** Task is the final target that the agent needs to achieve, for example, booking a flight ticket for Alice, recommending a restaurant for Bob, and so on. Formally, we use $\mathcal{T}$ to represent a task and label different tasks by subscripts in the following contents.

Definition 2 (Environment)**.** In a narrow sense, environment is the object that the agent needs to interact with to accomplish the task. For the examples in definition 1, the environments are Alice and Bob, who provide feedback on the agent's actions. More broadly, environment can be any contextual factors that influence the agent's decisions, such as the weather when booking flight tickets, the time and location when recommending restaurants, etc.

Definition 3 (Trial and Step)**.** To accomplish a task, the agent needs to interact with the environment. Usually, the agent first takes an action, and then the environment responds to this action. At last, the agent takes the next action based on the response. This process iterates until the task is finished. The complete agent-environment interaction process is called a **trial**, and each interaction turn is called a

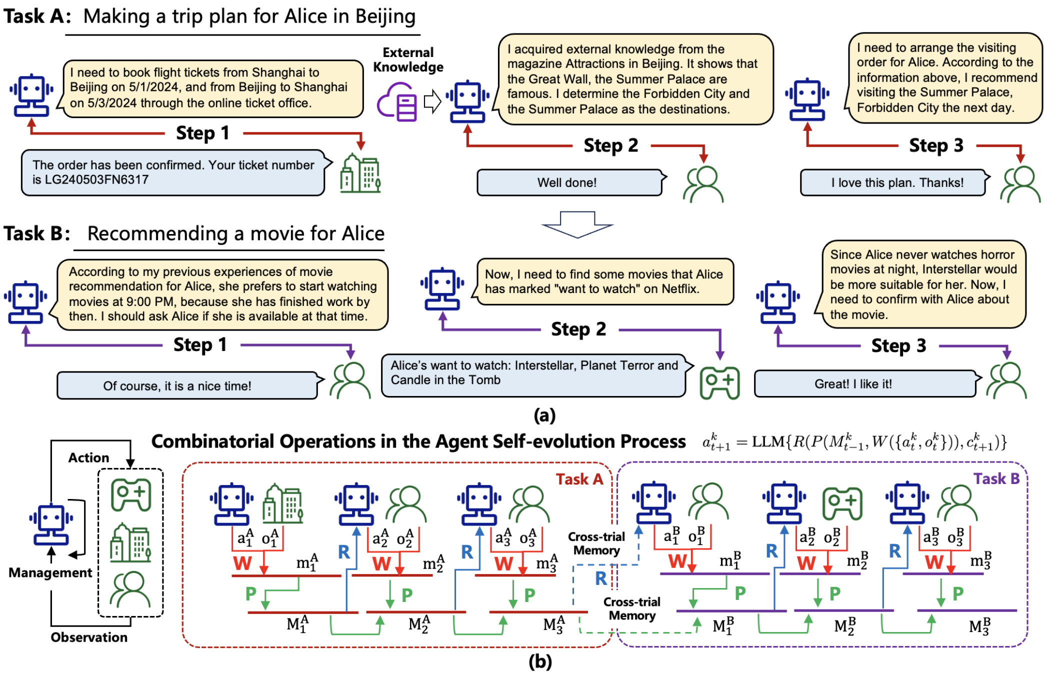

Task A: Making a trip plan for Alice in Beijing
I need to book flight tickets from Shanghai to Beijing on 5/1/2024, and from Beijing to Shanghai on 5/3/2024 through the online ticket office.
Step 1
The order has been confirmed. Your ticket number is LG240503FN6317
External Knowledge
I acquired external knowledge from the magazine Attractions in Beijing. It shows that the Great Wall, the Summer Palace are famous. I determine the Forbidden City and the Summer Palace as the destinations.
Step 2
Well done!
I need to arrange the visiting order for Alice. According to the information above, I recommend visiting the Summer Palace, Forbidden City the next day.
Step 3
I love this plan. Thanks!
Task B: Recommending a movie for Alice
According to my previous experiences of movie recommendation for Alice, she prefers to start watching movies at 9:00 PM, because she has finished work by then. I should ask Alice if she is available at that time.
Step 1
Of course, it is a nice time!
Now, I need to find some movies that Alice has marked "want to watch" on Netflix.
Step 2
Alice's want to watch: Interstellar, Planet Terror and Candle in the Tomb
Since Alice never watches horror movies at night, Interstellar would be more suitable for her. Now, I need to confirm with Alice about the movie.
Step 3
Great! I like it!
(a)
Combinatorial Operations in the Agent Self-evolution Process $a_{t+1}^k = \mathrm{LLM}\{R(P(M_{t-1}^k, W(\{a_t^k, o_t^k\})), c_{t+1}^k)\}$
Action
Management
Observation
Task A
Task B
a_1^A o_1^A a_2^A o_2^A a_3^A o_3^A
W
R
P
m_1^A m_2^A m_3^A
M_1^A M_2^A M_3^A
Cross-trial Memory
Cross-trial Memory
a_1^B o_1^B a_2^B o_2^B a_3^B o_3^B
m_1^B m_2^B m_3^B
M_1^B M_2^B M_3^B
(b)

Figure 3: (a) Examples of the potential trials in the agent-environment interaction process. (b) Illustration of the memory reading, writing, and management processes, where dotted lines mean that the cross-trial information can be incorporated into the memory module.

step. For each trial, the agent can take multiple steps to form a potential solution to the task. For each task, the agent can explore multiple trials to accomplish the task [5]. Formally, at step t, we use a_t and o_t to represent the agent action and the observed environment response, respectively. Then, a T-length trial can be represented as $\xi_T = \{a_1, o_1, a_2, o_2, ..., o_T, a_T\}$.

In the above definitions, task and environment are the most coarse-grained concepts, while step is the most fine-grained one. They together describe the complete agent-environment interaction process.

A Toy Example. Based on the above definitions, we show a toy example and use it throughout the paper to provide more intuitions. In this example, the agent has two tasks: (A) **Making a trip plan for Alice in Beijing from 5/1/2024 to 5/3/2024**, where the agent has to (i) buy a round-trip flight ticket, (ii) determine several attractions according to Alice's preferences, and (iii) arrange the visiting order of these attractions. (B) **Recommending a movie for Alice on 5/10/2024**, where the agent has to (i) determine the time that Alice is free to watch movies, and (ii) select a movie according to Alice's watching preference. For each task, we present a potential trial in the following (also illustrated in **Figure 3**(a)). For task (A), the trial is:

[Step 1] **Agent**: I need to book flight tickets from Shanghai to Beijing on 5/1/2024, and from Beijing to Shanghai on 5/3/2024 through the online ticket office. **Environment (Online Ticket Office)**: The order has been confirmed. The departure flight takes off at 9:00 AM, and your ticket number is EK240501SX9021. The returning flight takes off at 4:30 PM, and your ticket number is LG240503FN6317. Please arrive at the airport on time for your flight.

[Step 2] **Agent**: I have acquired external knowledge from the magazine *Attractions in Beijing*. It shows that *the Great Wall, the Forbidden City* and *the Summer Palace* are famous. According to Alice's preferences, I think she prefers ancient architecture, so I determine *the Forbidden City* and *the Summer Palace* as the destinations. **Environment (Alice)**: Well done!

[Step 3] **Agent**: Now, I need to arrange the visiting order for Alice. According to the information above, I recommend visiting *the Summer Palace* after she arrives in Beijing because it is suitable to spend a whole afternoon there and watch the sunset. Then, she can go to *the Forbidden City* the next day, which would give her plenty of time. Finally, she can return to Shanghai on the third day and take a rest. **Environment (Alice)**: I love this plan. Thanks!

For task (B), the trial is:

[Step 1] **Agent**: According to my previous experiences of movie recommendations for Alice, she prefers to start watching movies at 9:00 PM, because she has finished work by then. I should ask Alice if she is available at that time. **Environment (Alice)**: Of course, it is a nice time!

[Step 2] **Agent**: Now, I need to find some movies that Alice has marked "want to watch" on Netflix. **Environment (Netflix)**: Alice's want to watch: *Interstellar*, *Planet Terror* and *Candle in the Tomb*.

[Step 3] **Agent**: Since Alice never watches horror movies at night, *Interstellar* would be more suitable for her. Now, I need to confirm with Alice about the movie. **Environment (Alice)**: Great! I like it!

3.2 Narrow Definition of the Agent Memory

In a narrow sense, the memory of the agent is only relevant to the historical information within the same trial. Formally, for a given task, the historical information of the trial before step t is $\xi_t = \{a_1, o_1, a_2, o_2, ..., a_{t-1}, o_{t-1}\}$, and then the memory is derived based on ξ_t. In the above toy example, for task (A), the agent at [step 3] needs to arrange the visiting order for Alice; at this time, its memory contains the information about the selected attractions and arrival time in [step 1] and [step 2]. For task (B), the agent has to choose a movie for Alice at [step 3]; at this time, its memory contains the arranged time to watch films.

3.3 Broad Definition of the Agent Memory

In a broad sense, the memory of the agent can come from much wider sources, for example, the information across different trials and the external knowledge beyond the agent-environment interactions. Formally, given a series of sequential tasks $\{\mathcal{T}_1,\ \mathcal{T}_2, ...,\ \mathcal{T}_K\}$, for task $\mathcal{T}_k$, the memory information at step t comes from three sources: (1) the historical information within the same trial, that is, $\xi_t^k = \{a_1^k, o_1^k, ..., a_{t-1}^k, o_{t-1}^k\}$, where we add superscript k to label the task index. (2) The historical information across different trials, that is, $\Xi^k = \{\xi^1, \xi^2, ..., \xi^{k-1}, \xi^{k'}\}$, where ξ^j $(j \in \{1, ..., k-1\})$ represents the trials of task j[1], and $\xi^{k'}$ denotes the previously explored trials for task $\mathcal{T}_k$. (3) External knowledge, which is represented by D_t^k. The memory of the agent is derived based on $(\xi_t^k,\ \Xi^k,\ D_t^k)$. In the above toy example, for task (A), if there are several failed trials, that is, the feedback from Alice is negative, then these trials can be incorporated into the agent's memory to avoid future similar errors (corresponding to $\xi^{k'}$). In addition, for task (B), the agent may recommend movies relevant to the attractions that Alice has visited in task (A) to capture her recent preferences (corresponding to $\{\xi^1, \xi^2, ..., \xi^{k-1}\}$). In the agent decision process, it has also referred to the magazine *Attractions in Beijing* for making trip plans, which is the external knowledge (corresponding to D_t^k) for the current task $\mathcal{T}_k$.

3.4 Memory-assisted Agent-Environment Interaction

As mentioned at the beginning of **Section 3**, there are three key phases in the agent-environment interaction process. The agent memory module implements these phases through three operations including memory writing, memory management, and memory reading.

Memory Writing. This operation aims to project the raw observations into the actually stored memory contents, which are more informative [7] and concise [6]. It corresponds to the first phase of the agent-environment interaction process. Given a task $\mathcal{T}_k$, if the agent takes an action a_t^k at step t, and the environment provides an observation o_t^k, then the memory writing operation can be formally represented as:

$$m_t^k = W(\{a_t^k, o_t^k\}),$$

where W is a projecting function. m_t^k is the finally stored memory contents, which can be either natural languages or parametric representations. In the above toy example, for task (A), the agent is supposed to remember the flight arrangement and the decision of attractions after [step 2]. For task (B), the agent should memorize the fact that Alice hopes to watch movies at 9:00 PM, after [step 1].

Memory Management. This operation aims to process the stored memory information to make it more effective, for example, summarizing high-level concepts to make the agent more general-

[1] For each task, there can be multiple trials for exploring the final solution, and all of them can be incorporated into the memory.

izable [6], merging similar information to reduce redundancy [7], and forgetting unimportant or irrelevant information to remove its negative influence. This operation corresponds to the second phase of the agent-environment interaction process. Let M_{t-1}^k be the memory contents for task k before step t, and suppose m_t^k is the stored information at step t based on the above memory writing operation, then, the memory management operation can be represented by:

$$M_t^k = P(M_{t-1}^k, m_t^k),$$

where P is a function that iteratively processes the stored memory information. For the narrow memory definition, the iteration only happens within the same trial, and the memory is emptied when the trial is ended. For the broad memory definition, the iteration happens across different trials or even tasks, as well as the integrations of external knowledge. For task (B) in the above toy example, the agent can conclude that Alice enjoys watching science fiction movies in the evening, which can be used as a default rule to make recommendations for Alice in the future.

Memory Reading. This operation aims to obtain important information from the memory to support the next agent action. It corresponds to the third phase of the agent-environment interaction process. Suppose M_t^k is the memory contents for task k at step t, c_t^k is the context of the next action, then the memory reading operation can be represented by:

$$\hat{M}_t^k = R(M_t^k, c_{t+1}^k),$$

where R is usually implemented by computing the similarity between M_t^k and c_{t+1}^k [82]. $\hat{M}_t^k$ is used as parts of the final prompt to drive the agent's next action. For task (B) in the above toy example, when the agent decides on the final recommended movie in [Step 3], it should focus on the "want to watch" list in [Step 2] and select one from it.

Based on the above operations, we can derive a unified function for the evolving process from $\{a_t^k, o_t^k\}$ to a_{t+1}^k, that is:

$$a_{t+1}^k = \text{LLM}\{R(P(M_{t-1}^k, W(\{a_t^k, o_t^k\})), c_{t+1}^k)\},$$

where LLM is the large language model. The complete agent-environment interaction process can be easily obtained by iteratively expanding this function (see **Figure 3**(b) for an intuitive illustration).

Remark. This function provides a general formulation of the agent memorizing process. Previous works may use different specifications. For example, in [5], R and P are set as identical functions, and P only takes effect at the end of a trial. In Park et al. [83], R is implemented based on three criteria including similarity, time interval, and importance, and P is realized by a reflection process to obtain more abstract thoughts. In this section, we focus on the overall framework of the agent's memory operations. More detailed realizations of W, P, and R are deferred in **Section 5**.

4 Why We Need the Memory in LLM-based Agent

Above, we have introduced what is the memory of LLM-based agents. Before comprehensively presenting how to implement it, in this section, we briefly show why memory is necessary for building LLM-based agents, where we expand our discussion from three perspectives including cognitive psychology, self-evolution, and agent applications.

4.1 Perspective of Cognitive Psychology

Cognitive psychology is the scientific study of human mental processes such as attention, language use, memory, perception, problem-solving, creativity, and reasoning[2]. Among these processes, memory is widely recognized as an extremely important one [84]. It is fundamental for humans to learn knowledge by accumulating important information and abstracting high-level concepts [85], form social norms by remembering cultural values and individual experiences [86], take reasonable behaviors by imagining the potential positive and negative consequences [87], and among others.

