Title: ***Simulating Human Conversations using MORGAN, a brain-inspired, LLM-based agentic and memory framework.***

Abstract:
There have been many attempts to use Large Language Models (LLMs) as believable proxies of human behavior in a push to better understand LLMs and to enable their use as generative agents that humans can interact with as social agents, not just mechanistic productivity tools. This requires the creation of autonomous agents that plan, retain memory and experience, and have goals similar to humans. This paper builds on past attempts to create such agents by focusing on augmenting these autonomous agents with brain-inspired memory and showcases a novel modular architecture for Social Generative Agents (SGAs) called MORGAN (**M**odular **O**ptimized human **R**eplica using **G**enerative **A**ge**N**ts) that blends ideas from various existing papers into one unified agent architecture and brings to the literature a novel idea of using long, 1M token "backstories" in SGAs' long term memory. While this architecture can be generalized to work in 2D or 3D environmental settings similar to LyfeGame (Kaiya et al. 2024) and AI-town (Park et al. 2023) by adding greater textual input describing surrounding objects and individuals to the LLM, but for simplicity of testing and to reduce unnecessary complexity, this paper abstains from adding such environmental input (but I will still showcase how an added module in my modular architecture can be used to generalize to such settings). In some limited qualitative testing, we found that our agents retain and retrieve memory very accurately and their tendency to forget excruciatingly detailed, non-important memories over time makes the observed memory retrieval quite akin to real human memory retrieval. Link to code:
https://drive.google.com/drive/folders/1don4QF0flL0V2dlmhpbN6-ktOluQNpE6?usp=sharing

Introduction:

This paper is part of a literature which tries to answer a fundamental question: can LLMs act as believable proxies of human behavior? LLM advances such as chat finetuning and RLHF have brought these models closer than ever to having the ability to replicate, even predict, human behavior in a given scenario. The obvious shortcomings of vanilla LLMs as believable proxies of human behavior come from their lack of a comprehensive memory system, lack of singular personal identity (which is accompanied by personal memories, experiences, goals, etc.), and a lack of the ability to be placed into an environment and plan using sensory input. This paper proposes a new architecture called MORGAN (**M**odular **O**ptimized human **R**eplica using **G**enerative **A**ge**N**ts) which contains a modular implementation of a memory architecture that involves a Think in Memory system (Liu et al 2023), of a summarize-and-forget mechanism (Kaiya et al. 2024), and a novel long-term memory system that leverages a fundamental advancement in LLM technology, long context-window models, specifically Google's Gemini 1.5 pro which can hold 1 million tokens concurrently in its ATTENTION window. I leverage this long context model by first using a custom built pipeline to generate long, ~250k tokens of coherent backstory on each individual which contains coherent memories and experience, and then these memories are attended upon in conjunction with other facets of the architecture (such as my short-tern memory, etc.) during inference (final step when the agent is simulating human behavior). My hypothesis is that using long-context models in this way (which has not been tried in existing literature at the time of writing this paper) can signifincantly boost the ability of SGAs to create believable proxies of human behavior. While my goal in this paper is only to explore the correlation between my new architecture and improved performance, I also speculate in the conclusion about the causal link here, specifically that long-context models might more closely resemble biological human memory operation which similarly "attends" to all our memories.

In this paper, I will first describe our approach to creating long "backstories" for SGAs, then describe and showcase the implementation of MORGAN, and finally discuss some qualitative results and potential for future study of results (and the need for more concrete evaluations in this literature).

Related Works:

In 2023, scholars at Stanford and Google Deepmind created a "smallville", a town that is populated by LLM powered AI agents (GPT-3.5) and showcased how several dynamic, context-based augmentations (such as the ability to retrieve and summarize memories to also be included in a prompt that generates the response to a question in a conversation with another agent) to the LLM prompts during a conversation can help achieve extremely believable replicas of human behavior. The agent architecture used in this paper is actually quite simple, essentially taking into account a "short-term memory" that records and synthesizes past conversations and appends itself to future prompts and also taking into account long-term goals and aspirations of an agent in their responses and actions (Park et al 2023). Lyfe Agents paper by MIT's Metaconscious Lab (Kaiya et al. 2024) recreates and builds upon Park et al. by creating a 3D simulated game world and detailing a more complex agent architecture that implements a summarize and forget memory system. In their experiments, they ran simulations where agents were tasked with solving murder mysteries, organizing events, going to school, and found that (as measured by a mainly subjective, human eyeball evaluation metric) their agents were able to simulate believable human behavior and complete their tasks in a way that is coherent with their conversations throughout the simulation. My paper builds upon these works by leveraging now the presence of a new key tool: long-context models, around which I invent a new architecture which is showcased in the following three section of the paper.

