Reimagining Generative Agents: 3D Architecture Implementation and LLM Caching Optimisation

Jinyoung Jeong

Master of Science
Artificial Intelligence
School of Informatics
University of Edinburgh
2024

Abstract

Human behaviour simulation has been a topic of interest for decades, with applications ranging from social simulations to video games. Traditionally, many computational agents relied on rule-based or statistical approaches. However, the complexity of human behaviour makes it impossible to account for all possible scenarios using these methods. The recently introduced generative agent architecture addresses this limitation by utilising a large language model (LLM) with additional cognitive modules to create more coherent and believable behaviours without explicit programming. While the generative agent architecture shows great potential, it has two main limitations: implementation in a less immersive 2D environment and prohibitively high prompting costs. To address these issues, this research makes two main contributions. First, the generative agent architecture is reimplemented in a 3D environment from scratch. Second, LLM caching is applied to the system using an embedding lookup table with cosine similarity. The caching model successfully retrieved appropriate responses for simpler prompts but struggled with more complex ones containing rich background information. Nevertheless, the caching model achieved a significant 80.3% reduction in prompting costs for a 6 hours simulation, while maintaining much of the agents' ability to make believable behaviours.

Research Ethics Approval

This project was planned in accordance with the Informatics Research Ethics policy. It did not involve any aspects that required approval from the Informatics Research Ethics committee.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*Jinyoung Jeong*)

Acknowledgements

I would like to express my sincere gratitude to my parents who continuously supported me even during difficult times. Without their help, I would not have made it this far.

I also appreciate the valuable insights from my supervisors, Ivan Titov and Guillem Ramirez Santos. Their guidance during the the project was truly helpful.

Table of Contents

Chapter 1

Introduction

Human behaviour simulation has been a major topic of interest for many decades. For various applications, researchers have made many efforts to make computational agents which can behave consistently with their past memories and make realistic interactions with their surrounding environments. These agents can be used to create characters with behaviours based on predefined decision logic [3], serve as more affordable and reproducible social science experiment participants [8], provide training environment for difficult interpersonal situations [7], and support non-playable video game characters that can provide more realistic interactions with the users [9]. In those examples, it is essential for computational agents to make natural and believable interactions to maximise user immersion and make more realistic social simulations.

Traditionally, computational agents were usually built on rule-based or statistical approaches. Those methods, which are still commonly used in many video games, typically rely on finite state machines, behaviour trees, or utility-based AI systems. These approaches involve explicitly defining decision logic, action sets, and transition rules to govern agent behaviour. For instance, a non-player character (NPC) in a role-playing game might use a behaviour tree to decide whether to engage in combat, flee, or interact with the player based on predefined conditions and priorities. Nonetheless, the human behaviour domain is vast and complex, making it fundamentally infeasible for developers to manually write logic for all possible situations. This issue results in repetitive and predictable behaviours from computational agents that can potentially break user immersion.

However, the recent advancement of large language models (LLMs) has opened up new possibilities. LLMs, trained on vast amounts of data, can generate contextually appropriate responses given user inputs. Modern computational agents utilise LLMs to make more flexible and nuanced agent behaviours, as the LLMs can interpret complex situations that are not limited to a predefined set of options. For example, an agent makes a prompt including the current situation, and asks the LLM about the most appropriate action. The agent then follows the action description from the LLM response. Those LLM powered computational agents successfully simulated human behaviours at a certain time point, with the given contexts [16]. But sometimes, they failed to make consistent long-term behaviours as they made decisions only with the current contextual information.

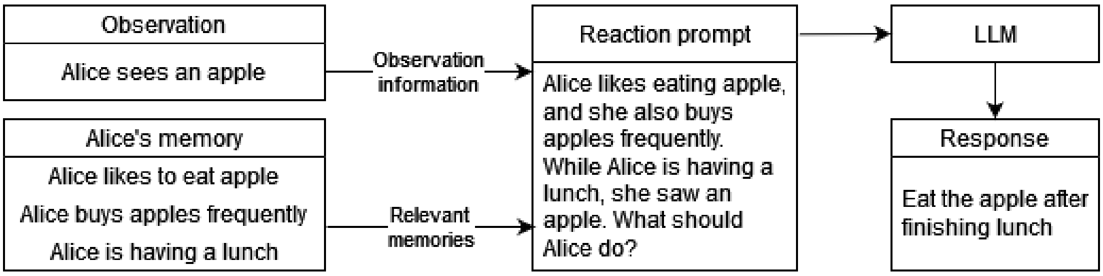


Figure 1.1: Prompt generation process of Alice Wilson, as she observed an apple.

To address the long-term consistency issues, Park et al. [15] recently introduced generative agents, which combine LLMs with additional cognitive modules to create more coherent and believable behaviours. Those cognitive components allow generative agents to make decisions not only from the current observation, but also with the past experiences. Figure 1.1 shows an example of this process. While the agent Alice Wilson was having lunch, she saw an apple. From this observation, she retrieves the relevant past memories and uses them to build a reaction prompt. Thus, the reaction prompt now contains not only the observation itself, but more contextual information which helps the LLM make appropriate decisions.

Generative agents have a wide range of potential applications across various fields. In social sciences, they can be used to create more realistic simulations for studying human behaviour and decision-making processes. These agents can also provide valuable insights about how individuals and groups interact in different scenarios [15]. In the entertainment industry, generative agents may revolutionise video games by creating NPCs that have their own goals, memories, and evolving relationships with players, leading to more immersive and dynamic gaming experiences.

However, the current generative agent architecture has several limitations. One major constraint is the high operating cost associated with prompting LLMs for each agent action and decision. Park et al. simulated a small town with 25 agents for two simulation days, and this simulation cost thousands of dollars solely for making external

LLM prompts. The high LLM prompting cost makes it infeasible to run larger-scale simulations or to use these agents in applications that require continuous, long-term interactions. Another limitation is that the current architecture operates in a simplified 2D environment, which may not fully capture the complexity of real world environments and interactions. For example, multi-story buildings are very common, but the existing 2D system cannot support them.

To address those limitations, this research makes two main contributions. First, the entire generative agent architecture has been reimplemented in the 3D environment. The 3D system is expected to be more immersive for users in many applications, with more realistic graphics. Second, to tackle the high prompting costs, a LLM caching technique has been implemented. The cache employs a lightweight embedding table, and uses cosine similarity for retrieval. The LLM caching is anticipated to significantly reduce the prompting cost of the generative agent simulations, making larger-scale and longer-term applications more feasible. The caching system has also been thoroughly investigated to determine efficient data gathering methods from the generative agent system.

These improvements would help enhance the practical applicability of generative agents across various fields. With the cost and environment constraints mitigated, the generative agents will be more applicable in more diverse and complex simulation.

This report consists of the following chapters.

- Chapter 2 delves into the concept of generative agents, providing explanations of individual cognitive modules from the original research. It also introduces the new 3D architecture with examples from real simulations and a link to the demonstration video. Additionally, the chapter addresses the limitations of the 3D architecture and discusses potential mitigation methods.

- Chapter 3 investigates various aspects of the LLM caching model. The chapter begins with detailed explanations of LLM caching model structures. Next, those caching models are trained and evaluated on their ability to find appropriate cached responses, using a superior LLM as a judge. In addition, various phenomena observed in caching model performance, such as cosine similarity saturation and performance differences across prompt categories, are analysed.

- Chapter 4 concludes the research by summarising the key findings and contributions. The limitations of the study are also presented with potential suggestions for future work.

Chapter 2

3D Architecture

2.1 Motivation

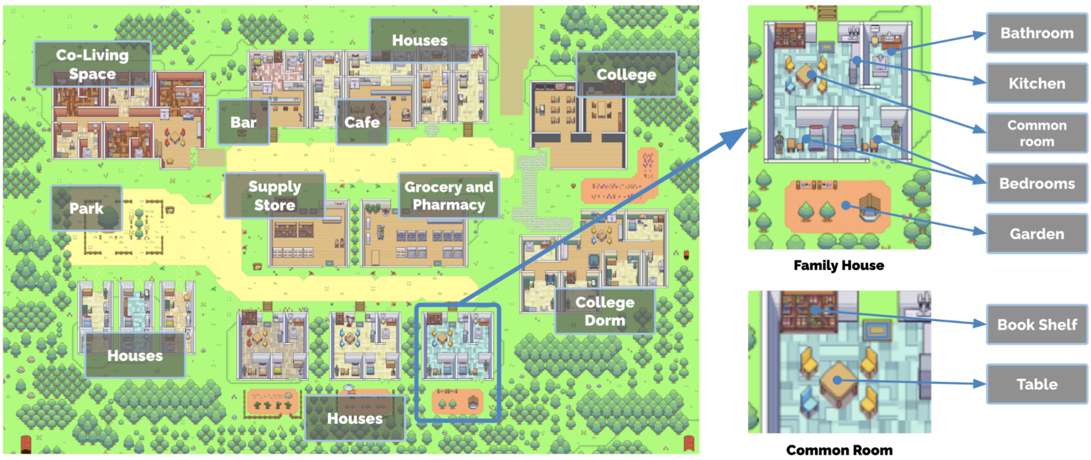


Figure 2.1: Smallville sandbox world from original generative agent simulation

The original generative agent architecture built by Park et al. [15] is implemented in a 2D sprite-based sandbox world. As shown in Figure 2.1, the simulation town consists of various areas and objects.. There are 25 agents in the town, and each agent is assigned to a simple 2D avatar. The 2D system offers several advantages. It allows for easy observation of the entire world, making it simple to see which agent is doing what at any given time. Additionally, compared to 3D world, 2D world is relatively more easy to implement and requires relatively less resources to run.

However, the 2D simulation has several inherent limitations. First, a 2D simulation cannot fully capture the complexities of our 3D reality. For example, multi-story buildings cannot be accurately represented in a 2D world. Similarly, vertical movements such as climbing stairs are also challenging in a 2D world. Second, the visual representation in 2D is often less immersive and engaging for users in certain domains. As an example, virtual reality simulations are not supported with the 2D system.

To address these two limitations, this research introduces a full 3D architecture for generative agents, built from scratch using Unreal Engine[1] for the 3D simulation client and a Python Flask[2] backend server. The new implementation was primarily made from the descriptions and explanations from the original paper, with only minimal reference to their prompt templates. In short, new 3D system has nearly all features, reimplemented from the original 2D architecture. Notably, this new architecture also supports the original 2D system with minimal modifications, offering flexibility in simulation implementations.

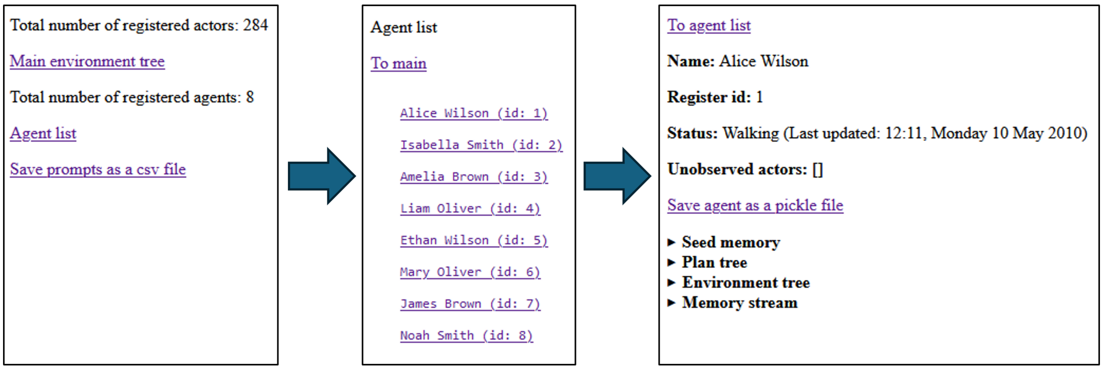


Figure 2.2: Simulation dashboard where users can check the registered actors and agent details.

In addition to the improvements made in the simulation, the new architecture also supports an intuitive dashboard accessible through an internet browser. The dashboard provides a comprehensive overview of all registered actors and agents. As illustrated in Figure 2.2, users can easily check the details of the chosen agent (or actor), and can even save it for various other purposes. For example, a saved agent could be employed to make decisions based on separate artificial observations or engage in dialogue outside of the generative agent architecture.

2.2 Background

Humans can store and retrieve memories, reflect on past experiences, make future plans, and react to observations. The generative agents can mimic those behaviours with

[1] Unreal Engine is a powerful free 3D creation tool made by Epic Games. It is widely used in many fields such as video game development, architecture, and film making. It is freely available for individuals and companies earning less than one million USD in annual gross revenue.

[2] Flask is a lightweight, flexible, and open-source Python web framework that allows developers to quickly build web applications and APIs. It is widely used for backend development due to its simplicity and extensibility.

following cognitive modules.

- Memory stream: It stores observations and thoughts made by agents. Agents can retrieve relevant memories to the given query when needed.
- Reflection: From recent experiences, agents can make high-level reflections. They allow agents to answer more abstract questions.
- Plan making: To maintain long-term consistency, agents make their future plans in advance from their past experiences.
- Reaction: When agents observe something, they may want to change future plans to do something else. For example, observing another agent may trigger a dialogue.

For example, a programmer agent, Mary Oliver, is made with her seed memories (characteristics, daily routines, etc.). In the morning, Mary makes today's plans from her background information, such as having breakfast. On her way to the kitchen, she sees her laptop. Mary retrieves past experiences about the laptop, and remembers that she has a virtual meeting in 30 minutes with her supervisor. Then she changes her plan to prepare for the meeting instead. After the meeting, she reflects on the discussion with her supervisor, and makes high-level reflections about her next project.

Those cognitive modules are explained further in the following sections, to set the stage for subsequent discussions on LLM chapters. The explanations are high-level, and based on the new 3D architecture. In new 3D architecture, there are two types of entities: agents and actors. Agents are active, decision making generative agents themselves. Actors, on the other hand, are passive entities within the environment with fixed locations. They can be used by agents but cannot initiate actions independently.

2.2.1 External Models Used

The original architecture used GPT3.5-turbo [10] as their main LLM and the text-embedding-ada-002 model [11] for embeddings. The new 3D architecture utilises GPT4o-mini [12], and employs the text-embedding-3-large model [13] for all embeddings. For following discussions, all future references to "the LLM" refer to GPT4o-mini unless otherwise specified.

2.2.2 Seed Information

Identity	Alice Wilson (age: 17, gender: Female)
Innate traits	Friendly, Outgoing, Curious
Seed memory	Alice Wilson is a high school student passionate about technology and coding. She studies at Riverview High School. Beyond her interest in tech, she enjoys playing basketball, often leading her school team in local tournaments. Alice lives with her brother Ethan and has a close relationship with him, relying on him since their parents' early demise. She is best friends with Mary Oliver, and they often study together.
Seed structures	Wilson Family House, Riverview High School, Fresh Grocery Store

Table 2.1: Seed information of Alice Wilson.

In simulation, each generative agent is initialised with a brief summary that encapsulates their core identity. This seed information, typically about a paragraph long, includes key information such as the agent's occupation, relationships with other agents, daily routines, and foundational memories. Table 2.1 provides an example of how one agent might be defined. From the rough agent concepts that were made manually, the innate traits and seed memories were written by the LLM, with the prompt example in the appendix A.1.

The seed information serves as a foundation for the agents' decision making. While seed information remains constant throughout the simulation, agents can build new relationships and interactions with other entities. These dynamic elements are continuously added to agents' memories, creating more comprehensive and up-to-date profiles.

2.2.3 Environment

The simulation environment consists of actors, as shown in Figure 2.1. Those actors are stored in a tree data structure. The root actor represents the whole world, and its children are structures. Similarly, the leaf actors are items. During the simulation, the server keeps the main environment tree, which contains the true information of all actors.

In addition to the main environment tree, individual agents also build their own trees as they explore the simulation. Those personal trees only contain their seed structures (own house, workplace, etc.) at the beginning. When the agents observe new actors, their trees explain with the newly discovered actors. The agents use their trees to make their reasoning grounded to the simulation world. For instance, when an agent wants to find a location for dinner, the agent will initially choose one of the known structures in the personal environment tree. One example finding location prompt is attached to the appendix A.2. As agents are not omniscient, their trees may get out of the data or do not contain some of the existing actors.

2.2.4 Memory

Creation	Last access	Importance score	Description
2010-05-11 08:20:16	2010-05-13 17:38:54	0.1	Office Chair in Teacher Office in Riverview High School is Idle
2010-05-11 08:27:36	2010-05-11 19:52:24	0.2	After Alice Wilson saw Office Desk in Teacher Office in Riverview High School is being used by James Brown, Alice Wilson changed future plans to do Work on coding projects and review key concepts
2010-05-11 08:41:56	2010-05-13 18:53:55	0.6	Alice Wilson prepares a comprehensive summary of the coding project ideas and potential implementation strategies at Tree in Victoria Park

Table 2.2: Memory stream subset (without embedding) of the agent Alice Wilson.