A major goal of LLM-based agents is to replace humans for accomplishing different tasks. To make agents behave like humans, following human's working mechanisms to design the agents is a natural and essential choice [88]. Since memory is important for humans, designing memory modules is also significant for the agents. In addition, cognitive psychology has been studied for a long time, so many effective human memory theories and architectures have been accumulated, which can support more advanced capabilities of the agents [89].

[2] https://en.wikipedia.org/wiki/Cognitive_psychology

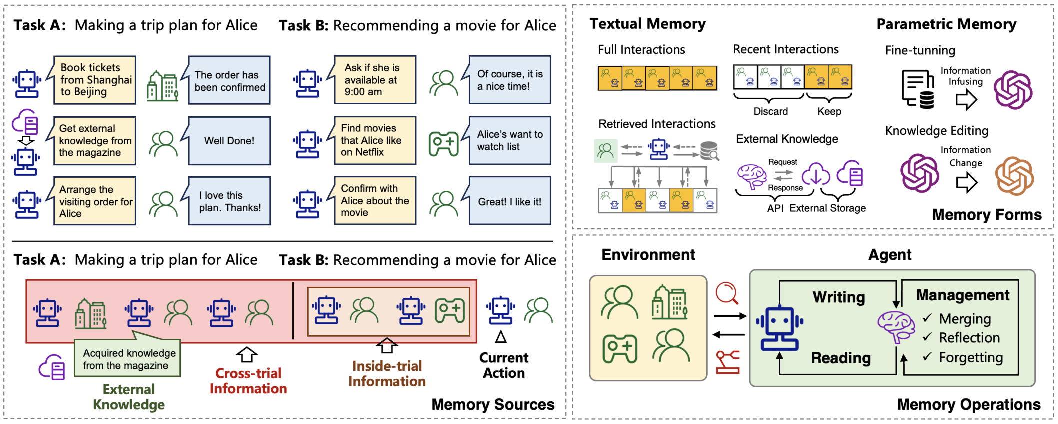

Task A: Making a trip plan for Alice
Task B: Recommending a movie for Alice
Book tickets from Shanghai to Beijing
The order has been confirmed
Ask if she is available at 9:00 am
Of course, it is a nice time!
Get external knowledge from the magazine
Well Done!
Find movies that Alice like on Netflix
Alice's want to watch list
Arrange the visiting order for Alice
I love this plan. Thanks!
Confirm with Alice about the movie
Great! I like it!
Task A: Making a trip plan for Alice
Task B: Recommending a movie for Alice
Acquired knowledge from the magazine
External Knowledge
Cross-trial Information
Inside-trial Information
Current Action
Memory Sources
Textual Memory
Full Interactions
Recent Interactions
Discard
Keep
Retrieved Interactions
External Knowledge
Request
Response
API
External Storage
Parametric Memory
Fine-tunning
Information Infusing
Knowledge Editing
Information Change
Memory Forms
Environment
Agent
Writing
Reading
Management
✓ Merging
✓ Reflection
✓ Forgetting
Memory Operations

Figure 4: An overview of the sources, forms, and operations of the memory in LLM-based agents.

4.2 Perspective of Self-Evolution

To accomplish different practical tasks, agents have to self-evolve in dynamic environments [90]. In the agent-environment interaction process, the memory is key to the following aspects: (1) **Experience accumulation**. An important function of the memory is to remember past error plannings, inappropriate behaviors, or failed experiences, so as to make the agent more effective for handling similar tasks in the future [91]. This is extremely important for enhancing the learning efficiency of the agent in the self-evolving process. (2) **Environment exploration**. To autonomously evolve in the environment, the agents have to explore different actions and learn from the feedback [92]. By remembering historical information, the memory can help to better decide when and how to make explorations, for example, focusing more on previously failed trials or actions with lower exploring frequencies [93]. (3) **Knowledge abstraction**. Another important function of the memory is to summarize and abstract high-level information from raw observations, which is the basis for the agent to be more adaptive and generalizable to unseen environments [82]. In summary, self-evolution is the basic characteristic of LLM-based agents, and memory is of key importance to self-evolution.

4.3 Perspective of Agent Applications

In many applications, memory is an indispensable component of the agent. For example, in a conversational agent, the memory stores information about historical conversations, which is necessary for the agent to generate the next response. Without memory, the agent does not know the context, and cannot continue the conversation [94]. In a simulation agent, memory is of great importance to make the agent consistently follow the role profiles. Without memory, the agent may easily step out of the role during the simulation process [95]. Both of the above examples show that the memory is not an optional component, but is necessary for the agents to accomplish given tasks.

In the above three perspectives, the first one reveals that the memory builds the cognitive basis of the agent. The second and third ones show that the memory is necessary for the agent's evolving principles and applications, which provide insights for designing agents with memory mechanisms.

5 How to Implement the Memory of LLM-based Agent

In this section, we discuss the implementation of the memory module from three perspectives: memory sources, memory forms, and memory operations. Memory sources refer to where the memory contents come from. Memory forms focus on how to represent the memory contents. Memory operations aim to process the memory contents. These three perspectives provide a comprehensive review of memory implementation methods, which is helpful for future research. For better demonstration, we present an overview of implementation methods in **Figure 4**.

5.1 Memory Sources

In previous works, the memory contents may come from different sources. Based on our formulation in **Section 3**, these sources can be classified into three categories, that is, the information inside a trial, the information across different trials, and the external knowledge. The former two are dynamically

Table 1: Summarization of the memory sources. We use ✓ and × to label whether or not the corresponding source is adopted in the model.

Models	Inside-trial Information	Cross-trial Information	External Knowledge
MemoryBank [6]	✓	×	×
RET-LLM [7]	✓	×	✓
ChatDB [96]	✓	×	✓
TiM [97]	✓	×	×
SCM [98]	✓	×	×
Voyager [99]	✓	×	×
MemGPT [100]	✓	×	×
MemoChat [94]	✓	×	×
MPC [101]	✓	×	×
Generative Agents [83]	✓	×	×
RecMind [102]	✓	×	✓
Retroformer [103]	✓	✓	✓
ExpeL [82]	✓	✓	✓
Synapse [91]	✓	✓	×
GITM [93]	✓	✓	✓
ReAct [104]	✓	×	✓
Reflexion [5]	✓	✓	✓
RecAgent [95]	✓	×	×
Character-LLM [105]	✓	×	✓
MAC [106]	✓	×	×
Huatuo [107]	✓	×	✓
ChatDev [1]	✓	×	×
InteRecAgent [108]	✓	×	✓
MetaAgents [109]	✓	×	×
TPTU [110, 111]	✓	×	✓
MetaGPT [112]	✓	✓	×
S^3 [2]	✓	×	×
InvestLM [113]	✓	×	✓

generated in the agent-environment interaction process (*e.g.*, task internal information), while the latter is static information outside the loop (*e.g.*, task external information). We summarize previous works on memory sources in **Table 1**.

5.1.1 Inside-trial Information

In the agent-environment interaction process, the historical steps within a trial are usually the most relevant and informative signals to support the agent's future actions. Almost all the previous works use this information as a part of the memory sources.

Representative Studies. Generative Agents [83] aims to simulate human's daily behaviors by using LLM-based agents. The memory of an agent is derived from the historical behaviors to achieve a target, for example, the collection of relevant papers when researching on a specific topic. MemoChat [94] aims to chat with humans, where the memory of the agent is derived based on the conversation history of a dialogue session. TiM [97] aims to enhance the agent's reasoning capability by self-generating multiple thoughts after accomplishing a task, which is used as the memory to provide more generalizable information. Voyager [99] focuses on building game agents based on Minecraft, where the memory contains executable codes of preliminary and basic actions to accomplish a task. It should be noted that the inside-trial information not only includes agent-environment interactions, but also contains interaction contexts, such as time and location information.

Discussion. The inside-trial information is the most obvious and intuitive source that should be leveraged to construct the agent's memory since it is highly relevant to the current task that the agent has to accomplish. However, relying solely on inside-trial information may prevent the agent from accumulating valuable knowledge from various tasks and learning more generalizable information. Thus, many studies also explore how to effectively utilize the information across different tasks to build the memory module, which is detailed in the following sections.

5.1.2 Cross-trial Information

For LLM-based agents, the information accumulated across multiple trials in the environment is also a crucial part of the memory, typically including successful and failed actions and their insights, such as failure reasons, common action patterns to succeed, and so on.

Representative Studies. One of the most prominent studies is Reflexion [5], which proposes verbal reinforcement learning for LLM-based agents. It derives the experiences from past trials in verbal form, and applies them in subsequent trials to improve the performance of the same task. Furthermore, Retroformer [103] fine-tunes the reflection model, enabling the agent to extract cross-trial information from past trials more effectively. In Synapse [91], the agents focus on solving the computer control tasks. Their memory can record cross-trial information through successful exemplars, which would be used as references on similar trials. In ExpeL [82], the agents are required to solve a collection of complex interactive tasks within the environment. They store and organize completed trajectories, and recall similar ones for the new task. In the recalled trajectories, successful cases will be compared with failed ones to identify the patterns to succeed.

Discussion. According to the accumulated memory of cross-trial information, the agents are able to accumulate experiences, which is important for their evolution. Based on the past experiences, the agents can adjust their actions based on the overall feedback of the whole process. In contrast to the inside-trial observations, which serve as short-term memory, the trial experiences can be considered as long-term memory. It utilizes feedback from different trials to support a wider range of agent trials, providing more prolonged experiential support for agents. However, the limitation lies in the fact that both inside-trial and cross-trial information require the agents to personally engage in agent-environment interactions, where external experiences and knowledge are not included.

5.1.3 External Knowledge

An important characteristic of LLM-based agents is that they can be directly communicated and controlled in natural languages. As such, LLM-based agents can easily incorporate external knowledge in textual forms (*e.g.*, Wikipedia [3]) to facilitate their decisions.

Representative Studies. In ReAct [104], the agents are required to answer questions about general knowledge by multiple reasoning steps. They can utilize Wikipedia APIs to obtain external knowledge if they lack information during these steps. GITM [93] intends to design agents in Minecraft, which can explore in complex and sparse-reward environments. The agents draw from the online Minecraft Wiki and craft recipes to provide an infinite source of knowledge for their navigation. CodeAgent [114] focuses on the repo-level code generation task, which commonly requires complex dependencies and extensive documentation. It designs a web search strategy for acquiring related external knowledge. ChatDoctor [115] adapts LLM-based agents to the medical domain. It fine-tunes an acquisition process to retrieve external knowledge from Wikipedia and medical databases.

Discussion. The external knowledge can be obtained from both private and public sources. It provides LLM-based agents with much knowledge beyond their internal environment, which might be difficult or even impossible for the agent to acquire by agent-environment interactions. Moreover, most external knowledge can be acquired by accessing the APIs of various tools dynamically in real time according to the task needs, thus mitigating the problem of outdated knowledge. Integrating external knowledge into the memory of LLM-based agents significantly expands their knowledge boundaries, providing them with unlimited, up-to-date, and well-founded knowledge for decision-making.

5.2 Memory Forms

In general, there are two forms to represent the memory contents: textual form and parametric form. In textual form, the information is explicitly retained and recalled by natural languages. In parametric

[3] `https://www.wikipedia.org`

Table 2: Summarization of the memory forms. We use ✓ and × to label whether or not the corresponding memory form is adopted in the model.

Models	**Textual Form**				**Parametric Form**	
	Complete	**Recent**	**Retrieved**	**External**	**Fine-tuning**	**Editing**
MemoryBank [6]	×	×	✓	×	×	×
RET-LLM [7]	×	×	✓	×	×	×
ChatDB [96]	×	×	✓	×	×	×
TiM [97]	×	×	✓	×	×	×
SCM [98]	×	✓	✓	×	×	×
Voyager [99]	×	×	✓	×	×	×
MemGPT [100]	×	✓	✓	×	×	×
MemoChat [94]	×	×	✓	×	×	×
MPC [101]	×	×	✓	×	×	×
Generative Agents [83]	×	×	✓	×	×	×
RecMind [102]	✓	×	×	×	×	×
Retroformer [103]	✓	×	×	✓	✓	×
ExpeL [82]	✓	×	✓	✓	×	×
Synapse [91]	×	×	✓	×	×	×
GITM [93]	✓	×	✓	✓	×	×
ReAct [104]	✓	×	×	✓	×	×
Reflexion [5]	✓	×	×	✓	×	×
RecAgent [95]	×	✓	✓	×	×	×
Character-LLM [105]	×	✓	×	×	✓	×
MAC [106]	×	×	×	×	×	✓
Huatuo [107]	✓	×	×	×	✓	×
ChatDev [1]	✓	×	×	×	×	×
InteRecAgent [108]	×	✓	✓	✓	×	×
MetaAgents [109]	×	×	✓	×	×	×
TPTU [110, 111]	✓	×	×	✓	×	×
MetaGPT [112]	✓	×	×	×	×	×
S^3 [2]	×	×	✓	×	×	×
InvestLM [113]	✓	×	×	×	✓	×

form, the memory information is encoded into parameters and implicitly influences the agent's actions. We summarize previous works on memory forms with their implementations in **Table 2**.

5.2.1 Memory in Textual Form

Textual form is currently the mainstream method to represent the memory contents, which is featured in better interpretability, easier implementation, and faster read-write efficiency. In specific, the textual form can be both non-structured representations like raw natural languages and structured information such as tuples, databases, and so on. In general, previous studies use the textual form memory to store four types of information including (1) complete agent-environment interactions, (2) recent agent-environment interactions, (3) retrieved agent-environment interactions, and (4) external knowledge. In the former three methods, the memory leverages natural languages to describe the information within the agent-environment interaction loop. In the former three types, they record the information inside the agent-environment interaction loop, while the last type leverages natural languages to store information outside that loop.

Complete Interactions. This method stores all the information of the agent-environment interaction history based on long-context strategies [116]. For the example in **Section 3.1**, the memory of the agent in task (A) after step 2 can be implemented by concatenating all the information before step 2, and the final textual form memory is: "Your memory is [Step 1] (Agent) ... (Online Ticket Office) ... [Step 2] ... Please infer based on your memory".

In the previous work, different models store the memory information using different strategies. For example, in LongChat [116], the agents focus on understanding natural languages in long-context scenarios. It fine-tunes the foundation model for better adapting to memorize complete interactions. Memory Sandbox [117] intends to alleviate the impact of irrelevant memory in conversations. It designs a transparent and interactive method to manage the memory of agents, which removes irrelevant memory before concatenating them as a prompt. Moreover, some efforts are dedicated to enhancing the capacity of LLMs to handle longer contexts [118, 119].

