Final Report for Computer Systems Research Lab
Simulation of Human Behavior Through LLM-Based Autonomous Agents

Raymond Fu and Gabriel Xu

Date: 6/2/2025

Yilmaz - Period 1

Table of Contents

Abstract

The advent of Large Language Models (LLMs) such as ChatGPT allows for a computer to generate human-like text. Utilizing such LLMs, accurate simulations of human behavior are possible. While existing solutions do exist, they are heavily limited by the money it takes to run LLMs multiple times over the course of a second through an API. In this paper, we introduce a solution which minimizes the amount of resources necessary, enabling it to run locally on a computer. Our project aims to bring accurate proxies of human behavior to the masses, allowing smaller companies and individuals to gain access to such technology by simply running it on their own computers.

I. Introduction

Human behavior is incredibly complex, so an accurate simulation of it comes with many nuances and problems. In implementations of these simulations, there are generally two main parts: the agents and the town. Agents are the people in our simulation. They take up a role in the simulated world, such as teacher, student, cashier, etc. The agents interact with each other, forming a realistic society. The town usually consists of buildings and roads, providing a very basic imitation of a real town.

At the basis of every simulation is a large language model (LLM). LLMs are natural language processing AI models that generate text given a prompt. Famous examples of LLMs include Chat-GPT.

One of the biggest problems in these simulations is memory. Human memory is very nuanced, including aspects such as memory storage, memory retrieval, and forgetting. Another problem is the problem of cost. While previous solutions used thousands of dollars in API tokens [5], our simulation is able to run locally on a client's machine, bringing the cost down to effectively 0.

II. Background

One previous solution we examined was Park et. al [3]. Park et. al featured one of the most intricate simulations. Their simulation had a planning module which we took inspiration from. This planning module would generate a plan for each agent at the beginning of each day, and this plan would evolve as time went on. For their implementation of memory, they used a memory stream that would record everything that happened at every moment of time in the simulation. When retrieving memories, the entire memory stream would be looked through by an LLM, racking up costs very quickly. Their simulation also featured a reflection module, which would condense memories into reflections of the agent. For example, memories of an agent studying could result in a reflection such as “Gabriel is very studious”.

Another solution that we examined is AgentSims by Lin et. al [8]. This simulation cut costs relative to Park et. al by using a vector-based memory storage system. This storage system would tag memories, and by doing so, drastically reduce the number of memories iterated per memory retrieval. Something unique about AgentSims is its user extensibility. It would allow a user to drag in agents and change the profile of agents while the simulation is running.

III. Applications:

By cutting down on the cost of running such a simulation, we allow smaller game developers, scientists, etc. to run such simulations. Although games initially came to mind with this project, it could also be used in fields of study such as psychology and behavioral science.

IV. Methods:

As shown in Figure 1, we use a planning system for our simulation. At the start of the day, each agent creates a general plan using the LLM. As each timestamp arrives, a more detailed plan is then developed. Another vital part of the project is the storage of memory. We used a system where every observation is shown, then separated into categories to make retrieval easier. It can be represented as a dictionary where each category (represented as a string) is mapped to a list of events containing conversations between agents. Our project heavily reduces the demand for computing power of LLMs by storing the memories of agents through a database system, reducing the need to query an LLM with all memories for each decision of an agent.