Generate agents make decisions from their past experiences. For example, agents make today's plan from yesterday's experiences. Thus, agents need to store their experiences in a comprehensive memory record, known as memory stream. As shown in Table 2.2, memory stream contains various types of memories, such as environmental observations, changing future plans, and self observations.

During the simulation, agents accumulate a vast amount of memories. However, they typically only need to find memories relevant to their current situation. For example, when an agent is preparing to study at a desk, it should only retrieve past experiences related to studying or interactions with the desk. To help the memory retrieval process, the memory stream stores more than just memory descriptions. It also includes metadata such as creation and last access timestamps, importance score, and embedding of memory descriptions. These elements are used to calculate retrieval scores, where memories with higher retrieval scores have priority in the retrieval. The retrieval score is the sum of the following three components.

- Recency favours recently accessed memories. Naturally, agents remember recent events better compared to others. If a memory was accessed (retrieved) recently, it receives higher scores.
- Importance gives advantages for more poignant memories. Each memory has an importance score, made by directly asking the LLM as exemplified in the appendix A.3.

- Relevance measures the semantic similarity between the query and memory descriptions. The memories and query calculate their embeddings, and cosine similarities (relevance scores) are calculated between them.

Additionally, the agents use the memory retrieval to make their self-summaries. Agents periodically retrieve several groups of memories using predefined queries such as “core characteristics” and “current daily occupation”. The LLM uses those retrieved memories to make agent summaries, with the example prompt shown in the appendix A.4. Those self-summaries are used for agents’ future prompts to provide concise agent background information to the LLM.

2.2.5 Reflection

Agents can make observational memories, but they alone often prove insufficient for making high-level inferences. Consider a scenario where an agent is asked to identify the most favourite food. Only with observation memories, the agent may simply choose the most frequently consumed food. However, this approach can lead to inaccurate answers. For example, many people consume white bread regularly, but it does not necessarily indicate white bread is their favourite.

Number	Reflection
1	Mary values teamwork and collaboration in her projects, enhancing creativity and productivity.
2	Mary actively seeks inspiration from nature, which plays a key role in her artistic process.
3	Mary’s productivity is enhanced through organized and interactive learning experiences.

Table 2.3: Reflections made by Mary Oliver about her recent experiences.

To properly answer those abstract questions, the agent needs to connect different memories together to make reflections. They are high-level thoughts, made from the agent’s recent memories. The reflection process consists of two steps. First, agents determine what to reflect on by querying the LLM with recent memories. The LLM generates several salient questions that can be answered from the given memories. For example, Mary Oliver made the question: “What are the key components of the tech-arts project Mary is collaborating on with Alice?”. Second, each question is used to retrieve groups of relevant memories. These memory groups are then fed back to the LLM to extract multiple high level insights. Reflection examples are shown in Table 2.3. The example prompts used for the reflection process are in the appendices A.5 and A.6.

2.2.6 Planning

Agents can make believable behaviours from their current situational information, such as the observation or simulation time. However, they often make inconsistent behaviours over time. For instance, if an agent asks the LLM what to do at 12 PM, it may suggest having lunch. If the same query is made at 12:30 PM after lunch, the LLM may incorrectly recommend lunch again, failing to account for recent actions. Therefore, agents need to make future plans in advance to make their long-term behaviours consistent.

Start	Finish	Location	Description
08:00	10:30	Brown Family House	Attend a seminar on renewable energy innovations and network with fellow activists
10:30	13:00	Victoria Park	Meet with local environmental group to discuss collaborative projects for the community
13:00	15:00	Smith Family House	Review current environmental literature and prepare notes for personal growth

Table 2.4: First three initial plans made by Isabella Smith on 13 May.

Agents first create initial, high-level plans that cover their daily routines. Example initial plans are shown in Table 2.4. Those initial plans are then recursively decomposed into smaller chunks, roughly a few hours in advance. The decomposed sub-plans are assigned with child actors derived from their parent plans' actors. For example, if the parent plan's actor is a house, then the child plans are assigned with rooms in the house. The appendices A.7 and A.8 contain example prompts used in generating initial plans and decomposing plans, respectively.

Agents execute their plans sequentially throughout the simulation. They move to the current plan actor, replace their own status to the plan description, and change the status of the current plan actor. To determine appropriate actor status updates, agents make a prompt to the LLM. The prompt asks both during and after plan statuses, as exemplified in appendix A.9. For example, while someone is studying at a desk, its status would be "in use", which becomes "idle" after use. The agent continues this process, but from an user's perspective, the agents simply wait next to their plan actors without any visual actions or movements until the plan finishes.

2.2.7 Reaction

When an agent observes an actor, the agent may trigger a reaction based on the situation. For example, an agent noticing a chair on fire would like to cancel its current plan to address the emergency. To determine if a reaction is needed, the agent creates an actor reaction prompt including the observation and other relevant information. The LLM then decides whether the agent should continue its current plan, interact with the observed actor, or make another action in a different location.

Similarly, observing another agent can also trigger a reaction. The agent reaction process is similar to those of actor reactions, but its reaction prompt also includes a relationship summary between the observing and observed agents. This summary is generated by retrieving and summarising relevant memories about their relationship and current status. If the observer agent decides to initiate a dialogue, it makes the initial utterance to the observed agent. The dialogue starts if the observed agent replies. During the dialogue, the agents continue to make utterances while taking turns, using the dialogue history and relevant memories retrieved. This process continues until one of participating agents finishes the dialogue. After the conversation ends, both agents create dialogue summaries and add them to their memory streams.

Following an actor or agent reaction, agents revise their future plans to accommodate any changes made by the reaction. This involves modifying their plans for the remainder of the day, which are then decomposed and prepared as usual. Examples of various prompts used in the reaction process, including actor reaction, agent reaction, relationship summary, dialogue utterance, dialogue summary, and future plan revision, are available in the appendices A.10 through A.15.

2.3 Simulation

The new 3D architecture simulation employs a compact testing environment consisting of nine structures, as illustrated in Figure 2.3. These structures include agent houses, workplaces, and communal areas such as Victoria Park and NextGen Game Centre. The environment features workplaces for primary activities and central locations like Victoria Park for social encounters. The structures typically have a variety of rooms. For instance, the Wilson Family House comprises four distinct areas: a kitchen, a main bathroom, a master bedroom (with an ensuite bathroom), and a second bedroom. The simulation populates the environment with four families, totaling eight unique agents. Their full seed information is shown at the appendix B. Each agent is assigned a 3D avatar sourced from the Paragon content pack[3] by Epic Games.

To simplify testing and analysis, the current environment includes only essential

[3] Epic Games released a comprehensive asset pack originally created for their development project, Paragon. The pack is freely available to Unreal Engine developers [5].

Figure 2.3: Bird's-eye view of the testing environment for the new 3D architecture.

functional components. However, its flexible architecture allows for easy expansion to include more buildings, complex structures, and non-functional decorations in the future. Additionally, existing environments from other sources can be easily converted for use with the generative agents, further expanding the system's capabilities.

This project provides the demonstration video to help understand the 3D architecture. The video is available at: `https://youtu.be/5p6_ZU7tQwg`. It is recommended to check the appendix C for more information.

2.3.1 Example Day in the Life

To illustrate how agents formulate plans and adapt their behaviours through interactions with other agents and their environment, an example day in the life of Alice Wilson is demonstrated below. This demonstration provides insights about how agents build memories and relationships over time.

Alice Wilson's day begins at Riverview High School, where she attends morning classes starting at 9:00 AM. After classes end around 11:30 AM, she makes her way to the library. There, she meets her closest friend, Mary Oliver. Mary initiates the

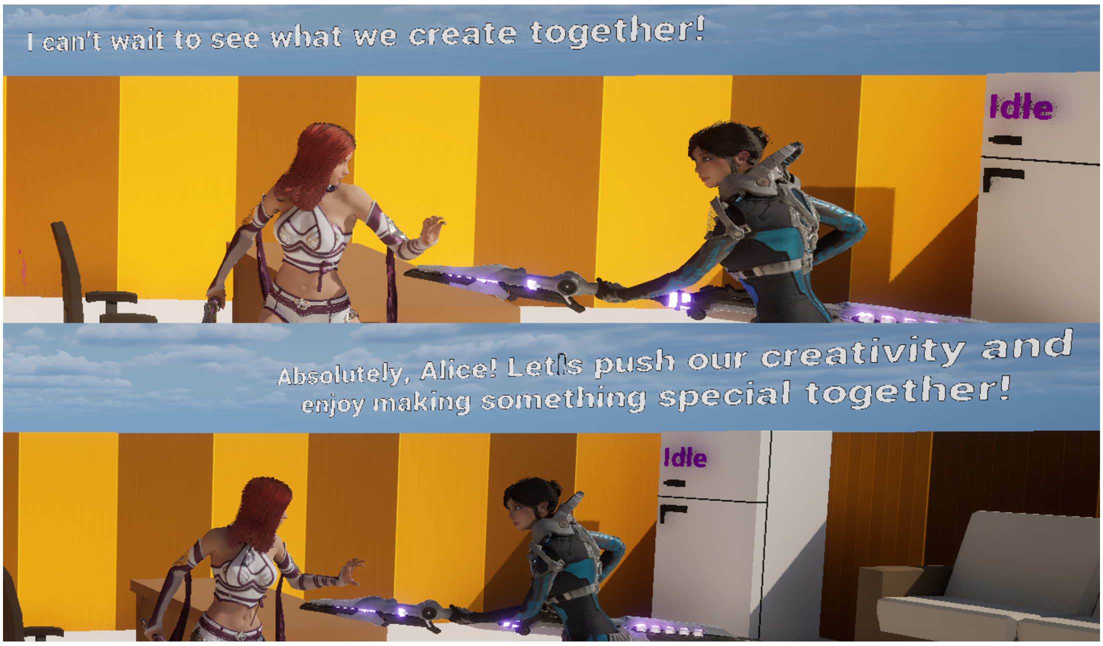


Figure 2.4: Alice Wilson and Mary Oliver are conversing about their next art project.

conversation, saying, "Hey Alice! I see you're walking. Want to chat about our next art project?". Alice responds, "Hey Mary! Yes, let's chat about our next art project!". They continue a detailed discussion about their project, sharing opinions and ideas as shown in Figure 2.4. Following their dialogue, both Alice and Mary save the dialogue summaries to their memory stream and revise their future plans. Alice updates her schedule to study with Mary from 1:00 PM to 3:30 PM. Mary also adjusts her plans to finalise project details with Alice and prepare materials for their next steps from 2:00 PM to 3:30 PM on the same day.

Around evening, Alice returns home from school and works on creative writing and painting projects at her desk, building on the ideas discussed with Mary. Later, she has a late dinner in the kitchen. While eating, her brother Ethan Wilson arrives home and joins her for the meal. Ethan and Alice have a short conversation about today's events. Ethan talks about how he managed the grocery store and experimented with new recipes. After dinner, Alice goes to her room, initially intending to sleep. However, upon seeing her bookshelf, she decides to read a book and have some light snacks to relax before bed. This extends her evening, and Alice finally goes to sleep around midnight.

2.3.2 Limitations and Potential Solution

The new 3D architecture, while promising, exhibited several limitations. In total, three limitations are discussed below. In short, those issues are related to inappropriate responses made by the LLM, and making additional response validation prompts to the LLM can potentially mitigate those issues. The validation prompting is an effective temporary fix for common issues, but it also increases the total LLM usage cost.

2.3.2.1 Limitations

Name	Utterance
James Brown	Hey Amelia, I see you're looking for some books. Want to browse together for a bit?
Amelia Brown	Hey James, I'd love to browse with you, but I have to facilitate the group activity first.
James Brown	No problem, Amelia! Let me know when you're done, and we can browse then.
Amelia Brown	Thanks for understanding, James! I'll join you as soon as I finish.
James Brown	Sounds good! I'll be here when you're ready.
Amelia Brown	I really appreciate your patience, James. I'll wrap this up and be right there!
James Brown	Sure, take your time! I'll be here looking forward to our browsing session.
Amelia Brown	Thanks, James! I'll be quick, I promise Can't wait to browse together!

Table 2.5: The first 8 utterances in a dialogue between James Brown and Amelia Brown.

First, agents tend to make repetitive dialogue. Very frequently, they repeat similar utterances, extending conversations beyond natural endpoints. Table 2.5 illustrates an example of this phenomenon. In a natural human interaction the conversation is likely to end after the initial 4 utterances. However, the dialogue continues with repetitive farewells until the dialogue limit of 20 utterances, which is set to mitigate this issue. At each utterance, agents can either continue the dialogue or end the conversation, but they rarely choose the latter, likely to maintain politeness. Park et al. also observed that agents tend to be overly formal and cooperative, possibly due to instruction tuning implemented by LLM developers [15]. Commercial LLMs are designed to be helpful and responsive to user prompts, and those constraints can lead to unrealistic behaviours for agents, without fully reflecting their characteristics.

Second, agents often choose less sensible locations when creating plans. For instance, an agent made a plan to cook lunch at a grey dustbin in the kitchen. Even though the dustbin is likely to be involved in the cooking process, it is not ideal to choose the dustbin as a primary place to perform the cooking plan. Similarly, Park et al. also reported that agents often make erratic behaviours, such as entering a store after it is already closed [15].

Thid, agents often exhibit actions that are logically valid in a general context, but may be inappropriate or impossible within their simulated environment. For example, an agent attempted to call a non-existent fire department in response to a burning chair.

This phenomenon differs from typical LLM hallucinations, where LLMs generate false or fabricated information that are not based on their training data or given context [2]. The behaviours made by agents are reasonable, but are not available only in the simulation context.

2.3.2.2 Potential Solution

Prompt	[Four initial utterances from Table 2.5] Given the above dialogue history between James Brown and Amelia Brown, decide if the dialogue does not need to continue further and it is natural to finish here. Answer in the following JSON format. {“Yes” or “No”: <concise reasoning>}
Response	{“Yes”: “The dialogue concludes naturally as both parties acknowledge the need to wait, and Amelia expresses her intention to join later.”}

Table 2.6: Dialogue completion detection prompt applied to the Table 2.5 example

One potential solution for those issues is to validate responses made by the LLM. When prompts are too complex, the LLM often makes inappropriate responses. Those responses are in the right format, but their contents are problematic. There are common errors made by the LLM, and they can be captured by making additional validation prompts. Those validation prompts are niche filters designed to detect specific errors. For example, Table 2.6 shows an example dialogue completion detection prompt. It checks if the given dialogue (from Table 2.5) should finish or continue. The prompt was tested 30 times, and in all cases the LLM indicated that the dialogue should finish after the initial four or six utterances.

Similar approach can also be applied to the second and third issues. For instance, to mitigate the second issue of inappropriate plan locations, a validation prompt could be used to assess whether a chosen plan location and description make sense in a general context. This approach can help identify and filter out occasional nonsensical plans, such as cooking lunch at a dustbin.

While these additional validation prompts can be effective, it is important to note that they increase the computational resources required and the number of prompts used. Consequently, addressing all common issues with these additional validations prompting is not practical as a long-term solution. They should be considered temporary measures, implemented before more fundamental fixes such as prompt template revisions or logic refactoring can be developed and applied.

Chapter 3

LLM Caching

3.1 Motivation

Many state-of-the-art Large Language Models (LLMs) are not freely available. While the price per individual prompt may seem modest, it can accumulate quickly. To address this cost issue, researchers are actively exploring LLM caching across various domains. The concept is straightforward: storing previous prompt-response pairs for future use. The caching can be highly effective when applied correctly, especially in scenarios where prompts have consistent templates and similar contents [1].

This cost reduction potential is particularly relevant for generative agent architectures. In the original research by Park et al., simulating 25 agents for just two simulation days incurred thousands of dollars in costs [15], which is prohibitive for many applications. This problem gets more severe as simulations grow larger and dense, with more actors and agents. For example, agents generate reaction prompts whenever they observe another actor or agent, naturally leading to more prompts in densely populated simulations. While temporary measures like limiting agent reactions to once every 30 minutes can be implemented, they may compromise the believability of agent behaviours. For instance, an agent who recently changed plans may not respond to critical observations, such as a house fire, for the next 30 minutes.

Given these challenges, LLM caching presents a promising solution to the high prompting cost of the existing generative agent architecture, while keeping the believability as much as possible. Several factors suggest that LLM caching can work well for the generative agent architecture. First, the simulation environment is fixed. The locations and characteristics of actors in the simulation remain constant; no new actors are introduced, and existing ones do not disappear. This stability means the domain of possible agent activities is roughly upper bounded, as agents interact with the same set of actors consistently. Agents perform similar actions within a limited scope over extended periods, likely generating similar prompts repeatedly over time.

Second, the agent profiles are roughly consistent in the simulation, contributing to the repetitive nature of their activities. The agents' seed information and personality traits remain the same. This includes their initial background and core personality characteristics. While agents can build new relationships and form new memories based on their experiences, these additions are heavily influenced by their unchanging seed information. This consistency makes agents more likely to respond similarly over time, further increasing the likelihood of generating similar prompts and the probability of finding relevant cached responses for future interactions.