While storing all the agent-environment interactions can maintain comprehensive information, obvious limitations exist in terms of computational cost, inference time, and inference robustness. Firstly, the fast-growing long-context memory in practice results in high computational cost during LLM inference, due to the quadratic growth of the time complexity of attention computation with sequence length. It thus requires much more computing resources and significantly increases inference latency, which hinders its practical deployment. What's more, with its fast growth, the memory length can easily exceed the upper bound of the sequence length during LLM's pretraining, which makes a truncation of memory necessary. Thus, it can lead to information loss due to the incompleteness of agent memory. Last but not least, it can lead to biases and unrobustness in LLM's inference. Specifically, a previous research [120] has shown that, the positions of text segments in a long context can greatly affect their utilization, so the memory in the long-context prompt can not be treated equally and stably. All these drawbacks show the need to design extra memory modules for LLM-based agents, rather than straightforwardly concatenating all the information into a prompt.

Recent Interactions. This method stores and maintains the most recently acquired memories using natural languages, thereby enhancing the efficiency of memory information utilization according to the Principle of Locality [121]. In task (B) of the example in **Section 3.1**, we can just remember Alice's preferences in the recent three years, and truncate the distant part, where the recent three years can be considered as the memory window size.

In previous studies, there are various strategies to store recent textual memories. For example, SCM [98] proposes a flash memory based on the cache mechanism, which preserves observations from the recent $t-1$ time steps, aimed at enhancing the recency of information. MemGPT [100] considers the agent as an operating system, which can dynamically interact with users through a natural interface. It designs the working context to hold recent histories, as a part of virtual context management. In RecAgent [95], the agents are designed to simulate user behaviors in movie recommendations. It stores some temporal information in short-term memory as an intermediate cache, which can simulate the memory mechanism of the human brain [122, 123]. These representative methods can dynamically update memories based on recent interactions, and pay more attention to the recent context that is important for the current stage.

Caching the memory according to recency is an effective way to enhance memory efficiency, and it enables agents to focus more on the recent information. However, in long-term tasks, this method fails to access key information from distant memories. It can result in the loss of potentially crucial information that is not within the immediate cache window. In other words, emphasizing on recency can inherently neglect earlier, yet critical information, thus posing challenges in scenarios requiring a comprehensive understanding of past events.

Retrieved Interactions. Unlike the above method which truncates memories based on time, this method typically selects memory contents based on their relevance, importance, and topics. It ensures the inclusion of distant but crucial memories in the decision-making process, thereby addressing the limitation of only memorizing recent information. In task (A) of the example in **Section 3.1**, Alice's preferences have been stored in the memory before this task. At [Step 2], the agent will retrieve the most relevant aspects of Alice's preferences from memory based on the query keyword "travel", obtaining Alice's scenic spot preference for ancient architectures. In general, retrieval methods will generate embeddings as indexes for memory entries during memory writing, along with recording auxiliary information to assist in retrieval. During memory reading, matching scores are calculated for each memory entry, and the top-K entries will be used for the decision-making process of agents.

In existing studies, most agents utilize retrieval methods to process the memory information. For example, Park et al. [83] first calculate the relevance between the current context and memory entries by cosine similarity, and obtain the importance and recency according to auxiliary information. MemoryBank [6] employs a dual-tower dense retrieval model to find related information from past conversations. Each memory entry is encoded into an embedding and subsequently indexed by FAISS [124] to improve the efficiency of retrieval. When reading memories, the current context will be encoded as representations to obtain the most relevant memory. Moreover, RET-LLM [7] intends to design a write-read memory module for general usage. It utilizes Locality-Sensitive Hashing (LSH) to retrieve tuples with relative entries in the database to provide more information. In addition, ChatDB [96] designs to utilize symbolic memory, and proposes to generate SQL statements to retrieve from database to obtain stored information.

The retrieval methods considerably depend on the accuracy and efficiency of obtaining expected information. An inaccurate retrieval strategy can potentially acquire unrelated information that is unhelpful for agent inference. And a heavy retrieval system can lead to large computational costs and long time latency, especially when handling massive information. Moreover, retrieval methods typically store homogeneous information inside the environment, where all the information is in a consistent form. For heterogeneous information outside the environment, it's difficult to directly apply the same method for memory storage.

External Knowledge. To obtain more information, some agents acquire external knowledge by invoking tools, with the aim of transforming additional relevant knowledge into their own memories for decision-making. For instance, accessing external knowledge through Application Programming Interface (API) is a common practice [104, 5]. Nowadays, abundant public information, such as Wikipedia and OpenWeatherMap[4], are available online (either free of charge or on a paying basis), and can be conveniently accessed through API calls. For instance, in [Step 2] of task (A) of the example in **Section 3.1**, external knowledge from the digital magazine is obtained with tool methods.

In existing models, Toolformer [125] proposes to teach LLM to use tools, which can acquire external knowledge for better solving tasks. Furthermore, ToolLLM [126] empowers Llama [127] with the ability to utilize more APIs in RapidAPI[5] and to enable multi-tool usage, which provides a general interface to extend agents' ability. In TPTU [110], the agents are incorporated in both task planning and tool usage, in order to tackle intricate problems. The follow-up work [111] further improves its ability extensively like retrieval. In ToRA [128], the agents are required to solve mathematical problems. They utilize imitation learning to improve their ability to use program-based tools.

The above methods significantly advance the capabilities of agents by allowing them to access external up-to-date and real-world information from diverse sources. However, the reliability of this information can be questionable due to potential inaccuracies and biases [18]. Furthermore, the integration of tools into agents demands a comprehensive understanding to interpret the retrieved information across various contexts, which can incur higher computational costs and complications in aligning external data with internal decision-making processes. Additionally, utilizing external APIs brings forth concerns regarding privacy, data security, and compliance with usage policies, necessitating rigorous management and oversight [18].

5.2.2 Memory in Parametric Form

An alternative type of approaches is to represent memory in parametric form. They do not take up the extra length of context in prompts, so they are not constrained by the length limitations of LLM context. However, the parametric memory form is still under-researched, and we categorize previous works into two types: fine-tuning methods and memory editing methods.

Fine-tuning Methods. Integrating external knowledge into the memory of agents is beneficial for enriching domain-specific knowledge on top of its general knowledge. To infuse the domain knowledge into LLMs, supervised fine-tuning is a common approach, which empowers agents with the memory of domain experts. It significantly improves the agent's ability to accomplish domain-specific tasks. In task (A) of the example in **Section 3.1**, the external knowledge of attractions from magazines can be fine-tuned into the parameters of LLMs prior to this task.

[4] `https://openweathermap.org`

[5] `https://rapidapi.com/hub`

In previous works, Character-LLM [105] focuses on the role-play circumstance. It utilizes supervised fine-tuning strategies with role-related data (*e.g.,* experiences), to endow agents with the specific traits and characteristics of the role. Huatuo [107] intends to empower agents with professional ability in the biomedical domain. It tries to fine-tune Llama [127] on Chinese medical knowledge bases. Besides, in order to create artificial doctors, DoctorGLM [129] fine-tunes ChatGLM [130] with LoRA [131], and Radiology-GPT [132] improves domain knowledge on radiology analysis by supervised fine-tuning on an annotated radiology dataset. Moreover, InvestLM [113] collects investment data and fine-tunes it to improve domain-specific abilities on financial investment.

The fine-tuning methods can effectively bridge the gap between general agents and specialized agents. It improves the capability of agents on the tasks that require high accuracy and reliability on domain-specific information. Nevertheless, fine-tuning LLMs for specific domains could potentially lead to overfitting, and it also raises concerns about catastrophic forgetting, where LLMs may forget the original knowledge because of updating their parameters. Another limitation of fine-tuning lies in the computational cost and time consumption, as well as the requirement of a large amount of data. Therefore, most fine-tuning approaches are applied to offline scenarios, and can seldom deal with online scenarios, such as fine-tuning with agent observations and trial experiences. Due to the frequent agent-environment interactions, it is unaffordable for the cost of backpropagation to fine-tune every step of the online and dynamic interactions.

Memory Editing Methods. Apart from the fine-tuning approaches, another type of methods for infusing memory into model parameters is knowledge editing [133, 134]. Unlike fine-tuning methods that extract patterns from certain datasets, knowledge editing methods specifically target and adjust only the facts that need to be changed. It ensures that unrelated knowledge remains unaffected. Knowledge editing methods are more suitable for small-scale memory adjustments. Generally, they have lower computational costs, making them more suitable for online scenarios. In our example of task (B), Alice always watches movies at 9:00 PM from the agent's memory, but she may recently change her work and would not be empty at 9:00 PM. If so, the related memory (such as routines at 9:00 PM) should be edited, which can be implemented by knowledge editing methods.

In previous studies, MAC [106] intends to design an effective and efficient memory adaptation framework for online scenarios. It utilizes meta-learning to substitute the optimization step. PersonalityEdit [135] focuses on editing the personality of LLMs and agents, where it changes their traits based on theories such as the big-five factor. MEND [134] utilizes the idea of meta-learning to train a lightweight model, which is capable of generating modifications for model parameters of a pre-trained language model. APP [136] studies whether adding a new fact leads to catastrophic forgetting of existing facts. It focuses on the impact of neighbor perturbation on memory addition. Moreover, KnowledgeEditor [133] trains a hyper-network to predict the modification of model parameters when injecting memory based on a learning-to-update problem formulation. Wang et al. [137] propose a new optimization target to change the poisoning knowledge of LLM, and maintain the general performance at the same time. For LLM-based agents, the agents can change bad memory by knowledge editing, which can be considered as a type of forgetting mechanism.

Knowledge editing methods provide an innovative way to update the information stored within the parameters of LLMs. By specifically targeting and adjusting the facts, these methods can ensure the non-targeted knowledge unaffected during updates, thus mitigating the issue of catastrophic forgetting. Moreover, the targeted adjustment mechanism allows for more efficient and less resource-intensive updates, making knowledge editing an appealing choice for high-precision and real-time modifications. However, despite these promising developments, computational costs of meta-training and the preservation of unrelated memories remain significant challenges.

5.2.3 Advantages and Disadvantages of Textual and Parametric Memory

Textual memory and parametric memory have their strengths and weaknesses respectively, making them suitable for different memory contents and application scenarios. In this section, we discuss the advantages and disadvantages of these two forms of memory from various aspects.

Effectiveness. The textual memory stores raw information about the agent-environment interactions, which is more comprehensive and detailed. However, it is constrained by the token limitation of LLM prompts, which makes the agent hard to store extensive information. In contrast, the parametric memory is not limited by the prompt length, but it may suffer from information loss when transforming texts into parameters, and the complex memory training can bring additional challenges.

Efficiency. For textual memory, each LLM inference requires to integrate memory into the context prompt, which leads to higher costs and longer processing times. In contrast, for parametric memory, the information can be integrated into the parameters of the LLM, eliminating the extra costs of these contexts. However, parametric memory takes additional costs in the writing process, but textual memory is easier to write, especially for small amounts of data. In a nutshell, textual memory is more efficient in writing, while parametric memory is more efficient in reading.

Interpretability. Textual memory is usually more explainable than the parametric one, since natural languages are the most natural and straightforward strategies for humans to understand, while parametric memory is commonly represented in latent space. Nevertheless, such explainability is obtained at the cost of information density. This is because the sequences of words in textual memory are represented in a discrete space, which is not as dense as continuous space in parametric memory.

In conclusion, the trade-offs between these two types of memories make them suitable for different applications. For example, for the tasks that require recalling recent interactions, like conversational and context-specific tasks, textual memory seems more effective. For the tasks that require a large amount of memory, or well-established knowledge, parametric memory can be a better choice.

5.3 Memory Operations

We separate the entire procedure of memory into three operations: memory writing, memory management, and memory reading. These three typically collaborate to achieve memory function, providing information for LLM inference. We summarize previous works on memory operations in **Table 3**.

5.3.1 Memory Writing

After the information is perceived by the agent, a part of it will be stored by the agent for further usage through the memory writing operation, and it is crucial to recognize which information is essential to store. Many studies choose to store the raw information, while others also put the summary of the raw information into the memory module.

Representative Studies. In TiM [97], the raw information will be extracted as the relation between two entities, and stored in a structured database. When writing into the database, similar contents will be stored in the same group. In SCM [98], it designs a memory controller to decide when to execute the operations. The controller serves as a guide for the whole memory module. In MemGPT [100], the memory writing is entirely self-directed. The agents can autonomously update the memory based on the contexts. In MemoChat [94], the agents summarize each conversation segment by abstracting the mainly discussed topics and storing them as keys for indexing memory pieces.

Discussion. Previous research indicates that designing the strategy of information extraction during the memory writing operation is vital [94]. This is because the original information is commonly lengthy and noisy. Besides, different environments may provide various forms of feedback, and how to extract and represent the information as memory is also significant for memory writing.

5.3.2 Memory Management

For human beings, memory information is constantly processed and abstracted in the brains. The memory in the agent can also be managed by reflecting to generate higher-level memories, merging redundant memory entries, and forgetting unimportant, early memories.

Representative Studies. In MemoryBank [6], the agents process and distill the conversations into a high-level summary of daily events, similar to how humans recall key aspects of their experiences. Through long-term interactions, they continually evaluate and refine their knowledge, generating daily insights into personality traits. In Voyager [99], the agents are able to refine their memory based on the feedback of the environment. In Generative Agents [83], the agents can reflect to get higher-level information, where the abstract thoughts are generated from agents. The reflection process will be activated when there are accumulated events that are enough to address. For GITM [93], in order to establish common reference plans for various situations, key actions from multiple plans are further summarized in the memory module.

Discussion. Most of the memory management operations are inspired by the working mechanism of human brains. With the strong capability of LLMs to simulate human minds, these operations can help the agents to better generate high-level information and interact with environments.

Table 3: Summarization of the memory operations. If a model does not have special designs on the memory operations, we use ○ to label it, otherwise, it is denoted by ✓. × means that the memory operations are not discussed in the paper.

Models	**Writing**	**Management**			**Reading**
		Merging	**Reflection**	**Forgetting**	
MemoryBank [6]	✓	✓	✓	✓	✓
RET-LLM [7]	✓	×	×	×	✓
ChatDB [96]	✓	×	✓	×	✓
TiM [97]	✓	✓	×	✓	✓
SCM [98]	✓	✓	×	×	✓
Voyager [99]	✓	×	✓	×	✓
MemGPT [100]	✓	×	✓	×	✓
MemoChat [94]	✓	×	×	×	✓
MPC [101]	✓	×	×	×	✓
Generative Agents [83]	✓	×	✓	✓	✓
RecMind [102]	○	×	×	×	✓
Retroformer [103]	✓	✓	✓	×	○
ExpeL [82]	✓	✓	✓	×	○
Synapse [91]	✓	×	×	×	✓
GITM [93]	○	✓	✓	×	✓
ReAct [104]	○	×	×	×	○
Reflexion [5]	✓	✓	✓	×	○
RecAgent [95]	✓	✓	✓	✓	✓
Character-LLM [105]	✓	×	×	×	○
MAC [106]	✓	✓	✓	×	✓
Huatuo [107]	✓	×	×	×	○
ChatDev [1]	✓	×	✓	×	✓
InteRecAgent [108]	✓	✓	✓	×	✓
MetaAgents [109]	✓	×	✓	×	✓
TPTU [110, 111]	○	×	✓	×	✓
MetaGPT [112]	✓	×	✓	×	✓
S^3 [2]	✓	×	✓	✓	✓
InvestLM [113]	✓	×	×	×	○

5.3.3 Memory Reading

When the agents require information for reasoning and decision-making, the memory reading operation will extract related information from memory for usage. Therefore, how to access the related information for the current state is important. Due to the massive quantity of memory entities, and the fact that not all of them are pertinent to the current state, careful design is required to extract useful information based on relevance and other task-orientated factors.