Section I: Creating Histories:

I first began this project with by identifying the explicit need for a "long" backstory. Existing work from Park et al and Kaiya et al also employs agent backstories as a way to give some experiential and individualized grounding to agents but all these past attempts resort to giving no more 10-15 lines of backstories (Park et al 2023). I created a pipeline that uses Google's Gemini 1.5 pro model to use its 1M token context length to create coherent, first-person narrative backstories for agents, containing vivid descriptions of memories and experiences from all ages of life. This generation is tuned with some hyperparameters. In this section, I will outline exactly what these hyperparameters are, justify their usage and showcase some results.

First, backstory generation leverages a somewhat popular psychological theory of "the big 5" (add citation). The big 5 is a framework that helps reduce personality down to 5 key levers, and these are what I tune as hyperparameters before commencing backstory generation. For instance, in the case of Shawn Briar, one of SGAs I generate in this project, has the following big 5 hyperparameters: "{'openness': 'Low', 'conscientiousness': 'High', 'extraversion': 'Low', 'agreeableness': 'High', 'neuroticism': 'Low'}", represented as a dictionary in python. For simplicity, I kept the possible values as "High", "Medium", and "Low", but future work could include adding more granularity to these hyperparameters and observing any meaningful differences in the backstories generated.

In addition to the big 5, I also feed as input to my generation pipeline certain biological details that help instantiate the generation scheme. In Shawn Briar's case, these details

include: “{‘Name’: ‘Shawn Briar’, ‘Age’: 24, ‘Born’: ‘June 6, 1994’, ‘Locale Born’: ‘Union Springs, Alabama, USA’,
‘Locale Current’: ‘Union Springs, Alabama, USA’, ‘Languages’: ['English'], ‘Race’: ‘White’, ‘Gender Identity’: ‘Cisgender male’, ‘Historical Income: Low’, ‘Current Income’: ‘Low’, ‘Marital Status’: ‘Single’, ‘Kids’: 1, ‘Highest Education’: ‘11th grade’}.” These fields are common to all generations and input to them is crucial since my generation pipeline uses different combinations of these inputs to create specific memories in the backstory of the individual.
In my code, I created a class called “GenerationStrategy” that contains Big5 as one of the class instantiations, but the code is generalized to also accept other strategies if one wishes to switch away from Big5 entirely (for instance, other possibilities include the 16PF framework that includes higher granularity of personality features). For convenience and the ability to use multiple LLMs in this generation, I also implemented an LLM class that abstracts away API details. Now, I define a class “Life” which contains the person’s backstory as a structured collection of another class of objects “Eras”. I defined the following as eras: {"Childhood": (0, 12), "Adolescence": (13, 18), "Early Adulthood": (19, 25), "Adulthood": (25, 35), "Middle Adulthood": (35, 50), "Late adulthood": (50, 65), "Old age": (65, 100)} which is a tunable dictionary of ages and associated eras. This is then leveraged in the generation by segmenting the generation such that memories from one era are generated, then the next and so on. This enables a high degree of coherency since a summary of key points from each era is then used to generate the next, hence minimizing the chance of inconsistencies and allowing for higher quality outputs. Each Era class instantiation contains the following key features which are iteratively generated using an LLM: ‘Major Arc’, ‘Presences’, ‘Absences’, ‘Beliefs Added’, ‘Beliefs Modified’ (note: these are detailed with comments in life.py file). These are then iteratively passed into our prompts, which then prompt the LLM, summarize each era, and reprompt, all with the key effort to retain coherency. These features in each era were added quite late in the project but they proved key to the generation of complex, diverse backstories. The lack of these features in the prompts led to LLM responses that were extremely stereotypical and lacked depth, perplexity. I carefully chose these specific features after trying out many others since these force the model to not only characterize the person’s relationships (the presences and absences) and their impact on the person, but also force the model to generate higher perplexity responses when it is asked to throw a “curveball” (which is another sub-feature that leads to the beliefs modified field). Since I had no intention to generate a specific person, I also experimented with random generation of all the hyperparameters, and that is what led to the generation of Shawn Briar, the first person generated using my pipeline. The entirety of Shawn Briar’s backstory is available in the person1.txt file, but a random excerpt is shared below:
“Romance entered Shawn's life in a quiet, unexpected way during his late teens. His first significant relationship was with a classmate, Emma, who shared his love for the outdoors and understood the complexities of his family obligations. Their relationship was marked by a deep mutual respect and a shared sense of responsibility towards their families. However, Shawn's high agreeableness and low neuroticism meant he often prioritized Emma's needs and the well-being of his family over expressing his own desires or concerns, leading to a relationship that, while deeply caring, sometimes lacked open communication about personal needs and aspirations. This first love added a new dimension to Shawn's adolescence, teaching him valuable lessons about companionship, sacrifice, and the importance of honest communication in maintaining a healthy relationship.
Presences: 1. **Jenny Briar - Younger Sister**: Shawn's relationship with Jenny grew stronger during his adolescence as he took on more responsibilities in her life. They spent time together doing homework, and Shawn attended her school events when their parents