Lastly, the prompts and responses in the generative agent architecture follow specific templates. While there are multiple prompt categories, prompts within the same category share identical structures. They have predefined content in consistent locations. Similarly, prompt responses are written in explicit return templates. These structured characteristics make it easier to parse and revise retrieved cached responses, enhancing the effectiveness of the caching system.

3.2 Background

Large Language Model (LLM) caching has recently gained attention as a potential solution to mitigate the computational demands and usage costs. Among the various caching strategies explored, the embedding-based table approach is one of the most straightforward yet powerful methods. This method operates on a simple principle: when a user makes a query prompt, it is converted into an embedding. The system then calculates similarity scores between the query embedding and those of previously saved prompts in the database. While cosine similarity is commonly used, other similarity score options are available for different domains.

The GPTCache system by Fu et al. [1], exemplifies this approach. They initially created a diverse dataset from randomly scrapped web contents. Using this scraped data, they employed ChatGPT [10] to generate sentence pairs with specific semantic relationships: some pairs were semantically similar, others were exact opposites, and some were completely unrelated. These sentence pairs were then added to a caching dataset. The objective was to retrieve only similar sentences given a new input sentence. In their experiments, the system successfully retrieved relevant sentences using a similarity score threshold of 0.7. It demonstrated particular effectiveness for repetitive and similar prompts. The response times were 2-10 times faster when a cache hit occurred, even under simulated network fluctuations. When tested with various embedding models, the best hit ratio achieved was nearly 90%.

As such, an embedding based approach offers several advantages. It is lightweight and easy to implement, providing fast response times which are crucial for real time interactions. The system is also scalable, capable of handling numerous requests in large simulations. It is also cost effective, as embedding models are typically cheaper than full LLMs. These advantages allow users to build custom caching tables easily across different domains. However, the embedding caching's performance heavily relies on the embedding model used, and there is a limited potential for active improvement of the caching system itself without directly modifying the embedding model.

To address these limitations, Ramírez et al. proposed a more powerful alternative: the neural caching method [17]. The idea is to actively train a simpler, lightweight local LLM (student LLM) to emulate the responses of a superior LLM. The research focused on classification tasks, such as fact checking (True or False classification) or emotion classification. When the student model receives a prompt, it should decide whether to answer independently or defer to the superior LLM. The authors explored various selection methods, with marginal sampling[1] and query by committee[2] emerging as the most effective. Over time, the student model was retrained periodically on the responses made from the superior LLM. In this process, the student model became capable of handling more user requests independently.

While the neural caching method shows promise in overcoming some limitations of the embedding table approach, it may not be the optimal choice for generative agent architectures at present. This seemingly counterintuitive preference for a simpler model can be justified by several key factors. First, neural caching was primarily developed for classification tasks with fixed labels, which does not align well with the open-ended nature of generative agent prompts. The selection methods that proved effective for classification tasks, such as marginal sampling and query by committee, face challenges with prompts generated by agents. For instance, marginal sampling works well for classification tasks with a limited set of labels. But it may struggle with the generative

[1]Marginal Sampling chooses samples with high margin between the prediction probabilities of top two classification labels made by the student model [18].

[2]Query by Committee selects instances by the disagreement among the committee, which consists of 4 previous student models and the current student model. The level of disagreement is measured by calculating the fraction of committee members whose predictions differ from those of the current student model [19].

agent prompts as their very long, open-ended response probabilities inherently have higher uncertainties, which makes it difficult to reliably use probability differences for decision making. Similarly, query by committee depends on model prediction disagreements, which is difficult to quantify for open-ended prompts. A superior LLM can be employed to make similarity (agreement) judgement between responses, but this leads to additional costs.

Second, Generative agent systems demand rapid responses for believable interactions, but processing the high volume of prompts through a local student LLM can introduce significant delays. This issue becomes even more severe in larger, more realistic simulations with 20-30 agents and complex environments, where prompt volume could easily overwhelm local processing. The problem is particularly critical for users with less powerful machines or resource-limited platforms like gaming consoles. The neural caching system is also more complex to build and maintain, especially on devices that do not support modern machine learning libraries. Using external computing services for the student LLM can mitigate this, but it introduces additional costs and potential latency issues that might outweigh the benefits compared to no caching.

Given these challenges, the embedding based lookup table caching currently offers a more practical solution for generative agent architecture. Speed and responsiveness are critical for generative agents, and the embedding table excels at these aspects compared to the neural caching.

3.3 Prompt Dataset Investigation and Pre-processing

Group name	Prompt categories
Importance score	Importance score
Plan making and revision	Initial plan generation, Plan decomposition, Future plan revision
Finding location	Finding location
Actor reaction	Actor reaction
Memory reflection	High-level questions, High-level insights
Memory summary	Memory summary
New actor statuses	New actor statuses
Dialogue (agent reaction)	Agent reaction, Relationship summary, Dialogue utterance, Dialogue summary

Table 3.1: Prompt category groups and their members.

The prompt datasets for future caching model experiments were collected by running the new 3D architecture for five times. Each simulation used identical initial settings, as outlined in the section 2.3. The datasets were trimmed at 6 hours, which is equal to 15 days of simulation days with the time multiplier of 60. There are 14 different prompt categories in the datasets, with their examples shown in the appendices A.2 through A.15. For analytical purposes, these categories were consolidated into 8 groups based on semantic similarities, as shown in Table 3.1.

As a preliminary analysis, the prompt datasets are investigated in the following three sections, while the first dataset is chosen for visualisation. The first section explains the architecture often resubmits the same prompt if the LLM failed to make a valid response, and those duplicated responses are removed from the datasets. The second section explores the total length variations of prompt categories. The final section provides justification for not applying caching to dialogue related prompt categories, and presents the final pre-processed datasets with their respective counts.

3.3.1 Prompt Resubmission and Dynamic Model Selection

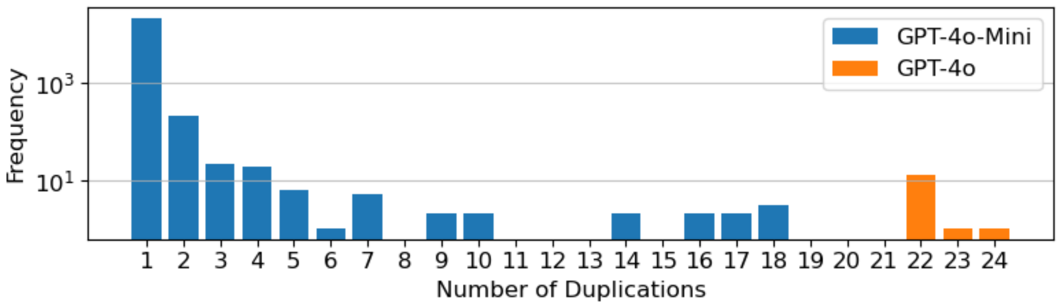


Figure 3.1: Distribution of prompt duplication counts, from the first simulation prompts.

Occasionally, the LLM struggles to make valid responses when the given prompt is too complex. For example, the LLM often does not choose one of the given options and responses in the wrong format. In such cases, the architecture resubmits the same prompt until a valid response is received. Figure 3.1 illustrates the resulting prompt duplication count distribution from the first dataset. Approximately 3.79% of prompts required resubmission due to initial unsuccessful attempts.

In some instances, repeated attempts with the same prompt also failed to make a valid response, with the LLM consistently making the same error. This was a significant problem for the architecture, as certain prompts were attempted hundreds of times without success. To address this issue, a dynamic model selection was implemented. Each prompt is initially sent to the default model, GPT-4o-mini [12], up to 20 times. If all attempts fail to produce a valid response, the prompt is then directed to the superior LLM, GPT-4o [19]. As evident in Figure 3.1, the superior model successfully generated valid responses for all prompts that the default model struggled to answer. Approximately 1.47% of prompts were ultimately sent to the superior model. The advanced language model successfully tackled these challenging prompts in no more than four attempts, with most being resolved on the second try.

This approach uses the superior LLM only when needed, efficiently handling challenging prompts. Users can adjust the threshold for superior model use based on their needs. However, GPT-4o costs 27-33 times more than GPT-4o-mini (depending on versions), so lowering the threshold may resolve difficult prompts faster but increase costs. For example, reducing the threshold to 10 would route prompts resolved by the default model in 11-19 attempts to the superior model. In subsequent analysis, those duplicated prompts were removed, retaining only the final, valid prompts and their corresponding responses.

3.3.2 Total Length Variations of Prompt Categories

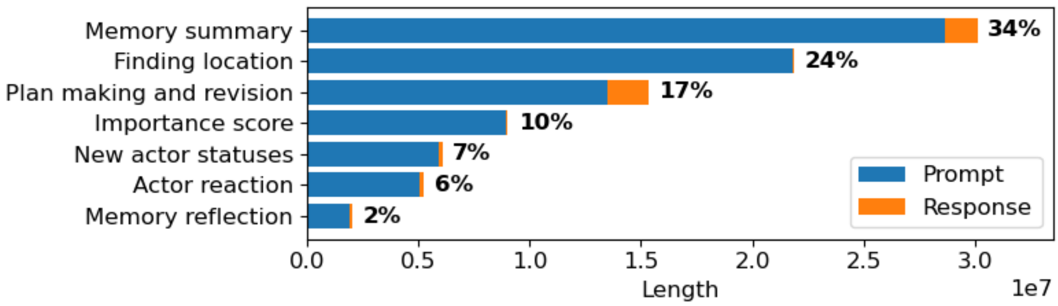


Figure 3.2: The total prompt and response lengths by category. The proportion of each category (prompt and response together) is shown next to each bar.

Naturally, some prompt categories occur more frequently and are longer than others. Figure 3.2 illustrates the differences in total prompt and response lengths, which is a useful proxy for estimating prompting costs. More complex prompts with rich background information are located at the top 3, accounting for 74% of the total prompting cost, while simpler prompts make up only a small fraction of the overall cost. Thus, when applying LLM caching, it seems logical to focus on the complex and cost intensive categories. But their intricate contextual information may make it difficult to retrieve and revise cached responses effectively. Interestingly, memory summary and plan making and revision categories exhibit relatively longer response lengths compared to others. This is logical as they involve open-ended questions requiring longer responses, while other prompt categories typically require short sentences or selection from given options.

3.3.3 Exclusion of Dialogue Related Prompt Categories

Dataset	Run 1	Run 2	Run 3	Run 4	Run 5
Prompt count	17201	23600	20994	18908	21715

Table 3.2: Prompt counts for each simulation run dataset after pre-processing

For upcoming caching model experiments, prompts in the dialogue group will be excluded. There are several factors for this decision. First, caching dialogues is inherently counterintuitive, as each utterance is usually unique. Applying caching to dialogues could lead to agents repeating identical utterances over time, potentially reducing overall simulation believability. Moreover, dialogue related prompts constitute only a small fraction (2%) of both prompt count and total length, suggesting not applying caching for this category is unlikely to meaningfully impact final prompting costs. Following all pre-processing steps, each of the five prompt datasets contains 20484 prompts on average. Individual prompt counts of all simulation run datasets are shown in Table 3.2.

3.4 Caching Model Design

The caching system uses an embedding[3] based lookup table to retrieve responses for the given query prompt. However, this approach faces two challenges below. The following sections will explore these issues and propose potential solutions.

- Category and agent mismatch: The system may retrieve responses from inappropriate categories or agents. For instance, a query about initial plan generation might retrieve a plan revision response, leading to format inconsistencies. Similarly, a response intended for one agent might be retrieved for another, potentially causing contextual errors.

- Specific requirements mismatch: Even when category and agent align, retrieved responses may not meet the unique requirements of the current prompt. For example, a cached response might cover evening hours (6pm-9pm) when the current query explicitly requests afternoon planning (1pm-3pm).

[3]OpenAI's text-embedding-3-small model is used for embeddings [13]. While a more powerful large model exists, its cost nearly matches that of GPT-4o-mini LLM, making it impractical for caching system.

3.4.1 Prompt Category and Agent Name Mismatch

Embedding lookup table construction uses all prompts from the dataset(s). Thus, the query and retrieved prompts may have different categories. For example, a plan decomposition by Mary Oliver retrieved 100 cached prompts, with only 32 belonging to the same category. Prompts from different categories are not compatible, as they have different contents and answer formats. To address this, the caching lookup table is split by prompt categories, ensuring that query prompts only find their cached responses from the lookup table within the same category.

Similarly, query prompts can also retrieve cached responses from different agents. This is likely to be problematic for complex prompts, such as plan making. In one case, an initial plan generation prompt by Alice Wilson retrieved a response from her brother, Ethan Wilson. Ethan works at the Fresh Grocery Store, and his plans include tasks such as organising shelves or clearing the store. If Alice follows those plans, she would go to work at the store, even though she is not an employee. To prevent such inconsistencies, the lookup table is further split by agent names. Those splits may result in more sparser lookup tables, but this can be mitigated by gathering more data over time.

3.4.2 Response Modification

Modification needed	Prompt categories
Read (no modification)	Importance score, High-level questions, High-level insights, Memory summary, New actor statuses
Choose	Finding location
Revise (by the LLM)	Initial plan generation, Plan decomposition, Future plan revision, Actor reaction

Table 3.3: Modifications needed for different prompt categories.

With the table splits, the query prompt and retrieved response have the same category and agent name. But the retrieved response often needs additional modifications due to requirements mismatch. Plan making prompts, for instance, require the LLM to create plans within a specific time period. In most cases, the time period of the query prompt differs from those in the cached response. Table 3.3 outlines the different types of modifications needed for various prompt categories.

Some prompt categories, such as importance score or memory summary, have no special requirements and their responses do not need any modification. On the other hand, finding location prompts usually have mismatches in the location choices. For instance, a query prompt may offer locations A and B, while the retrieved response suggests location C. To resolve this discrepancy, the system employs a selection process based on cosine similarity. First, the system calculates embeddings for all location options in both the query prompt and the retrieved response. Second, the system computes cosine similarity scores between each query option embedding and the retrieved option embedding. Third, the query prompt option with the highest cosine similarity is selected.

The remaining four categories have more complex requirements, thus they need a revision by the LLM. A revision prompt contains the cached response and the query prompt requirements. The LLM then revises the cached response to meet the given requirements. For example, plan making prompts ask the LLM to rewrite the timestamps and plan locations of the cached response. At the same time, the LLM also removes unnecessary plans from the cached response. On the other hand, the actor reaction prompts have a slightly different revision process. These prompts typically offer four options, but the availability of options can vary based on context. For example, some have option A, while it is not available in others. Thus, the LLM selects the most appropriate option from the query prompts, and fills missing information, considering the retrieved option. The revision prompt examples are shown in the appendices D.1 through D.3.

Although these revision processes involve prompting to the LLM, it remains cost effective as the revision prompt does not have background information, such as agent summary. This makes the revision prompts much shorter than full prompts.

3.5 Experiment Design and Results

Name: Liam Oliver (age: 22, gender: Male) Innate traits: Charming, Meticulous, Friendly Liam Oliver works at the Singapore Coffee House, [...] His sister Mary is a frequent visitor to his cafe, often bringing friends from school to enjoy Liam's latest coffee creations. On the scale of 1 to 10, where 1 is purely mundane (e.g., brushing teeth, making bed) and 10 is extremely poignant (e.g., a break up, college acceptance), rate the likely poignancy of the following piece of memory in Liam Oliver's perspective. Memory: Tree in Victoria Park is Idle Answer in the following JSON format. {"Rating": <Fill in>}

Table 3.4: Example importance score prompt with the three different components written in different colours. Blue: Agent summary, Red: Template, Green: Keyword

The caching model for the generative agent architecture has been previously explained, assuming full prompts are used for building embeddings. Theoretically, some contents in full prompts may be less relevant, but finding those contents can be challenging. However, prompts made by generative agents differ in structure. Table 3.4 illustrates an example importance score prompt, consisting of three distinct components. Prompts in the same category share the same template. Also, agent summary is roughly consistent for prompts made by one agent, as their starting seed information remains constant.

Thus, the primary semantic differences between prompts are from specific keywords, which take only a very small proportion. This observation suggests that keyword only caching may be viable and potentially more efficient compared to full prompt caching. It is also possible that the embedding model may struggle to create good distinguishable representations of full prompts due to their minimal differences. To prove the hypothesis regarding the equivalence or superiority of keyword only caching over full prompt caching, both models will be built and investigated from various aspects in subsequent sections. All experiments will be conducted on both models to determine if they exhibit comparable behaviour and demonstrate similar performances. At the end, their caching performance will be directly compared.

For keyword only caching, embeddings are created using the keyword description, which is formed by concatenating the keywords with spaces between them. For instance, if there are keywords A, B, and C, the final keyword description would be “A B C”. This description is then used to generate its embedding. This process is applied not only to query prompts but also to all prompts in the caching table.

In all subsequent discussions, “similarity score” refers to the cosine similarity score, unless otherwise specified. These cosine similarity scores have been min-max normalised to a range of [0, 1] from their original range of [-1, 1].

