AgentLens: Visual Analysis for Agent Behaviors in LLM-Based Autonomous Systems

Jiaying Lu ⓘ, Bo Pan ⓘ, Jieyi Chen ⓘ, Yingchaojie Feng ⓘ, Jingyuan Hu ⓘ, Yuchen Peng ⓘ,
and Wei Chen ⓘ, *Senior Member, IEEE*

Abstract—Recently, Large Language Model based Autonomous System (LLMAS) has gained great popularity for its potential to simulate complicated behaviors of human societies. One of its main challenges is to present and analyze the dynamic events evolution of LLMAS. In this work, we present a visualization approach to explore the detailed statuses and agents' behavior within LL-MAS. Our approach outlines a general pipeline that organizes raw execution events from LLMAS into a structured behavior model. We leverage a behavior summarization algorithm to create a hierarchical summary of these behaviors, arranged according to their sequence over time. Additionally, we design a cause trace method to mine the causal relationship between agent behaviors. We then develop *AgentLens*, a visual analysis system that leverages a hierarchical temporal visualization for illustrating the evolution of LLMAS, and supports users to interactively investigate details and causes of agents' behaviors. Two usage scenarios and a user study demonstrate the effectiveness and usability of our *AgentLens*.

Index Terms—Agent, autonomous system, LLM, visual analysis.

I. INTRODUCTION

AUTONOMOUS agents, as computational entities that possess a certain degree of autonomy [1], [2], are seen as a promising pathway toward achieving artificial general intelligence (AGI) [3], [4]. In recent years, owing to the breakthroughs in natural language processing [5], [6], [7] achieved by Large Language Models (LLM), the LLM-based autonomous agent has gained widespread adoption in both academia and industry [8], [9]. Built upon LLM-based agents, LLM-based autonomous systems (LLMAS) deploy multiple agents within a shared environment, enabling them to display behavior and social patterns akin to humans. This collective intelligence fosters emergent social dynamics, such as the formation of new relationships, diffusion of information, and the rise of coordination among agents [10]. Consequently, LLMAS exhibits significant potential in society simulation [10], [11], software engineering [12], [13], and scientific research [14].

However, monitoring and analyzing the dynamic evolution of LLMAS, including agents in LLMAS and event sequences undertaken by them, can be challenging due to the tremendous information generated during the system evolution and the inherent unpredictability of LLMs. The most straightforward approach for analyzing LLMAS is to inject logging code into LLMAS to trace agent events of interest and check the raw output logs in text format [15]. However, this approach requires expertise with specific LLMAS and is unintuitive for general users. To address this, many LLMAS projects provide a graphical representation of the simulation process [9], which is typically re-playable 2D [10], [16], [17], [18] or 3D video [19], [20], [21], [22]. By transforming a fixed sequence of intermediate simulation events into expressive visual recordings, users can digest that information more efficiently and intuitively. However, a re-playable recording with a fixed level of abstraction limits the flexibility of analysis for LLMAS. Even for a specific LLMAS and a fixed usage scenario, a user's short-term analysis target will change frequently during the analysis process. As the users' analysis target varies, the type, quantity, and granularity of agent events to be visualized also need to change. Moreover, analyzing the agent's behavior at a specific time point requires users to switch the recording back and forth to trace the cause and consequence of this behavior, which is tedious and unreliable.

This work thus presents a visualization approach to assist users in efficiently analyzing the evolving status and complex behaviors of agents within an LLMAS. To mitigate cognitive overload due to the profusion of data produced throughout the evolution of LLMAS, and to enhance adaptability for subsequent analytical processes, we introduce a general pipeline, which establishes a hierarchical behavior structure of agent entities and raw event sequences within the LLMAS operational records. The formulation of the structure is based on our survey of prevalent architectures within extant LLMAS, coupled with a design study that engaged 4 LLMAS developers and 4 layman users. We design an LLM-based algorithm for summarizing agent behavior that furnishes a hierarchical depiction of sequences of agent events. Additionally, we employ a cause trace method to unearth the causal linkages among disparate agent events. Based on the
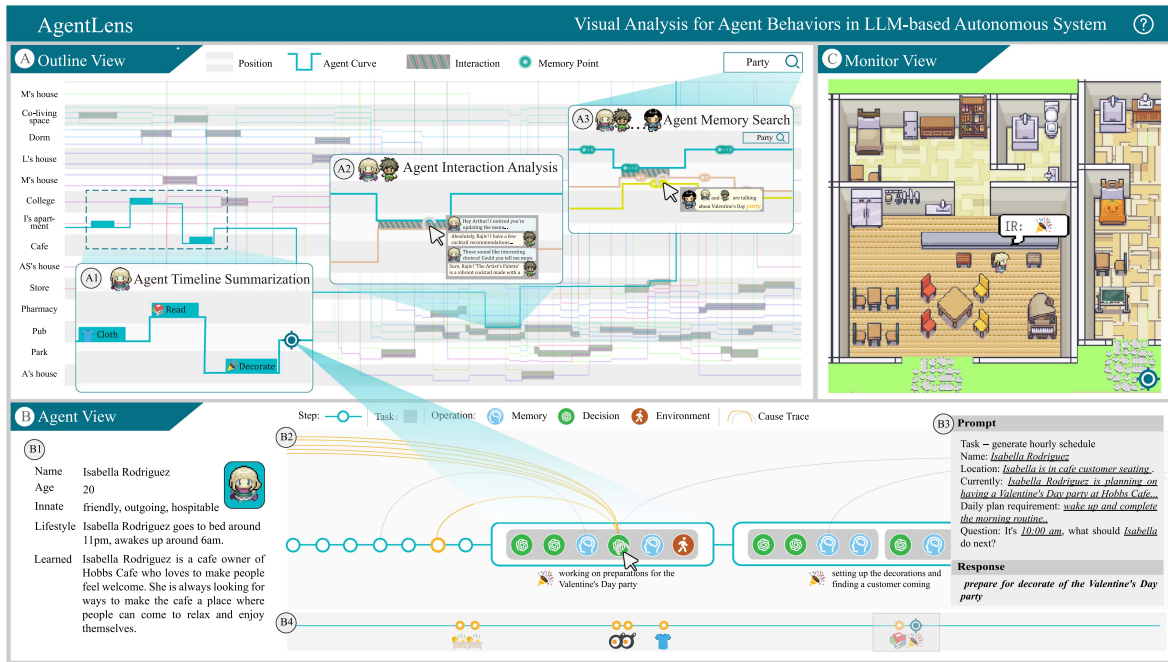
Fig. 1. The user interface of *AgentLens* comprises three views. The *Outline View* (A) displays the trajectory of each agent using different colored curves, enabling users to identify significant patterns or event summarization during the evolution of LLMAS. By clicking on a time step in each curve, users can further investigate it in the *Agent View* (B). It allows users to progressively reveal agent event information and trace the cause of specific agent behavior. The *Monitor View* (C) automatically adjusts the graphical representation of LLMAS based on the user's current point of interest.

extracted hierarchical structure, we then develop *AgentLens*, a visual analysis system designed to facilitate interactive analysis and exploration of agent behaviors in LLMAS.

AgentLens provides a multi-faceted perspective for LLMAS through its three distinct but interrelated views, each offering a different level of abstraction. The *Outline View* (Fig. 1, Ⓐ) illustrates the spatiotemporal trajectory of each agent with curves of different colors, aiding users in identifying notable agents or their intriguing behaviors throughout the evolution of LLMAS. Users can quickly scan agent behaviors at different granularity (Fig. 1, Ⓐ₁), identify agent interaction of interest (Fig. 1, Ⓐ₂), perform topic search (Fig. 1, Ⓐ₃), and click any time point on an agent curve to further investigate it in the *Agent View* (Fig. 1, Ⓑ). The *Agent View* allows users to progressively reveal agent event information on demand and trace the cause of certain agent behavior. The *Monitor View* (Fig. 1, Ⓒ) automatically adjusts the graphical representation of LLMAS for users based on their current point of interest in the *Outline View* or the *Agent View*. To evaluate the performance of *AgentLens*, we present two cases and conduct a user study with 14 participants to gather their feedback. The results indicate that *AgentLens* is capable of assisting users in the LLMAS evolution analysis and agent behaviors investigation.

The main contributions of our work are as follows:

- To the best of our knowledge, our work is the first visual analysis system that enables analysis and explorations of agent behaviors within LLMAS.
- We propose a general pipeline that establishes a hierarchical behavior structure from raw LLMAS execution events to facilitate downstream analysis.

- We conduct two cases and a user study to demonstrate the capabilities of our system. The evaluation results confirm the usefulness and effectiveness of the behavior structure and *AgentLens*.

II. RELATED WORK

A. LLM-Based Autonomous Agents

Franklin et al. [23] defined the agent as an entity situated in the environment that senses the environment and acts on it over time, in pursuit of its own agenda and so as to affect what it senses in the future. Possessing the ability to perform intelligent operations without human intervention, the autonomous agent remains a steadfast goal in artificial intelligence research [3], [24].

The progression of LLMs [6], [25] has underscored exceptional proficiency in areas of comprehension, reasoning, and language generation [26], which kindled optimism for continued advancements in the realm of autonomous agents. With the advent of LLMs, the study of LLM-based autonomous agents began to thrive. This includes enhancing agents' self-reflective capabilities [27], [28], implementing superior task decomposition strategies [29], and endowing the ability to utilize and create tools [30], [31], [32], [33]. There is also a vibrant development of applications of LLM-based agents in the open source community [15], [34], [35].

Recently, researchers have found that LLM-based agents can address a wider range of tasks through collaboration or competition. Camel [36] presented a framework that emphasizes the autonomous interaction between communicative agents. It is capable of creating varied, detailed instructions across

numerous tasks, thereby providing a platform for these agents to demonstrate their cognitive operations. Talebirad et al. [37] introduced a comprehensive framework for multi-agent collaboration based on LLMs. ProAgent [18] exhibited the distinctive ability for agents to foresee the upcoming decisions of collaborators and adjust their behaviors, enabling them to excel in cooperative reasoning tasks. Multi-Agent Debate (MAD) [38] introduced an approach in which several agents present their arguments collaboratively while a judge guides the discourse, enhancing agents' divergent thinking for deep-reflective tasks.

However, as the number and the intricacy of agents increase, the complexity of analyzing their behaviors escalates rapidly. While past works have focused on elevating the capabilities of LLM-based agents in emulating human-like behaviors, they often overlooked how to effectively analyze agent behaviors. In this work, we identify this research gap and present a visualization approach for analyzing agent behaviors in LLM-based multi-agent systems.

B. LLM-Based Autonomous System

By incorporating numerous LLM-based agents into a cohesive environment, the LLMAS is capable of handling diverse complex scenarios. For example, WebAgent [39] demonstrated the possibility of building agents that can complete the tasks on real websites following natural language instructions. ChatDev [12] and MetaGPT [13] experimented with software development in multi-agent communication settings. Zhang et al. [19] built embodied agents to cooperate effectively with humans. Park et al. [10] situates generative agents with unique characteristics in a societal context, in order to mimic human social behaviors.

Several task-independent frameworks designed for diverse usages have received considerable attention within the community. AgentVerse [17] dynamically assembled multi-agent teams tailored to task complexities, outperforming individual agents with adaptable team structures. AgentSims [16] offered a real-time evaluation platform for LLM-based agents, enabling adaptable configurations to facilitate the performance evaluation of different modules. AutoGen [40] fostered conversations among multiple agents and organized individual insights in a general manner, offering an interconnected manner to coordinate multiple agents within the LLMAS. MetaGPT [13] injects effective human workflows into multi-agent collaboration by encoding Standardized Operational Procedures (SOP) into prompts, underscoring the potential of incorporating human domain expertise into LLMAS. CGMI [11] replicated human interactions and imitated human routines in real-world scenarios, which enhances the realism of more humanized simulation of complex social scenarios.