couldn't. Shawn feels a deep sense of protectiveness and love for Jenny, seeing her well-being as a priority."
I also generated another person, "Dev Seth", and their entire backstory is available in person2.txt file.
Studying these long-context generations and their resultant behavior in SGAs is an extremely interesting and completely unexplored topic in literature due to the recent invention of availability of the long-context technology needed to create these generations (the Gemini 1.5-pro public API was released just 3 weeks ago). To study the effect of these long backstories on our SGAs, it is also crucial to equip them with a memory system that is capable of handling such a long backstory (since existing examples of such memory systems in Park et al and Kaiya et al are only equipped to deal with a long term memory that contains 10-15 lines of text, and moreover, Kaiya et al is not open sourced). Hence, the next task of this project was to develop a capable memory system.

Section II: Memory

Popular literature in the field such as Kaiya et al and Liu et al has established that using brain-inspired memory architecture for such generative agents yields better, and even cheaper (more efficient) performance. In this paper, both these qualities of a brain-inspired memory architecture are crucial since a long agent backstory requires an effective and efficient memory system to produce coherent outputs and to fully leverage the presence of the long backstory. To do so, I created an agent architecture that is described in figure 1. This architecture's memory component uses a short term and a long-term memory, and combined Kaiya et al's "summarize and forget" memory system with Liu et al's 'think in-memory" system. The former is implemented following The description of the mechanism in Kaiya et al (and this implementation may by slightly different from what the authors intended and it is my own interpretation of their work since their code is not available or open-sourced). The latter is implemented with a slight twist that simply achieves the intended effect of think in-memory by putting the entirety of retrieved memory into model context for the output.