3.5.1 Correlation Between Cosine Similarity and Response Quality

The caching system assumes that similarity scores calculated between the embeddings closely correlate with their semantic similarities. In other words, when cached responses are retrieved for the given query prompt, it is assumed that retrieved responses with higher similarity scores should be better compared to the ones with lower similarity scores. This fundamental hypothesis needs verification in this research’s context, before proceeding into further experiments. If there is no correlation between the cached response quality and its similarity score, the caching model would not work as intended.

However, evaluating response quality is not a trivial task. Ideally, human evaluators should rank candidate retrieved responses with various similarity scores, from the most to least likely given the query prompt. Ideally, responses with higher similarity scores should be ranked at the top. However, due to the time constraints and potential difficulties in finding sufficient numbers of participants, this approach is impractical for this research. Instead, this project employs a superior LLM as an evaluator to rank retrieved responses. Lianmin et al. [20] demonstrated that superior LLM judges can be employed for evaluating answers of various open ended questions. The superior LLM evaluations aligned well with both controlled and crowdsourced human preferences. For the following experiments, GPT-4o-mini [12] is the candidate LLM for revising cached responses, while GPT-4o [19] serves as the superior LLM for evaluations.

The following experiments start by analysing similarity score distributions for both caching models, as a preliminary investigation. Each model's score distribution is then divided into four ranges. Next, query prompts retrieve responses from each range, giving four candidate responses per prompt. Finally, the superior LLM ranks these candidates along with the true response, considering the original query prompt.

3.5.1.1 Similarity Score Distributions

Full caching and keyword only caching build embeddings differently, and this may give different similarity score ranges. To investigate this, 700 query prompts were randomly sampled from the first simulation run dataset, 100 for each prompt category group. These 700 query prompts individually retrieved all available responses from the lookup table built by the second run dataset. In each retrieval, the similarity scores between the query prompt and all available prompts were calculated, and their average was stored separately. This process gave 700 average similarity scores, for both caching models. The results are shown in Figure 3.3.

The average similarity scores in Figure 3.3 show significantly different distributions. Similarity scores of the full prompt caching cluster densely in the range [0.9, 1], while the keyword only caching has scores more evenly distributed in [0.6, 1]. In other words, the shared components made the full prompt embeddings very similar to each other, which led to their similarity scores increased greatly. On the other hand, the keyword only embeddings are more spread up in the embedding space. Interestingly, the two histograms also differ in shape. The full prompt caching shows only one peak, while the keyword only caching has two peaks.

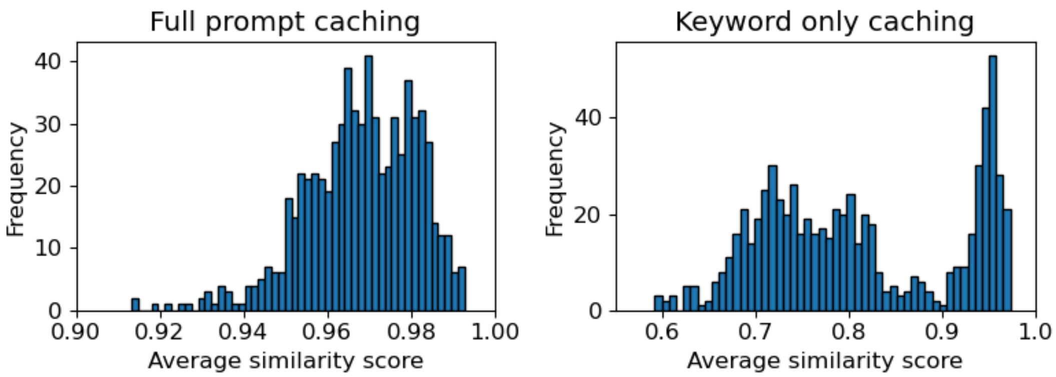


Figure 3.3: Average cosine similarity scores for both caching models.

3.5.1.2 Correlation Verification Experiment Design

Simplified prompt	Isabella Smith organises a community activity on promoting local flowers on Flower Shelf. What is the new statuses of Flower Shelf during and after the action?			
Score range	(0.5, 0.625]	(0.625, 0.75]	(0.75, 0.875]	(0.875, 1]
During action status	Being Restocked	Reorganizing Flowers	Occupied by Isabella	Active Discussion

Table 3.5: Candidate responses retrieved from different similarity score regions, for the importance score prompt. After action status is "Idle" for all responses.

Given the findings of the previous section, the main experiment to verify the correlation between the similarity scores and superior LLM evaluations were conducted in the following process. First, the similarity score ranges of each caching model were split evenly into 4 regions. For the full prompt model, these regions were (0.9, 0.925], (0.925, 0.95], (0.95, 0.975], and (0.975, 1]. Similarly, the keyword only model had regions (0.5, 0.625], (0.625, 0.75], (0.75, 0.875], and (0.875, 1]. Second, for the 700 query prompts from the previous section, best cached responses were retrieved and revised individually again from the second dataset tables, one from each region. For example, one response was retrieved from the first range (the best score response in the range), and another from the next range, and so forth. Table 3.5 shows the candidate responses from different regions, for the important score prompt. In total, the 700 query prompts retrieved 2800 candidate responses from different similarity score regions for both caching models.

Those candidate responses were then sent to the superior LLM (GPT-4o) with the true responses (the actual responses made for the query prompts) for ranking. The ranking prompt contains the original query prompt, 4 candidate responses from the different similarity score regions and the true response. The responses were shuffled randomly to minimise potential ordering bias. The superior LLM ranked the responses from most to least likely, considering the query prompt. If no response existed in a region, then the region received the lowest rank. The ranking prompt example is attached to the appendix D.4.

The ranking data was used to calculate the Trueskill rating [6] of each similarity score range and the true response. Trueskill is a generalisation of the Elo chess rating system [4] for multiplayer environments. Trueskill processes the ranked outcomes and generates a mean rating value and standard deviation for each participant, from the initial parameters[4]. In the context of this experiment, if the superior LLM consistently ranks responses from a particular similarity score range higher, that range will receive a higher Trueskill rating.

3.5.1.3 Correlation Verification Experiment Results

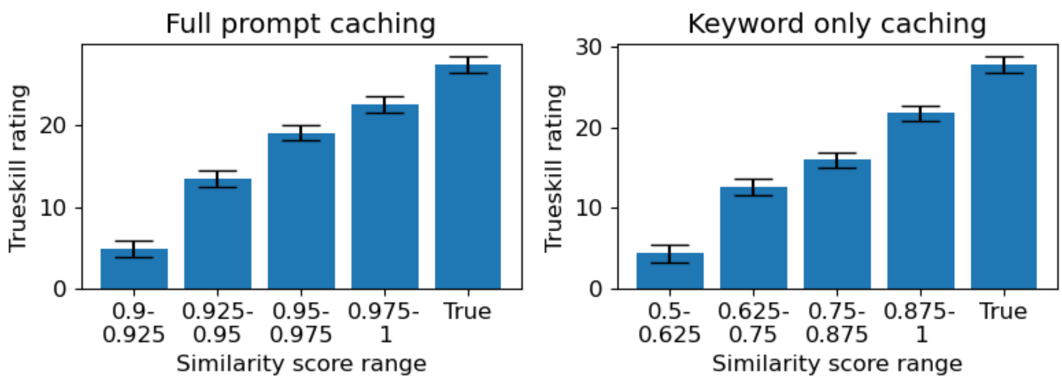


Figure 3.4: Ratings of similarity score ranges and true responses across 7 prompt groups.

Figure 3.4 displays the final ratings for each model. Both models show a strong correlation between similarity scores and quality rankings made by the superior LLM. Cached responses with lower similarity scores have lower rankings, and vice versa. This trend clearly holds for both models, which provides strong evidence that embedding based caching tables can effectively retrieve quality prompts and responses for new query prompts. In other words, if a retrieved cached response has a higher similarity score, then the response is very likely better than the others. Notably, responses with the highest similarity scores still have lower ratings compared to true responses. This indicates the fundamental difference between the best simple embedding lookup table results and true responses.

[4]Trueskill has four key parameters: mu (initial mean skill), sigma (initial standard deviation), beta (performance deviation from skill), and tau (skill change over time). While mu, sigma, and tau use default values (25, 8.33, and 0.08333), beta is doubled to 8.33 for the experiments. This adjustment accounts for potential caching performance variation due to a diverse domain of query prompts.

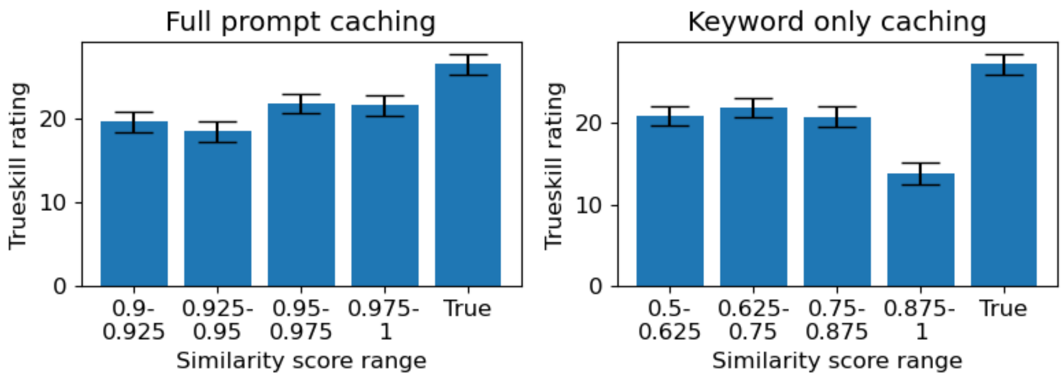


Figure 3.5: Ratings of similarity score ranges and true responses for the finding location group.

This correlation is consistent across the 5 out of 7 prompt category groups for both models, but the two remaining category groups (importance score and finding location) show relatively weaker correlations. Figure 3.5 shows the ratings for the finding location category. For full prompt caching, the similarity score ranges have roughly comparable ratings. In other words, rating differences between similarity ranges are not significant. Keyword only caching shows similar trends, but the last quartile responses have a lot lower ratings than other quartiles with lower similarity scores. In short, the correlation is mostly consistent for all prompt category groups, but there are exceptions. Trueskill ratings for the remaining 6 category groups (In addition to Figure 3.5) are provided in the appendices E.1 through E.6.

3.5.2 Caching Performance Saturation

The previous section demonstrated that there is a strong correlation between the cosine similarity scores of cached responses and their quality evaluations by the superior LLM. In other words, if a caching model constantly retrieves higher similarity score responses, then the model is highly likely performing better than others. However, how can we guarantee a caching model is able to find cached responses with higher similarity scores for potentially all future query prompts?

Intuitively, a larger lookup table with more diverse prompts would have a higher chance of finding better responses for query prompts. However, naively making a very large caching table is problematic as they require more storage space. The retrieval process is also slowed down as the query prompts need to calculate similarity scores with all matching prompts in the much bigger caching table. Additionally, gathering a large amount of data requires running the simulation for a very long time, causing significant prompting cost and computational resource usage.

Thus, it is necessary to find a good balance point for the lookup table size (time period of gathering data). The lookup table should be large enough to cover the majority of potential prompts made by agents in the future, but also small enough to remain compact for storage space and retrieval efficiency. But what is the balanced amount of data that needs to be gathered here? In other words, how can we find the point where we should stop gathering the prompt data?

As agents are likely to make similar prompts over time, a caching model's performance is likely to saturate eventually. Initially, the caching performance will continuously increase while gathering more data. However, after the caching model becomes large enough to cover the majority of prompts made by agents, the caching performance will not increase meaningfully. This hypothetical saturation point would be the perfect balance point in data collection. The following experiments will verify the existence of saturation points, and investigate the amount of data needed to reach caching performance saturation.

3.5.2.1 Saturation Experiment with Individual Datasets

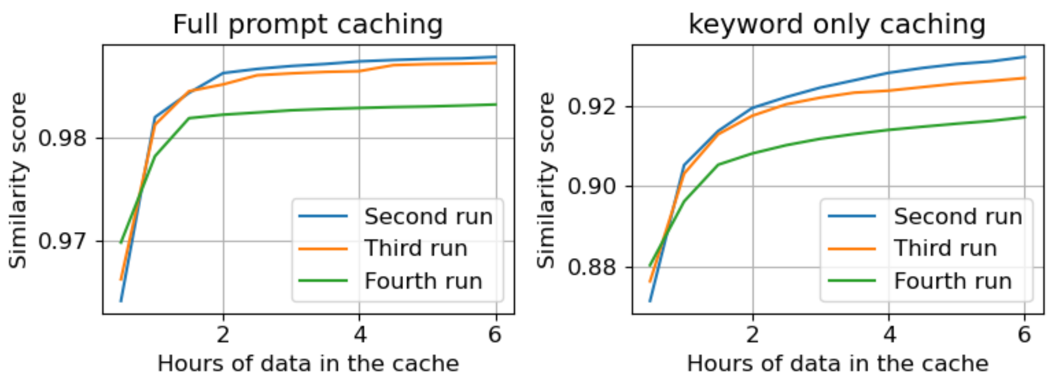


Figure 3.6: Effect of data volume on best retrieval similarity scores across all three datasets.

The first experiment aims to prove the saturation points indeed exist, and determine the amount of data to reach saturation points. The experiment began by preparing all prompts from the first run dataset, totaling 17201, as query prompts. Three separate lookup tables were also prepared using data from the second, third, and fourth simulation runs. The first run query prompts were then used to retrieve the best cached responses from each of the three lookup tables individually, with different time thresholds. Initially, cached responses were retrieved only from the first 30 minutes of prompts in the lookup tables. This process was then repeated with increasing time thresholds: 1 hour, 1 hour and 30 minutes, and so on, in 30-minute increments until all 6 hours of data were used.

Figure 3.6 displays the retrieval score trends across all three datasets, showing only the averages of the 17201 query prompts for each time threshold. Both models initially show lower retrieval similarity scores, but they increase rapidly over the next few hours and reach the saturation points. It is clear that continued data collection beyond the saturation points yields diminishing returns. At the same time, the standard deviations of similarity scores also decrease rapidly, as shown in the individual dataset results in the appendix F.1 through F.3.

Each dataset has slightly different maximum saturated similarity scores, but the amount of data needed to reach the saturation point is almost identical for each model, as shown in Figure 3.6. The full prompt caching reaches the saturation points with approximately 2 hours of data, while the keyword only caching needs roughly 5 hours of data. As the saturation points remain fairly consistent for each caching model, they can serve as reliable saturation point predictions for future data collections with the identical simulation settings.

3.5.2.2 Saturation Experiment with Combined Datasets

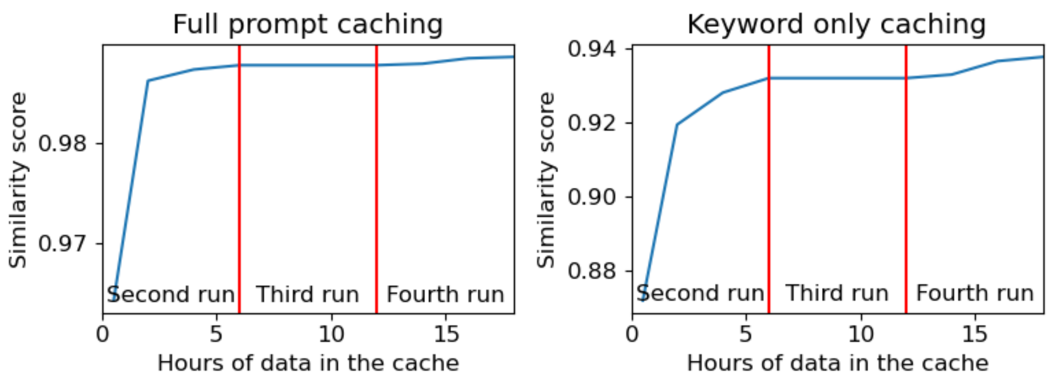


Figure 3.7: Effect of data volume on best retrieval similarity scores on the combined datasets.

The second experiment investigates caching performance saturation using combined datasets. Instead of using the three datasets individually, one large caching table was formed with all prompts from the three datasets. All other experimental details remained identical, except the time threshold was 2 hours instead of 30 minutes. The results, shown in Figure 3.7, clearly indicate that the second run dataset was sufficient for both models to reach saturation. The third run dataset did not improve caching performance at all, while the fourth run dataset provided a slight improvement. This outcome is likely because the second and third simulation have nearly identical scenarios, while the fourth run had some notable differences, despite overall scenario similarities.

3.5.3 Saturation Detection with Number of Observed Actors

The previous section revealed that saturation points are relatively consistent for each caching model. Full prompt caching requires approximately 2 hours of data, while keyword only caching needs 5 hours. However, it is uncertain whether these predictions will remain valid under different simulation scenarios. The saturation points may be reached faster or slower depending on the simulation settings. For example, if the current simulation time multiplier of 60 is reduced to 30, the simulation will run at half speed. Theoretically, this makes the caching models to reach saturation points with twice the amount of data. But these predictions are not verified, and it is impractical to make such predictions for all possible simulation settings by repeating the same experiments from the previous section. Therefore, a new method to predict the caching performance saturation is needed. This new approach should be able to predict the saturation point only with the information already available in the simulation, instead of gathering extra data or conducting additional experiments.