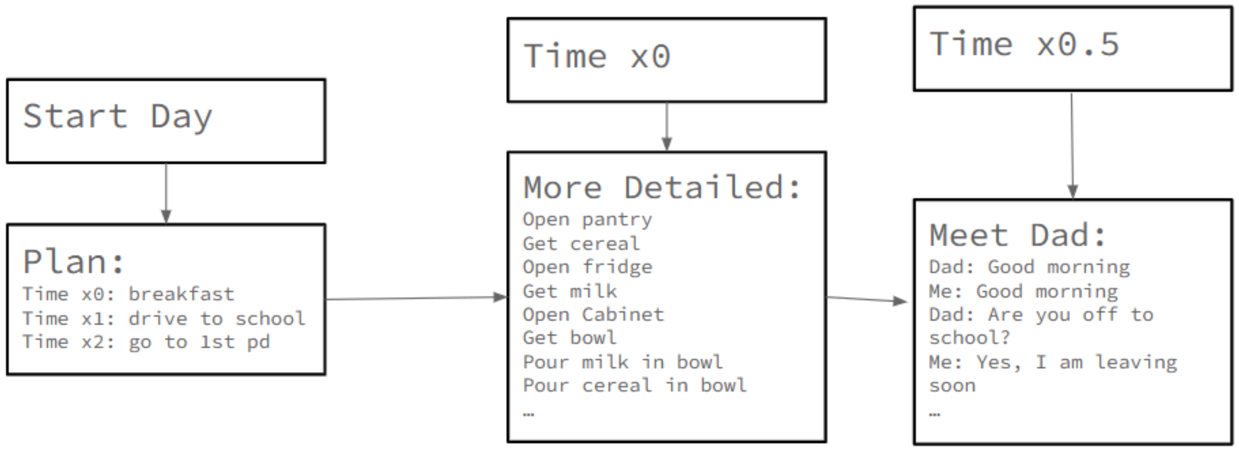


Figure 1. Example Task List Generated for Agent

The only input the project is taking in is a folder of text files containing the name and profile of each agent. The profile describes the background of the character each agent is simulating, including their job, personality, and hobbies. The program then includes this profile in each prompt to the LLM when determining what action each agent to take, creating tasks, movements, and dialogue for the agents at each time of day, which are then rendered as a video for a demo.

In our code, one thing we addressed was the occurrence of LLMs often having hallucinations when it outputs text that is nonsensical or unfaithful to the original prompt, likely influenced by a large influx of data that is difficult to fully process. In a simulation of human behavior, hallucinations would be an agent leaving its role, an agent responding nonsensically to another agent, or anything else that breaks their believability. To address this issue, we had to limit our LLMs to the amount of ticks it would take before they could be called again, as having too frequent calls to them would output less valuable messages, but also lag the simulation as the LLM would not respond in time while the simulation would try to keep moving forward. To execute this, we had an infinite loop that would update the conversation patterns and movements of each agent once their individual counter reached the set amount. This is partially why our implementation only ran for about a full day, as the simulation has a tendency to lag a lot once the amount of memories gets too large.

The buildings were positioned at preset locations such that there would be no overlap, and the LLM decided which buildings would be most important to include in the simulation. When the LLM included output that indicated an agent should move, we queried it once again to decide which building would be best to go to, using the difference between the goal location and the current location as a velocity to define the character's movement.

Conversations were initiated with a conversation check function where the LLM was asked if an agent should converse with another one once they got within range and if the current task and conversation history justified a conversation. When conversations did occur, their contents were rendered to the console using standard output, while the tasks agents were currently working on were rendered through individual dialogue boxes.

V. **Results**:

A video demo of our final result can be found here: https://drive.google.com/file/d/10ziM2JV48DpI5cUCwoU1WN9kHU-xD4NW/view?usp=drive_link. We were able to simulate a realistic day for a group of four people where they properly carried out the tasks that they planned. The agents can be observed to move to the locations associated with their tasks at the appropriate times, interacting and having conversations with other agents.

In the Demo, there are four agents with these profiles:

Alice: You are a student, and you enjoy playing basketball and video games

Mr. Smith: You are a teacher who teaches math and history, and enjoys reading

Bob: You are a cashier, and spends your free time watching TV

Charlie: You are a worker at a construction site and love spending quality time with your family

Figure 2. Agent Profiles in Demo

Here are some snapshots of points in the demo:

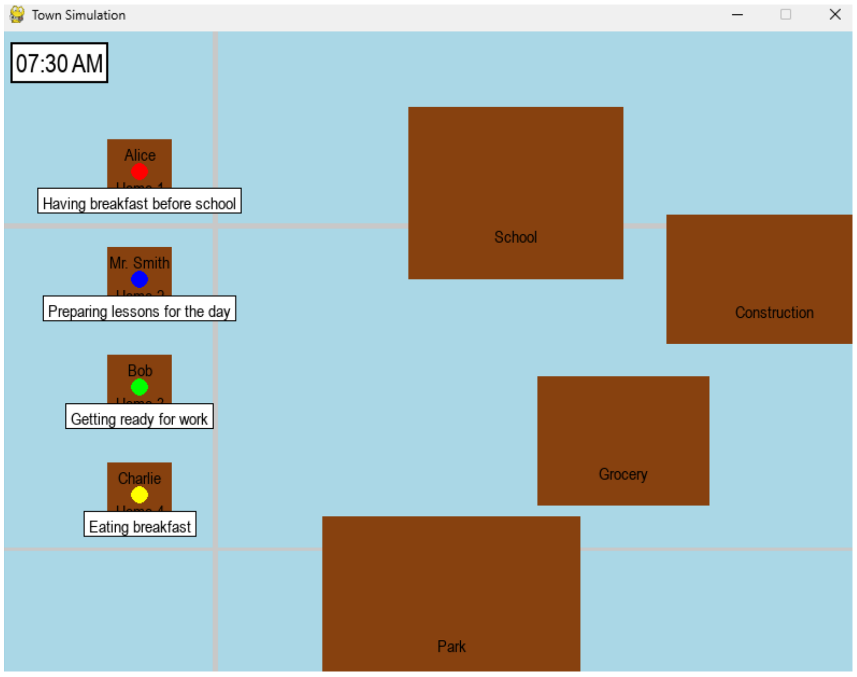
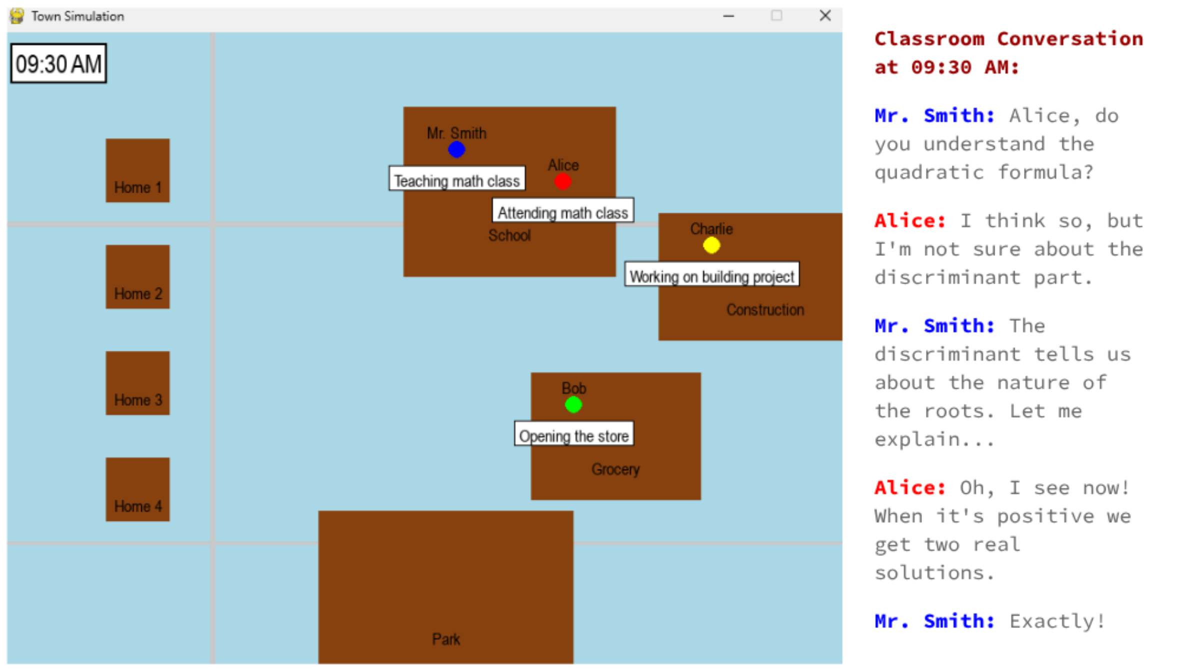


Figure 3. First Snapshot of Simulation



Classroom Conversation at 09:30 AM:

Mr. Smith: Alice, do you understand the quadratic formula?

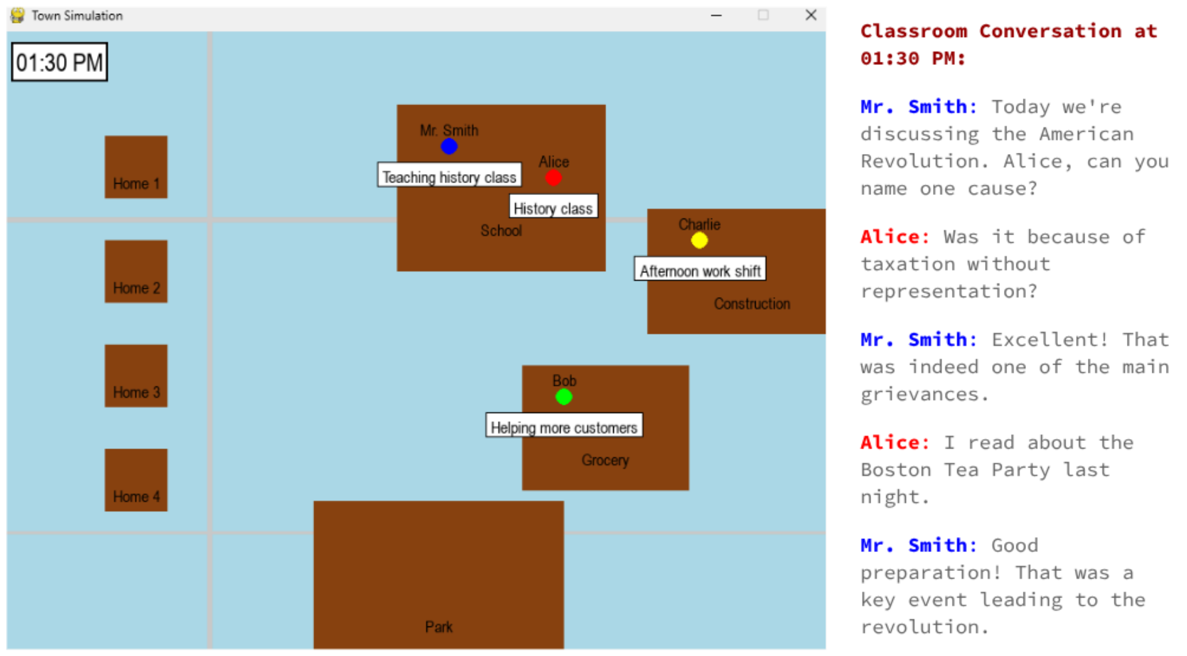
Alice: I think so, but I'm not sure about the discriminant part.