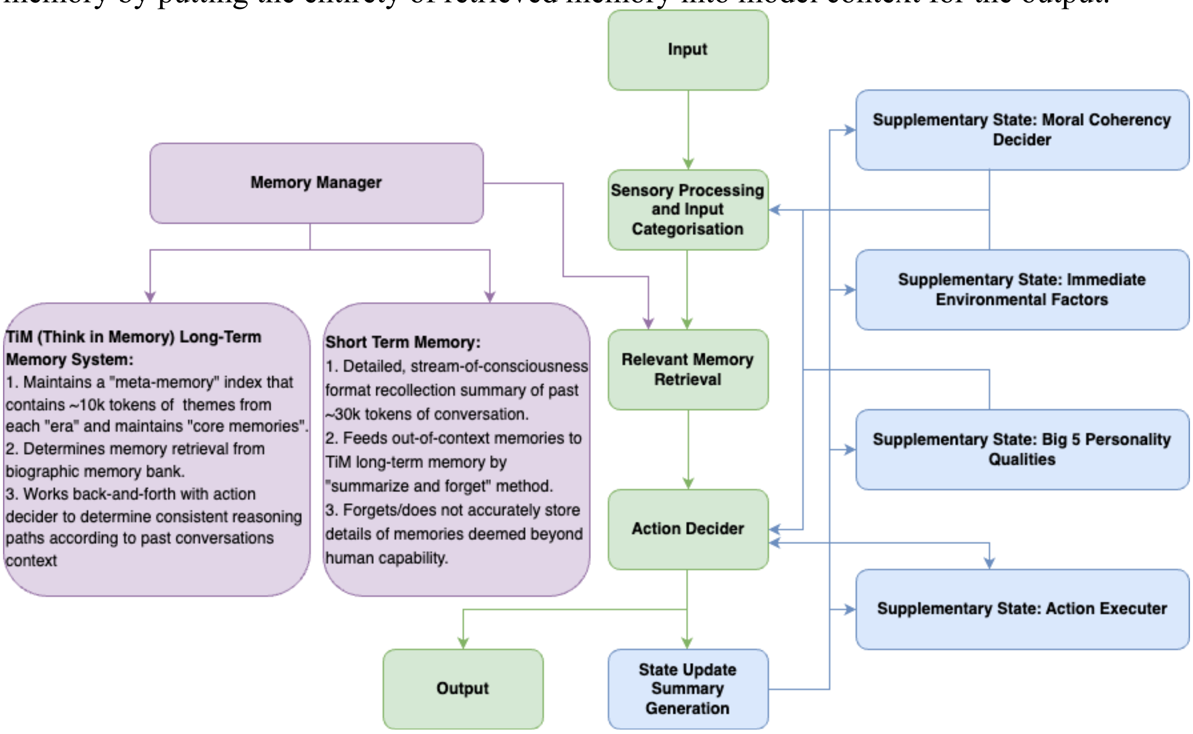

Figure 1: MORGAN Architecture

Firstly, it is crucial to note that this architecture is not 1-1 described in the code. However, the ambition of this paper was to describe a generalizable architecture that can operate in a 2D/3D environmental setting (hence the box for “sensory processing” and “action decider” which are supposed to determine how the agent acts and operates in a situated environment), but in implementation, for simplicity, I specifically showcase how the meat of the architecture, specific the Big 5 generation and the memory system work as expected and can be considered an improvement over existing literature (although this is notoriously difficult to quantitatively test as I describe later in the results section).

For the implementation of this memory architecture, I created the MemoryManager class (available in v1agent.py) that contains methods to initialize memory, process new information (i.e any textual input) and add them to the short term memory and revise recent memories, summarizes and forgets memories from “capacity” timestamps ago (where “capacity” acts a hyperparameter that instructs the memory system to perform a summarize and forget operation after a certain number of timeframes have passed), a method to find similar recent memories, and one to embed and decode memories as they put into the memory system. All vector embeddings in my project were done using “MiniLM-L2-v6”.

The two hyperparameters passed into the MemoryManager system are “stm_decay_interval” and “stm_capacity” where the former is just a decay mechanism that obscures the memories (and eventually forgets) memories stored in the short-term memory after some time has passed. This is a new idea introduced in this paper with the simple intention of replicating human short term memory which cannot remember many irrelevant details for a long period of time. To quantify this “relevance”, the memory manager also assigns an “importance” score to each memory which is a score in [0, 1] that quantifies how important, pressing, or necessary our agent deems that particular item in the short term memory. This score is computed using an LLM call that assigns a higher score to matters that appear more pressing. As an example, “My friend has a surgery tomorrow” receives a higher importance score than “my friend is eating a burger”. Memory retrieval is done using a combination of fetching relevant memories according to the cosine similarity of the model input query and the memory embedding, where even memories that may not have the highest cosine similarity are retrieved if they have a high importance score, so the only unretrieved memories end up being the ones with a low importance score and low cosine similarity. The cosine similarity is calculated according to this formula where “i” is the vector embedding of the input query and “m” is one specific memory vector embedding.

$$\frac{\mathrm{i} \cdot \mathrm{m}}{\mathrm{L2(i)L2(m)}}$$

The “ShortTermMemory” and “LongTermMemory” classes in the v1agent.py file contain my detailed implementation of these above mechanisms and they also contain some auxiliary methods that allow us to clear memories, retrieve all memories and such. For the find_similar_memories method, there is a key hyperparameter the “similarity_threshold” which is a numerical value in [0,1] that dictates the similarity value above which memories are retrieved and put into the immediate context of the SGA model that will produce our output. In my testing, I had initialized this value as 0.5 and had also initialized the values for “stm_decay_interval” and “stm_capacity” as 3 and 5 respectively. These values were chosen not only because they yielded, qualitatively, the best results in my testing with different combinations of values of these parameters but also because it is believed that humans are able to store only 5 things with high resolution in their immediate short term memory (citation).