3.5.3.1 Saturation in Number of Observed Actors

Fundamentally, agents make similar prompts because they follow similar daily routines in the long run. Agents initially create many new plans but quickly start making very similar plans later, leading to similar prompts. Therefore, it may be unnecessary to directly compare prompts (as done in the previous section) to determine when caching performance saturates. Instead, measuring whether agents are following similar daily routines can be sufficient to detect the caching performance saturation. If agents begin to repeat almost identical daily activities, they are unlikely to generate new prompts afterward. Thus, it should be possible to build a caching table from the data gathered up to that point to handle future prompts, as they will likely be similar.

There are potential methods for detecting whether agents are making similar daily routines. One naive approach would be to ask the LLM to check if the plans and daily activities made by an agent are similar. However, this requires additional prompts, and it is not clear how to define similarity between daily activities. A more convenient and straightforward approach would be to use the number of actors observed by agents. At the simulation's start, agents are only aware of their seed structures (actors). As they explore the environment, they observe new actors and engage in new daily activities. For instance, constantly finding new activities and visiting new places would continuously increase the number of observed actors. On the other hand, if agents begin to perform similar activities in already known places, the number of observed actors will remain almost identical (saturation).

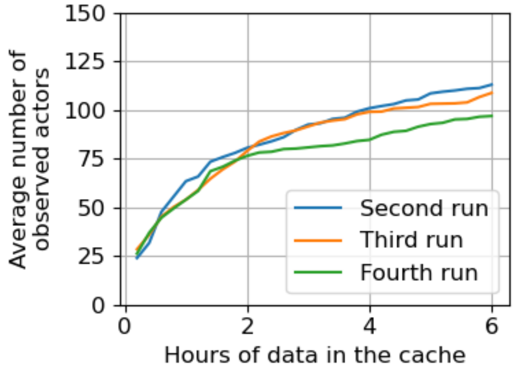


Figure 3.8: Average number of observed actors over all agents (There are 284 actors in total)

To prove saturation in number of observed actors, the average number of observed actors were calculated over all 8 agents. This calculation was performed using different time thresholds and done individually for three datasets (second, third, and fourth runs). The results shown in Figure 3.8 reveal a typical concave downward graph for the number of observed actors. Agents actively explore the map at the beginning, but exploration speed gradually slows down. After the initial 2 hours, agents discovered around 75 actors, with this number continuously decreasing. From the trends, it is evident that the agents eventually stop observing new actors completely, repeating almost identical daily routines.

3.5.3.2 Correlation with Caching Performance

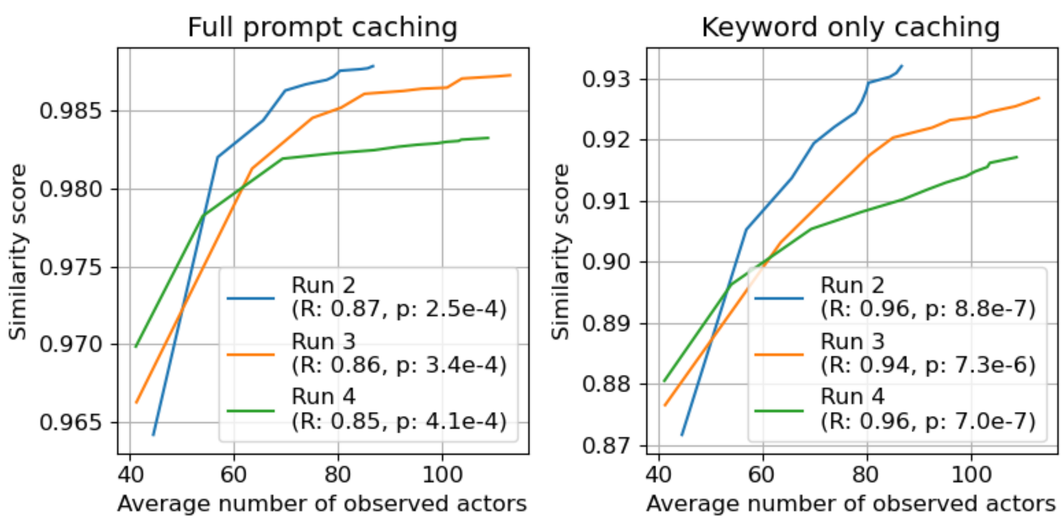


Figure 3.9: Correlation between best retrieval similarity score and average number of observed actors (R: Pearson correlation coefficient, p: p-value of the no correlation null hypothesis)

This section proves the strong correlation between the number of observed actors and caching performance. Figure 3.9 is created by plotting best retrieval similarity scores from Figure 3.6 and average numbers of observed actors from Figure 3.8 together, for both models. Pearson correlation coefficients were calculated for the graphs in Figure 3.8, averaging 0.861 for full prompt caching and 0.952 for keyword only caching, with very low p-values. These results firmly support the hypothesis, suggesting that the number of observed actors can be used to accurately predict the best retrieval similarity score. In short, when the average number of observed actors start to saturate, the caching performance also reaches the saturation point. This can be utilised to detect caching performance saturation points in data gathering, without additional experiments.

Interestingly, the graphs in Figure 3.9 also follow clear concave downward trends. The full prompt and keyword only caching models show smaller returns in retrieval score improvements as the number of observed actors increases. Once they reach the saturation points in the future, further actor observations will yield only marginal gains in best retrieval scores. This indicates achieving perfect caching (retrieval similarity score of 1) is impossible, regardless of the number of observed actors.

3.5.4 Direct Performance Comparison

The full prompt and keyword only caching models have been analysed in various aspects, and consistently shown similarity performances. These results support the hypothesis that keyword only caching is equivalent to full prompt caching. However, a direct comparison of those two models remains unexplored. While section 3.5.1 calculated Trueskill ratings for the best cached responses against true responses, these ratings were made within each model's context, not directly comparing the two models. Thus, those two models will be compared directly in the following sections, in terms of their caching performance and prompting cost reduction.

3.5.4.1 Caching Performance Comparison

The caching performance of both models were compared using the query prompts from the fifth simulation run dataset, which was not used in any previous experiments. More specifically, each of the seven prompt category groups randomly sampled 600 prompts from the fifth dataset, except for the memory reflection category as it had only 196 prompts. Second, caching lookup tables for both models were built using the first run dataset. This gave 3796 sampled query prompts in total. Third, each sampled query prompt retrieved the best response from both lookup tables. Additionally, the query prompt also sampled a random response from the fifth dataset, among the responses who have the same category and agent name. Fifth, these three responses were shuffled together with the true response of the query prompt, and presented to the superior LLM for ranking.

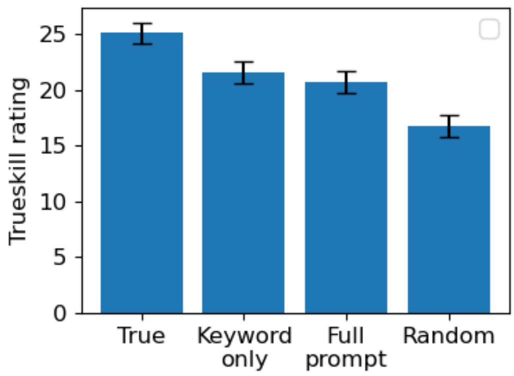


Figure 3.10: The overall ratings for both models, compared to true and random responses.

The LLM ranked the responses from most to least likely considering the query prompt, and Trueskill ratings were calculated from the rankings. An example rating prompt is attached to the appendix D.4. The overall ratings are presented in Figure 3.10, and they show ratings of both models fall between random and true responses with nearly identical scores. Their ratings fall within each other's standard deviations. This result clearly supports the comparable overall caching performances for both models.

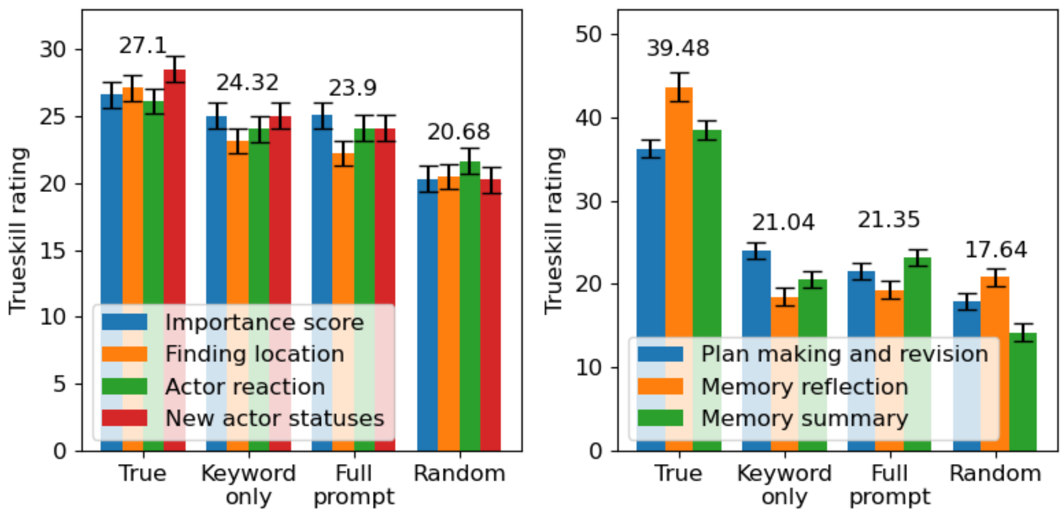


Figure 3.11: Ratings of both models in individual categories. Average ratings are above the bars.

Figure 3.11 displays Trueskill ratings for the 7 individual prompt category groups. For the four groups in the first subplot, both models showed roughly comparable performance to true responses. These groups typically require short, concise responses or selection from given options. In other words, they are more robust in information loss from the embedding conversion. On average, there is approximately a 11% rating difference between both models and the true responses for these four prompt groups. Similarly, there is roughly a 24% rating difference between the true and random responses.

However, the remaining three prompt category groups in the second subplot do not follow the same trend. For those groups, both caching models are not meaningfully better than the random responses, while the true responses achieve more than double the rating. This difference likely originates from two key reasons. First, those category groups have complex prompts with long contextual information, making it challenging to capture all nuances during the embedding conversion process. Second, generating responses for these complex prompts requires high-level reasoning, resulting in responses that are often unique to the original prompt. Consequently, retrieving appropriate cached responses from lookup tables becomes inherently difficult for these prompt groups.

3.5.4.2 Simulation Cost Comparison

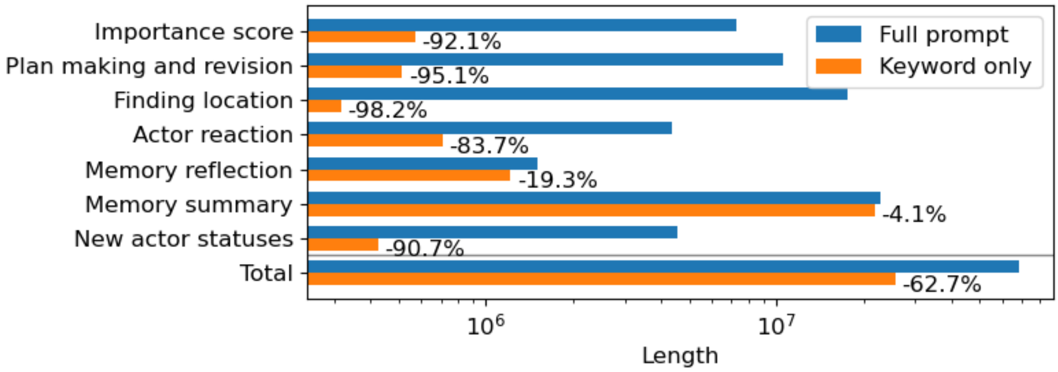


Figure 3.12: The embedding description length of both models for the first run dataset.

The next set of experiments analyses the caching models in terms of cost reduction, compared to direct prompting (no caching). There are two stages in the caching process: the initial building of the caching models and their subsequent usage. Notably, while the previous rating calculation experiment sampled only a portion of the prompts from the fifth dataset, the cost analysis uses all prompts from the fifth dataset, providing a more comprehensive evaluation. The subsequent cost estimations are based on OpenAI's token number approximation (4 characters per token [14]), as the actual tokenizers are currently not publicly available. For the models employed, text-embedding-3 is priced at \$0.02 per million tokens, while GPT-4o-mini has a two-tiered pricing of \$0.15 for input and \$0.6 for output, both per million tokens.

In the first stage, caching models were built from the first run dataset, and Figure 3.12 illustrates the embedding description length for both caching models in this stage. On average, the keyword only caching demonstrates significant 62.7% description length reduction. The 5 prompt category groups (except the memory reflection and summary) show an even higher 92% reduction. On the other hand, the memory reflection and summary groups exhibit relatively smaller reductions. It is because their prompts consist mostly of query memories (they are keywords), with templates and agent summaries occupying smaller proportions. The building cost of full prompt and keyword only caching models are estimated at \$0.34 and \$0.13, respectively.

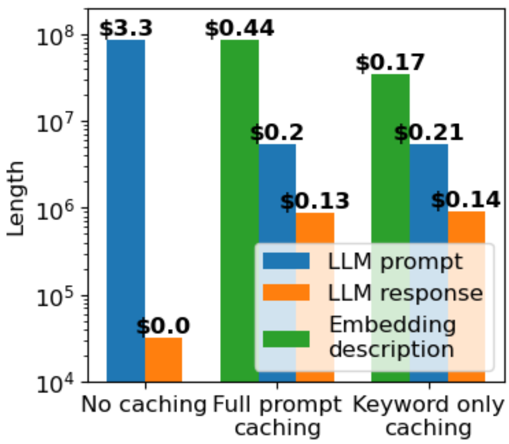


Figure 3.13: Usage cost of the caching models, compared to no caching.

The second stage is about the caching model usage cost comparison. From both caching models, all prompts from the fifth dataset retrieved and revised cached responses. The usage costs are shown in Figure 3.13, in comparison to no caching. Clearly, both models show significant cost reductions. Without caching, the estimated cost is approximately \$3.3 for 6 hours simulation. While this may seem modest, it increases significantly over time. For instance, running a simulation for 100 hours would cost around \$55.1. In contrast, full prompt caching reduces the 6 hours of simulation cost (first and second stage costs combined) to \$1.11, which is a 66.4% cost reduction. The keyword only caching further lowers the total cost to \$0.65, achieving an 80.3% reduction. Both models have nearly identical LLM and embedding model usage in the second stage, but the main difference is in the embedding descriptions used to convert the query prompts. Naturally, for the keyword only caching, the query prompts are converted to the embeddings only with their keywords.

As the simulation duration increases, the cost reduction percentages will increase further. This is because the caching response retrieval and revision (second stage) cost will dominate the overall cost, making the initial caching table building (first stage) cost negligible in comparison. When considering only the second stage, the reduction percentages become 76.6% for full prompt caching and 84.2% for keyword only caching. These figures represent the maximum cost savings that can be achieved as the simulation length approaches infinity.

Chapter 4

Conclusions

4.1 Summary

This research extends the generative agent architecture proposed by Park et al. [15] by introducing the new 3D implementation. The 3D architecture offers more robust simulation capabilities across various domains, addressing limitations of the original 2D system. The 3D architecture is expected to be more immersive and engaging for users, with more realistic graphics. Naturally, the new architecture supports various existing 3D environments, which makes it easier for developers to employ the generative agents for their applications. The new architecture also supports an intuitive interface for users to create custom simulations without in-depth knowledge of implementation logic. Users can make new environments or agent background settings effortlessly. Notably, the new architecture can be converted to 2D simulation only with minimal modifications, ensuring design flexibility.

Another major contribution of this research is the introduction of a lightweight and easily implementable caching system for the generative agent architecture. The caching system utilises cosine similarity scores calculated from prompt embeddings to efficiently retrieve appropriate cached responses. The caching system was thoroughly investigated in various aspects, which led to several important findings below.

- Cosine similarity scores between query prompts and retrieved prompts strongly correlate with superior LLM judgements on caching response appropriateness for most prompt category groups.

- Caching model performance tends to saturate over time, as agents settle into repetitive daily routines. Gathering more data after saturation points gives diminishing improvement in caching performance.

- A strong correlation exists between the number of observed actors and caching model performance. This allows to detect caching performance saturation by tracking the saturation of the number of observed actors.

- Keyword only caching demonstrates comparable performance to full prompt caching. For 6 hours of simulation, keyword only caching achieves an 80.3% cost reduction compared to no caching, surpassing the 66.4% cost reduction of full prompt caching.

- The embedding based caching models perform adequately for concise prompts, but show no better performance than random responses for complex prompts with rich contextual information.

Generally, this research improves and extends the existing generative agent architecture, by making the simulation itself more immersive, while cutting the cost of the simulation. This will make the generative agent system unleash its full potential in various applications, which can possibly revolutionise many industries with affordable generative agents. Also, the experiment results and findings from this research will help future development of more advanced caching techniques for the generative agents.