Previous LLMAS research has primarily focused on constructing more universal frameworks or designing for specific domains, yet there has been a noticeable lack of emphasis on the analysis methods of parallel behaviors among agents within LLMAS. Contemporary LLMAS predominantly depend on conventional methods for surveillance and analysis. MetaGPT [13] utilizes log outputs for record maintenance, while Park et al. [10]

adopts panoramic videos for observation, providing detailed maps with agent avatars to denote their locations and behaviors. Distinct from preceding efforts, our work offers an interactive visual system that hierarchically organizes events, facilitating users in quickly grasping the happenings within LLMAS.

C. Event Sequence Visualization

Data featuring time-based event sequences is widespread and can be found in various sectors, including healthcare records [41], [42], [43], career design [44], [45] and social interactions [46], [47], [48]. In these fields, distinct types of time-stamped events are sequentially organized, each relevant to a particular subject or entity. While earlier methods [49], [50] have been geared toward simpler, low-dimensional data, the data sets encountered in real-world scenarios frequently display a higher level of complexity, calling for more comprehensive analytical ideas and methods.

A substantial number of research on event sequence visualization is notably correlated with fields where there is a prevalent demand for event information condensation, such as in the realm of social media data [51], the sphere of smart manufacturing [52], and the study of anomalous user behaviors [53]. Guo et al. [54] proposed an organizational framework for event sequences to summarize the common goal of different properties with great heterogeneity. EventThread [44] focuses on visualization and cluster analysis, providing an interactive interface for browsing and summarizing event sequence data. Building on past frameworks of condensing events and visualizing them, we focused on the behavioral patterns of LLM agents and proposed an LLM-driven approach to handle non-structured natural language-based event sequences.

Event sequence visualization has highly relevant applications in the realm of collective behavior analysis, which aligns closely with the focus of our research, both referring to activities conducted by a temporary and unstructured group of people [47], [55], [56]. In the field of social media, collective actions emerge from the collaborative efforts of users engaged in disseminating information and navigating through virtual spaces. A variety of sophisticated visual analytics methodologies have been introduced to scrutinize these group dynamics. R-map [57], Socialwave [58], FluxFlow [59] and Google+ ripples [60] are specifically tailored to examine the mechanics of information propagation, while Maqui [61] and Frequence [46] offers insights into the complexities of human mobility within this context.

While existing research has made significant contributions to the field, there's a growing need to address the increasingly complex behaviors and interactions that call for the advancement of autonomous systems. Our work introduces event sequence visualization as an integral tool for the analysis and exploration of LLMAS.

III. OVERVIEW

A. Common Architecture of LLMAS

To ensure maximum compatibility with various LLMAS, we survey LLMAS-related papers [27], [28], [62], [63], [64], [65],
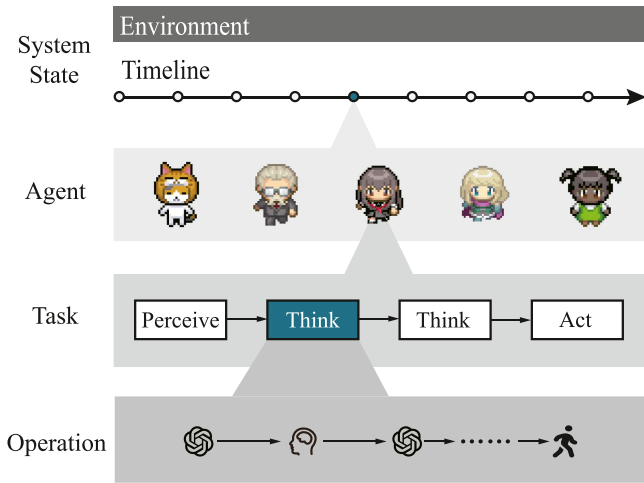
Fig. 2. The common architecture abstracted from existing LLMAS consists of four layers: system states, agents, tasks, and operations.

[66], [67], [68], [69] as well as some projects [70], [71], [72], [73], [74], [75], [76], [77], [78], [79] with high stars in open source communities published before August 31, 2023. We analyzed their system architectures and components, based on which we abstract a common architecture (as shown in Fig. 2) for LLMAS. The **system state** in LLMAS provides the environmental information at any time point. At each time point, each **agent** executes its own **task**, which consists of several atomic **operations**. A raw event is generated whenever an operation is executed by an agent, thereby advancing the evolution of LLMAS.

System State provides a comprehensive understanding of the environment. By acquiring the environmental information from the system state, agents can comprehend the current context and conditions. For example, the system state can inform agents about object locations and environmental properties, which significantly impact their decision-making and planning processes. In addition, the system state governs the timelines of each agent, ensuring events by different agents are temporally aligned.

Agents are autonomous entities with cognitive abilities and action capability. By performing various types of tasks, agents can interact with the environment and gradually change the system state to achieve their goals. Additionally, agents can communicate and collaborate with each other. They can share their knowledge and exchange messages to accomplish more complex duties.

Tasks are typically customized for the usage scenarios. A sequence of operations with a common goal can be grouped as a task. Extending prior research that has focused on different scenarios for agents, we classify tasks into three categories: Perceive, Think, and Act. In *Perceive* tasks, the agent obtains perception of the external system. Such perception includes sensing the environment (virtual, real, or external resources), as well as perceiving other agents. In *Think* tasks, the agent engages in decision-making, reasoning, planning, and other behaviors based on external perception and its own memory. In *Act* tasks, the agent interacts with the external system by providing outputs,

including text outputs, virtual actions, or specific invocations such as tool usage.

Operations are the basic units for Tasks. Operations can be classified based on their target, including Environmental Operations, Memory Operations, and Decision Operations. *Environment Operation*s execute interactions toward the external system, including other agents and the environment defined by LLMAS. *Memory Operation*s involve storing and updating the memory of an agent. *Decision Operation*s are for decision-making and action planning, where LLM-based agents typically utilize LLMs for decision operations.

B. Design Requirement

Our study focuses on users involved in analyzing, exploring, and monitoring LLMAS. Our primary goal is to create a system that enhances users' comprehension of LLMAS. We recruited 4 developers highly familiar with LLMAS and 4 users who have a basic understanding of LLMAS and have previously utilized such systems.

To identify the design requirements, We asked participants to explore the behaviors of agents in Reverie,[1] a typical autonomous system consisting of 25 LLM-based agents. We requested participants to actively explore and delve into the identification of agent behaviors that intrigued them, as well as to investigate the underlying causes or consequences. To facilitate this, we encouraged participants to "think aloud", articulating the information they sought and the type of assistance they desired throughout the process. We then conducted the first interview with them to collect their feedback on the whole exploration process. At the same time, we maintain regular contact with them to keep them updated on the design requirements. Based on their feedback, and combined with the survey on existing LLMAS work in Section III-A, the following 4 design requirements can be summarized.

R1. Provide suitable generality of information for different analysis targets: During the evolution of LLMAS, a significant volume of information is continuously generated, which is overwhelming for users to comprehend. While the current 2D graphical interface of Reverie provides a fixed visual abstraction, many users express their desire to change the generality of presented information to better match their current analysis target. For instance, users want to scan summarized agent traces across a large time scale when they analyze the long-term relationship among several agents, while they prefer a detailed presentation of an agent's operations when they analyze how the agent performs a certain task. Therefore, the system should provide users with flexible levels of abstraction for the generated information of LLMAS, and allows users to reveal details according to their analysis target.

R2. Present agents' transition of physical location and thought content: The physical and mental changes of agents play a vital role in driving and reflecting the evolution of the entire LLMAS. Nevertheless, currently, users can only stare at the re-playable recording to see if there is a location transition

---

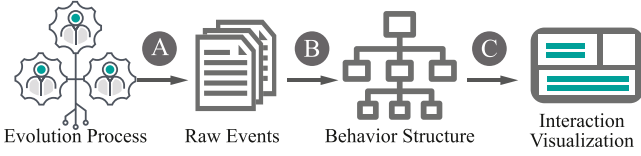[1]https://reverie.herokuapp.com/arXiv_Demo/#

Fig. 3.   The workflow of our approach consists of three major steps. (A) Collect raw execution log of events from the LLMAS evolution process. (B) Establish a behavior structure with hierarchical summarization and a cause trace method. (C) Provide an interactive user interface for visual exploration and analysis.
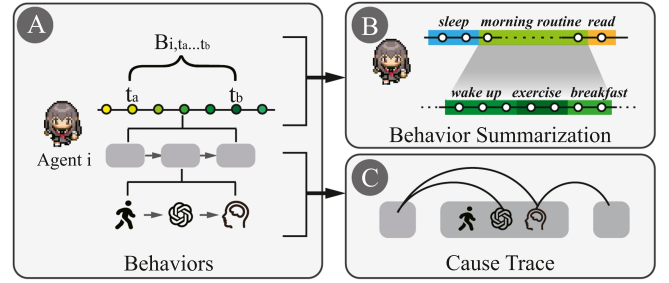

Fig. 4.   The behavior structure is established through a three-step pipeline: (A) We organize raw events into behaviors, (B) summarize and segment behaviors for an agent, and (C) trace causal relationships among behaviors.

of the agent and check the raw execution log to find when the agent starts to think about a certain idea, which is inefficient and error-prone. Therefore, the system should provide visual emphasis on agents' transition of location and highlight the time points the agent starts to think about a topic the user wishes to explore.

R3. Underscore possible causes of agent behaviors: When users become interested in a certain behavior of the agent, they usually want to investigate the cause or consequence of this behavior. However, an agent's behavior can be influenced not only by its current perception and thoughts but also by the memory of its past behavior. It is tedious and unreliable for users to switch the replayable recording back and forth to locate the cause of the behaviors of certain agents. Therefore, the system should provide a mechanism to mine the possible causes of an agent's behaviors and highlight them for users' investigation.

R4. Explicate the context of LLM invocation: LLM plays a crucial role as the core of the LLMAS, which is frequently invoked to make cognitive decisions for agents. To inform the background for making a certain decision, the preceding contextual information is organized in a specific manner with a customized template and then sent as a prompt to the LLM. Therefore, to help users understand how and why a decision is made by an agent, the system should present the decisions made by LLM and explicate the context of its invocation. Moreover, it is desirable to provide visual enhancement to help users trace how the context information is collected from previous agent behaviors.

C. Approach Overview

In alignment with the aforementioned design requirements, we designed *AgentLens*, a proof-of-concept system dedicated to visualizing agent behaviors during the LLMAS evolution. The workflow of our approach is depicted in Fig. 3. Users can utilize logging codes to log their LLMAS evolution process and capture raw events executed by agents. Based on these raw events, we establish a hierarchical structure to summarize agent behaviors in different granularity and trace possible causal relationships among their behaviors (Section IV). A user interface and a series of interactions are provided to support interactive exploration and analysis of the agent behaviors in LLMAS (Section V).

IV. BEHAVIOR STRUCTURE ESTABLISHMENT

In this section, we introduce a pipeline designed to establish the hierarchical behavior structure from raw events generated

during the evolution of LLMAS. It facilitates the generation of structured data for visualization, achieved via summarization and causal analysis of agent behaviors. As shown in Fig. 4, the pipeline consists of three steps: (A) processing the raw events and organizing them into behaviors based on the common architecture shown in Fig. 2 (**R1**), (B) summarizing these behaviors and segmenting them in accordance with their semantic implications. (**R1**, **R2**), and (C) tracing the cause between these behaviors by analyzing the correlations among original events (**R3**, **R4**).