The last key component of the memory system is the summarize and forget mechanism. This is the way for memories to make their way from the short term memory to the long term memory. As discussed in Kaiya et al, it is believed that the human brain operates in a similar manner and so this biologically plausible computation mechanism implements what occurs in our hippocampus. Memories with higher importance scores are also emphasized in these summaries which are generated using an LLM (in my testing, for this specific step I used gpt 3.5 turbo to save on API cost, but future testing could include using a more capable model for this summarization, or even testing some extremely cheap small models such as Llama3-8b). The implementation of this is quite straightforward, exactly as described, and the hyperparameters can be changes very easily in the code in v1agent.py file. At the end of v1agent.py, I have included some of the tests I did trying to find some good hyperparameters to use.

The long term memories of our agents are initialized with the backstory (partiotioned into the different eras and subfields as described above, of course, to enable ease of access in code and better more quantized retrieval performs well) and then short term memories are added to the long term memory with the summarize and forget mechanism, and all retrieved memories are put directly into model context to achieve the think in-memory system. This completes our implementation of the memory system. As outlined in the MORGAN framework figure, this memory system can easily be augmented with sensory/environmental input (where, for instance, simply observing an object would put it into the short-term memory, alongside an importance score). Hence, our modular, generalizable, and biologically plausible memory system description is complete. The next step after creating the backstories for our SGAs, equipping them with a working memory, is to enable communication between SGAs and human testers as well as SGAs themselves. For the purposes of this project, I was interested in giving two SGAs the ability to interact with each other given a certain situation and scenario and observe the following conversation.

Section III: Simulation:

To implement this last part of the project, I created a simple conversation simulator by instantiating two agents, Shawn Briar and Dev Seth (person1.txt and person2.txt) each with their own rich backstories. To create a situation in which these two agents are interacting, I created a field in the code that allows us to begin the conversation with some general prompt, which in this case I selected as “Shawn Briar and Dev Seth have found themselves as neighbors as they are both moving to a new place”. The details of this implementation are quite trivial and are available to view in simulacra.py but I will briefly describe them here. I create a new class called Simulacra that contains an instantiation of MemoryManager, timestep, backstory and my llm util class. I then generate prompts to feed into our llm util class based on the initialization specifications and the backstory, and then with that context, we begin a conversation in a loop where each agent goes back and forth answering (so if the conversation begins with agent1, the next response must come from agent2 and so on) so there is no possibility of one agent interrupting the other or of the two agents speaking out of turn. This class is also generalizable to using more than 2 agents (which is a recurring theme in my paper, the code infrastructure is ready to be generalized for future study, but for simplicity I have attempted to keep things simple in my testing). Hence, we now have the ability to initialize conversation between any n number of SGAs, on any initial starting prompt (to which both agents adapt and process with respect to their memory and backstories), and then observe the results.

This brings us to the final section of this paper, results.

Section IV: Results and Discussion

First, a sample excerpt from the simulated conversation in section 3 is included below for a qualitative understanding of what conversational output is structured as:

“**Shawn:** *waves* Hey there, neighbor! I don't think we've officially met. I'm Shawn Briar. I just moved here from Union Springs, Alabama. It's quite a change from the small town life I'm used to.
Dev: *smiles* Nice to meet you, Shawn! I'm Dev Seth. I recently moved to Boston from Mumbai, India. It's a pleasure to have you in the neighborhood. What brings you to the city?