Mr. Smith: The discriminant tells us about the nature of the roots. Let me explain...

Alice: Oh, I see now! When it's positive we get two real solutions.

Mr. Smith: Exactly!

Figure 4. Second Snapshot of Simulation



Classroom Conversation at 01:30 PM:

Mr. Smith: Today we're discussing the American Revolution. Alice, can you name one cause?

Alice: Was it because of taxation without representation?

Mr. Smith: Excellent! That was indeed one of the main grievances.

Alice: I read about the Boston Tea Party last night.

Mr. Smith: Good preparation! That was a key event leading to the revolution.

Figure 5. Third Snapshot of Simulation

VI. **Limitations**:

While the novelty of our project is the fact that no money needs to be spent in order to run the simulation, money quickly became our most pressing limitation. Through the development of the simulation, we needed to constantly balance performance and complexity of the simulation. In the simulation, we were limited to four agents and a very rudimentary imitation of a town.

VII. **Conclusion**:

We accomplished our goal of creating a simulation of human behavior and society without any monetary costs.

VIII. **Future Work and Recommendations**:

In the future, we would like to experiment with different environments. While we only tested in this very basic town, we would like to test more complex environments, such as a city. We would also like to try using such methods of behavioral simulation in a video game. For others who want to continue on our progress, one idea would be to properly implement the reflection system by occasionally aggregating the memories into a prompt for the LLM that will provide an updated profile text file. Another idea would be to implement the idea of forgetting less relevant and older memories to prune the size and fix certain memory issues.

Something that we wish we spent less time on was the game itself. When we first started, we debated between using PyGame and Godot, an open-source and free game engine. Although our final project was done using PyGame, we believe that using an actual game engine could have made it look nicer. However, in the end, the project is meant to be focused on the simulation itself rather than the presentation of it. For the people that work on this project in the future, try to spend minimal time on the game aspect.

Another important aspect in our completion of the project was our computer. We were able to run the Llama 3.1 locally because of our computer, as it will not run on a school computer and many other machines. For reference, we used a 4070 Super with 32 GB of memory. If the machine is not up to par, effort spent on the project will be for naught.

With regards to the LLM itself, it could be interesting to tweak its parameters, seeing how the agents behave differently with different parameters. Due to time constraints, we were unable to do such experimentation.

IX. **Materials:**

For the LLM, we use Llama 3.1 with the Ollama [2] Python package. The project is run entirely locally on a computer. We use PyGame [3], a Python game development library, to develop the environment/visuals.

References

[1] Fu, R. & Xu, G. (2025). GitHub Link. https://github.com/Almondbubby/syslab.
[2] Ollama (2024). https://ollama.com/.
[3] PyGame (2024). https://www.pygame.org/wiki/about.
[4] Wanjun, Z. et al. (2023). MemoryBank: Enhancing Large Language Models with Long-Term Memory. https://arxiv.org/pdf/2305.10250.
[5] Park, J. et al. (2023). Generative Agents: Interactive Simulacra of Human Behavior. https://arxiv.org/pdf/2304.03442.
[6] Wang, Z. et al. (2023). Self-consistency Improves Chain of Thought Reasoning in Language Models. https://arxiv.org/pdf/2203.11171.
[7] Wei, J. et al. (2023). Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. https://arxiv.org/pdf/2201.11903.
[8] Lin, J. et al. (2023). AgentSims: An Open-Source Sandbox for Large Language Model Evaluation. https://arxiv.org/pdf/2308.04026.

Appendix

I. Link to Code: https://drive.google.com/file/d/15K52eV2CqgqOrNBaJLVUvjTDsuP-SP1z/view?usp=drive_link