A. Behavior Definition

During the evolution of LLMAS, multiple raw events are generated, creating large, often chaotic, and obscure text logs with the scaling of agent populations. To streamline downstream analysis and visualization efforts, we defined agent behaviors as structured representations that encapsulate the sequence of raw events (**R1**).

Drawing upon the system state adopted by most LLMAS architectures, we denote the timeline T to represent the states and events of agents at various time points within environments. For each time point t on the timeline, we can define the tuple T_t as follows:

$$T_t = \langle e_{t-1}, \bigcup a_{t-1}[i], \bigcup s_t[i] \rangle \tag{1}$$

where e_{t-1} denotes the environment state at the previous point $t-1$ before time t. $a_{t-1}[i]$ represents the agent state of the i-th agent at $t-1$, encompassing its position within the environment as well as individual status indicators such as hunger levels, mood values, etc. In various LLMAS, a_t encompasses a diverse array of attributes. $s_t[i]$ denotes the set of indivisible $\bigcup o_{t,i}[k]$ (operation informed in Section III-A) executed by the i-th agent at t, and k denotes the operation index. Following these definitions, the indivisible minimal events occurring within an LLMAS are transformed into operations o, which are bound to a specific time point, agent, and task(*e.g.* perceive, think, act). However, these low-level events can be irrelevant or redundant for high-level analysis targets. For a specific agent, there may exist hundreds of events at a single time point t, which imply factual (*e.g.* duplicate segments generated by prompt construction) and semantic (*e.g.* repeated biased interpretations of the same observation) duplications.
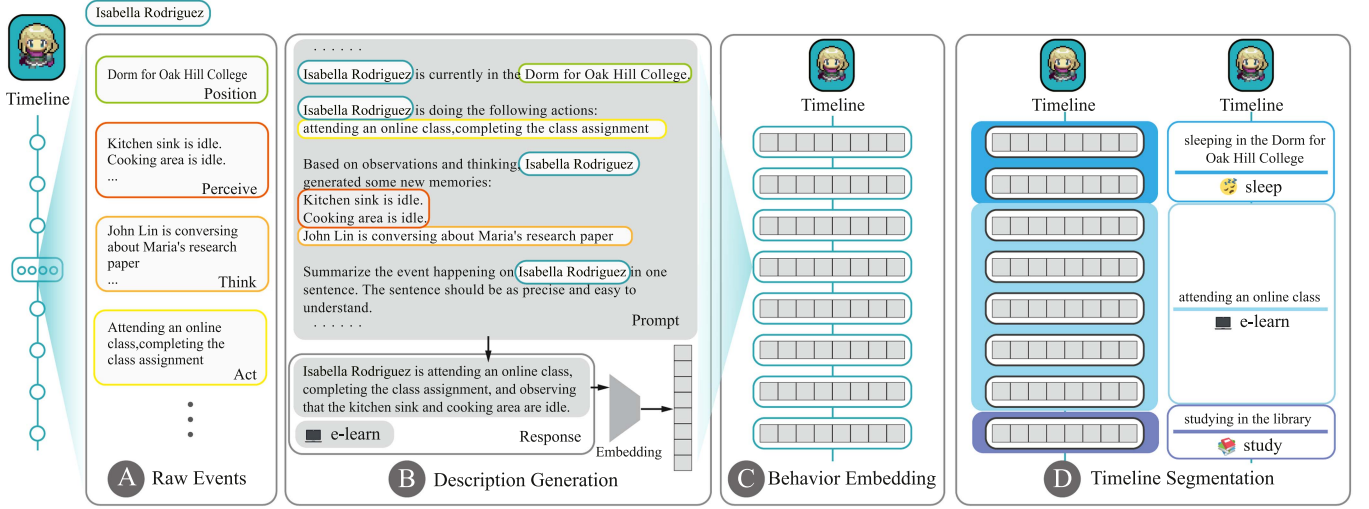
Fig. 5. The agent behavior is summarized in four stages: (A) Raw Events: acquire raw events from the logs to detail the occurrences involving the agent along the timeline, including the agent's location, actions, memory, and conversations. (B) Description Generation: organize the raw events and employ models such as LLMs to generate concise descriptions of the behaviors. (C) Behavior Embedding: translate the behavior descriptions into a sequence of textual embedding vectors. (D) Timeline Segmentation: involve the detection of change points within the sequence of behavior vectors, followed by the corresponding segmentation of the agent's timeline.

To address these problems, we synthesize events on T for each agent into their behaviors:

$$B_{i,t_0\cdots t_1} = \bigcup_{t\in[t_0,t_1]} s_{t,i} \qquad (2)$$

It refers to the set of operations performed by the i-th agent across the subsequence $[t_0, t_1]$ within the temporal series T.

B. Behavior Summarization

In various LLMAS, operations manifest in different forms, such as text, images, and even physical behaviors in the factory environment. Meanwhile, new behaviors of those agents are continuously generated as T is increasing. The multiplicity of manifestation and the extensive aggregation of behaviors can obscure the visualization system's interpretation, thereby impeding the exploration of an agent's internal causality. Therefore, we propose a behavior summarization method. As shown in Fig. 5, we (1) outline behaviors that encapsulate a singular time point into a succinct description (Fig. 5, A → B), (2) utilize text embedding to capture underlying semantics within the behavior(Fig. 5, B → C), (3) utilize a change point detection method to divide the sequence of behaviors and abstract each sub-sequence of behavior(Fig. 5, C → D). Ultimately, we can summarize a multitude of small behaviors into several noteworthy behaviors with segmented timelines.

Description Generation: We incorporate an external text summarization model, which acts as a standalone LLM agent that operates independently of LLMAS. All annotated descriptions are concatenated to form a comprehensive model input (*i.e.* prompts for LLM). Given this long text sequence as input, the summarization model generates a succinct behavior description, significantly reducing the information length while maintaining the original meaning (shown in Fig. 5, B, from *Prompt* to

Response). Concurrently, we prompt that the summarization model yields a highly abstract description of the behavior, employing both textual and emoji symbols. Textual descriptions serve as the foundation for the forthcoming embedding model and emoji symbols are conceived to facilitate subsequent visualization.

Behavior Embedding: We further utilize all summarized behavior descriptions, embedding them to better grasp the latent semantics, including the inherent similarities and hierarchical relationships. To maximize the efficiency of the encoding schema, we adopt the text-embedding model[2] pretrained on large-scale internet text data, renowned for its superior performance, cost-effectiveness, and simplicity of use. The summarized behavior descriptions are then each encoded into a 1536-dimensional vector, constituting the sequence E_{agent} for each agent. With these powerful embeddings, we can uncover the semantic similarity of a single behavior, thereby unlocking the potential to tackle a myriad of complex text sequence analyses.

Timeline Segmentation: Considering the data characteristics of the embedding sequence e and our design requirements, we employ the Window-based change point detection (WIN) algorithm [80] with the cosine distance measure to segment the sequence. This approach is suitable for real-time or streaming data contexts, as it allows for incremental updates in response to the arrival of new data and exhibits insensitivity to short-term and frequent fluctuations.

Firstly, to compare two embedding vectors e_x and e_y ($e_x, e_y \in E_{\text{agent}}$) with dimension $d = 1536$, we use the cosine similarity $k_{cosine} : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ (shown in (3)) as the kernel function [81] , where $\langle \cdot, \cdot \rangle$ and $\| \cdot \|$ are the Euclidean scalar product

---

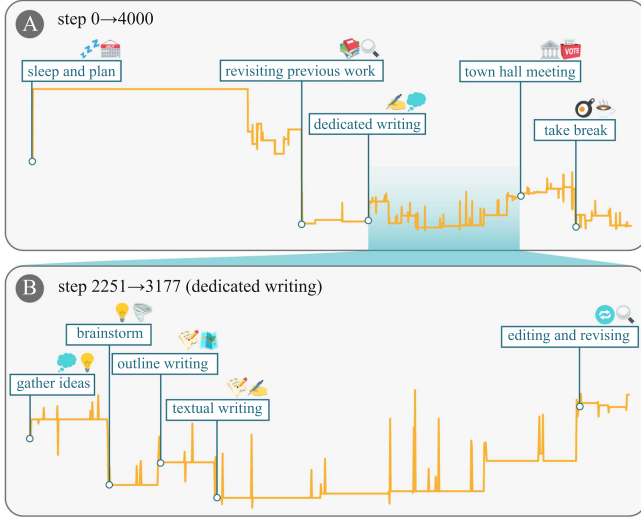[2]https://platform.openai.com/docs/guides/embeddings

Fig. 6. The timeline segmentation results of an agent in Reverie. The x-axis denotes the timeline, while the y-axis corresponds to the value of the principle PCA component of agent behavior embedding at each time point.

and norm respectively:

$$k(e_x, e_y) := \frac{\langle e_x | e_y \rangle}{\|e_x\| \|e_y\|} \quad (3)$$

Then we recall the cost $c(\cdot)$ deriving from $k(\cdot, \cdot)$ as (4), where $e_{a..b}$ is the subsequence $\{e_{a+1}, e_{a+2}, \ldots, e_b\} \subseteq E$:

$$c(e_{a..b}) = \sum_{t=a+1}^{b} k(e_t, e_t) - \frac{1}{b-a} \sum_{s,t=a+1}^{b} k(e_s, e_t) \quad (4)$$

WIN utilizes two sliding windows that traverse the data stream. By comparing the statistical properties of the signals within each window, a discrepancy measure is obtained based on the cost function c:

$$d(e_{u..v}, e_{v..w}) = c(e_{u..w}) - c(e_{u..v}) - c(e_{v..w}) \quad (5)$$

The discrepancy d means the cost gain of splitting the subsequence $e_{u..w}$ at the index v. If the boundary v is a change index within the window $u..w$, the discrepancy d will be significantly higher. After a sequential peak search of d, we have a series of time points $t_1^* < t_2^* < \ldots < t_K^*$. Certain features of the embedding sequence change suddenly at these points. We utilize the abstraction of t_i, encompassing both textual and emoji symbol descriptions, to aggregate the behaviors of the agent from t_i to t_{i+1}.

Fig. 6 provides an illustrative example of the timeline segmentation process. Here we try to segment the timeline of a writer agent in the Reverie environment. The agent's entire morning schedule is shown in (Fig. 6, A), spanning from midnight to noon, encompassing 4000 time points (0→4000) on the timeline. To facilitate an intuitive understanding of the segmentation result, we conducted principal component analysis (PCA) on the embedding of behavior at each time point, and used the y-axis to encode the values of the primary PCA components, resulting in the orange δ line plot presented in Fig. 6.

As we can see in (Fig. 6, A), by applying the segmentation algorithm (with N=5 as an example), this period is summarized into five main behaviors ("sleep and plan", "revisiting previous work", etc.). Moreover, if we re-apply the timeline segmentation algorithm to the "dedicated writing" behavior, which spans time points 2251→3177 (Fig. 6, B) on the timeline, we can further divide it into five sub-behaviors ("gather ideas", "brainstorm", etc.). Note that all these sub-behaviors can be considered as "dedicated writing", while exhibiting more subtle distinctions among them.

Another observation to note is that in the line plot formed by PCA principal component values, there are some peaks. These peaks occur because the agent executes specific operations at this time point, such as generating new memories or perceiving new objects. However, these operations do not have a lasting impact on the agent's ongoing behavior. Therefore, they are usually regarded as tiny behaviors contained in their parent behavior.