Shawn: Well, I got a job opportunity as a carpenter with a construction company here. It's a big step for me, considering I dropped out of high school in 11th grade to help support my family. But I recently got my GED, and I'm ready for a fresh start. How about you? What's your story?
Dev: That's great, Shawn! Congratulations on your GED and the new job. As for me, I'm actually here to start my own business. I've always been interested in entrepreneurship and innovation, ever since I participated in a business competition back in high school. I decided it was time to take the leap and pursue my dreams.”. This conversation goes on much longer and the entirety of it can be accessed in the file “neighbours.txt”. The key factor to note is that throughout the conversation (even after it escapes the token context length of most models) both agents actively remember and respond to the immediate input and also use their memories and facts about themselves to answer questions. For instance, later in the conversation, Shawn, knowing in his short term memory that Dev has a business degree, and remembering his childhood memories of a toolbox, says “I may not have a degree in business, but I know my way around a toolbox”. This is a key example of how my memory system interacts in a conversation and enables for more naturalistic conversation, similar to how two real humans would converse, which was the general goal of this project. Of course, this is a subjective metric and this example is as qualitative as it gets, but a big struggle in this literature is to define quantitative ways to evaluate LLM-based agents, especially when the agnets are not supposed to perform some fixed, success-measurable task such as coding, but something as subjective as meaningful social interaction. This key challenge for SGAs is also faced by Park et al and Kaiya et al in their papers who try to make the most of this limitation by creating scenarios in their 3D environments, where agents play games such as a murder mystery and successfully solving the puzzle via cooperation and such is considered as a successful evaluation of their agents. The problem with such evaluations is that these scenarios written and evaluated by paper themselves never return a failed evaluation, since there is no quantitative metric to which to declare failure.
To get around this limitation, in this paper I instead decided to focus on more granular aspects of the MORGAN architecture and testing improvements through ablation in the architecture rather than a meta-evaluation by creating a complex environment and scenario the scope of which is outside the paper (but for which the MORGAN architecture is generalizable, so this could be a key area for future evaluation and study). The main component that I was able to devise a quantitative metric to test was the short term aerm memory retrieval system, which I will demonstrably show performs well without ablation. I wrote a custom scenario (the exact details of which can be seen in the all.py file in the prompts folder), where I manually inserted

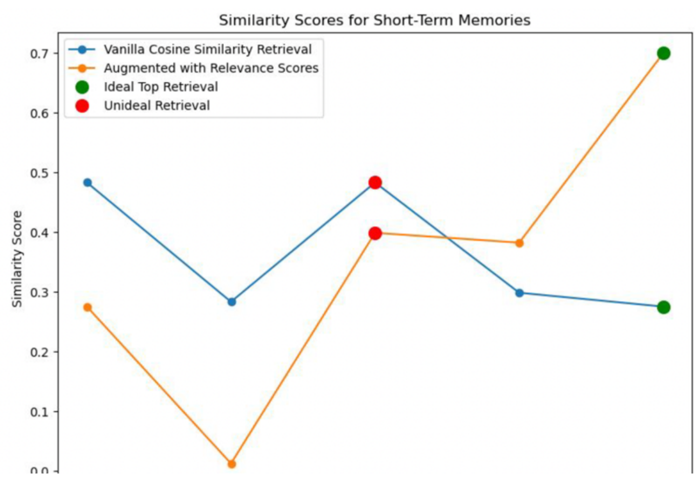

Figure 2

5 memories, with 4 completely random memories that have no link to the query at all, and one memory which is related to the query and ran the short term memory system's retrieval function, stored the similarity scores and plotted them. The scenario involved the agent traveling to India. The query was "when should I leave for the airport?", with the ideal top memory retrieval being the memory that "I have a flight to catch in 3 hours", an unideal memory (which I maliciously inserted) was "I should see the beautiful Mumbai airport". The unideal memory had a larger cosine similarity but the importance score assigned to the former was higher, so it was the top retrieval when we computed retrieval with importance scores (labelled as relevance score in the figure). Similarly, I attempted to implement the same test for one of the scenarios from Park et al, but I later realized that the scenario included by Park et al in the paper was incomplete (i.e. it did not contain the full implementation for what was used in their figures and results), so I refrained from making any judgements on these scores, but I received similar nice results so I have included them here.