Representative Studies. In ChatDB [96], the memory reading operation is executed by the SQL statements. These statements will be generated by agents as a series of Chain-of-Memory in advance. In MPC [101], the agents can retrieve relevant memory from the memory pool. This method also proposes to provide Chain-of-Thought examples for ignoring certain memory. ExpeL [82] utilizes the Faiss [124] vector store as the pool of memory, and obtains the top-K successful trajectories that share the highest similarity scores with the current task.

Discussion. To some extent, the memory reading and writing operations are collaborative, and the forms of memory writing greatly influence the methods of memory reading. For the forms of textual

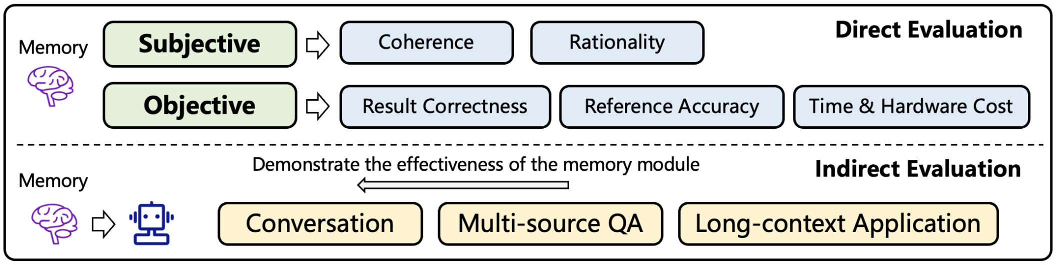


Figure 5: An overview of the evaluation methods of the memory module.

memory, most previous works use the text similarity and other auxiliary information for reading. For the forms of parametric memory, existing models may just utilize the updated parameters for inference, which can be seen as an implicit reading process.

6 How to Evaluate the Memory in LLM-based Agent

How to effectively evaluate the memory module remains an open problem, where diverse evaluation strategies have been proposed in previous works according to different applications. To clearly show the common ideas of different evaluation methods, in this section, we summarize a general framework, which includes two broad evaluation strategies (see **Figure 5** for an overview), that is, (1) direct evaluation, which independently measures the capability of the memory module. (2) indirect evaluation, which evaluates the memory module via end-to-end agent tasks. If the tasks can be effectively accomplished, the memory module is demonstrated to be useful.

6.1 Direct Evaluation

This type of approaches regards the memory of the agents as a stand-alone component and evaluates its effectiveness independently. Previous studies can be categorized into two classes: subjective evaluation and objective evaluation. The subjective evaluation aims to measure memory effectiveness based on human judgments, which can be widely used in the scenarios that lack objective ground truths. Objective evaluation assesses memory effectiveness based on numerical metrics, which makes it easy to compare different memory modules.

6.1.1 Subjective Evaluation

In subjective evaluation, there are two key problems, that is, (1) what aspects should be evaluated and (2) how to conduct the evaluation process. To begin with, the following two aspects are the most common perspectives leveraged to evaluate the memory module.

Coherence. This aspect refers to whether the recalled memory is natural and suitable for the current context. For example, if the agent is making a plan for Alice's travel, the memory should be related to her preference for traveling rather than working. In previous works, Modarressi et al. [7] study whether the memory module could provide proper references among the ever-changing knowledge. Liang et al. [98] present some examples to demonstrate the relation between the current query and historical memory. Zhong et al. [6] and Liu et al. [97] assess the coherence of responses that integrate context and retrieved memory by scoring labels. Lee et al. [101] focus on the contradiction between the recalled memory and contexts.

Rationality. This aspect aims to evaluate whether the recalled memory is reasonable. For example, if the agent is asked to answer "Where is the Summer Palace", the recalled memory should be "The Summer Palace is in Beijing" rather than "The Summer Palace is on the Moon". In previous works, Lee et al. [101] ask crowd workers to directly score the rationality of the retrieved memory. Zhong et al. [6] and Liu et al. [97] recruit human evaluators to check if the memory contains reasonable answers for the current question.

As for how to conduct the evaluation process, there are two important problems. The first one is how to select the human evaluators. In general, the evaluators should be familiar with the evaluation task, which ensures that the labeling results are convincing and reliable. In addition, the backgrounds of the evaluators should be diverse to remove subjective biases of specific human groups. The second problem is how to label the outputs of the memory module. Usually, one can either directly score the results [6] or make comparisons between two candidates [95]. The former can obtain absolute and quantitative evaluation results, while the latter can remove the labeling noises when independently scoring each candidate. In addition, the granularity of the ratings should also be carefully designed. Too coarse ratings may not effectively discriminate the capabilities of different memory modules, while too fine-grained ones may bring more effort for the workers to make judgments.

In general, subjective evaluation can be used in a wide range of scenarios, where one just needs to define the evaluation aspects and let recruited workers make judgments. This method is usually more explainable since the workers can provide the reasons for their judgments. However, subjective evaluation is costly due to the need to employ human evaluators. Additionally, different groups of evaluators may have various biases, making the results difficult to reproduce and compare.

6.1.2 Objective Evaluation

In objective evaluation, previous work usually defines numeric metrics to evaluate the effectiveness and efficiency of the memory module.

Result Correctness. This metric measures whether the agent can successfully answer pre-defined questions directly based on the memory module. For example, the question could be "Where did Alice go today?" with two choices "A: the Summer Palace" and "B: the Great Wall". Then, the agent should choose the correct answer based on the problem and its memory. The agent-generated answer will be compared with the ground truth. Formally, the accuracy can be calculated as

$$\text{Correctness} = \frac{1}{N}\sum_{i=1}^{N} \mathbb{I}\left[a_i = \hat{a}_i\right],$$

where N is the number of problems, a_i represents the ground truth for the i-th problem, $\hat{a}_i$ means the answer given by the agent, and $\mathbb{I}\left[a_i = \hat{a}_i\right]$ is the matching function commonly represented as

$$\mathbb{I}\left[a_i = \hat{a}_i\right] = \begin{cases} 1 & \text{if } a_i = \hat{a}_i, \\ 0 & \text{if } a_i \neq \hat{a}_i. \end{cases}$$

In previous works, Hu et al. [96] construct questions from past histories with annotated ground truths and calculate the accuracy of whether the recalled memory could match the correct answers. Similarly, Packer et al. [100] generate questions and answers that can only be derived from past sessions, and compare the responses from the agents with the ground truths to calculate the accuracy.

Reference Accuracy. This metric evaluates whether the agent can discover relevant memory contents to answer the questions. Different from the above metric, which focuses on the final results, reference accuracy cares more about the intermediate information to support the agent's final decisions. In specific, it compares the retrieved memory with the pre-prepared ground truth. For the above problem of "Where did Alice go today?", if the memory contents include (A) "Alice had lunch with friends at *Wangfujing* today." and (B) "Alice had roast duck for lunch", then a better memory module should select (A) as a reference to answer the question. Usually, researchers leverage F1-score to evaluate the reference accuracy, which is calculated as

$$\text{F1} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}},$$

where the precision and recall scores are calculated as $\text{Precision} = \frac{\text{TP}}{\text{TP+FP}}$ and $\text{Recall} = \frac{\text{TP}}{\text{TP+FN}}$. The TP represents the number of true positive memory contents, FP means the number of false positive memory contents, and FN indicates the number of false negative memory contents. In previous works, Lu et al. [94] utilize F1-score to evaluate the retrieval process of the memory, and Zhong et al. [6] focus on assessing whether related memory can be successfully retrieved.

Result Correctness and Reference Accuracy are both utilized to evaluate the effectiveness of the memory module. Beyond effectiveness, efficiency is also an important aspect, especially for real-world applications. Therefore, we describe the evaluation of efficiency as follows.

Time & Hardware Cost. The total time cost includes the time leveraged for memory adaption and inference. The adaptation time refers to the time of memory writing and memory management, while the inference time indicates the time latency of memory reading. In specific, the difference from the end time to the start time of memory operations can be considered as the time consumption. Formally, the average time consumption of each type of operation can be represented as

$$\Delta\text{time} = \frac{1}{M}\sum_{i=1}^{M} t_i^{\text{end}} - t_i^{\text{start}},$$

where M represents the number of these operations, t_i^{end} means the end time of the i-th operation, and t_i^{start} indicates the start time of that operation. As for the computation overhead, it can be evaluated by the peak GPU memory allocation. In previous works, Tack et al. [106] utilize the peak memory allocation and adaptation time to assess the efficiency of memory operations.

Objective evaluation offers numeric strategies to compare different methods of memory, which is important to benchmark this field and promote future developments.

6.2 Indirect Evaluation

Besides the above method that directly evaluates the memory module, evaluating via task completion is also a popular evaluation strategy. The intuition behind this type of approaches is that if the agent can successfully complete a task that highly depends on memory, it suggests that the designed memory module is effective. In the following parts, we present several representative tasks that are leveraged to evaluate the memory module in indirect ways.

6.2.1 Conversation

Engaging in conversations with humans is one of the most important applications of agents, where memory plays a crucial role in this process. By storing context information in memory, the agents allow users to experience personalized conversations, thus improving users' satisfaction. Therefore, when other parts of the agents are determined, the performance of the conversation tasks can reflect the effectiveness of different memory modules.

In the context of conversation, consistency and engagement are two commonly used methods to evaluate the effectiveness of the agents' memory. Consistency refers to how the response from agents is consistent with the context because dramatic changes should be avoided during the conversation. For example, Lu et al. [94] evaluate the consistency of agents on interactive dialogues, using GPT-4 to score on the responses from agents. Engagement refers to how the user is engaged to continue the conversation. It reflects the quality and attraction of agents' responses, as well as the ability of agents to craft the personas for current conversations. For example, Lee et al. [101] assess the engagingness of responses by SCE-p score, and Packer et al. [100] utilize CSIM score to evaluate the memory effect on increasing engagement of users.

6.2.2 Multi-source Question-answering

Multi-source questing-answering can comprehensively evaluate the memorized information from multiple sources, including inside-trial information, cross-trial information, and external knowledge. It focuses on the integration of memory utilization from various contents and sources.

In previous works, Yao et al. [104] evaluate the memory that integrates information from the task trial and the external knowledge from Wikipedia. Then, Shinn et al. [5] and Yao et al. [103] further include the cross-trial information of the same task, where the memory is permitted to obtain more experiences from previous failed trials. Moreover, Packer et al. [100] allow agents to utilize the memory from multi-document information for question-answering.

By evaluating multi-source question-answering tasks, the memory of agents can be examined on the capability of content integration from various sources. It also reveals the issue of the memory contradiction due to multiple information sources, and the problem of updated knowledge, which can potentially affect the performance of the memory module.

6.2.3 Long-context Applications

Beyond the above general applications, in many scenarios, LLM-based agents have to make decisions based on extremely long prompts. In these scenarios, the long prompts are usually regarded as the memory contents, which play an important role in driving agent behaviors.

In previous works, Huang et al. [19] organize a comprehensive survey for long-context LLMs, which provides a summary of evaluation metrics on long-context scenarios. Moreover, Shaham et al. [138] propose a zero-shot benchmark for evaluating agents' understanding of long-context natural languages. As for specific long-context tasks, long-context passage retrieval is one of the important tasks for evaluating the long-context ability of agents. It requires agents to find the correct paragraph in a long context that corresponds to the given questions or descriptions [139]. Long-context summarization is another representative task. It requests agents to formulate a global understanding of the whole context, and summarizes it according to the descriptions, where some metrics on matching scores like ROUGE can be utilized to compare the results with ground truths.

The evaluation of long-context applications provides broader approaches to assess the function of memory in agents, focusing on practical downstream scenarios. The comprehensive benchmarks [138, 140] also provide an objective assessment for the ability of long-context understanding.

6.2.4 Other Tasks

In addition to the above three types of major tasks for indirect evaluation, there are also some other metrics in general tasks that can reveal the effectiveness of the memory module.

Success rate refers to the proportion of tasks that agents can successfully solve. For Yao et al. [104], Shinn et al. [5] and Zhao et al. [82], they assess how many spacial tasks can be correctly completed through reasoning and memory in AlfWorld [141]. In Zhu et al. [93], they evaluate the success rate of producing different items in Minecraft to show the effect of memory. Moreover, Shinn et al. [5] measure the success rate of passed problems by generated codes, and Zheng et al. [91] calculate the success rate of computer control and accuracy of element selection to show the function of trajectory-as-exemplar memory. Exploration degree typically appears in exploratory games, which reflects the extent that agents can explore the environment. For example, Wang et al. [99] compare the numbers of distinct items explored in Minecraft to reflect the skill learning in memory.

In fact, nearly all the memory-equipped agents can evaluate the effect of memory by ablation studies, comparing the performance between with/without memory modules. The evaluation on specific scenarios can better reflect the significance of memory for the downstream applications practically.

6.3 Discussions

Compared with direct evaluation, indirect evaluation via specific tasks can be easier to conduct, since there are already many public benchmarks. However, the performance on tasks can be attributed to various factors, and memory is only one of them, which may make the evaluation results biased. By direct evaluation, the effectiveness of the memory module can be independently evaluated, which improves the reliability of the evaluation results. However, to our knowledge, there are no open-sourced benchmarks tailored for the memory modules in LLM-based agents.

7 Memory-enhanced Agent Applications

Recently, LLM-based agents have been investigated across a wide variety of scenarios, facilitating societal advancement. In general, most LLM-based agents are equipped with memory modules. However, the specific effects undertaken by these memory components, the particular information they store, and the implementation methods they use, vary across different applications. In order to provide insights for the design of memory functionalities in LLM-based agents, in this section, we review and summarize how memory mechanisms are manifested in LLM-based agents across various application scenarios. In specific, we categorize them into several classes: role-playing and social simulation, personal assistant, open-world games, code generation, recommendation, expert systems in specific domains, and other applications. The summarization is shown in **Table 4**.

7.1 Role-playing and Social Simulation

Role-playing represents a classic application of LLM-based agents, where memory plays a crucial role inside the agents. It endows roles with distinct characteristics, differentiating them from one another. Many previous studies have explored methods for constructing role memories [105, 143, 145–147]. Shao et al. [105] construct the memory of roles by experience uploading, which utilizes SFT to inject memory into model parameters. Li et al. [143] enhance large language models for role-playing via an improved prompt and the character memory extracted from scripts, where user queries and

Table 4: Summarization of memory-enhanced agents applications.