C. Cause Tracing

Within a complex timeline, any agent event is influenced by both its internal memory and interactions with the external environment. By tracing the causal factors of these events, users can gain valuable insights into agent behaviors (**R3**) and LLM invocation for decision-making (**R4**), thereby improving the credibility and interpretability of LLMAS.

Existing works [10] primarily rely on log debugging to explicitly reveal the origins of agents' operations. However, these methods place an additional cognitive burden on users due to the need for manual tracing and often fail to capture implicit causal relationships. For instance, current thinking can be influenced by observations over a long time steps. To efficiently trace the behavior causes, we propose a two-fold provenance tracing method to mine the causal relationships between underlying events within the behaviors.

Explicit Causes: It refers to the distinct and observable causal relationships that can be directly discerned from raw event logs, explicitly delineating the direct influence relationships between operations. For example, in open-source agent creation frameworks like Langchain [34] and AgentVerse [17], mechanisms have been implemented to index attributes of agent memory, facilitating direct backtracking to the relevant source operations upon the invocation of an agent's memory. When such explicit causal chains are completed in LLMAS, users can thus obtain these records through raw event logs and transmit them to *AgentLens*. *AgentLens* utilizes these logs as input to facilitate the analysis of downstream tasks for users.

Implicit Causes: Throughout the evolution of LLMAS, the agents' invocations of historical operations are not always documented, but rather are expressed through complex intermediate variables or latent patterns within the program. To capture these implicit causal relationships, we conduct relevance detection based on the text similarities (as in (3)) between the textual log of these operations themselves, thereby revealing the latent connections between events. To strike a balance between uncovering potential causal relationships and preventing information overload for users, we define a similarity threshold δ. For a

certain operator o_{res} at time point j, if the similarity between it and another operator o_{src} at time point i (*s.t.* $i \leq j$) exceeds δ, we consider o_{src} as one of the potential causes of o_{res}.

After the extraction of both explicit and implicit causes among operations is completed, we have ascertained every possible pair $< o_{src}, o_{res} >$. The connections between operations can be elevated to the connection between the corresponding behaviors in a bottom-up fashion, in accordance with the definition of behavior outlined in Section IV-A.

V. USER INTERFACE

The user interface is composed of three views. The *Outline View* (Fig. 1, B) visualizes how the agents' activity, interaction, and environment change over time, allowing users to analyze the evolution process of the LLMAS. Once the user becomes interested in certain behaviors of any agent, they can check its details and trace its cause from the *Agent View* (Fig. 1, C). During the exploration process, the visualization of LLMAS will synchronously switch to the corresponding agent and time point to support intuitive perception and verification in *Monitor View* (Fig. 1, D).

A. Outline View

Outline View serves as a springboard for exploration, providing a suitable generality of information (**R1**) to assist users in efficiently discovering noteworthy patterns or behaviors of interest during the evolution of the LLMAS.

Agent Timeline Summarization: Every agent has its individual behaviors (*e.g.* what it is perceiving, thinking, and acting) at each time point. When users double-click on the view, all selected agent curves will be automatically summarized into N (we set N = 10 during experiments) segments using the behavior summarization algorithm proposed in Section IV-B. Users can click the start of a segment to check details about what is happening during this period of timeline. If users desire a more granular behavior representation (**R1**), they can zoom in to a specific region by scrolling the mouse wheel. The system will then re-summarize the timeline based on the currently visible area (Fig. 1, A_1).

Agent Interaction Analysis: Each agent in the *Outline View* is represented as a uniquely colored curve, whose x-axis encodes the system time point and y-axis encodes the location of the agent, depicting the transition of the location of each agent (**R2**). When several agents are in the same time and location, they can have interactions (*e.g.* conversations, collaborations, or conflicts) with each other. Since these interactions usually play a crucial role in affecting the LLMAS's evolution, we highlight them by filling the area among the corresponding segment of agent curves. Users can click an interaction area of interest to check the integration details (Fig. 1, A_2). Drawing inspiration from previous work of storytelling [82], [83], we enforce agent curves to get closer if there is an interaction among them.

*Agent Memory Search:* Sometimes users want to conduct exploration about when and how the agents start to have thoughts about a specific topic (**R2**). Therefore, we provide a search box in the top right corner of the view, allowing users to add keywords related to the topic they want to explore. Whenever a keyword is added, the points on the agent curves corresponding to time points associated with relevant memory will be highlighted (Fig. 1, A_3).

B. Agent View

When users notice a specific phenomenon or behavior from the *Outline View* and wish to further explore it, they can click on the corresponding time point on an agent curve to access more details (**R1**) in the *Agent View*.

Agent Characteristic: A complex LLMAS typically contains agents with different characteristics. For example, agents might be assigned different roles and goals, which are usually realized through prompt engineering or LLM fine-tuning. Since these details are important for users to understand and infer an agent's behavior, we display them on the left panel of the *Agent View* (Fig. 1, B_1).

Time Point Revealing: On the right panel of *Agent View*, we provide users with a timeline (Fig. 1, B_2) to help investigate the behavior of the selected agent during this period of time, which is a detailed counterpart of the agent curve in *Outline View*. Users can click a time point icon to reveal descriptions (summarized using the method shown in Fig. 5, A - B) and the task-level events performed by this agent at this time point (**R1**). They can click a task icon to further reveal the operators involved in performing this task (**R1**). As discussed in Section III-A, the operators can be classified into Environmental Operations, Memory Operations, and Decision Operations based on the target. Therefore, we use different icons to represent operators of different type: If the user clicks , a description panel will pop up to show the invocation context of LLM to make the decision (Fig. 1, B_2) (**R4**); If the user clicks , a description panel will pop up to show the texts stored into the memory at this operation; If the user clicks , a description panel will pop up to show what the agent is perceiving from or act on the environment.

Cause Tracing: In addition to obtaining detailed behavioral information about agents, users also need to locate and analyze the reasons behind these agent behaviors. Whenever the user clicks an operator icon in the *Agent View*, the system will utilize the cause trace method described in Section IV-C to find previous operators that potentially have an intrinsic relationship with the current operation and highlight their corresponding time point on the *Agent View* (**R3**). We use edges with orange color to connect the selected operator and their predecessors. Since the agent behaviors could be affected by previous operations a long time ago, we provide users with a mini-map to visualize the point of the current operation and its related predecessors across the whole timeline (Fig. 1, B_4) (**R1**). Based on this mini-map, users can switch back and forth between the cause and result across the timeline more easily.

C. Monitor View

LLMAS typically provides a graphical representation of the dynamic simulation. It could be re-playable for 2D video or 3D, contingent upon the LLMAS evolution logs provided by

the user for *AgentLens*. This visual representation transforms abstract simulation data into perceptually friendly visual elements, which helps users understand LLMAS and verify their analysis more intuitively. However, manually switching between different locations and time points can be tedious and interrupt the user's analysis flow. Therefore, we provide the *Monitor View* to support fluent adjustment of the panoramic visualization of LLMAS (Fig. 1, Ⓒ) based on users' current focus and demand for context.

Focus Switching: Whenever the user clicks a time point on agent curve from the *Outline View* or a time point from the *Agent View*, the *Monitor View* will automatically switch to the location of that agent at that time point, providing a corresponding concrete visualization to complement the other two views (**R1**).

Context Revealing: The *Monitor View* also supports spatial and temporal context revealing to help users better comprehend the current focus point. As for the spatial context, the user can scroll the mouse wheel to adjust the level of scope, ranging from a macroscopic view of the entire LLMAS to a microscopic focus on a single agent. As for the temporal context, whenever the user changes the focus point from time point A to time point B, they can right-click the mouse to replay a fast-forward recording of that period of time in the *Monitor View*.

VI. USAGE SCENARIOS

A. Scenario A: Information Diffusion

This case demonstrates how our system helps users understand the patterns of agent behaviors in LLMAS. In the initialization phase, the user adds the information "*Organize Valentine's Day party at Hobbs Coffee on the evening of February 14th*" to the characteristic (Fig. 7, A_1) of the agent Isabella Rodriguez (IR) and wishes to observe the evolution of the system on February 13th.

To focus on the theme of the party, the user searches for the occurrence of the keyword "*party*" (Fig. 7, Ⓑ) in the agent's memory and follows IR's timeline for observation. The user discovers that the message primarily spreads during IR's conversations with others. Furthermore, the user finds the "*party*" memory highlight surfacing in the conversation between Ayesha Khan (AK) and John Smith (JS). Upon examining their dialogue (Fig. 7, B_1), it is revealed that the message is from AK to JS while there is no prior knowledge of the "*party*" message in AK's settings (Fig. 7, A_2). In order to delve into the underlying cause, the user selects the time point when AK initiates the conversation with JS, employing the *Agent View* to obtain detailed insights (Fig. 7, Ⓒ). The user expands the time point (Fig. 7, C_1) and traces the cause of one of the decision operations (Fig. 7, C_2). It is highly probable that AK's decision to discuss "*party*" with JS has its historical roots in a conversation between IR and AK that took place some time ago. Finally, the user reverts to the *Outline View*, confirming that a conversation concerning the "*party*" has indeed occurred between IR and AK (Fig. 7, B_2), during which IR extends an invitation to AK to participate in the party preparation.
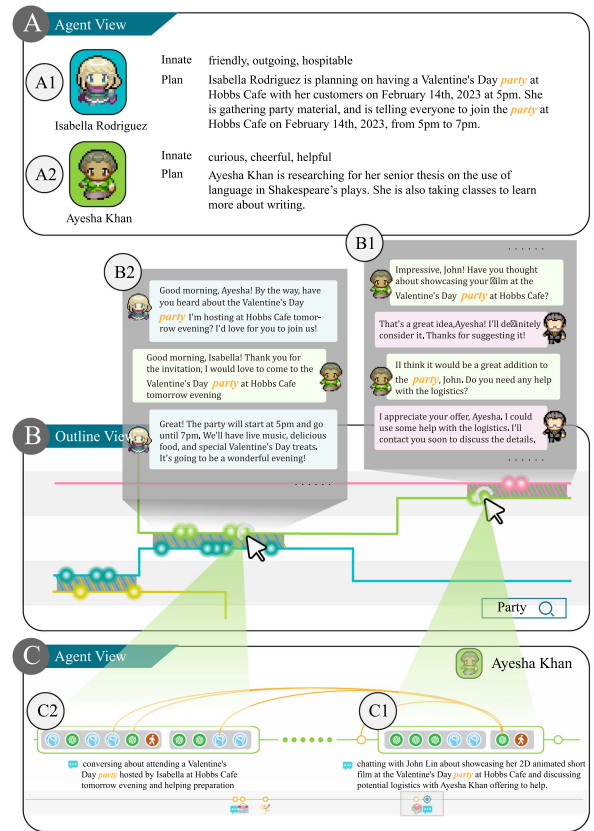

Fig. 7. The first usage scenario showcases the support *AgentLens* provides to the user in exploring social patterns like **Information Diffusion**. (A) The user gleans the characteristics of each agent via the *Agent View*. (B) Proceeding to the *Outline View*, the user searches for the keyword "*party*", discovering several related memory points generated in several conversations. (C) Utilizing the *Agent View*, the user delves into the origins of these conversational patterns.

With the assistance of *AgentLens*, the user successfully pinpoints an instance of information diffusion from a primary disseminator IR to a secondary one AK, then gradually diffusing towards other agents. From the *Agent View*, users discover that with the increase in both secondary propagators and the number of conversations related to "*party*", the speed of "*party*" diffusion throughout the small town significantly accelerates.