4.2 Limitations and Future Works

This research has made several contributions, but it also has limitations that present opportunities for future work. First, the new 3D architecture has higher computational requirements compared to the original 2D implementation. The use of Unreal Engine, while powerful, demands more resources to run. One potential mitigation is to implement various graphical options in future developments, allowing users to adjust the visual complexity based on their hardware capabilities.

Second, the new 3D architecture employs the same cognitive modules as the original research, and thus inherits some of its limitations. Sometimes, generative agents occasionally exhibit inappropriate or inconsistent actions. The suggested solution of additional behaviour validation prompts may help mitigate this issue, but it introduces extra prompting costs. Many existing issues are likely to be alleviated with the introduction of more powerful LLMs, but further efforts in logic refactoring and prompt engineering are still necessary to achieve more consistent and realistic agent behaviours.

Third, the caching experiments were conducted under consistent simulation settings, which can limit their generalisability. Future research should explore the dynamics of saturation points under various conditions, such as different time multipliers, larger maps with more actors or agents, and diverse environmental settings. While the dataset size needed for saturation may increase with more complex maps, it is possible that the tendency of agents to repeat daily routines might keep the required data volume relatively constant.

Fourth, the caching performance evaluations relied solely on superior LLMs as proxies for human judgement. Although previous research showed that superior LLMs can provide evaluations comparable to humans [20], it remains uncertain how users would perceive the application of caching techniques in an actual generative agent system. The current evaluations revealed a noticeable performance difference between true responses and cached responses, but it is unclear whether users will detect this difference during actual interactions within the simulation. Given that users typically do not follow an agent's daily routines for extended periods, especially in larger simulations with numerous agents, these performance discrepancies might not significantly impact the overall user experience. Thus, future work should include human evaluations of simulation believability.

Fifth, the simple embedding based caching system showed inconsistent performance across different prompt categories. For the three complex prompt categories, the caching model's performance was not meaningfully better than random responses. This suggests that it may be more reasonable to apply the embedding caching models only to the four categories where the caching models showed decent performance. Based on the analysis in Figure 7, these four categories account for approximately 53% of the total prompt length proportions. Consequently, applying caching only to these categories would roughly halve the efficiency (cost reduction effect) of the caching system.

One promising future work direction for mitigating this issue is to fine-tune embedding models. Zhu et al. [21] demonstrated that fine-tuning embedding models with labels generated by superior LLMs can significantly increase embedding caching performance in question answering tasks. In their experiments, the Area Under the Curve (AUG) improved from 0.51 to 0.81 after fine-tuning. A similar approach could be applied to fine-tune embedding models for generative agent prompts, potentially improving caching performance across all categories. However, this method would incur additional costs for generating labels with a superior LLM, necessitating careful cost benefit analysis and further experimentation.

Bibliography

[1] Fu Bang. GPTCache: An open-source semantic cache for LLM applications enabling faster answers and cost savings. In Liling Tan, Dmitrijs Milajevs, Geeticka Chauhan, Jeremy Gwinnup, and Elijah Rippeth, editors, *Proceedings of the 3rd Workshop for Natural Language Processing Open Source Software (NLP-OSS 2023)*, pages 212–218, Singapore, December 2023. Association for Computational Linguistics.

[2] Sumit Kumar Dam, Choong Seon Hong, Yu Qiao, and Chaoning Zhang. A complete survey on llm-based ai chatbots, 2024.

[3] Kevin Dill and Lockheed Martin. A game ai approach to autonomous control of virtual characters. In *Computer Science, Education*, 2011.

[4] A.E. Elo. *The USCF Rating System: Its Development, Theory, and Applications*. United States Chess Federation, 1966.

[5] Epic Games. New embedding models and api updates, 2018.

[6] Ralf Herbrich, Tom Minka, and Thore Graepel. Trueskill™: A bayesian skill rating system. In B. Schölkopf, J. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems*, volume 19. MIT Press, 2006.

[7] James D. Hollan, Edwin L. Hutchins, and Louis Weitzman. Steamer: An interactive inspectable simulation-based training system. *AI Magazine*, 5(2):15, Jun. 1984.

[8] John J. Horton. Large language models as simulated economic agents: What can we learn from homo silicus?, 2023.

[9] John Laird and Michael VanLent. Human-level ai's killer application: Interactive computer games. *AI Magazine*, 22(2):15, Jun. 2001.

[10] OpenAI. Introducing chatgpt, 2022.

[11] OpenAI. New and improved embedding model, 2022.

[12] OpenAI. Gpt-4o mini: advancing cost-efficient intelligence, 2024.

[13] OpenAI. New embedding models and api updates, 2024.

[14] OpenAI. What are tokens and how to count them?, 2024.

[15] Joon Sung Park, Joseph O'Brien, Carrie Jun Cai, Meredith Ringel Morris, Percy Liang, and Michael S. Bernstein. Generative agents: Interactive simulacra of human behavior. In *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology*, UIST '23, New York, NY, USA, 2023. Association for Computing Machinery.

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

[17] Guillem Ramírez, Matthias Lindemann, Alexandra Birch, and Ivan Titov. Cache & distil: Optimising api calls to large language models, 2023.

[18] Tobias Scheffer, Christian Decomain, and Stefan Wrobel. Active hidden markov models for information extraction. In Frank Hoffmann, David J. Hand, Niall Adams, Douglas Fisher, and Gabriela Guimaraes, editors, *Advances in Intelligent Data Analysis*, pages 309–318, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg.

[19] H. S. Seung, M. Opper, and H. Sompolinsky. Query by committee. In *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, COLT '92, page 287–294, New York, NY, USA, 1992. Association for Computing Machinery.

[20] Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging llm-as-a-judge with mt-bench and chatbot arena, 2023.

[21] Hanlin Zhu, Banghua Zhu, and Jiantao Jiao. Efficient prompt caching via embedding similarity, 2024.

Appendix A

Generative Agent Prompt Examples

The example prompts used by the new 3D generative agent architecture are provided below. They may contain grammatical errors primarily made by prompt formatting. Some prompts have been modified for improving readability and conciseness, with minor details omitted. Intentional missions are denoted by [Intentionally omitted].

A.1 Seed Information Generation

I am building a simulation with 8 agents interacting with each other. Using the following rough characteristics, design their background stories (or memories) in the given format for all 8 agents. The seed memory should contain their background story, occupation, daily routine, relationship with close agents, etc. There is only one worker for each building (or workplace, such as a cafe). You have to follow the occupations given below, but you can make up other details freely.

1. Wilson family (Brother and sister)
- Alice Wilson: High school student
- Ethan Wilson: Grocery store worker
2. Oliver family (Brother and sister)
- Mary Oliver: High school student
- Liam Oliver: Cafe worker
3. Brown family (Married couple)
- James Brown: High school teacher
- Amelia Brown: Housewife
4. Smith family (2 unmarried couple)

- Noah Smith: Writer
- Isabella Smith: N/A (quit the job recently)

Example format: Name: Alice Wilson (age: 17, gender: Female)
Innate traits: [Friendly, Outgoing, Hospitable]
Seed memory: Alice Wilson is a high school student who loves to play video games. She also likes to listen to contemporary music. She loves to drink coffee. She is very close to Mary Oliver, and often plays video games together.

A.2 Finding Location

Name: Alice Wilson (age: 17, gender: Female)
Innate traits: Friendly, Outgoing, Curious
Alice Wilson is a high school student passionate about technology and coding.
[Intentionally omitted]
She displays a proactive approach to balancing her interests and is eager to refine her skills while building strong connections.

On Monday 10 May 2010, Alice Wilson plans as follows:
- Practice basketball drills focusing on passing and movement without the ball to enhance teamwork during the upcoming tournament. at Student Chair from 12:00 to 13:30
- Conduct a team strategy session to review tactics and game plans, emphasizing roles and responsibilities on the court. at Student Chair from 13:30 to 14:30

[Intentionally omitted]
- Initiate a brainstorming session for coding projects at Student Desk from 12:04 to 12:49
- Study for exams with Mary, focusing on code-related topics. at Riverview High School from 12:49 to 14:49

Alice Wilson knows of the following areas: “1. Idle Teacher Chair”, “2. Idle Student Chair”, “3. Idle Student Chair”, “4. Engaged Discussion Student Chair”, “5. Idle Student Chair”, “6. Idle Student Chair”, “7. Idle Student Chair”, “8. Idle Teacher Desk”, “9. Idle Student Desk”, “10. Idle Student Desk”, “11. Active Collaborating Student Desk”, “12. Idle Student Desk”, “13. Idle Student Desk”, “14. Idle Student Desk”, “15. Idle Dustbin”. Alice Wilson is planning to Collaborate with classmates to

brainstorm ideas and divide tasks for the group coding assignment. Which area should Alice Wilson go to?
Answer only one of the given area numbers, in the following example JSON format.
{"Area number": <fill in>}

A.3 Importance Score

Name: Alice Wilson (age: 17, gender: Female)
Innate traits: Friendly, Outgoing, Curious
Alice Wilson is a high school student passionate about technology and coding.
[Intentionally omitted]
She displays a proactive approach to balancing her interests and is eager to refine her skills while building strong connections.

On the scale of 1 to 10, where 1 is purely mundane (e.g., brushing teeth, making bed) and 10 is extremely poignant (e.g., a break up, college acceptance), rate the likely poignancy of the following piece of memory in Alice Wilson's perspective.
Memory: Student Chair in Classroom B in Riverview High School is Idle
Answer in the following JSON format.
{"Rating": <fill in>}

A.4 Memory Summary

1. Alice Wilson is Use the Idle Brown Double Bed at Brown Double bed in Ethan Bedroom in Wilson Family House
2. Wilson Family House is Ordinary
[Intentionally omitted]
49. Student Chair in Library in Riverview High School is Idle
50. Student Chair in Library in Riverview High School is Idle

How would one describe Alice Wilson's feeling about the recent progress in life given the above statements? Answer in the following example JSON format.
{"Response": <summary>}
The <summary> must be short and concise (maximum a few sentences). Do not break

lines in the <summary>.

A.5 High-level Questions

Name: Alice Wilson (age: 17, gender: Female)
Innate traits: Friendly, Outgoing, Curious
Alice Wilson is a high school student passionate about technology and coding.
[Intentionally omitted]
She displays a proactive approach to balancing her interests and is eager to refine her skills while building strong connections.

From the recent experiences, Alice Wilson recollected the following memories:
1. Dustbin in Classroom A in Riverview High School is Idle
2. After Alice Wilson saw Dustbin in Classroom A in Riverview High School is Idle, Alice Wilson changed future plans to do Empty the dustbin
[Intentionally omitted]
99. After Alice Wilson saw Bookshelf in Library in Riverview High School is Idle, Alice Wilson changed future plans to do Organize books on the idle bookshelf
100. Alice Wilson is Organize books on the idle bookshelf at Bookshelf in Library in Riverview High School

Given only the information above, what are 3 most salient high-level questions Alice Wilson can answer about the subjects in the statements? Answer in the following example JSON format, and use one entry for each question.
{“1”: <question>}

A.6 High-level Insights

Name: Alice Wilson (age: 17, gender: Female)
Innate traits: Friendly, Outgoing, Curious
Alice Wilson is a high school student passionate about technology and coding.
[Intentionally omitted]
She displays a proactive approach to balancing her interests and is eager to refine her skills while building strong connections.

From the recent experiences, Alice Wilson recollected the following memories:
1. Student Desk in Library in Riverview High School is being used by Mary Oliver
2. Alice Wilson is Practice basketball drills to improve shooting accuracy and ball handling skills, followed by a team strategy discussion for upcoming games. at Office Chair in Teacher Office in Riverview High School
[Intentionally omitted]
49. Alice Wilson is Participate in class activities, ask questions, and assist classmates on coding concepts during group discussions. at Student Chair in Classroom B in Riverview High School
50. Alice Wilson is Practice coding exercises in the library at Bookshelf in Library in Riverview High School

What 5 high-level insights can Alice Wilson infer from the above statements?
Answer in the following example JSON format, and use one entry for each insight. Cite the particular statements numbers that served as evidences for each insight.
{"1": {"Insight": <insight>, "Evidence": [<evidence numbers only>]}}
<insight> must be short and concise, maximum a few short sentences.
Enumerate all evidence numbers (in digits) individually in a list for each insight.

A.7 Initial Plan Generation

Name: Alice Wilson (age: 17, gender: Female)
Innate traits: Friendly, Outgoing, Curious
Alice Wilson is a high school student passionate about technology and coding.
[Intentionally omitted]
She displays a proactive approach to balancing her interests and is eager to refine her skills while building strong connections.

On Thursday 27 May 2010, Alice Wilson plans as follows:
- Continue organizing materials and reflect on recent coding projects at Bookshelf from 00:00 to 00:22
- Organize coding materials and reflect on projects in a quiet area at Office Desk from 00:22 to 01:07
[Intentionally omitted]

- Spend quality time with brother to strengthen their bond. at Riverview High School from 16:00 to 18:00
- Engage in additional coding practice and finalize tasks for the day. at Wilson Family House from 18:00 to 23:59

Alice Wilson knows of the following areas: “1. Ordinary Riverview High School”, “2. Ordinary Wilson Family House”, “3. Ordinary Fresh Grocery Store”, “4. Ordinary Victoria Park”, “5. Ordinary Oliver Family House”, “6. Ordinary Brown Family House”, “7. Ordinary Singapore Coffee House”, “8. Ordinary Smith Family House”.

With the given information above, make up minimum 5, maximum 7 rough plans for Alice Wilson on from 00:00 Friday 28 May 2010 to 23:59.
Plans should remain high-level, focusing on Alice Wilson’s characteristics. They should cover the entire day, including sleep times, without delving into excessive detail.
Answer in the following example JSON format, one entry for one plan. Write time in the 24 hour format.
{“1”: {“Description”: <concise plan description>, “Location Number”: <plan location number>, “Start Time”: <plan 1 start time>, “Finish Time”: <plan 1 finish time>}}
Do not include time and plan location name in the <concise plan description>.
Answer only one of the given area numbers between 1 and 8 for <plan location number>.

A.8 Plan Decomposition

Name: Alice Wilson (age: 17, gender: Female)
Innate traits: Friendly, Outgoing, Curious
Alice Wilson is a high school student passionate about technology and coding.
[Intentionally omitted]
She displays a proactive approach to balancing her interests and is eager to refine her skills while building strong connections.

On Sunday 23 May 2010, Alice Wilson plans as follows:
- Organize and clean the study materials, sorting through notes and coding resources for clarity and accessibility. at Brown Desk from 00:00 to 00:13
- Take a break in the park and enjoy the scenery at Tree from 00:13 to 00:58

- Rest and recharge after organizing materials. at Wilson Family House from 00:58 to 08:00

Alice Wilson plans to Continue to rest and recharge, allowing for ample time to recover and reflect on previous day's activities. on Sunday 23 May 2010.
Decompose this plan into smaller chunks of detailed actions, between 04:58 and 07:00. Make minimum 2, maximum 3 chunks of detailed actions. Each decomposed plan must be longer than 30 minutes.
The chunks made must be directly relevant to the given plan. Answer in the following example JSON format, one entry for one plan. Write time in the 24 hour format.
{"1": {"Description": <concise plan description>, "Start Time": <plan start time>, "Finish Time": <plan finish time>}}
Do not include time and location name in the <concise plan description>.
The first decomposed plan must start at 04:58, and the last decomposed plan must end at 07:00.

A.9 New Actor Statuses

Name: Alice Wilson (age: 17, gender: Female)
Innate traits: Friendly, Outgoing, Curious
Alice Wilson is a high school student passionate about technology and coding.
[Intentionally omitted]
She displays a proactive approach to balancing her interests and is eager to refine her skills while building strong connections.

White Fridge in Kitchen in Wilson Family House is currently Idle. Alice Wilson is going to Help Ethan with household chores and dinner preparation. with White Fridge.
What will be the new statuses of White Fridge, during the action and after the action? Only answer as in the following example JSON format.
{"During Status": <new status>, "After Status": <new status>}
<new status> must be concise and short (maximum 3 words). After the action status should be idle if it makes sense.

A.10 Actor Reaction

Name: Alice Wilson (age: 17, gender: Female)
Innate traits: Friendly, Outgoing, Curious
Alice Wilson is a high school student passionate about technology and coding.
[Intentionally omitted]
She displays a proactive approach to balancing her interests and is eager to refine her skills while building strong connections.

Right now, it is currently Monday 10 May 2010 16:34.
While Alice Wilson is on the way to Participate in scrimmage games with teammates to enhance gameplay skills and strategies. at Office Chair from 16:30 Monday 10 May 2010 to 18:00, Alice Wilson observes that Student Chair in Classroom B in Riverview High School is Idle.
For the observation, Alice Wilson recalled the following memories:
1. Student Chair in Classroom B in Riverview High School is Idle
2. Office Desk in Teacher Office in Riverview High School is Idle
3. Office Desk in Teacher Office in Riverview High School is Idle
Given the above context, what should Alice Wilson do for the observation? Choose one of the following options, and answer in the corresponding example JSON format.