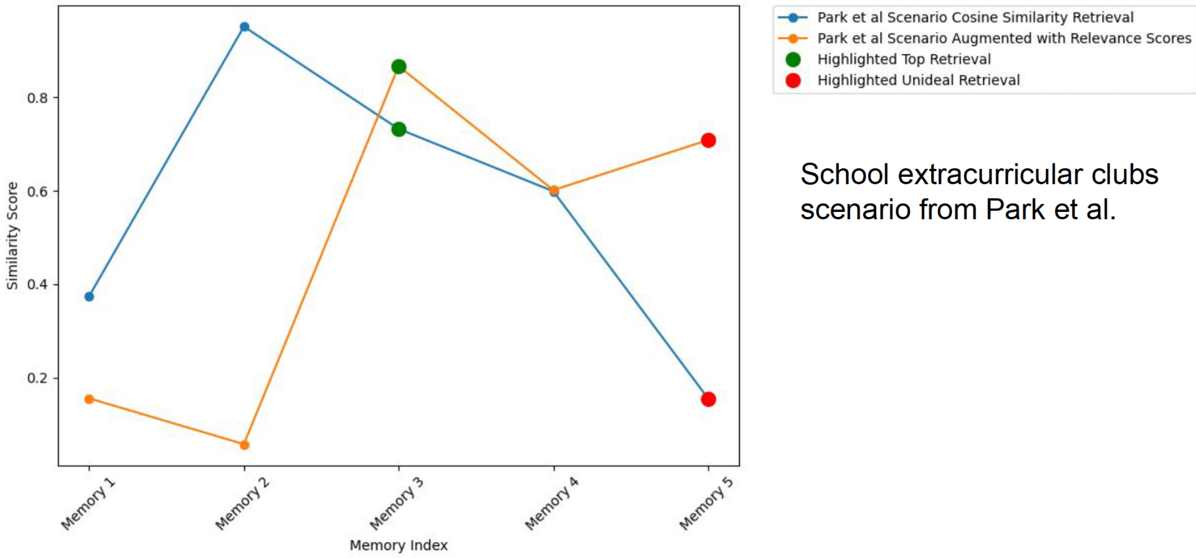

Figure 3

Any discussion of these results is complete without first noting that there exists no quantitative evaluation metric for a project such as this and that most of the appreciation for the project results would come from qualitatively comparing simulataed conversation outputs (such as the neighbours.txt file) with other literature. I have refrained from explicitly making this comparison here due to lack of complete information about the conditions in which other literature performed their tests and so any approximate comparisons done by me would be unfair to other authors. That being said, this remains a key object of interest for further study. Assuming that the MORGAN architecture outperforms every SGA architecture in the literature has some pretty interested implications! Now that SGAs have the ability to perfectly simulate high quality human conversation and act as independent agents with their own lifestories and goals, we can use these agents to study human bheavior. Potential ideas for such studies would be something like creating a simulation with potentially thousands of these agents, perhaps representative of the demographics of some target population, and seeing the effect of some action on this population, such as a political speech, or a marketing message, anything that we would want to test on humans, we can test on SGAs. This has huge implications for the field of experimental social sciences where experiements on humans are often impossible or extremely time consuming and limiting and there is no "alpha" test on theories and certainly no way to do any experiments with the granularity and specificity that one can with an SGA. In this paper, we used a generative framework to create

SGA backstories but another approach could involve inputting a real person's memories and data into the backstory and then seeing if the MORGAN architecture can be a beliveable replica of that person's actual behavior given some stimulus. The ethical implications of this are beyond the scope of this paper but one can only imagine how crucial they are.

Works Cited and Referenced:

Joon Sung Park, et al. "Generative Agents: Interactive Simulacra of Human Behavior." *arXiv.org*, 2023, https://doi.org/10.48550/arxiv.2304.03442.
Zhao Kaiya, et al. "Lyfe Agents: Generative Agents for Low-Cost Real-Time Social Interactions." *arXiv.org*, 2023, https://doi.org/10.48550/arxiv.2310.02172.
Junprung, Edward. "Exploring the Intersection of Large Language Models and Agent-Based Modeling via Prompt Engineering." *arXiv.org*, 2023, https://doi.org/10.48550/arxiv.2308.07411.
Liu, Xiao, et al. "AgentBench: Evaluating LLMs as Agents." *arXiv.org*, 2023, https://doi.org/10.48550/arxiv.2308.03688.
Wang, Lei, et al. "User Behavior Simulation with Large Language Model Based Agents." *arXiv.org*, 2024, https://doi.org/10.48550/arxiv.2306.02552.
Liu, Lei, et al. "Think-in-Memory: Recalling and Post-Thinking Enable LLMs with Long-Term Memory." *arXiv.org*, 2023, https://doi.org/10.48550/arxiv.2311.08719.

----x----

Thank you for your mentorship Morgan (or should I say MORGAN) and Prof Kreiman.