B. Scenario B: Unexpected Social Patterns

In this scenario, the user uncovers an unexpected pattern of information diffusion: **eavesdropping.**

During the observation of the "*party*" propagation process (Fig. 8, A_1), the user discovers that Sam Moore (SM) forms relevant memories without engaging in any direct conversation. According to the event summary, SM is in the process of writing his novel when this memory is formed. The user hovers on this memory point about the "*party*", learning that the memory formed by SM at this time is "*IR and Giorgio Moore (GM) are talking about Valentine's Day party*". From the visual representation, the user observes that IR, SM, and GM are in the same room at this moment, a fact that is corroborated by the *Monitor View* (Fig. 8, A_2). The user infers that SM comes to know about the "*party*" by eavesdropping on others' conversations.
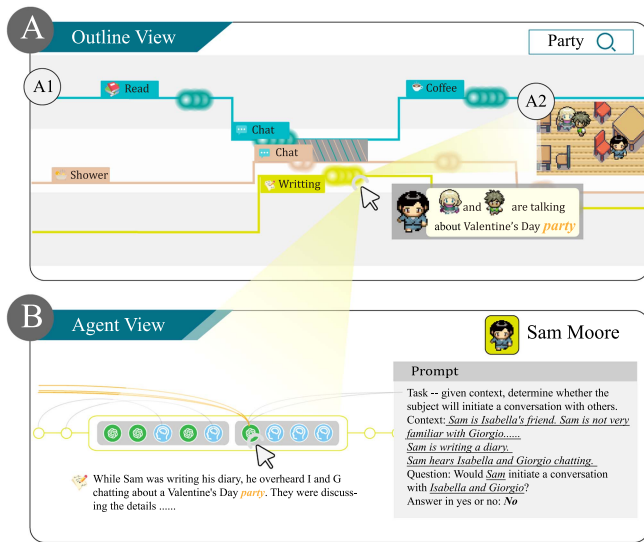
Fig. 8. The second usage scenario presents how *AgentLens* aids users in explaining an unexpected agent behavior. (A) The user identifies some unexpected agent behaviors in *Outline View*, like an agent participating in information dissemination without engaging in a related conversation. Upon validation through *Monitor View*, the user determines that this pattern corresponds to the eavesdropping behavior of the agent. (B) The user uses *Agent View* to investigate the reasons behind the agent's reluctance to participate in the discussion. Finally, the user discovers that a certain decision operation at the time point results in the behavior.

The user seeks to investigate why SM does not join the conversation. The user expands the corresponding time point(Fig. 8, (B)) in the *Agent View* and identifies the *Decision Operation* that determines SM's choice not to participate in the discussion. The prompt dispatches to the LLM incorporated agent settings pertaining to SM, like "*SM is IR's friend*" and "*Sam is not very familiar with GM*", in addition to the immediate observations made by SM, such as "*IR and GM are presently engaged in a conversation*" among other pieces of prompt input. It is the response returned by the LLM, based on the prompt, making the decision for SM's subsequent action that he determines not to join the conversation.

VII. USER EVALUATION

We conducted a user study to evaluate the performance of *AgentLens* in enhancing LLMAS analysis. The study was specially designed to assess the comprehensive efficiency, effectiveness, and usability of the system. We also examine the analytical support provided by our system compared to a baseline system, which replicates the visualization approach in existing LLMAS works.

A. Participants

To prevent participants from having prior knowledge of the system before evaluation, we recruited 14 new participants (denoted as P1-P14) from a local university who had not been involved in the design requirements phase of this study, thereby enhancing the assessment validity and the results generalizability. These participants have diverse academic backgrounds, with most being undergraduate and graduate students from fields such

No.	Task	Accuracy		Time Consuming (s)	
T1	What is the characteristic of the agent?		14/14		8.02
			14/14		12.03
T2	What is the first behavior the agent does after waking up?		14/14		44.50
			14/14		88.78
T3	What is the earliest conversation that occurred in a certain position?		14/14		20.00
			13/14		92.20
T4	How many conversations did a certain agent have in a day, and with whom?		13/14		38.17
			13/14		81.60
T5	Where did a certain agent first come up with a certain memory?		14/14		17.83
			12/14		29.42
T6	Why would a certain agent have such an memory?		13/14		51.29
			8/14		180.67

■ AgentLens ■ Baseline

Fig. 9. Statistical result of the accuracy and time consumption for participants completing individual behavior analysis tasks using both *AgentLens* and the baseline system.

as computer science, software engineering, and sociology. Some of them are developers with a high level of expertise in LLMAS, while others only have had direct interaction with LLMAS.

B. Baseline Systems

A baseline system[3] has been set up for direct comparison with our proposed system. Both the baseline system and our system utilize the log data generated by Reverie [10], which records the interactions and memory logs of agents within the system during the simulation process.

The baseline provides a view for replaying past events with plain text descriptions of agent settings and behaviors, which simulates a typical LLMAS panoramic visualization. Firstly, it features a monitoring interface that uses a flat map as the background. This allows users to replay and observe the agent positions and behavior descriptions at different time points through a timeline. Secondly, the system offers a textual representation of the current events for each agent, including the agent's location, the action in progress, and the ongoing dialogue (if any). Finally, the system also provides a pure textual display of all events in each agent's evolutionary process, encompassing the agent's personality, complete memory records, and event sequences. These features enable users to understand the agent behaviors and status and delve into their evolutionary process.

C. Procedure and Tasks

Introduction (10 min): Initially, we provided a concise overview of the research, including the motivation and methodology. We then collected basic personal information from them, including their gender, age, and occupation. In addition, we obtained authorization to record their behaviors during the subsequent task analysis. Finally, we describe the characteristics of the individual views in both baseline and *AgentLens* in detail and demonstrate their practical use in a specific scenario.

Task-Based Analysis (40 min): In this stage, participants were required to undertake 2 groups of analytical tasks (refer to Figs. 9 and 10), designed to evaluate the system's overall effectiveness

[3]https://reverie.herokuapp.com/arXiv_Demo/#

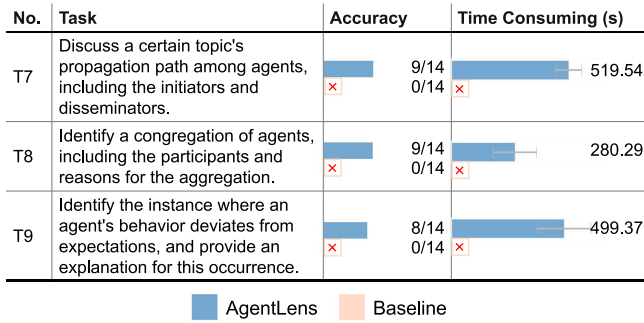| No.	Task	Accuracy	Time Consuming (s)
T7	Discuss a certain topic's propagation path among agents, including the initiators and disseminators.	9/14 ✗ 0/14 ✗	519.54
T8	Identify a congregation of agents, including the participants and reasons for the aggregation.	9/14 ✗ 0/14 ✗	280.29
T9	Identify the instance where an agent's behavior deviates from expectations, and provide an explanation for this occurrence.	8/14 ✗ 0/14 ✗	499.37

■ AgentLens   ■ Baseline

Fig. 10. Statistical result of the accuracy and time consumption for emergent phenomena identification tasks using both *AgentLens* and the baseline system.

and usability. Participants were required to fulfill tasks for each system, with the duration and accuracy of task completion being recorded. To obviate the potential for participants to replicate responses through memorization [84], the sequence in which the two systems were presented was randomized. Each task was uniquely tailored for both systems while ensuring an equivalent level of challenge.

Semi-Structured Interview (30 min): To enhance the evaluation of the method and interface efficacy, we utilized the five-point Likert scale in an 8-item questionnaire. Additionally, we employed the System Usability Scale (SUS) [85] to evaluate the usability of *AgentLens*. Participants were asked to rate each question from 1 (strongly disagree) to 5 (strongly agree) to gauge their agreement levels. During the questionnaire process, we encouraged participants to speak freely to uncover the reasoning behind their ratings.

D. Task Completion Analysis

For the task-based analysis, we conducted a quantitative comparison between *AgentLens* and the baseline, focusing on accuracy and task completion time. We developed two distinct groups of evaluation tasks to assess the efficacy of 2 systems for the analysis of agent behaviors (Fig. 9) and the identification of emergent phenomena arising from such behaviors (Fig. 10).

1) Individual Behavior Analysis: T1 - T6 in Fig. 9 are designed with elicit concise answers, requiring participants to rapidly comprehend the fundamental characteristics and behaviors of agents. Based on the analytical target, we categorize this set of tasks into 3 classifications. Participants exhibit varying levels of accuracy and time expenditure across tasks, however, there was a notable improvement in task accuracy ($p = 1.2e - 3$) and reduction in time consumption ($p = 1.2e - 3$) with *AgentLens*.

Single-Agent Analysis (T1 - T2): This set of tasks focuses on the system's enhancement of simple information analysis about individual agents. Without compromising task accuracy, *AgentLens* decreased time consumption by 33% for T1 ($\mu_{AgentLens} = 8.02, \mu_{baseline} = 12.03$) and by 50% for T2 ($\mu_{AgentLens} = 44.50, \mu_{baseline} = 88.78$) compared to the baseline system. The visual representation of agent characteristics in the *Agent View* eliminates the need for search operations in T1. Furthermore, the event summarization method helps participants

quickly identify agent behaviors, eliminating the need to sift through complex log records to complete T2.

Multi-Agent Analysis (T3 - T4): This set of tasks demonstrates the system's effect in assisting participants with the analysis of interactions between agents. It is noteworthy that one participant failed in both two tasks using the baseline system due to his incorrect agent selection. *AgentLens* reduced time consumption by 78.3% for T3 ($\mu_{AgentLens} = 20.00, \mu_{baseline} = 92.20$) and 53.2% for T4 ($\mu_{AgentLens} = 38.17, \mu_{baseline} = 81.60$). The visual encoding in *AgentLens*, particularly in the *Outline View*, allowed participants to quickly derive answers by observing agent interactions including dialogues and cohabitation instances.

Behavior Cause Analysis (T5 - T6): In this set of tasks, *AgentLens* demonstrated marked improvements over the baseline in facilitating the exploration of the cause of agent behaviors. While a part of the participants quickly obtained answers using the baseline in T5, *AgentLens* still provided a 39.4% improvement with the topic search feature ($\mu_{AgentLens} = 17.83, \mu_{baseline} = 29.42$). T6 presented a significant challenge for the baseline, with over 42% of participants notably failing to complete the task. P9 commented, "*In the ton of plain text logs, I can't find any connection between the events at all.*" However, with the cause trace feature in *Agent View*, *AgentLens* demonstrated a substantial 71.6% improvement in it ($\mu_{AgentLens} = 51.29, \mu_{baseline} = 180.67$).

2) Emergent Phenomena Identification: T7-T9 in Fig. 10 are designed to correspond to three categories of emergent phenomena arising from agent autonomy, which is not explicitly pre-programmed in LLMAS. These tasks are more complex for the participants, requiring back-and-forth exploration and analysis through multiple steps. We invited evaluators to assess the accuracy of the participant's responses. Concurrently, we observe that *AgentLens* demonstrates capabilities in complex analytical tasks that the traditional baseline failed to achieve, particularly in the exploration of emergent behaviors arising from agent autonomy.

Topic Propagation (T7): Participants are tasked with identifying the propagation path of a specific topic, such as "a Valentine's Day party will be held" or "someone is preparing the selection for mayor". Nearly all participants consider the task to be impossible while utilizing the baseline, as "this task is akin to searching for a needle in a haystack" (P11). When utilizing *AgentLens*, the majority of participants swiftly opted for the *Agent Memory Search* within the *Outline View* to conduct searches on the propagated topics. Leveraging the representation of *Agent Interaction Analysis* within the view, participants could easily explore the propagation paths. Although the propagation path participants were asked to identify has multiple branches and complex scenarios, 9 participants completed the task using *AgentLens*.