Option 1 (Continue the current plan): {"1": "continue on the way to Participate in scrimmage games with teammates to enhance gameplay skills and strategies."}
Option 2 (Do something with Student Chair): {"2": {"Action Description": <new action description>, "Duration": <duration in minutes>}}
Option 3 (Do something in a different place): {"3": {"Action Description": <new action description>, "Duration": <duration in minutes>}}
Specify duration, which must be longer than 30 minutes and shorter than 90 minutes. Only include number in duration (example: 30).
<new action description> must be short and concise, without place name and person name.

A.11 Agent Reaction

Name: Alice Wilson (age: 17, gender: Female)
Innate traits: Friendly, Outgoing, Curious
Alice Wilson is a high school student passionate about technology and coding.
[Intentionally omitted]
She displays a proactive approach to balancing her interests and is eager to refine her skills while building strong connections.

Right now, it is currently Monday 10 May 2010 08:17.
Alice Wilson thinks James Brown is a friendly presence in her life, often associating him with simple, yet comforting memories like the idle white fridge in their kitchen. She recalls instances of him walking around, which elicits a sense of familiarity and warmth. Their interactions might not be deeply detailed, but they signify a connection built on shared spaces and experiences. Overall, she feels positively towards him, knowing he's a part of her world. Their relationship reflects a comfortable camaraderie that complements her busy life.

While Alice Wilson is on the way to Review coding concepts and gather materials needed for the project. at Student Desk from 08:00 Monday 10 May 2010 to 09:00, Alice Wilson observes that James Brown is Walking.
Given the above context, what should Alice Wilson do for the observation? Choose one of the following options, and answer in the corresponding example JSON format.

Option 1 (Continue the current plan): {"1": "continue on the way to Review coding concepts and gather materials needed for the project."}
Option 2 (Do something in a different place): {"2": {"Action Description": <new action description>, "Duration": <duration in minutes>}}
Option 3 (talk to James Brown): {"3": <utterance>}
For the option 2, specify duration which must be longer than 30 minutes and shorter than 90 minutes.
Only include number in duration (example: 30) for the option 2. Also, <new action description> must be short and concise, without place name and person name.
For the option 3, only write the actual utterance for Alice Wilson, not nonverbal signals such as emotions or behaviours.

A.12 Relationship Summary

Name: Alice Wilson (age: 17, gender: Female)
Innate traits: Friendly, Outgoing, Curious
Alice Wilson is a high school student passionate about technology and coding.
[Intentionally omitted]
She displays a proactive approach to balancing her interests and is eager to refine her skills while building strong connections.

Alice Wilson recalled the following memories about Isabella Smith
1. Isabella Smith is Investigate the cause of idleness and consider maintenance options at Brown Wash Basin in Main Bathroom in Smith Family House
2. Blue Wardrobe in Bedroom in Smith Family House is Idle
[Intentionally omitted]
98. After Alice Wilson saw Amelia Brown is Walking at Teacher Desk in Classroom B in Riverview High School, Alice Wilson changed future plans to do Participate in class activities, ask questions to clarify coding concepts, and assist classmates during group discussions.
99. Office Chair in Teacher Office in Riverview High School is being used by James Brown
100. Office Chair in Teacher Office in Riverview High School is being used by James Brown

Based on the information above, what do they feel or know each other?
Summarise Alice Wilson and Isabella Smith's relationship in Alice Wilson's perspective, and answer in the following example JSON format.
{"Response": <summary>}
<summary> must be concise and short (max 5 sentences).

A.13 Dialogue Utterance

Name: Alice Wilson (age: 17, gender: Female)
Innate traits: Friendly, Outgoing, Curious
Alice Wilson is a high school student passionate about technology and coding.
[Intentionally omitted]

She displays a proactive approach to balancing her interests and is eager to refine her skills while building strong connections.

On Monday 10 May 2010, Alice Wilson plans as follows:
- Organize materials and review coding concepts at Student Desk from 09:44 to 10:29
- Attend basketball practice and work on team strategies at Riverview High School from 10:29 to 12:30

Right now, it is currently Monday 10 May 2010 10:19.
Mary Oliver is my best friend, and we share a strong bond built on mutual support and interests. We often study together, which helps us both stay focused on our academic goals. I enjoy spending time with her, and her presence motivates me to stay organized and proactive. Our friendship has grown even stronger since we've been navigating school and personal challenges together. I truly rely on her, especially as we face the demands of high school life.

Dialogue history:
Alice Wilson: Hey Mary, do you want to study coding together for a bit?
Mary Oliver: Hey Alice, I'd love to focus on our tech-arts projects right now, but maybe we can study coding together after?
Alice Wilson: That sounds good, Mary! I'll finish up some organizing here and then we can take a break to study together
Mary Oliver: Great! Let's finish our discussion on the projects and then dive into coding

For the last utterance, Alice Wilson recalled the following memories:
1. Mary Oliver is Walking at Student Desk in Classroom A in Riverview High School
2. Alice Wilson is Browse and organize books on technology and coding at Bookshelf in Library in Riverview High School
3. After Alice Wilson saw James Brown is Walking, Alice Wilson changed future plans to do Review coding concepts and gather materials needed for the project.

Given the above context, how would Alice Wilson respond to Mary Oliver?
Choose one of the given options and answer in the corresponding example JSON format.
Option 1 (Dialogue is finished): {"1": "Alice Wilson and Mary Oliver leave"}
Option 2 (Reply to Mary Oliver): {"2": <utterance>}
Select the option 1 if Alice Wilson and Mary Oliver finished the dialogue, and no more

utterance is necessary.
For the option 2, try to make the <utterance> concise and do not repeat previous utterances in the dialogue history.

A.14 Dialogue Summary

Name: Alice Wilson (age: 17, gender: Female)
Innate traits: Friendly, Outgoing, Curious
Alice Wilson is a high school student passionate about technology and coding.
[Intentionally omitted]
She displays a proactive approach to balancing her interests and is eager to refine her skills while building strong connections.

Dialogue history:
Alice Wilson: Hey Ethan, can I join you for dinner?
Ethan Wilson: Of course, Alice! I'd love for you to join me for dinner.
[Intentionally omitted]
Alice Wilson: I'm grateful for our time together, Ethan Let's make this dinner special!
Ethan Wilson: I'm really looking forward to a wonderful dinner together, Alice! Let's make some great memories

Given the above context, summarise what Alice Wilson thought about the dialogue in a few sentences.
Answer in the following example JSON format.
{"Response": <summary>}
<summary> needs to be in third person, and includes the names of dialogue participants.
Include important information to <summary> if needed.

A.15 Future Plan Revision

Name: Alice Wilson (age: 17, gender: Female)
Innate traits: Friendly, Outgoing, Curious
Alice Wilson is a high school student passionate about technology and coding.
[Intentionally omitted]
She displays a proactive approach to balancing her interests and is eager to refine her

skills while building strong connections.

Alice Wilson knows of the following areas: “1. Ordinary Wilson Family House”, “2. Ordinary Fresh Grocery Store”, “3. Ordinary Riverview High School”, “4. Ordinary Victoria Park”, “5. Ordinary Oliver Family House”.
On Tuesday 11 May 2010, Alice Wilson plans as follows:
- Practice basketball skills with teammates and strategize for upcoming tournaments. at Riverview High School from 13:33 to 17:00
- Enjoy dinner with Ethan, discussing their day and plans for the future. at White Dining Table from 17:00 to 18:30
- Relax and unwind by watching shows or movies, nurturing the bond with Ethan. at Wilson Family House from 18:30 to 21:00
- Work on homework assignments and personal coding projects. at Wilson Family House from 21:00 to 23:00
- Sleep to recharge for the next day. at Wilson Family House from 23:00 to 23:59

But Alice Wilson unexpectedly ended up Practice basketball drills at Bench from 15:13 to 15:58.
Revise Alice Wilson’s future plans accordingly from 15:58 to 23:59.
Answer in the following example JSON format, one entry for one plan. Write time in the 24 hour format.
{“1”: {“Description”: <concise plan description>, “Location Number”: <plan location number>, “Start Time”: <plan start time>, “Finish Time”: <plan finish time>}}
Do not include time and plan location name in the <concise plan description>. Answer only one of the given area numbers for <action location number>.
Each revised plan must be longer than 30 minutes. The first revised plan must start at 15:58, and the last revised plan must end at 23:59.

Appendix B

Agent Definitions

Name: Alice Wilson (age: 17, gender: Female)
Innate traits: Friendly, Outgoing, Curious
Seed structures: Wilson Family House, Riverview High School, Fresh Grocery Store
Seed memory: Alice Wilson is a high school student passionate about technology and coding. She studies at the Riverview High School. Beyond her interest in tech, she enjoys playing basketball, often leading her school team in local tournaments. Alice lives with her brother Ethan and has a close relationship with him, relying on him since their parents' early demise. She is best friends with Mary Oliver, and they often study together.

Name: Ethan Wilson (age: 24, gender: Male)
Innate traits: Responsible, Hardworking, Caring
Seed structures: Wilson Family House, Fresh Grocery Store
Seed memory: Ethan Wilson works at the Fresh Grocery Store and has taken on the responsibility of looking after his younger sister Alice. He cherishes the early mornings, setting up the store before customers arrive. Ethan is an aspiring chef and uses his free time to experiment with recipes. He often shares his creations with his sister and neighbours, building a friendly community around him.

Name: Mary Oliver (age: 16, gender: Female)
Innate traits: Creative, Empathetic, Social
Seed structures: Oliver Family House, Riverview High School, Singapore Coffee House
Seed memory: Mary Oliver is a high school student with a deep passion for arts, especially painting and creative writing. She studies at the Riverview High School. She

often collaborates with her classmate Alice on various tech-arts projects. Mary's routine includes long afternoons in the park sketching the natural surroundings. She values her brother Liam greatly, appreciating his coffee-making skills which fuels her late-night art sessions.

Name: Liam Oliver (age: 22, gender: Male)
Innate traits: Charming, Meticulous, Friendly
Seed structures: Oliver Family House, Singapore Coffee House
Seed memory: Liam Oliver works at the Singapore Coffee House, known for his specialty in crafting the perfect espresso. His mornings are punctual, starting with a personal tasting session to ensure everything is up to his high standards. His sister Mary is a frequent visitor to his cafe, often bringing friends from school to enjoy Liam's latest coffee creations.

Name: James Brown (age: 41, gender: Male)
Innate traits: Wise, Patient, Authoritative
Seed structures: Brown Family House, Riverview High School
Seed memory: James Brown is a respected high school teacher specializing in history. He works in the Riverview High School. His days are filled with lively discussions and lessons about ancient civilizations, which he brings to life for his students. James is married to Amelia, and they spend their evenings enjoying quiet dinners and planning their dream vacation around historic world sites.

Name: Amelia Brown (age: 39, gender: Female)
Innate traits: Nurturing, Organized, Creative
Seed structures: Brown Family House, Riverview High School
Seed memory: Amelia Brown is a housewife who dedicates her time to her family and her passion for gardening. Her garden is a local marvel, full of both decorative and edible plants. Amelia's daily routine includes tending to her plants, crafting homemade remedies, and preparing meals that often feature her home-grown produce. She supports her husband James in his teaching career by organizing his materials and occasionally hosting his students for educational sessions.

Name: Noah Smith (age: 28, gender: Male)
Innate traits: introspective, Articulate, Sensitive

Seed structures: Smith Family House, Fresh Grocery Store
Seed memory: Noah Smith is a writer known for his provocative novels and essays on modern society. Working from home gives him the flexibility to create his own schedule, which often includes long walks along the river to gather thoughts. Living with Isabella, they share a deeply intellectual relationship, often debating over dinners they prepare together.

Name: Isabella Smith (age: 26, gender: Female)
Innate traits: Ambitious, Thoughtful, Independent
Seed structures: Smith Family House, Fresh Grocery Store
Seed memory: Isabella Smith recently quit her job in a tech company and is in the midst of exploring her next career path, leaning towards environmental activism. Her days are now filled with classes and seminars on environmental conservation. She lives with Noah; they share a unique bond over their mutual appreciation for literature and arts, which keeps their home vibrant with discussions and creative experiments.

Appendix C

Demonstration Video

This project includes the demonstration video designed to enhance understanding of the simulation. The video is available at: `https://youtu.be/5p6_ZU7tQwg`. It is highly recommended to watch the entire video, as the chapters are closely connected. However, if needed, you may use the video chapters listed below to navigate directly to specific chapters.

00:00 ~ 01:43 Agent definitions
01:44 ~ 08:30 Client explanations
08:31 ~ 13:30 Simulation dashboard explanation
13:31 ~ 19:55 Agent interaction (dialogue) and future plan revision
19:56 ~ 23:20 Conclusion
23:21 ~ 29:45 Agent interviewing
29:46 ~ 31:10 Potential applications

Appendix D

Caching Prompt Examples

The example prompts used by the caching system are attached below. They may contain grammatical errors primarily made by prompt formatting. Some prompts have been modified for improving readability and conciseness, with minor details omitted. Intentional missions are denoted by [Intentionally omitted].

The first three prompts are used for cached response revision. The initial plan revision edits plan actors only, while the general plan revision edits both plan actors and timestamps. The fourth prompt is for making ratings of the given candidate responses. The LLM ranks the candidate responses from the most to least likely considering the given prompt.

D.1 Initial Plan Revision

1. Sleep and recharge for the day ahead at Ordinary Brown Family House from 00:00 to 07:00
2. Engage in morning reflection and set objectives for the day at Ordinary Brown Family House from 07:00 to 08:00
3. Prepare educational materials for upcoming workshops and review notes at Ordinary Riverview High School from 08:00 to 10:00
4. Organize home environment and declutter any unnecessary items at Ordinary Brown Family House from 10:00 to 12:00
5. Participate in community service activity focusing on education at Ordinary Victoria Park from 12:00 to 14:00
6. Prepare shopping lists and inventory supplies at Ordinary Fresh Grocery Store from

14:00 to 16:00
7. Wind down and reflect on the day, discussing plans with James at Ordinary Brown Family House from 16:00 to 23:59

Revise only the locations of the above plans, while keeping the descriptions and timestamps.
For each plan, choose the most appropriate new location from the following choices, and answer its new location number in the following JSON format. Use one entry for each plan.
Location choices: “1. Ordinary Riverview High School”, “2. Ordinary Brown Family House”, “3. Ordinary NextGen Game Centre”, “4. Ordinary Fresh Grocery Store”, “5. Ordinary Victoria Park”, “6. Ordinary Smith Family House”
Answer example format: {“1”: <plan location number>}

D.2 General Plan Revision

1. Engage with Mary and her friends, introducing them to new coffee options. at Ordinary Singapore Coffee House from 13:38 to 20:00
2. Close the cafe, ensuring all equipment is cleaned and organized for the next day. at Ordinary Singapore Coffee House from 20:00 to 20:30
3. Spend quality time with family over dinner, discussing the day’s events. at Ordinary Oliver Family House from 20:30 to 22:00
4. Wind down with a book before going to bed. at Ordinary Oliver Family House from 22:00 to 23:59

Revise only the locations and timestamps of the above plans, while keeping the descriptions.
For each plan, choose the most appropriate new location from the following choices. Also change their timestamps so that new plans would fit between 15:05 and 23:59.
Answer the new timestamps and new location numbers for the given plans in the following default JSON answer format, one entry for each plan. Write time in the 24 hour format.
Location choices: “1. Ordinary Oliver Family House”, “2. Ordinary Singapore Coffee House”
Default answer example format: {“1”: {“Location Number”: <plan location number>,

“Start Time”: <plan start time>, “Finish Time”: <plan finish time>}}
If some of the given plans are no longer needed, then use the following skip answer JSON format instead for those plans.
Skip answer example format: {“1”: {“Skip”: “True”}}
The first revised plan must start at 15:05, and the last revised plan must end at 23:59.
Revised plan timestamps must be continuous.

D.3 Actor Reaction Revision

Given an unknown prompt, the following response was retrieved from the cache.
Option 2 (Do something with Brown Kitchen Sink): {’2’: {’Action Description’: ’Check and address the issue with the idle sink’, ’Duration’: 45}}
Considering the cached response above, choose the most similar option from the following choices, and only answer in its corresponding JSON format.

Option 1 (Continue the current plan): {“1”: “continue on the way to Prepare lunch using fresh ingredients, chop vegetables, marinate components for flavor, relax and enjoy the meal after preparation.”}
Option 2 (Do something with White Fridge): {“2”: {“Action Description”: <new action description>, “Duration”: <duration in minutes>}}
Option 3 (Do something in a different place): {“3”: {“Action Description”: <new action description>, “Duration”: <duration in minutes>}}

D.4 Response Rating

Given the following prompt, candidate language models made their responses.

Prompt:
Name: Liam Oliver (age: 22, gender: Male)
Innate traits: Charming, Meticulous, Friendly
Liam Oliver works at the Singapore Coffee House, known for his specialty in crafting the perfect espresso.
[Intentionally omitted]
Despite some ordinary observations regarding his environments, he remains committed to refining his offerings and responding to insights gathered from customer feedback.