Applications	Models	Applications	Models
Role-playing	Character-LLM [105]	Code Generation	RTLFixer [142]
	ChatHaruhi [143]		GameGPT [144]
	RoleLLM [145]		ChatDev [1]
	NarrativePlay [146]		MetaGPT [109]
	CharacterGLM [147]		CodeAgent [114]
Social Simulation	Generative Agents [83]	Recommendation	RecAgent [95]
	Lyfe Agents [148]		InteRecAgent [108]
	S^3 [2]		RecMind [102]
	MetaAgents [109]		AgentCF [149]
	WarAgent [150]	Medicine	Huatuo [107]
Personal Assistant	MemoryBank [6]		DoctorGLM [129]
	RET-LLM [7]		Radiology-GPT [132]
	MemoChat [94]		Wang et al. [151]
	MemGPT [100]		EHRAgent [152]
	MPC [101]		ChatDoctor [115]
	AutoGen [153]	Finance	InvestLM [113]
	ChatDB [96]		TradingGPT [154]
	TiM [97]		QuantAgent [155]
	SCM [98]		FinMem [156]
Game	Voyager [99]		Koa et al. [157]
	GITM [93]	Science	Chemist-X [158]
	JARVIS [159]		ChemDFM [160]
	LARP [161]		MatChat [162]

chatbot's responses are concatenated to form a sequence as memory. Wang et al. [145] infuse role-specific knowledge and episode memories into LLM-based agents, where context QA pairs are concatenated to form episode memory. Zhao et al. [146] aim to generate human-like responses, guided by personality traits extracted from narratives, which can be stored and retrieved by relevance and importance. Zhou et al. [147] generate character-based dialogues for different roles and empower LLM-based agents with corresponding styles by SFT.

Social simulation is basically an extension of role-playing, which focuses more on multi-agent modeling. The memory module is an important component for such applications, which helps to accurately simulate human dynamic behaviors. In previous studies, Kaiya et al. [148] propose a Summarize-and-Forget memory mechanism for better self-monitoring in social scenarios. Gao et al. [2] focus on social network simulation systems. Each agent in the system has a memory pool, which consists of diverse user messages from online platforms to identify the user. Li et al. [163] maintain conversation contexts, encompassing the economic environment and agent decisions from previous months, in order to simulate the impact of broad macroeconomic trends on agents' decision-making and to make the agents grasp market dynamics. Li et al. [109] simulate the job-seeking scenario in human society, where the memory of agents includes profiles and goals initially and is further enriched with other information, like dialogues and personal reflections. Hua et al. [150] simulate the decisions and consequences of the participating countries in the wars, where the conversations of agents are continuously maintained into memory.

There are several insights in designing an agent's memory for role-play and social simulation. First, the memory should be consistent with the roles' characteristics, which can be used to identify each role and distinguish it from the others. This is crucial for improving the realism of role-play and the diversity of social simulation. Second, the memory should appropriately influence the subsequent actions of the agent to ensure the consistency and rationality of its behaviors. Additionally, for humanoid agents, their memory mechanisms should align with the features of human memory, such as forgetting and long/short-term memory, which should refer to the theories of cognitive psychology.

7.2 Personal Assistant

LLM-based agents are well-suited for creating personal assistants, such as agents capable of engaging in long-term conversations with users [94, 101, 153], as well as those tasked with automatically seeking information [164]. These agents often need to memorize previous dialogues to maintain the consistency, and remember critical styles and events to generate more personalized and relevant responses. Lu et al. [94] maintain the context consistency for dialogues by saving contents and information of conversations, which helps to find proper relevant information by retrieval. Lee et al. [101] summarize conversations to extract important information, store it, and retrieve it for future inference. Pan et al. [164] focus on information-seeking tasks, which design memory modules to store user's context information, and empower external knowledge with tool usage. Wu et al. [153] retain important context as memory to maintain conversation consistency.

In summary, most memory implementations for personal assistants adopt retrieval methods in textual form, because they are better at finding relevant information from pieces of conversations. For the memory storage, the agent should remember the factual information during user-agent interactions, as well as the personal style of users, in order to generate responses that are tailored to the user's situation. Additionally, when recalling memories, the agent should identify and retrieve the memory that is relevant to the current query and context. This principle can enable the agent to correctly understand the user's requirement, and maintain the consistency in conversations.

7.3 Open-world Game

For games and open-world exploration, LLM-based agents always maintain post observations as task contexts, and store experiences in previous successful trials. By leveraging past experiences, agents can avoid making the same mistakes repeatedly and achieve a high-level understanding of environments, thus exploring more effectively. Some of them can acquire external databases or APIs to obtain general knowledge [99, 93, 159, 161]. Wang et al. [99] save obtained skills into memory for further usage in Minecraft. Zhu et al. [93] store and retrieve successful trajectories as examples for similar tasks, and utilize external Minecraft Wiki by API calls. Wang et al. [159] construct multimodal memory as a knowledge library and provide examples for prompt by retrieving interactive experiences. Yan et al. [161] maintain working memory for decision-making, save and retrieve relevant past experiences, and implement external datasets for general knowledge.

In summary, no matter inside-trial or cross-trial information, the key aspect of memory is to reflect on past interactions and draw experiences that can be applied to the subsequent exploration. In addition to accumulating experience through self-involving trials, absorbing external knowledge as part of the agent's memory is also an important way to enhance the exploratory capabilities of the agent.

7.4 Code Generation

In the scenario of code generation, LLM-based agents can search relevant information from the memory, thereby obtaining more knowledge for development. They can save previous experiences for future problems, and also maintain context in conversational development interfaces [142, 144, 1, 109]. Tsai et al. [142] construct an external non-parametric memory database, which stores the compiler errors and human expert instructions for automatic syntax error fixing. In [144], personal information will be stored in the memory, and helps in retaining context and knowledge for decision-making. Qian et al. [1] adopt multi-agents to develop software, where each role maintains a memory to store the past conversations with other roles. Li et al. [109] also focus on software development, and the agent can retrieve its historical records preserved in memory when errors occur. Zhang et al. [114] can search relevant information when they face problems on code generation.

By leveraging external resources, the agents can learn from code-related knowledge and store it into their memory, thereby enhancing the capabilities of code generation. In addition, the memory can improve the continuity and consistency in code generation. By integrating contextual memory, the agent can better understand the requirements for software development, thereby enhancing the coherence of the generated code. Furthermore, the memory is also crucial for the iterative optimization of code, as it can identify the developer's targets based on the histories.

7.5 Recommendation

In the field of recommendation, some previous works focus on simulating users in recommender systems [95, 108], where the memory can represent the user profiles and histories in the real world. Others try to improve the performance of recommendation, or provide other formats of recommendation interfaces [149, 102]. Wang et al. [95] simulate user behaviors in recommendation scenarios to generate data for recommender systems, and the agents store past observations and insights into a hierarchical memory. In Huang et al. [108], the memory in LLM-based agents can archive the user's conversational history over extended periods, as well as capture the most recent dialogues pertinent to the current prompt, to simulate interactive recommender systems. It also uses an actor-critic reflection to improve the robustness of agents. Item agents and user agents are equipped with different memories in [149], where item agents are endowed with dynamic memory modules designed to capture and preserve information pertinent to their intrinsic attributes and the inclinations of their adopters. For user agents, the adaptive memory updating mechanism plays a pivotal role in aligning the agents' operations with user behaviors and preferences. Wang et al. [102] memorize individualized user information like reviews or ratings for items, and acquire domain-specific knowledge and real-time information by web searching tools.

For both simulating users in recommender systems and capturing their preferences, retaining personalized information through memory is essential. A critical challenge lies in how to align the personalized information and feedback with LLMs, and store them into the memory of agents. It is also an important task for bridging the gap between conventional recommendation models and LLMs.

7.6 Expert System in Specific Domains

Medicine Domain. In the field of medicine, most of the previous works empower LLM-based agents with external knowledge in their memory [107, 129, 132, 151, 115]. Wang et al. [107] fine-tune LLaMA [127] with medical knowledge graph CMeKG [165] in QA form, in order to enhance their medical domain knowledge. Xiong et al. [129] adopt LoRA [131] to efficiently fine-tune on foundation models for healthcare. Wang et al. [151] empower LLM-based agents to acquire text-based external knowledge as reasoning reference. Besides, Shi et al. [152] build memory upon the most relevant successful cases from past experiences, and use similarity metric for the retrieval of relevant questions in the medicine domain.

Finance Domain. Some previous works also apply LLM-based agents in the finance domain, whose memory can store financial knowledge [113], market information [154, 156], and successful experiences [157, 155]. Yang et al. [113] construct financial investment dataset to fine-tune LLaMA [127] to empower knowledge on investment. Li et al. [154] design a layered-memory structure to store different types of marketing information. Wang et al. [155] record the ongoing interaction like exchanges and information to ensure consistent response, and record prior outputs as experiences for retrieving relevant examples to provide a diverse learning context for agents. Koa et al. [157] store past price movement and explanations, and generate reflections on previous trials. Yu et al. [156] adopt a layered memory mechanism to provide abundant information for reasoning.

Science. In the domain of science, some existing works design LLM-based agents with a large amount of knowledge in memory to solve problems [158, 160, 162]. Chen et al. [158] include molecule database and online literature as external knowledge for memory in LLM-based agents, and retrieve them when they need related information. Zhao et al. [160] and Chen et al. [162] empower domain knowledge by fine-tuning in Chemistry and structured materials respectively.

To build an expert system based on agents in a specific vertical domain, it is necessary to retain the domain-specific knowledge in their memory. However, there are several challenges. First, domain knowledge is specialized and requires higher accuracy, leading to difficulties in constructing memory storage. Second, domain knowledge is often time-sensitive, which can become outdated in the future. Therefore, the memory needs to be partially updated when some of the knowledge has been out-of-date. Furthermore, the substantial volume of domain knowledge makes it difficult to recall from memory based on the current query.

7.7 Other Applications

There are some other applications of memory in LLM-based agents. Wang et al. [166] focus on the task of cloud root cause analysis, using memory to store framework rules, task requirements, tools documentation, few-shot examples, and agent observations. Qiang et al. [167] solve the problem of ontology matching. The agents save conversational dialogues and construct a rational database for retrieving external knowledge. Wen et al. [168] investigate autonomous driving, whose memory module is constructed by a vector database and contains the experiences from past driving scenarios. Wang et al. [169] propose to improve user acceptance testing, which employs a self-reflection mechanism. After each trial, the operation agent summarizes the conversation and updates the memory pool, until the goal of the current step is accomplished.

For different applications, the focus of memory varies, as it inherently serves the downstream tasks. Therefore, the design should also consider the requirements of tasks.

8 Limitations & Future Directions

8.1 More Advances in Parametric Memory

At present, the memory of LLM-based agents is predominantly in textual form, especially for contextual knowledge such as observation records, trial experiences, and textual knowledge databases. Although textual memory possesses the advantages of being interpretable and easy to expand and edit, it also implies a sacrifice in efficiency compared to parametric memory. Essentially, parametric memory boasts a higher information density, expressing semantics through continuous real-number vectors in a latent space, whereas textual memory employs a combination of tokens in a discrete space for semantic expression. Thus, parametric memory offers a richer expressive space, and its soft encoding is more robust compared to the hard-coded form of token sequences. Additionally, parametric memory is more storage-efficient, where it does not require the explicit storage of extensive texts, similar to a knowledge compression process. As for the memory management, such as merging and reflection, parametric memory does not necessarily design manual rules like textual memory does, but can employ optimization methods to learn these processes implicitly. Moreover, pluggable parametric memory is similar to a digital life card, capable of endowing agents with the requisite characteristics. For example, Huatuo [107] aims to enhance agents with expertise in the biomedical field by refining the Llama [127] model on Chinese medical knowledge bases. MAC [106] is designed to create a parametric memory adaptation framework suitable for online settings, employing meta-learning techniques to replace the traditional optimization phase.

Although parametric memory holds great prospects, it currently faces numerous challenges. Foremost among these is the issue of efficiency: how to effectively transform textual information into parameters or modifications of parameters is a critical question. Presently, researchers can transfer vast amounts of domain knowledge into the parameters of LLMs by SFT. However, it is time-consuming and requires extensive text corpus, making it unsuitable for situational knowledge. One viable approach is to employ meta-learning to let models learn to memorize. For example, MEND [134] leverages the method of meta-learning to train a compact model that has the ability to produce adjustments for the parameters of a pre-trained language model. Moreover, the lack of interpretability associated with parametric memory can be a hindrance, especially in domains requiring high levels of trust, such as medicine. Therefore, enhancing the credibility and interpretability of parametric memory is an urgent issue that needs to be addressed.

8.2 Memory in LLM-based Multi-agent Applications

The exploration of memory mechanisms within LLMs has burgeoned into the dynamic domain of multi-agent systems (MAS), marking significant advancements in the realms of synchronization, communication, and the management of information asymmetry. One pivotal aspect that emerges in the cooperative scenarios is memory synchronization among agents. This process is fundamental for establishing a unified knowledge base, ensuring consistency in decision-making across different agents. For example, Chen et al. [170] emphasize the significance of integrating synchronized memory modules for multi-robot collaboration. Another important aspect is the communication among agents, which heavily relies on memory for maintaining context and interpreting messages. For example, Mandi et al. [171] illustrate memory-driven communication frameworks that foster a common understanding among agents. In addition to cooperative scenarios, some studies also focus on competitive scenarios, and the information asymmetry becomes a crucial issue [172].

Looking ahead, the advancement of memory in LLM-based MAS is poised at the confluence of technological innovation and strategic application. It beckons the exploration of novel memory modules that can further enhance agent synchronization, enable more effective communication, and provide strategic advantages in information-rich environments. The development of such memory models would not only necessitate addressing the current challenges of memory integration and management, but also explore the untapped potentials of memory in facilitating more robust, intelligent, and adaptable MAS. As evidenced by pioneering research, the evolving landscape of LLM-based MAS sets a promising stage for future innovations in memory utilization and management. This exploration is expected to unravel new dimensions of memory integration, pushing the boundaries of what is currently achievable and setting new benchmarks in the realm of MAS.

8.3 Memory-based Lifelong Learning

Lifelong learning is an advanced topic in artificial intelligence, extending the learning capabilities of agents across their life-long span [173]. Agents can continuously interact with their environment, persistently observe environments, and acquire external knowledge, enabling a mode of enhancement like humans. The memory of an agent is key to achieving lifelong learning, as it needs to learn to store and apply the past observations. Lifelong learning in LLM-based agents holds significant practical value, such as in long-term social simulations and personal assistance. However, it also faces several challenges. Firstly, lifelong learning is temporal, necessitating that an agent's memory captures temporality. This temporality could cause interactions between memories, such as memory overlap. Furthermore, due to the extended period of lifelong learning, it needs to store a vast amount of memories and retrieve them when needed, possibly incorporating a certain mechanism for forgetting.