Agent Congregation (T8): Participants are required to identify a congregation phenomenon, defined as more than three agents engaging in the same behavior at the same location, and participants should explain the reason behind it. While using the baseline, participants were compelled to conduct extended observations and iterative replays of the recorded video. Despite locating the participants of the aggregation, they remained unable to

No.	Question	Avg.	Score Distribution
Q1	The event summary is informative.	4.43	8, 6
Q2	The cause trace feature is expected and credible.	4.00	2, 10, 2
Q3	The hierarchical structure is well-structured and appropriately granular.	4.43	8, 6
Q4	The Outline View helps me analyze the behavior of an agent.	4.64	5, 9
Q5	The Outline View helps me understand interactions among agents.	4.71	4, 10
Q6	The Monitor View helps me validate my observation.	4.57	1, 4, 9
Q7	The Agent View helps me analyze the agent characteristic.	4.29	2, 6, 6
Q8	The Agent View helps me trace the event cause from the agent perspective with clarity.	4.57	6, 8

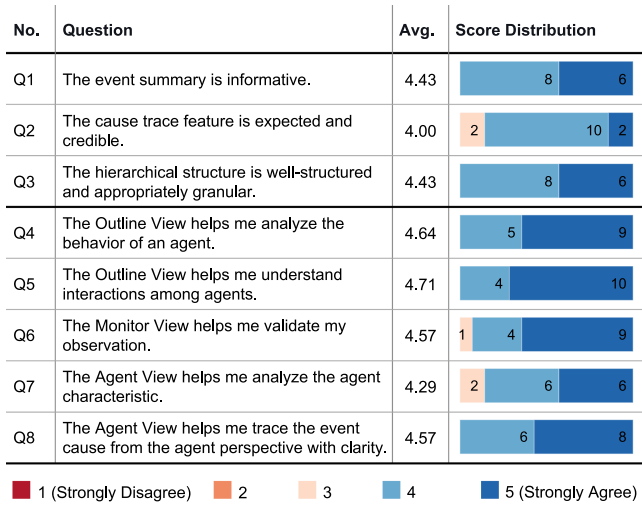■ 1 (Strongly Disagree)　■ 2　□ 3　■ 4　■ 5 (Strongly Agree)

Fig. 11. The questionnaire with results showing the efficacy of our method and interfaces.

ascertain the underlying causes of the phenomena. Through the interactivity among the three views of *AgentLens*, particularly the design of *Monitor View* and *Outline View*, participants were able to rapidly detect aggregation phenomena. Coupled with the method of behavior summarization, 9 participants successfully provided explanations for the aggregations.

Unexpected Behavior (T9): Participants were tasked with identifying and rationalizing unexpected agent behaviors across two systems. When using the baseline system, they noted that agent behaviors appeared uniformly logical and coherent. Additionally, the requisite alternation between observing multiple agents hindered their analytical process, thereby increasing the difficulty of detecting unexpected phenomena. With the assistance of *AgentLens*, this task became more manageable. P5 identified through *Outline View* that "agent RP did not leave his room throughout the entire day." He traced the cause using *Agent View* and discovered that the agent had received a plan that did not require leaving the house from LLM during the planning phase for that day. Another participant P8 noticed in *Agent View* that agent TT was able to observe the activities of agent IR in the adjacent room, and this observation influenced TT's subsequent decisions. The user suggested that this phenomenon should be addressed in the LLMAS, as in human society, individuals do not possess the ability to see through walls.

E. Semi-Structured Interview Analysis

We posed 8 interview questions in Fig. 11) and a SUS questionnaire(Fig. 12) to participants. Evaluating the results of the questionnaire with feedback obtained during the interview, we reported the performance of *AgentLens* including its effectiveness and usability, offering insights into its practical application.

1) Pipeline Effectiveness: All participants agreed that the event summary is **informative** (Q1) and helpful. P10 commented, *"The summaries are quite accurate. I can quickly locate the events and understand the evolution of an agent throughout the day with the help of the story-like subheadings."* P1 felt

impressed with the way of summarizing the agent's status, *"like having an agent helping me monitor this LLMAS"*.

Most participants agreed that the results of the **cause trace** met their expectations (Q2). They are willing to utilize the traced events to help analyze their interested events. For instance, P3 intended to incorporate the agent characteristics into the cause trace process. P5 pointed out that the cause trace served to *"unveil the black box of agent behavior"*.

The hierarchical structure received unanimous endorsement from all participants (Q3). They all admitted that the hierarchical structure elucidated the level at which they could retrieve information. Especially in the analysis of complex phenomena, the behavior hierarchical structure can *"effectively reduce information density"*(P6) and *"help me quickly focus on key phenomena"*(P10). Nonetheless, P12, who was relatively inexperienced with the LLMAS, expressed a need for more *"user-oriented guidance"*.

2) Visual Effectiveness: The *Outline View* was appreciated by the participants for **agents behavior analysis** (Q4). It helps participants circumvent the risk of *"getting lost in the complex and chaotic agent lines"* (P1) by summarizing and visualizing the agent's status. The interactive design, such as the click-to-highlight and view-details features, is *"remarkably user-friendly and intuitive"* (P11). In addition, the encoding of **interactions among agents** also received positive feedback from users (Q5). The gray box, which intertwines two lines to represent agent dialogues, *"stands out right away"* (P2). Some participants (P5, P7) indicated that they were accustomed to first spotting interesting agent dialogues in the relatively compact view, then zooming in to delve into more details. P7, who completed the task of identifying congregation phenomenon(T8) expeditiously, attributes the success to *"the visualization is trying to aggregate the curves of agents who are interacting with each other."* P9 commented, *"If I can dynamically adjust the positions of the agents in the view, the layout can better match my expectation"*.

The *Monitor View* was found to be useful for **validating the observation** (Q6). Several participants indicated that after observing the *Monitor View*, they gained more confidence in the results of their analysis. P10 mentioned, *"The monitor screen adjusts as I shift my focus in different views, kind of like video software, but it offers much more details than regular video playback."* P10 commended the interaction of this view in relation to the other two especially in complex tasks, *"This interactive responsiveness is beneficial during my iterative analysis process."* P5 suggested that the *Monitor View* could be more beneficial if it could *"display the location information of other unfocused agents"*.

The *Agent View* provides strong support for participants to analyze **individual agent characteristics** (Q7) and the **causal relationships** between agent behaviors (Q8). When observing agents of interest, they can *"quickly understand the agent's personality and style of action"* (P1). P6 said, *"The retrospective analysis is intuitive, but the individual timeline is too long. It would be better if I could explore the causes without having to drag the view around."* P4 praised the minimap in the *Agent View*, *"When I was trying to understand agent behaviors, I love using the minimap's navigation. It helped me find the causal*
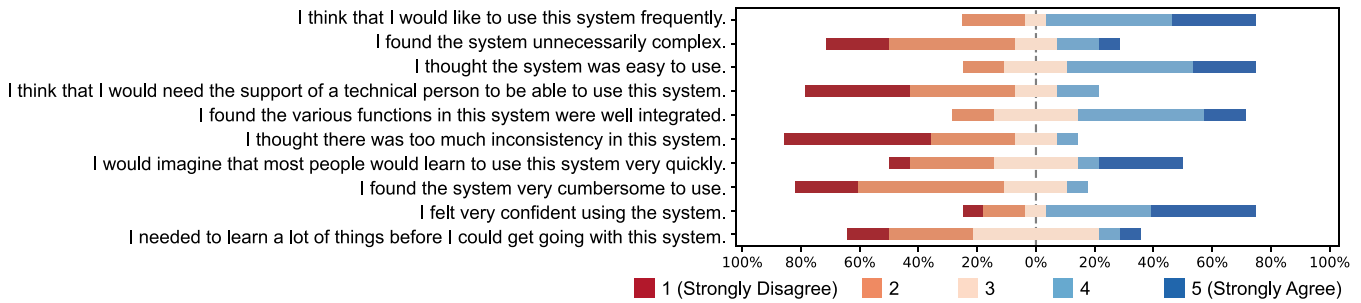
Fig. 12. The SUS questionnaire with results showing the usability of *AgentLens*.

links fast with those cool summary emojis." P13 commented, *"Developers should think about adding the agent view to their projects. Without it, agent behaviors might not seem convincing."*

3) Usability: We employed the SUS questionnaire to assess the system usability, thereby reporting users' cognitive load with *AgentLens*. Several developers among the participants conveyed not only their intent to use *AgentLens* in the future but also to consider its integration within their LLMAS development, which has significantly encouraged us.

Overall, participants provided positive comments on the usability. P9 lauded the workflow of *AgentLens*, *"I thoroughly enjoyed the freedom of exploration the system facilitated."*. P13 noted, *"The interaction is very fluid"*, but revealed a longing for automated assistance during complex analytical tasks: *"It would be perfect if the system could understand the type of task I want to analyze from just a few of my clicks."* Moreover, participants expressed their confidence and enjoyment when using *AgentLens*. However, several participants indicated that the system necessitates a measure of preliminary technical knowledge, despite acknowledgment from P2 that *"this is principally due to the intrinsic complexity of LLMAS itself."*

Ultimately, we achieved an average score of 67.5 on the SUS questionnaire(refer to Fig. 12), which we find exhilarating. However, it also serves as a reminder of the necessity for future optimization.

VIII. DISCUSSION

In this section, we commence by encapsulating the lessons collected from the user feedback, including providing comparisons within an agent and enabling modifications for system configurations. Subsequently, we deliberate on the generalizability, as well as the limitations and future work.

A. Lessons Learned

Providing Comparison Within an Agent: During the evaluation process, we recorded some specific interaction patterns among the users, although they did not actively mention them in the interview. Some users frequently analyzed the behaviors of a single agent across various temporal intervals. For instance, they compared the behaviors of an agent at 8 a.m. on February 13 with those at the same time on February 14. To facilitate this, they typically delved into the *Outline View* to explore the events associated with the agent at these two distinct time points. Observing disparate agent behaviors across separate days, users

inferred the existence of certain agent behavior patterns. This discovery inspires us to further investigate strategies for visually "folding" the agent's timeline, such as overlaying two periods of the timeline, thereby aiding users in rapidly comparing and encapsulating the agent's behavior patterns.

Enabling Modifications for System Configurations: Participants appreciated the aid provided by the novel behavior summarization method proposed in our study, which effectively mitigates information overload. Nevertheless, some users demonstrated an interest in understanding how these summaries are generated. They endorsed the summarization method after we clarified the details, as in Fig. 5. However, they still gave specific requirements, such as customizing the source of the summary contents. For example, one participant exhibited indifference towards the agent's location information. Such feedback motivates us to enable users to tailor the extraction pipeline in future research, thereby enhancing the usability of the exploratory analysis in a user-centric manner.

B. Generalizability

Our work builds upon the existing LLMAS, designed for the surveillance and analysis of agent behaviors. While we conduct our research based on Reverie, it can be seamlessly integrated into other LLMAS analysis processes. Moreover, the key components of our system, such as the *Outline View* and *Agent View*, are decoupled from the LLMAS implementations. The *Monitor View* is a representation of the replay monitor ubiquitous in most LLMAS. Developers can easily provide their own monitoring snapshots to populate this view. Therefore, our work is general to various LLMAS and can be used directly by developers in their LLMAS.