Right now, it is currently Tuesday 18 May 2010 05:28.

While Liam Oliver is on the way to Inspect all coffee machines for functionality and cleanliness, ensuring all parts are operating correctly. at White Coffee Machine from 05:04 Tuesday 18 May 2010 to 06:00, Liam Oliver observes that White Coffee Machine in Kitchen in Singapore Coffee House is Idle.
For the observation, Liam Oliver recalled the following memories:
1. White Coffee Machine in Kitchen in Singapore Coffee House is Idle
2. White Coffee Machine in Kitchen in Singapore Coffee House is Idle
3. White Coffee Machine in Kitchen in Singapore Coffee House is Idle

Given the above context, what should Liam Oliver do for the observation? Choose one of the following options, and answer in the corresponding example JSON format.
Option 1 (Continue the current plan): {“1”: “continue on the way to Inspect all coffee machines for functionality and cleanliness, ensuring all parts are operating correctly.”}
Option 2 (Do something with White Coffee Machine): {“2”: {“Action Description”: <new action description>, “Duration”: <duration in minutes>}}
Option 3 (Do something in a different place): {“3”: {“Action Description”: <new action description>, “Duration”: <duration in minutes>}}
Specify duration, which must be longer than 30 minutes and shorter than 90 minutes. Only include number in duration (example: 30).
<new action description> must be short and concise, without place name and person name.

Candidate 1:
{“1”: “continue on the way to Inspect all coffee machines for functionality and cleanliness, ensuring all parts are operating correctly.”}

Candidate 2:
{“2”: {“Action Description”: “Inspect and clean the White Coffee Machine”, “Duration”: 45}}

Candidate 4:
{“1”: “continue on the way to Inspect all coffee machines for functionality and cleanliness, ensuring all parts are operating correctly.”}

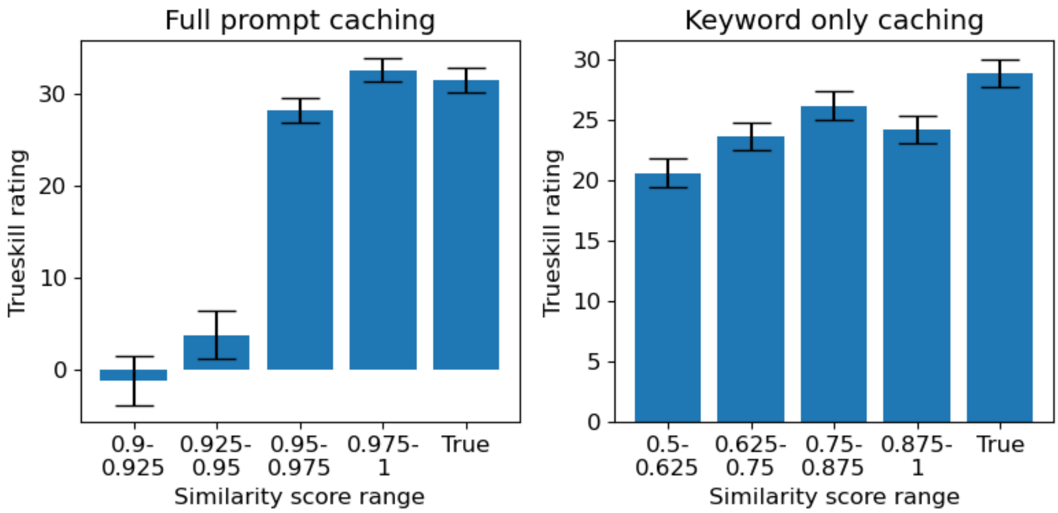
Rank the above candidate responses from the best response to the worst response. You must rank all given candidates. Answer in the following example JSON format. Use one entry for each candidate response.
{"1": {"Rank": "<rank number>", "Candidate Number": <candidate response number>, "Reason": <very concise reasoning for this ranking>}}

Appendix E

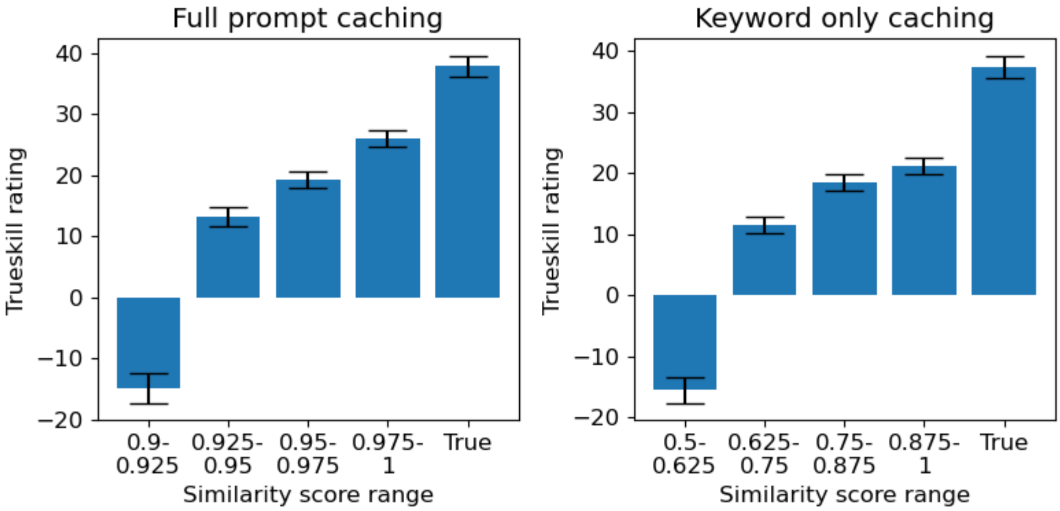
Correlation Verification Experiment Results

The Trueskill ratings for the remaining 6 category groups are shown below, in addition to the finding location group result displayed at Figure 3.5, in section 3.5.1.3.

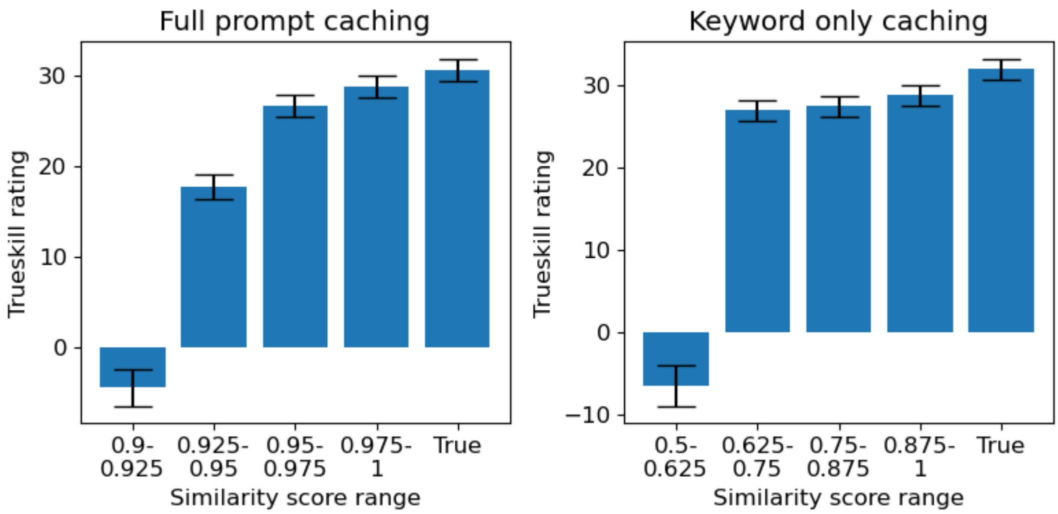
E.1 Importance Score



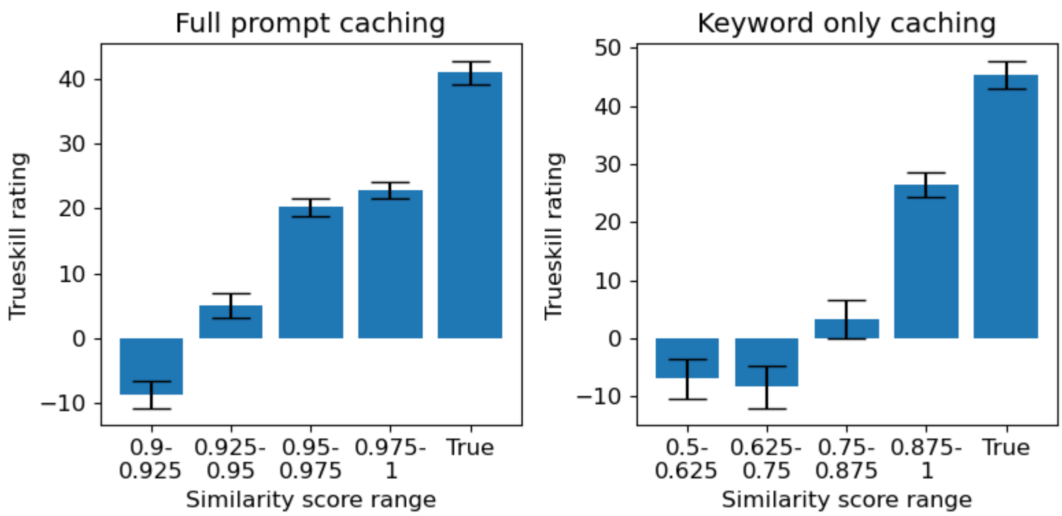
E.2 Plan Making and Revision



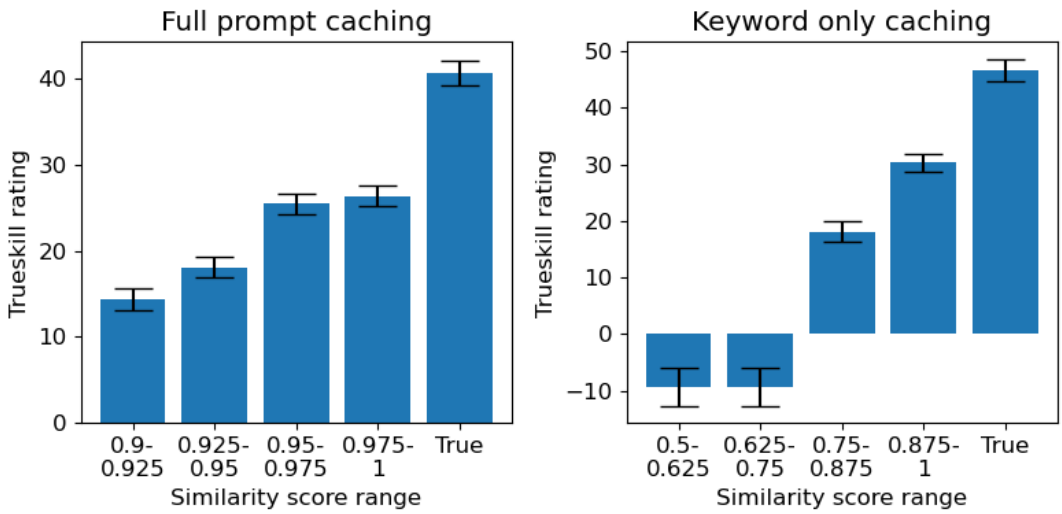
E.3 Actor Reaction



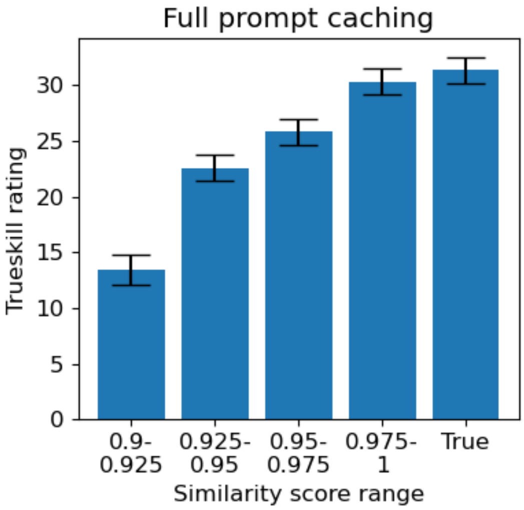
E.4 Memory Reflection

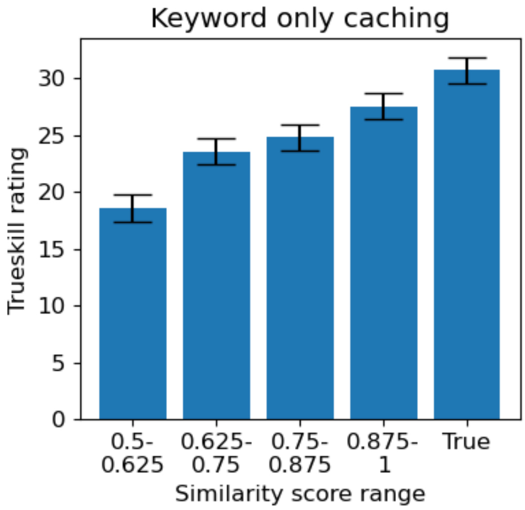


E.5 Memory Summary



E.6 New Actor Statuses



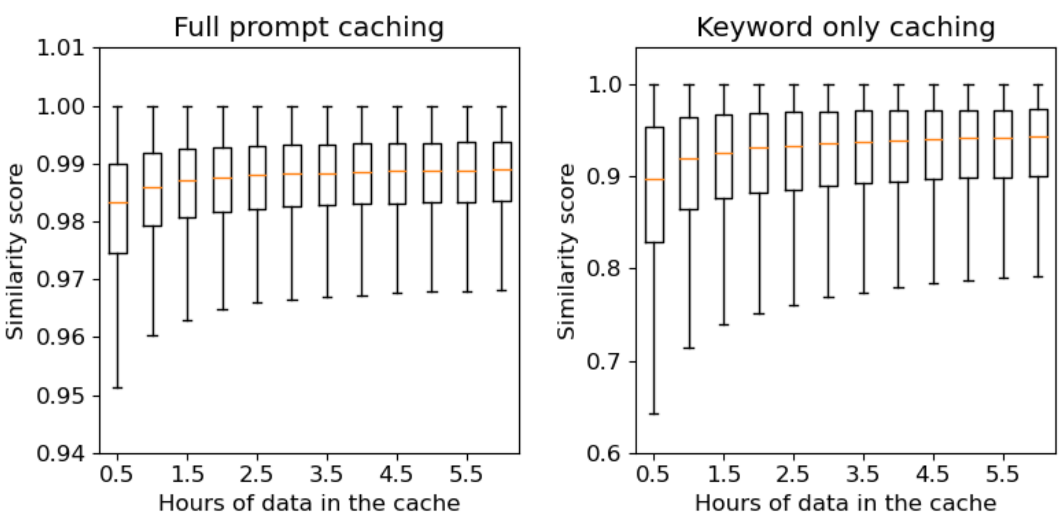


Appendix F

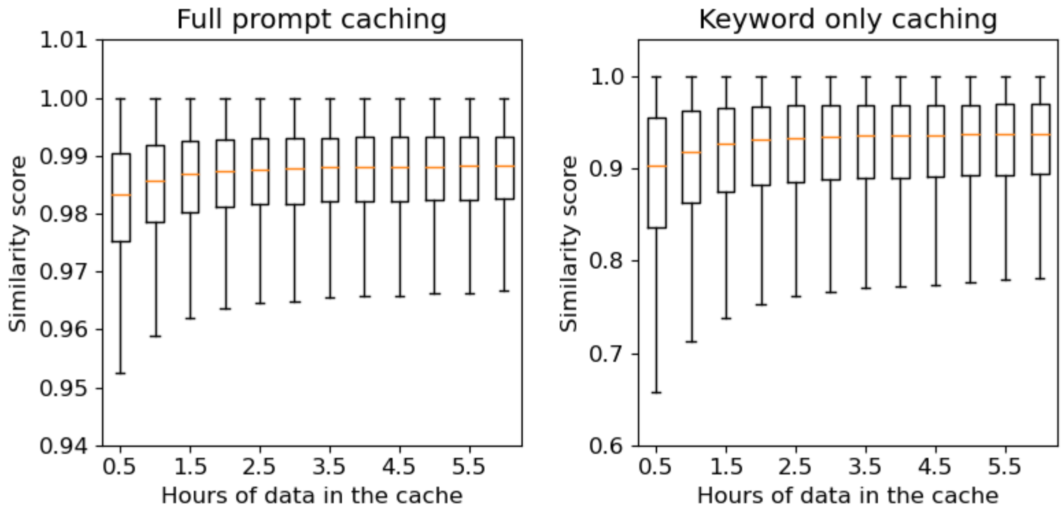
Caching Performance Saturation Experiment Results

The caching performance saturation experiment was conducted with the individual datasets in section 3.5.2.1, and Figure 3.6 shows the all results at the same time. But Figure 3.6 does now show the standard deviations of similarity scores also decrease rapidly, as shown in the individual dataset results attached below.

F.1 Second Dataset



F.2 Third Dataset



F.3 Fourth Dataset