8.4 Memory in Humanoid Agent

A humanoid agent refers to an agent designed to exhibit behaviors consistent with humans, thereby facilitating applications in social simulation, studies of human behavior, and role-playing. Unlike task-oriented agents where greater capability is typically preferred, the proficiency of a humanoid agent should closely mimic that of humans. Consequently, the memory of humanoid agents should align with human cognitive processes, adhering to psychological principles such as memory distortion and forgetfulness. Additionally, humanoid agents should possess knowledge boundaries, meaning that their knowledge should correspond to that of the entity they replicate. For instance, in role-playing scenarios, an agent embodying a child should not possess an understanding of advanced mathematical concepts or other complex knowledge beyond what is typical for that age [174].

9 Conclusion

In this survey, we provide a systematical review on the memory mechanism of LLM-based agents, where we focus on three key problems including "What is", "Why do we need" and "How to design and evaluate" the memory module in LLM-based agents. To show the importance of the agent's memory, we also present many typical applications, where the memory module plays an important role. We believe this survey can offer valuable references for newcomers to this domain, and also hope it can inspire more advanced memory mechanisms to enhance LLM-based agents.

Acknowledgement

We thank Lei Wang for his proofreading and valuable suggestions to this survey.

References

[1] Chen Qian, Xin Cong, Cheng Yang, Weize Chen, Yusheng Su, Juyuan Xu, Zhiyuan Liu, and Maosong Sun. Communicative agents for software development. *arXiv preprint arXiv:2307.07924*, 2023.

[2] Chen Gao, Xiaochong Lan, Zhihong Lu, Jinzhu Mao, Jinghua Piao, Huandong Wang, Depeng Jin, and Yong Li. S^3: Social-network simulation system with large language model-empowered agents. *arXiv preprint arXiv:2307.14984*, 2023.

[3] Lei Wang, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, Jingsen Zhang, Zhiyuan Chen, Jiakai Tang, Xu Chen, Yankai Lin, et al. A survey on large language model based autonomous agents. *arXiv preprint arXiv:2308.11432*, 2023.

[4] Zhiheng Xi, Wenxiang Chen, Xin Guo, Wei He, Yiwen Ding, Boyang Hong, Ming Zhang, Junzhe Wang, Senjie Jin, Enyu Zhou, et al. The rise and potential of large language model based agents: A survey. *arXiv preprint arXiv:2309.07864*, 2023.

[5] Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik R Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

[6] Wanjun Zhong, Lianghong Guo, Qiqi Gao, and Yanlin Wang. Memorybank: Enhancing large language models with long-term memory. *arXiv preprint arXiv:2305.10250*, 2023.

[7] Ali Modarressi, Ayyoob Imani, Mohsen Fayyaz, and Hinrich Schütze. Ret-llm: Towards a general read-write memory for large language models. *arXiv preprint arXiv:2305.14322*, 2023.

[8] Shengyu Zhang, Linfeng Dong, Xiaoya Li, Sen Zhang, Xiaofei Sun, Shuhe Wang, Jiwei Li, Runyi Hu, Tianwei Zhang, Fei Wu, et al. Instruction tuning for large language models: A survey. *arXiv preprint arXiv:2308.10792*, 2023.

[9] Tianhao Shen, Renren Jin, Yufei Huang, Chuang Liu, Weilong Dong, Zishan Guo, Xinwei Wu, Yan Liu, and Deyi Xiong. Large language model alignment: A survey. *arXiv preprint arXiv:2309.15025*, 2023.

[10] Yufei Wang, Wanjun Zhong, Liangyou Li, Fei Mi, Xingshan Zeng, Wenyong Huang, Lifeng Shang, Xin Jiang, and Qun Liu. Aligning large language models with human: A survey. *arXiv preprint arXiv:2307.12966*, 2023.

[11] Yang Liu, Yuanshun Yao, Jean-Francois Ton, Xiaoying Zhang, Ruocheng Guo Hao Cheng, Yegor Klochkov, Muhammad Faaiz Taufiq, and Hang Li. Trustworthy llms: a survey and guideline for evaluating large language models' alignment. *arXiv preprint arXiv:2308.05374*, 2023.

[12] Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, and Haofen Wang. Retrieval-augmented generation for large language models: A survey. *arXiv preprint arXiv:2312.10997*, 2023.

[13] Song Wang, Yaochen Zhu, Haochen Liu, Zaiyi Zheng, Chen Chen, et al. Knowledge editing for large language models: A survey. *arXiv preprint arXiv:2310.16218*, 2023.

[14] Yunzhi Yao, Peng Wang, Bozhong Tian, Siyuan Cheng, Zhoubo Li, Shumin Deng, Huajun Chen, and Ningyu Zhang. Editing large language models: Problems, methods, and opportunities. *arXiv preprint arXiv:2305.13172*, 2023.

[15] Peng Wang, Ningyu Zhang, Xin Xie, Yunzhi Yao, Bozhong Tian, Mengru Wang, Zekun Xi, Siyuan Cheng, Kangwei Liu, Guozhou Zheng, et al. Easyedit: An easy-to-use knowledge editing framework for large language models. *arXiv preprint arXiv:2308.07269*, 2023.

[16] Zhangyin Feng, Weitao Ma, Weijiang Yu, Lei Huang, Haotian Wang, Qianglong Chen, Weihua Peng, Xiaocheng Feng, Bing Qin, et al. Trends in integration of knowledge and large language models: A survey and taxonomy of methods, benchmarks, and applications. *arXiv preprint arXiv:2311.05876*, 2023.

[17] Ningyu Zhang, Yunzhi Yao, Bozhong Tian, Peng Wang, Shumin Deng, Mengru Wang, Zekun Xi, Shengyu Mao, Jintian Zhang, Yuansheng Ni, et al. A comprehensive study of knowledge editing for large language models. *arXiv preprint arXiv:2401.01286*, 2024.

[18] Yujia Qin, Shengding Hu, Yankai Lin, Weize Chen, Ning Ding, Ganqu Cui, Zheni Zeng, Yufei Huang, Chaojun Xiao, Chi Han, et al. Tool learning with foundation models. *arXiv preprint arXiv:2304.08354*, 2023.

[19] Yunpeng Huang, Jingwei Xu, Zixu Jiang, Junyu Lai, Zenan Li, Yuan Yao, Taolue Chen, Lijuan Yang, Zhou Xin, and Xiaoxing Ma. Advancing transformer architecture in long-context large language models: A comprehensive survey. *arXiv preprint arXiv:2311.12351*, 2023.

[20] Xindi Wang, Mahsa Salmani, Parsa Omidi, Xiangyu Ren, Mehdi Rezagholizadeh, and Armaghan Eshaghi. Beyond the limits: A survey of techniques to extend the context length in large language models. *arXiv preprint arXiv:2402.02244*, 2024.

[21] Saurav Pawar, SM Tonmoy, SM Zaman, Vinija Jain, Aman Chadha, and Amitava Das. The what, why, and how of context length extension techniques in large language models–a detailed survey. *arXiv preprint arXiv:2401.07872*, 2024.

[22] Jiayang Wu, Wensheng Gan, Zefeng Chen, Shicheng Wan, and S Yu Philip. Multimodal large language models: A survey. In *2023 IEEE International Conference on Big Data (BigData)*, pages 2247–2256. IEEE, 2023.

[23] Shezheng Song, Xiaopeng Li, and Shasha Li. How to bridge the gap between modalities: A comprehensive survey on multimodal large language model. *arXiv preprint arXiv:2311.07594*, 2023.

[24] Davide Caffagni, Federico Cocchi, Luca Barsellotti, Nicholas Moratelli, Sara Sarto, Lorenzo Baraldi, Marcella Cornia, and Rita Cucchiara. The (r) evolution of multimodal large language models: A survey. *arXiv preprint arXiv:2402.12451*, 2024.

[25] Shukang Yin, Chaoyou Fu, Sirui Zhao, Ke Li, Xing Sun, Tong Xu, and Enhong Chen. A survey on multimodal large language models. *arXiv preprint arXiv:2306.13549*, 2023.

[26] Guangji Bai, Zheng Chai, Chen Ling, Shiyu Wang, Jiaying Lu, Nan Zhang, Tingwei Shi, Ziyang Yu, Mengdan Zhu, Yifei Zhang, et al. Beyond efficiency: A systematic survey of resource-efficient large language models. *arXiv preprint arXiv:2401.00625*, 2024.

[27] Zhongwei Wan, Xin Wang, Che Liu, Samiul Alam, Yu Zheng, Zhongnan Qu, Shen Yan, Yi Zhu, Quanlu Zhang, Mosharaf Chowdhury, et al. Efficient large language models: A survey. *arXiv preprint arXiv:2312.03863*, 1, 2023.

[28] Xupeng Miao, Gabriele Oliaro, Zhihao Zhang, Xinhao Cheng, Hongyi Jin, Tianqi Chen, and Zhihao Jia. Towards efficient generative large language model serving: A survey from algorithms to systems. *arXiv preprint arXiv:2312.15234*, 2023.

[29] Lingling Xu, Haoran Xie, Si-Zhao Joe Qin, Xiaohui Tao, and Fu Lee Wang. Parameter-efficient fine-tuning methods for pretrained language models: A critical review and assessment. *arXiv preprint arXiv:2312.12148*, 2023.

[30] Xunyu Zhu, Jian Li, Yong Liu, Can Ma, and Weiping Wang. A survey on model compression for large language models. *arXiv preprint arXiv:2308.07633*, 2023.

[31] Canwen Xu and Julian McAuley. A survey on model compression and acceleration for pretrained language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 10566–10575, 2023.

[32] Wenxiao Wang, Wei Chen, Yicong Luo, Yongliu Long, Zhengkai Lin, Liye Zhang, Binbin Lin, Deng Cai, and Xiaofei He. Model compression and efficient inference for large language models: A survey. *arXiv preprint arXiv:2402.09748*, 2024.

[33] Seungcheol Park, Jaehyeon Choi, Sojin Lee, and U Kang. A comprehensive survey of compression algorithms for language models. *arXiv preprint arXiv:2401.15347*, 2024.

[34] Yupeng Chang, Xu Wang, Jindong Wang, Yuan Wu, Linyi Yang, Kaijie Zhu, Hao Chen, Xiaoyuan Yi, Cunxiang Wang, Yidong Wang, et al. A survey on evaluation of large language models. *ACM Transactions on Intelligent Systems and Technology*, 2023.

[35] Zishan Guo, Renren Jin, Chuang Liu, Yufei Huang, Dan Shi, Linhao Yu, Yan Liu, Jiaxuan Li, Bojian Xiong, Deyi Xiong, et al. Evaluating large language models: A comprehensive survey. *arXiv preprint arXiv:2310.19736*, 2023.

[36] Jingfeng Yang, Hongye Jin, Ruixiang Tang, Xiaotian Han, Qizhang Feng, Haoming Jiang, Bing Yin, and Xia Hu. Harnessing the power of llms in practice: A survey on chatgpt and beyond. *arXiv preprint arXiv:2304.13712*, 2023.

[37] Yutao Zhu, Huaying Yuan, Shuting Wang, Jiongnan Liu, Wenhan Liu, Chenlong Deng, Zhicheng Dou, and Ji-Rong Wen. Large language models for information retrieval: A survey. *arXiv preprint arXiv:2308.07107*, 2023.

[38] Derong Xu, Wei Chen, Wenjun Peng, Chao Zhang, Tong Xu, Xiangyu Zhao, Xian Wu, Yefeng Zheng, and Enhong Chen. Large language models for generative information extraction: A survey. *arXiv preprint arXiv:2312.17617*, 2023.

[39] Angela Fan, Beliz Gokkaya, Mark Harman, Mitya Lyubarskiy, Shubho Sengupta, Shin Yoo, and Jie M Zhang. Large language models for software engineering: Survey and open problems. *arXiv preprint arXiv:2310.03533*, 2023.

[40] Junjie Wang, Yuchao Huang, Chunyang Chen, Zhe Liu, Song Wang, and Qing Wang. Software testing with large language models: Survey, landscape, and vision. *IEEE Transactions on Software Engineering*, 2024.

[41] Zibin Zheng, Kaiwen Ning, Yanlin Wang, Jingwen Zhang, Dewu Zheng, Mingxi Ye, and Jiachi Chen. A survey of large language models for code: Evolution, benchmarking, and future trends. *arXiv preprint arXiv:2311.10372*, 2023.

[42] Fanlong Zeng, Wensheng Gan, Yongheng Wang, Ning Liu, and Philip S Yu. Large language models for robotics: A survey. *arXiv preprint arXiv:2311.07226*, 2023.

[43] Can Cui, Yunsheng Ma, Xu Cao, Wenqian Ye, Yang Zhou, Kaizhao Liang, Jintai Chen, Juanwu Lu, Zichong Yang, Kuei-Da Liao, et al. A survey on multimodal large language models for autonomous driving. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 958–979, 2024.

[44] Zhenjie Yang, Xiaosong Jia, Hongyang Li, and Junchi Yan. A survey of large language models for autonomous driving. *arXiv preprint arXiv:2311.01043*, 2023.

[45] Kai He, Rui Mao, Qika Lin, Yucheng Ruan, Xiang Lan, Mengling Feng, and Erik Cambria. A survey of large language models for healthcare: from data, technology, and applications to accountability and ethics. *arXiv preprint arXiv:2310.05694*, 2023.

[46] Hongjian Zhou, Boyang Gu, Xinyu Zou, Yiru Li, Sam S Chen, Peilin Zhou, Junling Liu, Yining Hua, Chengfeng Mao, Xian Wu, et al. A survey of large language models in medicine: Progress, application, and challenge. *arXiv preprint arXiv:2311.05112*, 2023.

[47] Benyou Wang, Qianqian Xie, Jiahuan Pei, Zhihong Chen, Prayag Tiwari, Zhao Li, and Jie Fu. Pre-trained language models in biomedical domain: A systematic survey. *ACM Computing Surveys*, 56(3):1–52, 2023.

[48] Yinheng Li, Shaofei Wang, Han Ding, and Hang Chen. Large language models in finance: A survey. In *Proceedings of the Fourth ACM International Conference on AI in Finance*, pages 374–382, 2023.

[49] Tianyu He, Guanghui Fu, Yijing Yu, Fan Wang, Jianqiang Li, Qing Zhao, Changwei Song, Hongzhi Qi, Dan Luo, Huijing Zou, et al. Towards a psychological generalist ai: A survey of current applications of large language models and future prospects. *arXiv preprint arXiv:2312.04578*, 2023.

[50] Lei Li, Yongfeng Zhang, Dugang Liu, and Li Chen. Large language models for generative recommendation: A survey and visionary discussions. *arXiv preprint arXiv:2309.01157*, 2023.

[51] Jianghao Lin, Xinyi Dai, Yunjia Xi, Weiwen Liu, Bo Chen, Xiangyang Li, Chenxu Zhu, Huifeng Guo, Yong Yu, Ruiming Tang, et al. How can recommender systems benefit from large language models: A survey. *arXiv preprint arXiv:2306.05817*, 2023.