Our system's capabilities extend beyond LLMAS analysis and can be applied to a wide range of applications, such as the analysis of multi-person communities and the development of open-world games. For the analysis of multi-person communities, the *Outline View* and *Monitor View* can assist in simultaneously examining numerous actions on multiple subject timelines. This enables analysts to rapidly comprehend the main behaviors of different entities and their interactions. Within the realm of open-world games, the incorporation of the *Outline View* can aid players in exploring non-player characters (NPC) behaviors in an immersive manner. Game developers can also utilize the *Agent View* to analyze and optimize the NPCs in the development stages, fostering the creation of more intelligent NPCs.

C. Limitations and Future Work

Despite the encouraging performance of *AgentLens*, there are several limitations and potential areas for further research.

Provide a More Flexible Interface: The current layout of the agent line and position block in the *Outline View* is pre-computed. Despite considerable efforts to minimize the crossover of lines, it remains difficult to avoid, particularly as the number of agents and the evolutionary timespan of LLMAS increase. One of our future tasks is to provide a more flexible layout for the *Outline View*, automatically reorganizing the view based on the user's interest regarding agent events.

Allow Users to Modify Pre-Configured Settings: AgentLens introduces a set of pre-configured settings for users, such as the granularity of *Timeline Segmentation* and the similarity threshold for *Cause Trace*. These configurations optimize the exploration experience for users, making better trade-offs between the intricate nature of the information and its succinct presentation. Nonetheless, some users expressed a desire to modify these presets during the analysis process to facilitate more flexible exploration. To accommodate these needs, we plan to incorporate a customizable preset panel for users in our system.

Support Interactive Exploration Among Different Agent Execution Strategies: In this work, we focus on facilitating users' exploration and analysis of the LLMAS operational process. However, this process is significantly influenced by agent execution strategies like planning methods and memory mechanisms. For example, the agent may choose to first make a high-level plan to divide tasks into several sub-tasks that can be completed in different orders, or choose to adopt a depth-first strategy that adaptively changes its target based on the incoming information. While the design of an effective agent planning strategy is attracting an increasing amount of research attention [17], [86], [87], [88], how to interactively analyze the effect of different planning strategies in LLMAS is still unexplored. Moreover, analyzing the influence of agent memory mechanisms on the agent execution process is an area of considerable interest. While currently the agent memory mechanisms are usually hard-coded in the LLMAS program, allowing users to interactively modify the agent's memory content or recall strategies and visually examine its downstream effects could be crucial for better understanding and optimizing LLMAS.

Extend to Multimodal LLMAS: Text-based interaction has been widely adopted in most existing LLMAS [10], [12], [16] in which agents are predicated on textual perception and decision-making. Even embodied agents [19], [20] typically transmute the perceived multimodal data like imagery and auditory inputs into a textual format for later processing. However, with the popularity of multimodal LLMs [6], [89], the future may see the emergence of LLMAS in which agents genuinely perceive, think, and act based on multimodal data. Future work can explore how the agents interact with multimodal data (e.g., image interpretation [90] and creation [91]) in this authentic multimodal LLMAS.

IX. CONCLUSION

This work presents a visualization approach for LLMAS, addressing the challenge of analyzing complex agent behaviors during LLMAS evolution. We introduce a general pipeline that establishes a hierarchical behavior structure from the raw execution events of LLMAS, including a behavior summarization algorithm and a cause-tracing method. Our system, *AgentLens*, offers an intuitive and hierarchical representation of the evolution of multiple agents, enabling users to interactively investigate behavior details and causes. Through two usage scenarios and a user study, we have demonstrated the performance of our pipeline and visual designs.

REFERENCES

[1] G. A. Agha, *ACTORS - A Model of Concurrent Computation in Distributed Systems*. Cambridge, MA, USA: MIT Press, 1990.

[2] H. S. Nwana, "Software agents: An overview," *Knowl. Eng. Rev.*, vol. 11, pp. 205–244, Jul. 1996.

[3] M. J. Wooldridge and N. R. Jennings, "Intelligent agents: Theory and practice," *Knowl. Eng. Rev.*, vol. 10, no. 2, pp. 115–152, Jun. 1995.

[4] M. Hutter, *Universal Artificial Intelligence: Sequential Decisions Based on Algorithmic Probability*. Berlin, Germany: Springer Science & Business Media, 2004.

[5] L. Ouyang et al., "Training language models to follow instructions with human feedback," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, Cambridge, MIC Press, 2022, pp. 27730–27744.

[6] OpenAI, "GPT-4 technical report," Mar. 2023, arXiv: *2303.08774*.

[7] J. Wei et al., "Emergent abilities of large language models," *Trans. Mach. Learn. Res.*, vol. 2022, pp. 1–30, 2022.

[8] L. Wang et al., "A survey on large language model based autonomous agents," *Frontier Comput. Sci.*, vol. 18, no. 186345, pp. 1–26, Mar. 2024.

[9] Z. Xi et al., "The rise and potential of large language model based agents: A survey," Sep. 2023, arXiv: *2309.07864*.

[10] J. S. Park, J. O'Brien, C. J. Cai, M. R. Morris, P. Liang, and M. S. Bernstein, "Generative agents: Interactive simulacra of human behavior," in *Proc. 36th Annu. ACM Symp. User Interface Softw. Technol.*, 2023, pp. 1–22.

[11] J. Shi, J. Zhao, Y. Wang, X. Wu, J. Li, and L. He, "CGMI: Configurable general multi-agent interaction framework," 2023, arXiv: *2308.12503*.

[12] C. Qian et al., "Communicative agents for software development," Jul. 2023, arXiv: *2307.07924*.

[13] S. Hong et al., "MetaGpt: Meta programming for multi-agent collaborative framework," in *Proc. Int. Conf. Learn. Representations*, 2024, pp. 1–29.

[14] D. A. Boiko, R. MacKnight, and G. Gomes, "Emergent autonomous scientific research capabilities of large language models," Apr. 2023, arXiv: *2304.05332*.

[15] Gravitas, "AutoGPT," 2023. Accessed: Mar. 01, 2024. [Online]. Available: https://github.com/Significant-Gravitas/AutoGPT

[16] J. Lin, H. Zhao, A. Zhang, Y. Wu, H. Ping, and Q. Chen, "AgentSims: An open-source sandbox for large language model evaluation," Aug. 2023, arXiv: *2308.04026*.

[17] W. Chen et al., "AgentVerse: Facilitating multi-agent collaboration and exploring emergent behaviors in agents," Aug. 2023, arXiv: *2308.10848*.

[18] C. Zhang et al., "ProAgent: Building proactive cooperative AI with large language models," 2023, arXiv: *2308.11339*.

[19] H. Zhang et al., "Building cooperative embodied agents modularly with large language models," in *Proc. Int. Conf. Learn. Representations*, 2024, pp. 1–29.

[20] X. Zhu et al., "Ghost in the Minecraft: Generally capable agents for open-world environments via large language models with text-based knowledge and memory," 2023, arXiv: *2305.17144*.

[21] D. Hafner, J. Pasukonis, J. Ba, and T. P. Lillicrap, "Mastering diverse domains through world models," 2023, arXiv: *2301.04104*.

[22] A. Mirchev, B. Kayalibay, P. van der Smagt, and J. Bayer, "Variational state-space models for localisation and dense 3D mapping in 6 DoF," in *Proc. Int. Conf. Learn. Representations*, 2021, pp. 1–23.

[23] S. Franklin and A. C. Graesser, "Is it an agent, or just a program?: A taxonomy for autonomous agents," in *Proc. Int. Workshop Agent Theories Archit. Lang.*, Budapest, Hungary, Springer, 1996, pp. 21–35.

[24] S. Bubeck et al., "Sparks of artificial general intelligence: Early experiments with GPT-4," Mar. 2023, arXiv: *2303.12712*.

[25] H. Touvron et al., "LLaMA: Open and efficient foundation language models," Feb. 2023, arXiv: *2302.13971*.

[26] T. Brown et al., "Language models are few-shot learners," *Adv. Neural Inf. Process. Syst.*, vol. 33, pp. 1877–1901, 2020.

[27] S. Yao et al., "ReAct: Synergizing reasoning and acting in language models," in *Proc. Int. Conf. Learn. Representations*, 2023, pp. 1–33.

[28] S. Noah, C. Federico, G. Ashwin, N. K. R, and Y. Shunyu, "Reflexion: Language agents with verbal reinforcement learning," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, Cambridge, MIT Press, 2023, pp. 8634–8652.

[29] S. Yao et al., "Tree of thoughts: Deliberate problem solving with large language models," 2023, arXiv: *2305.10601*.

[30] T. Schick et al., "ToolFormer: Language models can teach themselves to use tools," 2023, arXiv: *2302.04761*.

[31] Y. Qin et al., "Tool learning with foundation models," 2023, arXiv: *2304.08354*.

[32] Y. Qin et al., "Toolllm: Facilitating large language models to master 16000+ real-world apis," 2023, arXiv: *2307.16789*.

[33] C. Qian, C. Han, Y. R. Fung, Y. Qin, Z. Liu, and H. Ji, "CREATOR: Disentangling abstract and concrete reasonings of large language models through tool creation," 2023, arXiv: *2305.14318*.

[34] H. Chase, "Langchain," 2022. [Online]. Available: https://github.com/hwchase17/langchain

[35] Y. Nakajima, "BabyAGI," 2023. [Online]. Available: https://github.com/yoheinakajima/babyagi

[36] G. Li, H. A. A. K. Hammoud, H. Itani, D. Khizbullin, and B. Ghanem, "CAMEL: Communicative agents for "mind" exploration of large language model society," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, Cambridge, MIT Press, 2023.

[37] Y. Talebirad and A. Nadiri, "Multi-agent collaboration: Harnessing the power of intelligent LLM agents," 2023, arXiv: *2306.03314*.

[38] T. Liang et al., "Encouraging divergent thinking in large language models through multi-agent debate," May 2023, arXiv: *2305.19118*.

[39] R. Nakano et al., "WebGPT: Browser-assisted question-answering with human feedback," 2021, arXiv: *2112.09332*.

[40] Q. Wu et al., "AutoGen: Enabling Next-Gen LLM applications via multi-agent conversation framework," Aug. 2023, arXiv: *2308.08155*.

[41] R. Guo et al., "Comparative visual analytics for assessing medical records with sequence embedding," *Vis. Inform.*, vol. 4, no. 2, pp. 72–85, 2020.

[42] Z. Jin, S. Cui, S. Guo, D. Gotz, J. Sun, and N. Cao, "CarePre: An intelligent clinical decision assistance system," *ACM Trans. Comput. Healthcare*, vol. 1, no. 1, pp. 1–20, 2020.

[43] C. B. Nielsen, S. D. Jackman, I. Birol, and S. J. M. Jones, "ABySS-Explorer: Visualizing genome sequence assemblies," *IEEE Trans. Vis. Comput. Graphics*, vol. 15, no. 6, pp. 881–888, Nov./Dec. 2009.

[44] S. Guo, K. Xu, R. Zhao, D. Gotz, H. Zha, and N. Cao, "EventThread: Visual summarization and stage analysis of event sequence data," *IEEE Trans. Vis. Comput. Graphics*, vol. 24, no. 1, pp. 56–65, Jan. 2018.

[45] S. Guo, Z. Jin, D. Gotz, F. Du, H. Zha, and N. Cao, "Visual progression analysis of event sequence data," *IEEE Trans. Vis. Comput. Graphics*, vol. 25, no. 1, pp. 417–426, Jan. 2019.

[46] A. Perer and F. Wang, "Frequence: Interactive mining and visualization of temporal frequent event sequences," in *Proc. ACM 19th Int. Conf. Intell. User Interfaces*, 2014, pp. 153–162.

[47] Y. Han, A. Rozga, N. Dimitrova, G. D. Abowd, and J. T. Stasko, "Visual analysis of proximal temporal relationships of social and communicative behaviors," *Comput. Graph. Forum*, vol. 34, no. 3, pp. 51–60, 2015.

[48] N. Cao, Y. Lin, F. Du, and D. Wang, "Episogram: Visual summarization of egocentric social interactions," *IEEE Comput. Graph. Appl.*, vol. 36, no. 5, pp. 72–81, Sep./Oct. 2016.

[49] F. Fischer, J. Fuchs, P. Vervier, F. Mansmann, and O. Thonnard, "VisTracer: A visual analytics tool to investigate routing anomalies in traceroutes," in *Proc. 9th Int. Symp. Vis. Cyber Secur.*, 2012, pp. 80–87.

[50] W. Li, M. Wang, and J. H Chow, "Real-time event identification through low-dimensional subspace characterization of high-dimensional synchrophasor data," *IEEE Trans. Power Syst.*, vol. 33, no. 5, pp. 4937–4947, Sep. 2018.

[51] Y. Wu, N. Cao, D. Gotz, Y. Tan, and D. A. Keim, "A survey on visual analytics of social media data," *IEEE Trans. Multimedia*, vol. 18, no. 11, pp. 2135–2148, Nov. 2016.

[52] F. Zhou et al., "A survey of visualization for smart manufacturing," *J. Vis.*, vol. 22, no. 2, pp. 419–435, 2019.

[53] Y. Shi, Y. Liu, H. Tong, J. He, G. Yan, and N. Cao, "Visual analytics of anomalous user behaviors: A survey," *IEEE Trans. Big Data*, vol. 8, no. 2, pp. 377–396, Apr. 2022.

[54] Y. Guo, S. Guo, Z. Jin, S. Kaul, D. Gotz, and N. Cao, "Survey on visual analysis of event sequence data," *IEEE Trans. Vis. Comput. Graphics*, vol. 28, no. 12, pp. 5091–5112, Dec. 2022.

[55] X. Yuan, Z. Wang, Z. Liu, C. Guo, H. Ai, and D. Ren, "Visualization of social media flows with interactively identified key players," in *Proc. IEEE Conf. Vis. Analytics Sci. Technol.*, 2014, pp. 291–292.

[56] Y. Wu, S. Liu, K. Yan, M. Liu, and F. Wu, "OpinionFlow: Visual analysis of opinion diffusion on social media," *IEEE Trans. Vis. Comput. Graphics*, vol. 20, no. 12, pp. 1763–1772, Dec. 2014.

[57] S. Chen, S. Li, S. Chen, and X. Yuan, "R-Map: A map metaphor for visualizing information reposting process in social media," *IEEE Trans. Vis. Comput. Graphics*, vol. 26, no. 1, pp. 1204–1214, Jan. 2020.

[58] G. Sun, T. Tang, T. Peng, R. Liang, and Y. Wu, "SocialWave: Visual analysis of spatio-temporal diffusion of information on social media," *ACM Trans. Intell. Syst. Technol.*, vol. 9, no. 2, pp. 1–23, 2018.

[59] J. Zhao, N. Cao, Z. Wen, Y. Song, Y. Lin, and C. Collins, "#FluxFlow: Visual analysis of anomalous information spreading on social media," *IEEE Trans. Vis. Comput. Graphics*, vol. 20, no. 12, pp. 1773–1782, Dec. 2014.

[60] F. B. Viégas et al., "Google+Ripples: A native visualization of information flow," in *Proc. Int. Conf. World Wide Web*, Springer, 2013, pp. 1389–1398.

[61] P. Law, Z. Liu, S. Malik, and R. C. Basole, "MAQUI: Interweaving queries and pattern mining for recursive event sequence exploration," *IEEE Trans. Vis. Comput. Graphics*, vol. 25, no. 1, pp. 396–406, Jan. 2019.

[62] S. N. Dambekodi, S. Frazier, P. Ammanabrolu, and M. O. Riedl, "Playing text-based games with common sense," 2020, *arXiv: 2012.02757*.

[63] M. J. Hausknecht, P. Ammanabrolu, M. Côté, and X. Yuan, "Interactive fiction games: A colossal adventure," in *Proc. Conf. Assoc. Advance. Artif. Intell.*, 2020, pp. 7903–7910.

[64] R. Liu et al., "Training socially aligned language models in simulated human society," 2023, arXiv: *2305.16960*.

[65] D. Driess et al., "PaLM-E: An embodied multimodal language model," in *Proc. ACM Int. Conf. Mach. Learn.*, 2023, pp. 8469–8488.

[66] S. Paul, A. Roy-Chowdhury, and A. Cherian, "AVLEN: Audio-visual-language embodied navigation in 3D environments," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, Cambridge, MIT Press, 2022, pp. 6236–6249.

[67] J. S. Park, L. Popowski, C. J. Cai, M. R. Morris, P. Liang, and M. S. Bernstein, "Social simulacra: Creating populated prototypes for social computing systems," in *Proc. 35th Annu. ACM Symp. User Interface Softw. Technol.*, 2022, pp. 1–18.

[68] C. Gao et al., "S: Social-network simulation system with large language model-empowered agents," 2023, arXiv: *2307.14984*.

[69] R. Williams, N. Hosseinichimeh, A. Majumdar, and N. Ghaffarzadegan, "Epidemic modeling with generative agents," 2023, arXiv: *2307.04986*.

[70] A. O'Gara, "Hoodwinked: Deception and cooperation in a text-based game for language models," 2023, arXiv: *2308.01404*.

[71] L. Fan et al., "MineDojo: Building open-ended embodied agents with internet-scale knowledge," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, Cambridge, MIT Press, 2022, pp. 18343–18362.

[72] G. Wang et al., "Voyager: An open-ended embodied agent with large language models," 2023, arXiv: *2305.16291*.

[73] C. Lynch et al., "Interactive language: Talking to robots in real time," 2022, arXiv: *2210.06407*.

[74] D. Surís, S. Menon, and C. Vondrick, "ViperGPT: Visual inference via python execution for reasoning," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2023, pp. 11854–11864.

[75] K. Nottingham et al., "Do embodied agents dream of pixelated sheep: Embodied decision making using language guided world modelling," in *Proc. Int. Conf. Mach. Learn.*, 2023, pp. 26311–26325.

[76] J. Cui, Z. Li, Y. Yan, B. Chen, and L. Yuan, "ChatLaw: Open-source legal large language model with integrated external knowledge bases," 2023, arXiv: *2306.16092*.

[77] W. Huang et al., "Inner monologue: Embodied reasoning through planning with language models," in *Proc. 6th Annu. Conf. Robot Learn.*, Auckland, New Zealand, PMLR, 2022, pp. 1769–1782.

[78] Y. Shen, K. Song, X. Tan, D. Li, W. Lu, and Y. Zhuang, "HuggingGPT: Solving AI tasks with ChatGPT and its friends in hugging face," 2023, arXiv: *2303.17580*.

[79] Z. Wang, S. Mao, W. Wu, T. Ge, F. Wei, and H. Ji, "Unleashing the emergent cognitive synergy in large language models: A task-solving agent through multi-persona self-collaboration," Jul. 2023, arXiv: *2307.05300*.

[80] C. J. Chu, "Time series segmentation: A sliding window approach," *Inf. Sci.*, vol. 85, no. 1/3, pp. 147–173, Jul. 1995.

[81] C. Truong, L. Oudre, and N. Vayatis, "Selective review of offline change point detection methods," *Signal Process.*, vol. 167, Feb. 2020, Art. no. 107299.

[82] M. Ogawa and K. Ma, "Software evolution storylines," in *Proc. ACM 5th Int. Symp. Softw. Vis.*, 2010, pp. 35–42.

[83] Y. Tanahashi and K. Ma, "Design considerations for optimizing storyline visualizations," *IEEE Trans. Vis. Comput. Graphics*, vol. 18, no. 12, pp. 2679–2688, Dec. 2012.

[84] Y. Feng et al., "XNLI: Explaining and diagnosing NLI-based visual data analysis," *IEEE Trans. Vis. Comput. Graphics*, to be published, doi: 10.1109/TVCG.2023.3240003.

[85] B. John, "SUS: A retrospective," *J. Usability Stud.*, vol. 8, no. 2, pp. 29–40, 2013.

[86] X. Team, "XAgent: An autonomous agent for complex task solving," 2023. [Online]. Available: https://github.com/OpenBMB/XAgent

[87] R. Team, "Research agent," 2023. [Online]. Available: https://github.com/mukulpatnaik/researchgpt

[88] Y. Zheng, C. Ma, K. Shi, and H. Huang, "Agents meet OKR: An object and key results driven agent system with hierarchical self-collaboration and self-evaluation," Nov. 2023, arXiv: 2311.16542.

[89] Z. Yang et al., "The dawn of LMMs: Preliminary explorations with GPT-4V(ision)," 2023, arXiv: 2309.17421.

[90] F. Yingchaojie et al., "iPoet: Interactive painting poetry creation with visual multimodal analysis," *J. Vis.*, vol. 25, no. 3, pp. 1–15, 2022.

[91] Y. Feng et al., "PromptMagician: Interactive prompt engineering for text-to-image creation," *IEEE Trans. Vis. Comput. Graphics*, vol. 30, no. 1, pp. 295–305, Jan. 2024.

Yingchaojie Feng received the BE degree in software engineering from the Zhejiang University of Technology, China in 2020. He is currently working toward the PhD degree with the State Key Lab of CAD&CG, Zhejiang University, China. His research interests include data visualization, human-computer interaction, and natural language processing. For more details, please refer to https://yingchaojiefeng.github.io/.

Jingyuan Hu is currently working toward the undergraduate degree with the Chu Kochen Honors College, Zhejiang University. His research interests include visualization and visual analytics.

Yuchen Peng received the BE degree in computer science and technology from the Zhejiang University, China in 2022. He is currently working toward the PhD degree with the State Key Laboratory of Blockchain and Data Security from the Zhejiang University. His research interests include database system and data management in machine learning.

Jiaying Lu received the BE degree in computer science and technology from the Zhejiang University, China in 2022. She is currently working toward the master's degree with the State Key Lab of CAD&CG, Zhejiang University, China. Her research interests include LLM agent and visual analytics.

Bo Pan received the BS degree in electrical and computer engineering from the University of Illinois Urbana-Champaign and Zhejiang University in 2022. He is currently working toward the PhD degree with the State Key Lab of CAD&CG, Zhejiang University, China. His research interests include visualization and deep learning.

Wei Chen (Senior Member, IEEE) is a professor with the State Key Lab of CAD&CG, Zhejiang University. His current research interests include visualization and visual analytics. He has published more than 80 IEEE/ACM Transactions and IEEE VIS papers. He actively served in many leading conferences and journals, like IEEE PacificVIS steering committee, ChinaVIS steering committee, paper cochairs of IEEE VIS, IEEE PacificVIS, IEEE LDAV and ACM SIGGRAPH Asia VisSym. He is an associate editor of *IEEE Transactions on Visualization and Computer Graphics*, *IEEE TBG*, *ACM Transactions on Intelligent Systems and Technology*, *IEEE Transactions on Systems, Man, and Cybernetics*, *IEEE Transactions on Intelligent Vehicles*, *IEEE Computer Graphics and Applications*, *Frontiers of Computer Science*, and *Journal of Visualization*. More information can be found at: http://www.cad.zju.edu.cn/home/chenwei.

Jieyi Chen received the BE degree from the Zhejiang University of Technology, China in 2023. She is currently working toward the master's degree with the State Key Lab of CAD&CG, Zhejiang University, China. Her research interests include visualization and visual analytics.