[52] Wenjie Wang, Xinyu Lin, Fuli Feng, Xiangnan He, and Tat-Seng Chua. Generative recommendation: Towards next-generation recommender paradigm. *arXiv preprint arXiv:2304.03516*, 2023.

[53] Yue Zhang, Yafu Li, Leyang Cui, Deng Cai, Lemao Liu, Tingchen Fu, Xinting Huang, Enbo Zhao, Yu Zhang, Yulong Chen, et al. Siren's song in the ai ocean: a survey on hallucination in large language models. *arXiv preprint arXiv:2309.01219*, 2023.

[54] Lei Huang, Weijiang Yu, Weitao Ma, Weihong Zhong, Zhangyin Feng, Haotian Wang, Qianglong Chen, Weihua Peng, Xiaocheng Feng, Bing Qin, et al. A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions. *arXiv preprint arXiv:2311.05232*, 2023.

[55] Vipula Rawte, Amit Sheth, and Amitava Das. A survey of hallucination in large foundation models. *arXiv preprint arXiv:2309.05922*, 2023.

[56] Hongbin Ye, Tong Liu, Aijia Zhang, Wei Hua, and Weiqiang Jia. Cognitive mirage: A review of hallucinations in large language models. *arXiv preprint arXiv:2309.06794*, 2023.

[57] Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Ye Jin Bang, Andrea Madotto, and Pascale Fung. Survey of hallucination in natural language generation. *ACM Computing Surveys*, 55(12):1–38, 2023.

[58] SM Tonmoy, SM Zaman, Vinija Jain, Anku Rani, Vipula Rawte, Aman Chadha, and Amitava Das. A comprehensive survey of hallucination mitigation techniques in large language models. *arXiv preprint arXiv:2401.01313*, 2024.

[59] Xuhui Jiang, Yuxing Tian, Fengrui Hua, Chengjin Xu, Yuanzhuo Wang, and Jian Guo. A survey on large language model hallucination via a creativity perspective. *arXiv preprint arXiv:2402.06647*, 2024.

[60] Isabel O Gallegos, Ryan A Rossi, Joe Barrow, Md Mehrab Tanjim, Sungchul Kim, Franck Dernoncourt, Tong Yu, Ruiyi Zhang, and Nesreen K Ahmed. Bias and fairness in large language models: A survey. *arXiv preprint arXiv:2309.00770*, 2023.

[61] Hadas Kotek, Rikker Dockum, and David Sun. Gender bias and stereotypes in large language models. In *Proceedings of The ACM Collective Intelligence Conference*, pages 12–24, 2023.

[62] Yingji Li, Mengnan Du, Rui Song, Xin Wang, and Ying Wang. A survey on fairness in large language models. *arXiv preprint arXiv:2308.10149*, 2023.

[63] Haiyan Zhao, Hanjie Chen, Fan Yang, Ninghao Liu, Huiqi Deng, Hengyi Cai, Shuaiqiang Wang, Dawei Yin, and Mengnan Du. Explainability for large language models: A survey. *ACM Transactions on Intelligent Systems and Technology*, 2023.

[64] Yifan Yao, Jinhao Duan, Kaidi Xu, Yuanfang Cai, Eric Sun, and Yue Zhang. A survey on large language model (llm) security and privacy: The good, the bad, and the ugly. *arXiv preprint arXiv:2312.02003*, 1, 2023.

[65] Erfan Shayegani, Md Abdullah Al Mamun, Yu Fu, Pedram Zaree, Yue Dong, and Nael Abu-Ghazaleh. Survey of vulnerabilities in large language models revealed by adversarial attacks. *arXiv preprint arXiv:2310.10844*, 2023.

[66] Seth Neel and Peter Chang. Privacy issues in large language models: A survey. *arXiv preprint arXiv:2312.06717*, 2023.

[67] Victoria Smith, Ali Shahin Shamsabadi, Carolyn Ashurst, and Adrian Weller. Identifying and mitigating privacy risks stemming from language models: A survey. *arXiv preprint arXiv:2310.01424*, 2023.

[68] Zhichen Dong, Zhanhui Zhou, Chao Yang, Jing Shao, and Yu Qiao. Attacks, defenses and evaluations for llm conversation safety: A survey. *arXiv preprint arXiv:2402.09283*, 2024.

[69] Badhan Chandra Das, M Hadi Amini, and Yanzhao Wu. Security and privacy challenges of large language models: A survey. *arXiv preprint arXiv:2402.00888*, 2024.

[70] Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, et al. A survey of large language models. *arXiv preprint arXiv:2303.18223*, 2023.

[71] Muhammad Usman Hadi, Rizwan Qureshi, Abbas Shah, Muhammad Irfan, Anas Zafar, Muhammad Bilal Shaikh, Naveed Akhtar, Jia Wu, Seyedali Mirjalili, et al. A survey on large language models: Applications, challenges, limitations, and practical usage. *Authorea Preprints*, 2023.

[72] Bonan Min, Hayley Ross, Elior Sulem, Amir Pouran Ben Veyseh, Thien Huu Nguyen, Oscar Sainz, Eneko Agirre, Ilana Heintz, and Dan Roth. Recent advances in natural language processing via large pre-trained language models: A survey. *ACM Computing Surveys*, 56(2): 1–40, 2023.

[73] Grégoire Mialon, Roberto Dessì, Maria Lomeli, Christoforos Nalmpantis, Ram Pasunuru, Roberta Raileanu, Baptiste Rozière, Timo Schick, Jane Dwivedi-Yu, Asli Celikyilmaz, et al. Augmented language models: a survey. *arXiv preprint arXiv:2302.07842*, 2023.

[74] Xu Huang, Weiwen Liu, Xiaolong Chen, Xingmei Wang, Hao Wang, Defu Lian, Yasheng Wang, Ruiming Tang, and Enhong Chen. Understanding the planning of llm agents: A survey. *arXiv preprint arXiv:2402.02716*, 2024.

[75] Taicheng Guo, Xiuying Chen, Yaqi Wang, Ruidi Chang, Shichao Pei, Nitesh V Chawla, Olaf Wiest, and Xiangliang Zhang. Large language model based multi-agents: A survey of progress and challenges. *arXiv preprint arXiv:2402.01680*, 2024.

[76] Yuanchun Li, Hao Wen, Weijun Wang, Xiangyu Li, Yizhen Yuan, Guohong Liu, Jiacheng Liu, Wenxing Xu, Xiang Wang, Yi Sun, et al. Personal llm agents: Insights and survey about the capability, efficiency and security. *arXiv preprint arXiv:2401.05459*, 2024.

[77] Pengyu Zhao, Zijian Jin, and Ning Cheng. An in-depth survey of large language model-based artificial intelligence agents. *arXiv preprint arXiv:2309.14365*, 2023.

[78] Yuheng Cheng, Ceyao Zhang, Zhengwen Zhang, Xiangrui Meng, Sirui Hong, Wenhao Li, Zihao Wang, Zekai Wang, Feng Yin, Junhua Zhao, et al. Exploring large language model based intelligent agents: Definitions, methods, and prospects. *arXiv preprint arXiv:2401.03428*, 2024.

[79] Zane Durante, Qiuyuan Huang, Naoki Wake, Ran Gong, Jae Sung Park, Bidipta Sarkar, Rohan Taori, Yusuke Noda, Demetri Terzopoulos, Yejin Choi, et al. Agent ai: Surveying the horizons of multimodal interaction. *arXiv preprint arXiv:2401.03568*, 2024.

[80] Yingqiang Ge, Yujie Ren, Wenyue Hua, Shuyuan Xu, Juntao Tan, and Yongfeng Zhang. Llm as os (llmao), agents as apps: Envisioning aios, agents and the aios-agent ecosystem. *arXiv preprint arXiv:2312.03815*, 2023.

[81] Ning Ding, Yujia Qin, Guang Yang, Fuchao Wei, Zonghan Yang, Yusheng Su, Shengding Hu, Yulin Chen, Chi-Min Chan, Weize Chen, et al. Delta tuning: A comprehensive study of parameter efficient methods for pre-trained language models. *arXiv preprint arXiv:2203.06904*, 2022.

[82] Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. Expel: Llm agents are experiential learners. *arXiv preprint arXiv:2308.10144*, 2023.

[83] Joon Sung Park, Joseph O'Brien, Carrie Jun Cai, Meredith Ringel Morris, Percy Liang, and Michael S Bernstein. Generative agents: Interactive simulacra of human behavior. In *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology*, pages 1–22, 2023.

[84] Robert L Solso and Jerome Kagan. *Cognitive psychology*. Houghton Mifflin Harcourt P, 1979.

[85] Fergus IM Craik and Robert S Lockhart. Levels of processing: A framework for memory research. *Journal of verbal learning and verbal behavior*, 11(6):671–684, 1972.

[86] Selma Leydesdorff. *Memory cultures: Memory, subjectivity and recognition*. Routledge, 2017.

[87] Philip Nicholas Johnson-Laird. *Mental models: Towards a cognitive science of language, inference, and consciousness*. Number 6. Harvard University Press, 1983.

[88] John E Laird. *The Soar cognitive architecture*. MIT press, 2019.

[89] Ron Sun. *Duality of the mind: A bottom-up approach toward cognition*. Psychology Press, 2001.

[90] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[91] Longtao Zheng, Rundong Wang, Xinrun Wang, and Bo An. Synapse: Trajectory-as-exemplar prompting with memory for computer control. In *NeurIPS 2023 Foundation Models for Decision Making Workshop*, 2023.

[92] Ali Montazeralghaem, Hamed Zamani, and James Allan. A reinforcement learning framework for relevance feedback. In *Proceedings of the 43rd international acm sigir conference on research and development in information retrieval*, pages 59–68, 2020.

[93] Xizhou Zhu, Yuntao Chen, Hao Tian, Chenxin Tao, Weijie Su, Chenyu Yang, Gao Huang, Bin Li, Lewei Lu, Xiaogang Wang, et al. Ghost in the minecraft: Generally capable agents for open-world enviroments via large language models with text-based knowledge and memory. *arXiv preprint arXiv:2305.17144*, 2023.

[94] Junru Lu, Siyu An, Mingbao Lin, Gabriele Pergola, Yulan He, Di Yin, Xing Sun, and Yunsheng Wu. Memochat: Tuning llms to use memos for consistent long-range open-domain conversation. *arXiv preprint arXiv:2308.08239*, 2023.

[95] Lei Wang, Jingsen Zhang, Hao Yang, Zhiyuan Chen, Jiakai Tang, Zeyu Zhang, Xu Chen, Yankai Lin, Ruihua Song, Wayne Xin Zhao, Jun Xu, Zhicheng Dou, Jun Wang, and Ji-Rong Wen. When large language model based agent meets user behavior analysis: A novel user simulation paradigm, 2023.

[96] Chenxu Hu, Jie Fu, Chenzhuang Du, Simian Luo, Junbo Zhao, and Hang Zhao. Chatdb: Augmenting llms with databases as their symbolic memory. *arXiv preprint arXiv:2306.03901*, 2023.

[97] Lei Liu, Xiaoyan Yang, Yue Shen, Binbin Hu, Zhiqiang Zhang, Jinjie Gu, and Guannan Zhang. Think-in-memory: Recalling and post-thinking enable llms with long-term memory. *arXiv preprint arXiv:2311.08719*, 2023.

[98] Xinnian Liang, Bing Wang, Hui Huang, Shuangzhi Wu, Peihao Wu, Lu Lu, Zejun Ma, and Zhoujun Li. Unleashing infinite-length input capacity for large-scale language models with self-controlled memory system. *arXiv preprint arXiv:2304.13343*, 2023.

[99] Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models. *arXiv preprint arXiv:2305.16291*, 2023.

[100] Charles Packer, Vivian Fang, Shishir G Patil, Kevin Lin, Sarah Wooders, and Joseph E Gonzalez. Memgpt: Towards llms as operating systems. *arXiv preprint arXiv:2310.08560*, 2023.

[101] Gibbeum Lee, Volker Hartmann, Jongho Park, Dimitris Papailiopoulos, and Kangwook Lee. Prompted llms as chatbot modules for long open-domain conversation. *arXiv preprint arXiv:2305.04533*, 2023.

[102] Yancheng Wang, Ziyan Jiang, Zheng Chen, Fan Yang, Yingxue Zhou, Eunah Cho, Xing Fan, Xiaojiang Huang, Yanbin Lu, and Yingzhen Yang. Recmind: Large language model powered agent for recommendation. *arXiv preprint arXiv:2308.14296*, 2023.

[103] Weiran Yao, Shelby Heinecke, Juan Carlos Niebles, Zhiwei Liu, Yihao Feng, Le Xue, Rithesh Murthy, Zeyuan Chen, Jianguo Zhang, Devansh Arpit, et al. Retroformer: Retrospective large language agents with policy gradient optimization. *arXiv preprint arXiv:2308.02151*, 2023.

[104] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*, 2022.

[105] Yunfan Shao, Linyang Li, Junqi Dai, and Xipeng Qiu. Character-llm: A trainable agent for role-playing. *arXiv preprint arXiv:2310.10158*, 2023.

[106] Jihoon Tack, Jaehyung Kim, Eric Mitchell, Jinwoo Shin, Yee Whye Teh, and Jonathan Richard Schwarz. Online adaptation of language models with a memory of amortized contexts. *arXiv preprint arXiv:2403.04317*, 2024.

[107] Haochun Wang, Chi Liu, Nuwa Xi, Zewen Qiang, Sendong Zhao, Bing Qin, and Ting Liu. Huatuo: Tuning llama model with chinese medical knowledge. *arXiv preprint arXiv:2304.06975*, 2023.

[108] Xu Huang, Jianxun Lian, Yuxuan Lei, Jing Yao, Defu Lian, and Xing Xie. Recommender ai agent: Integrating large language models for interactive recommendations. *arXiv preprint arXiv:2308.16505*, 2023.

[109] Yuan Li, Yixuan Zhang, and Lichao Sun. Metaagents: Simulating interactions of human behaviors for llm-based task-oriented coordination via collaborative generative agents. *arXiv preprint arXiv:2310.06500*, 2023.

[110] Jingqing Ruan, Yihong Chen, Bin Zhang, Zhiwei Xu, Tianpeng Bao, Guoqing Du, Shiwei Shi, Hangyu Mao, Xingyu Zeng, and Rui Zhao. Tptu: Task planning and tool usage of large language model-based ai agents. *arXiv preprint arXiv:2308.03427*, 2023.

[111] Yilun Kong, Jingqing Ruan, Yihong Chen, Bin Zhang, Tianpeng Bao, Shiwei Shi, Guoqing Du, Xiaoru Hu, Hangyu Mao, Ziyue Li, et al. Tptu-v2: Boosting task planning and tool usage of large language model-based agents in real-world systems. *arXiv preprint arXiv:2311.11315*, 2023.

[112] Sirui Hong, Xiawu Zheng, Jonathan Chen, Yuheng Cheng, Jinlin Wang, Ceyao Zhang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, Liyang Zhou, et al. Metagpt: Meta programming for multi-agent collaborative framework. *arXiv preprint arXiv:2308.00352*, 2023.

[113] Yi Yang, Yixuan Tang, and Kar Yan Tam. Investlm: A large language model for investment using financial domain instruction tuning. *arXiv preprint arXiv:2309.13064*, 2023.

[114] Kechi Zhang, Jia Li, Ge Li, Xianjie Shi, and Zhi Jin. Codeagent: Enhancing code generation with tool-integrated agent systems for real-world repo-level coding challenges. *arXiv preprint arXiv:2401.07339*, 2024.

[115] Li Yunxiang, Li Zihan, Zhang Kai, Dan Ruilong, and Zhang You. Chatdoctor: A medical chat model fine-tuned on llama model using medical domain knowledge. *arXiv preprint arXiv:2303.14070*, 2023.

[116] Dacheng Li, Rulin Shao, Anze Xie, Ying Sheng, Lianmin Zheng, Joseph Gonzalez, Ion Stoica, Xuezhe Ma, and Hao Zhang. How long can context length of open-source llms truly promise? In *NeurIPS 2023 Workshop on Instruction Tuning and Instruction Following*, 2023.

[117] Ziheng Huang, Sebastian Gutierrez, Hemanth Kamana, and Stephen MacNeil. Memory sandbox: Transparent and interactive memory management for conversational agents. In *Adjunct Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology*, pages 1–3, 2023.

[118] Arka Pal, Deep Karkhanis, Manley Roberts, Samuel Dooley, Arvind Sundararajan, and Siddartha Naidu. Giraffe: Adventures in expanding context lengths in llms. *arXiv preprint arXiv:2308.10882*, 2023.

[119] Szymon Tworkowski, Konrad Staniszewski, Mikołaj Pacek, Yuhuai Wu, Henryk Michalewski, and Piotr Miłoś. Focused transformer: Contrastive training for context scaling, 2023.

[120] Nelson F Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. Lost in the middle: How language models use long contexts. *arXiv preprint arXiv:2307.03172*, 2023.

[121] Peter J Denning. The locality principle. *Communications of the ACM*, 48(7):19–24, 2005.

[122] Hermann Ebbinghaus. Memory: A contribution to experimental psychology, trans. *HA Ruger & CE Bussenius. Teachers College.[rWvH]*, 1885.

[123] Jaap MJ Murre and Joeri Dros. Replication and analysis of ebbinghaus' forgetting curve. *PloS one*, 10(7):e0120644, 2015.

[124] Jeff Johnson, Matthijs Douze, and Hervé Jégou. Billion-scale similarity search with gpus. *IEEE Transactions on Big Data*, 7(3):535–547, 2019.

[125] Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. *arXiv preprint arXiv:2302.04761*, 2023.

[126] Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, et al. Toolllm: Facilitating large language models to master 16000+ real-world apis. *arXiv preprint arXiv:2307.16789*, 2023.

[127] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

[128] Zhibin Gou, Zhihong Shao, Yeyun Gong, Yujiu Yang, Minlie Huang, Nan Duan, Weizhu Chen, et al. Tora: A tool-integrated reasoning agent for mathematical problem solving. *arXiv preprint arXiv:2309.17452*, 2023.

[129] Honglin Xiong, Sheng Wang, Yitao Zhu, Zihao Zhao, Yuxiao Liu, Qian Wang, and Dinggang Shen. Doctorglm: Fine-tuning your chinese doctor is not a herculean task. *arXiv preprint arXiv:2304.01097*, 2023.

[130] Aohan Zeng, Xiao Liu, Zhengxiao Du, Zihan Wang, Hanyu Lai, Ming Ding, Zhuoyi Yang, Yifan Xu, Wendi Zheng, Xiao Xia, et al. Glm-130b: An open bilingual pre-trained model. *arXiv preprint arXiv:2210.02414*, 2022.

[131] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.

[132] Zhengliang Liu, Aoxiao Zhong, Yiwei Li, Longtao Yang, Chao Ju, Zihao Wu, Chong Ma, Peng Shu, Cheng Chen, Sekeun Kim, et al. Radiology-gpt: A large language model for radiology. *arXiv preprint arXiv:2306.08666*, 2023.

[133] Nicola De Cao, Wilker Aziz, and Ivan Titov. Editing factual knowledge in language models. *arXiv preprint arXiv:2104.08164*, 2021.

[134] Eric Mitchell, Charles Lin, Antoine Bosselut, Chelsea Finn, and Christopher D Manning. Fast model editing at scale. *arXiv preprint arXiv:2110.11309*, 2021.

[135] Shengyu Mao, Ningyu Zhang, Xiaohan Wang, Mengru Wang, Yunzhi Yao, Yong Jiang, Pengjun Xie, Fei Huang, and Huajun Chen. Editing personality for large language models. 2023.

[136] Jun-Yu Ma, Jia-Chen Gu, Ningyu Zhang, and Zhen-Hua Ling. Neighboring perturbations of knowledge editing on large language models. *arXiv preprint arXiv:2401.17623*, 2024.

[137] Mengru Wang, Ningyu Zhang, Ziwen Xu, Zekun Xi, Shumin Deng, Yunzhi Yao, Qishen Zhang, Linyi Yang, Jindong Wang, and Huajun Chen. Detoxifying large language models via knowledge editing. *arXiv preprint arXiv:2403.14472*, 2024.

[138] Uri Shaham, Maor Ivgi, Avia Efrat, Jonathan Berant, and Omer Levy. Zeroscrolls: A zero-shot benchmark for long text understanding. *arXiv preprint arXiv:2305.14196*, 2023.

[139] Yushi Bai, Xin Lv, Jiajie Zhang, Hongchang Lyu, Jiankai Tang, Zhidian Huang, Zhengxiao Du, Xiao Liu, Aohan Zeng, Lei Hou, et al. Longbench: A bilingual, multitask benchmark for long context understanding. *arXiv preprint arXiv:2308.14508*, 2023.

[140] Dacheng Li, Rulin Shao, Anze Xie, Ying Sheng, Lianmin Zheng, Joseph Gonzalez, Ion Stoica, Xuezhe Ma, and Hao Zhang. How long can context length of open-source llms truly promise? In *NeurIPS 2023 Workshop on Instruction Tuning and Instruction Following*, 2023.

[141] Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. Alfworld: Aligning text and embodied environments for interactive learning. *arXiv preprint arXiv:2010.03768*, 2020.

[142] YunDa Tsai, Mingjie Liu, and Haoxing Ren. Rtlfixer: Automatically fixing rtl syntax errors with large language models. *arXiv preprint arXiv:2311.16543*, 2023.

[143] Cheng Li, Ziang Leng, Chenxi Yan, Junyi Shen, Hao Wang, Weishi Mi, Yaying Fei, Xiaoyang Feng, Song Yan, HaoSheng Wang, et al. Chatharuhi: Reviving anime character in reality via large language model. *arXiv preprint arXiv:2308.09597*, 2023.

[144] Dake Chen, Hanbin Wang, Yunhao Huo, Yuzhao Li, and Haoyang Zhang. Gamegpt: Multi-agent collaborative framework for game development. *arXiv preprint arXiv:2310.08067*, 2023.

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

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

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

[148] Zhao Kaiya, Michelangelo Naim, Jovana Kondic, Manuel Cortes, Jiaxin Ge, Shuying Luo, Guangyu Robert Yang, and Andrew Ahn. Lyfe agents: Generative agents for low-cost real-time social interactions. *arXiv preprint arXiv:2310.02172*, 2023.

[149] Junjie Zhang, Yupeng Hou, Ruobing Xie, Wenqi Sun, Julian McAuley, Wayne Xin Zhao, Leyu Lin, and Ji-Rong Wen. Agentcf: Collaborative learning with autonomous language agents for recommender systems. *arXiv preprint arXiv:2310.09233*, 2023.

[150] Wenyue Hua, Lizhou Fan, Lingyao Li, Kai Mei, Jianchao Ji, Yingqiang Ge, Libby Hemphill, and Yongfeng Zhang. War and peace (waragent): Large language model-based multi-agent simulation of world wars. *arXiv preprint arXiv:2311.17227*, 2023.

[151] Haochun Wang, Sendong Zhao, Zewen Qiang, Zijian Li, Nuwa Xi, Yanrui Du, MuZhen Cai, Haoqiang Guo, Yuhan Chen, Haoming Xu, et al. Knowledge-tuning large language models with structured medical knowledge bases for reliable response generation in chinese. *arXiv preprint arXiv:2309.04175*, 2023.

[152] Wenqi Shi, Ran Xu, Yuchen Zhuang, Yue Yu, Jieyu Zhang, Hang Wu, Yuanda Zhu, Joyce Ho, Carl Yang, and May D Wang. Ehragent: Code empowers large language models for complex tabular reasoning on electronic health records. *arXiv preprint arXiv:2401.07128*, 2024.

[153] Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Shaokun Zhang, Erkang Zhu, Beibin Li, Li Jiang, Xiaoyun Zhang, and Chi Wang. Autogen: Enabling next-gen llm applications via multi-agent conversation framework. *arXiv preprint arXiv:2308.08155*, 2023.

[154] Yang Li, Yangyang Yu, Haohang Li, Zhi Chen, and Khaldoun Khashanah. Tradinggpt: Multi-agent system with layered memory and distinct characters for enhanced financial trading performance. *arXiv preprint arXiv:2309.03736*, 2023.

[155] Saizhuo Wang, Hang Yuan, Lionel M Ni, and Jian Guo. Quantagent: Seeking holy grail in trading by self-improving large language model. *arXiv preprint arXiv:2402.03755*, 2024.

[156] Yangyang Yu, Haohang Li, Zhi Chen, Yuechen Jiang, Yang Li, Denghui Zhang, Rong Liu, Jordan W Suchow, and Khaldoun Khashanah. Finmem: A performance-enhanced llm trading agent with layered memory and character design. *arXiv e-prints*, pages arXiv–2311, 2023.

[157] Kelvin JL Koa, Yunshan Ma, Ritchie Ng, and Tat-Seng Chua. Learning to generate explainable stock predictions using self-reflective large language models. *arXiv preprint arXiv:2402.03659*, 2024.

[158] Kexin Chen, Junyou Li, Kunyi Wang, Yuyang Du, Jiahui Yu, Jiamin Lu, Lanqing Li, Jiezhong Qiu, Jianzhang Pan, Yi Huang, Qun Fang, Pheng Ann Heng, and Guangyong Chen. Chemist-x: Large language model-empowered agent for reaction condition recommendation in chemical synthesis, 2024.

[159] Zihao Wang, Shaofei Cai, Anji Liu, Yonggang Jin, Jinbing Hou, Bowei Zhang, Haowei Lin, Zhaofeng He, Zilong Zheng, Yaodong Yang, et al. Jarvis-1: Open-world multi-task agents with memory-augmented multimodal language models. *arXiv preprint arXiv:2311.05997*, 2023.

[160] Zihan Zhao, Da Ma, Lu Chen, Liangtai Sun, Zihao Li, Hongshen Xu, Zichen Zhu, Su Zhu, Shuai Fan, Guodong Shen, et al. Chemdfm: Dialogue foundation model for chemistry. *arXiv preprint arXiv:2401.14818*, 2024.

[161] Ming Yan, Ruihao Li, Hao Zhang, Hao Wang, Zhilan Yang, and Ji Yan. Larp: Language-agent role play for open-world games. *arXiv preprint arXiv:2312.17653*, 2023.

[162] Zi-Yi Chen, Fan-Kai Xie, Meng Wan, Yang Yuan, Miao Liu, Zong-Guo Wang, Sheng Meng, and Yan-Gang Wang. Matchat: A large language model and application service platform for materials science. *Chinese Physics B*, 32(11):118104, 2023.

[163] Nian Li, Chen Gao, Yong Li, and Qingmin Liao. Large language model-empowered agents for simulating macroeconomic activities. *arXiv preprint arXiv:2310.10436*, 2023.

[164] Haojie Pan, Zepeng Zhai, Hao Yuan, Yaojia Lv, Ruiji Fu, Ming Liu, Zhongyuan Wang, and Bing Qin. Kwaiagents: Generalized information-seeking agent system with large language models. *arXiv preprint arXiv:2312.04889*, 2023.

[165] Odma Byambasuren, Yunfei Yang, Zhifang Sui, Damai Dai, Baobao Chang, Sujian Li, and Hongying Zan. Preliminary study on the construction of chinese medical knowledge graph. *Journal of Chinese Information Processing*, 33(10):1–9, 2019.

[166] Zefan Wang, Zichuan Liu, Yingying Zhang, Aoxiao Zhong, Lunting Fan, Lingfei Wu, and Qingsong Wen. Rcagent: Cloud root cause analysis by autonomous agents with tool-augmented large language models. *arXiv preprint arXiv:2310.16340*, 2023.

[167] Zhangcheng Qiang, Weiqing Wang, and Kerry Taylor. Agent-om: Leveraging large language models for ontology matching. *arXiv preprint arXiv:2312.00326*, 2023.

[168] Licheng Wen, Daocheng Fu, Xin Li, Xinyu Cai, Tao Ma, Pinlong Cai, Min Dou, Botian Shi, Liang He, and Yu Qiao. Dilu: A knowledge-driven approach to autonomous driving with large language models. *arXiv preprint arXiv:2309.16292*, 2023.

[169] Zhitao Wang, Wei Wang, Zirao Li, Long Wang, Can Yi, Xinjie Xu, Luyang Cao, Hanjing Su, Shouzhi Chen, and Jun Zhou. Xuat-copilot: Multi-agent collaborative system for automated user acceptance testing with large language model. *arXiv preprint arXiv:2401.02705*, 2024.

[170] Yongchao Chen, Jacob Arkin, Yang Zhang, Nicholas Roy, and Chuchu Fan. Scalable multi-robot collaboration with large language models: Centralized or decentralized systems? *arXiv preprint arXiv:2309.15943*, 2023.

[171] Zhao Mandi, Shreeya Jain, and Shuran Song. Roco: Dialectic multi-robot collaboration with large language models. *arXiv preprint arXiv:2307.04738*, 2023.

[172] Jonathan Light, Min Cai, Sheng Shen, and Ziniu Hu. From text to tactic: Evaluating llms playing the game of avalon. *arXiv preprint arXiv:2310.05036*, 2023.

[173] Bing Liu. Lifelong machine learning: a paradigm for continuous learning. *Frontiers of Computer Science*, 11:359–361, 2017.

[174] Gati V Aher, Rosa I Arriaga, and Adam Tauman Kalai. Using large language models to simulate multiple humans and replicate human subject studies. In *International Conference on Machine Learning*, pages 337–371. PMLR, 2023.