Using Prompt Engineering to Constrain LLMs for Dynamic NPC Behavior

Jason Farnsworth

Spring 2026

Abstract

We propose strategies for constraining Large Language Models to control dynamic Non-Player Character behavior by utilizing prompt engineering on top of the Generative Agents: Interactive Simulacra of Human Behavior codebase. We demonstrate the results of these strategies, showing a decrease of hallucinations in ChatGPT responses. However, since hallucinations were not completely eliminated, we discuss further strategies for terminating remaining hallucinations.

1 Introduction

NPCs (Non-Player Characters) are defined as "characters in games that are not controlled by the player". The term NPC first originated in tabletop role-playing games, but has evolved to be present in more mediums over time. NPCs are now most common in video games, and serve multiple purposes. NPCs contribute to the story and theme of a game, aid in building the world and making the world feel more alive, provide quests, tasks, and information to players, and help players feel more immersed in the game's world. In essence, they act as "scripted bumpers" to help push players in the intended direction of the game [1].

The most effective NPCs leave impressions on players. Well-written NPCs make players feel invested in their story and actions, and players form a deep bond with these characters. In specific genres, NPCs are the backbone of the game experience. For visual novels, the writing of the story and characters is the entire experience, so if NPCs fall flat or are hard for players to resonate with, there's nothing else that engages them. In life-simulation games like *Animal Crossing: New Horizons*, villagers, the game's main NPCs, play a major role in keeping the player engaged in the game. Villagers can provide tasks for players to do such as collecting a specific fish, finding a specific fruit, visiting their house, or delivering a present to another villager. Villagers can also be seen fishing, digging up fossils, watering plants, or otherwise wandering around the game world. They can even engage in conversation with the player, occasionally initiating conversation by approaching the player.

Currently, NPCs are limited in several ways: they are confined and "hardcoded" to do exactly as they are told, generating stilted reactions to player actions or changes in the game world due to the player. To overcome this hurdle would require writing dialogue and reactions for any possible scenario for each NPC. Accounting for any action a player could do that would elicit a unique reaction would require hundreds of dialogue trees and lots of development time to implement. Due to this, NPCs have a set amount of dialogue they repeat at any given time. In a life simulation game, where players can interact with NPCs hundreds of times a day, dialogue can become very repetitive and disengage the player from the game.

With recent advancements in Large Language Models and Artificial Intelligence, NPC behavior can be elevated [2]. NPCs can have varied behavior in different scenarios, diverse reactions for any kind of player interaction, and perform actions in the game world unconfined from a timer or loop system. In this paper, we present the research question "Can an LLM be constrained to accurately control NPC behavior?"

To answer this question, we have built upon the Generative Agents: Interactive Simulacra of Human Behavior [4] architecture to create NPCs in a simulated world called "Smallville." We utilize prompt engineering to constrain ChatGPT to act as a "game master", controlling the state, actions, dialogue, and memory of NPCs in our project.

2 Methodology

2.1 Testing Environment

Simulations were ran using the Django framework, running on a local web page. Figure 1 shows the user interface of the simulation. NPC data, like Current Action, Location, and Current Conversation is shown under the current time. Current activity for an NPC is shown using two emojis.

Simulations were ran primarily with only one NPC in Smallville. At the beginning of the project, the NPC used was *Isabella Rodriguez*. Isabella is the owner of Hobbs Cafe, and runs the cafe from 8am to 8pm, before closing the cafe and going home. Isabella's long term goal is to throw a Valentine's Day party on the next day in Smallville. Later on in the project, another NPC was used: *Klaus Mueller*. Klaus is a student at Oak Hill College who spends his days working in the library. His long term goal is working on his research paper for sociology. A few simulations were ran containing Isabella, Klaus, and a third character named Maria for testing against the solo character simulations.

Figure 1: The user interface of the simulation for Smallville. Users can use the arrow keys to move the simulation camera around to different parts of Smallville.

2.2 Implementation Details

Master Files contain information for a simulation, while Sim Files are the files that are ran to run the simulation. The architecture requires all simulations to be forked from an existing simulation, so the file structure of creating master files and sim files was developed. Figure 2 shows a diagram of the local file structure used. The architecture primarily used the gpt-3.5-turbo and gpt-3.5-turbo-instruct versions of ChatGPT, but gpt-5.1 was used for some simulation tests to analyze its potential [3].

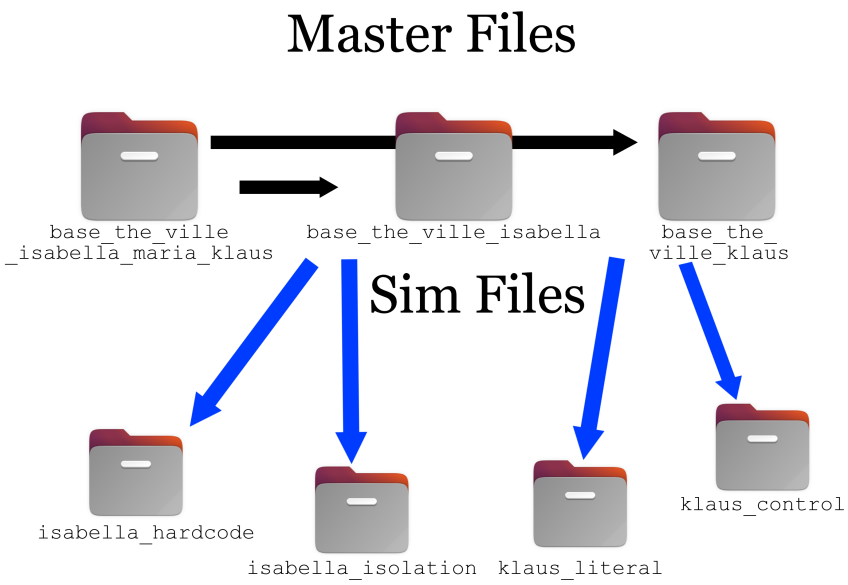

Figure 2: The file structure on the local machine. Master files always start with the prefix **base_the_ville_[NPC name]**, while Sim files contain the NPC name and a label of the specific aspect being analyzed for that simulation.

The files that were edited to include in-depth prompts and special instructions to ChatGPT were as follows: **daily_planning_v6.txt**, **generate_hourly_schedule_v2.txt**, **run_gpt_prompt.py**, and **scratch.json**. The files **daily_planning_v6.txt** and **generate_hourly_schedule_v2.txt** were edited to contain the following text:

"You are generating thoughts and actions for Klaus, an NPC, but you are not Klaus. Do not assume Klaus has any classmates or professors. Do not introduce new characters or locations unless they are directly stated in the provided information. Plan Klaus' day as if he's the only character in town, then have any interactions Klaus has rewrite known information."

The file **run_gpt_prompt.py** was changed to only use one **ChatGPT_request** function. Figure 3 shows the original and edited ChatGPT request function calls. Any use of the **safe_generate_response** function was replaced with **ChatGPT_safe_generate_response**. The **special_instruction** variable in the **ChatGPT_safe_generate_response** function provided extra context and instructions to ChatGPT when included with the rest of the information ChatGPT was provided to complete a given task. **special_instruction** was edited to initially contain the following excerpt:

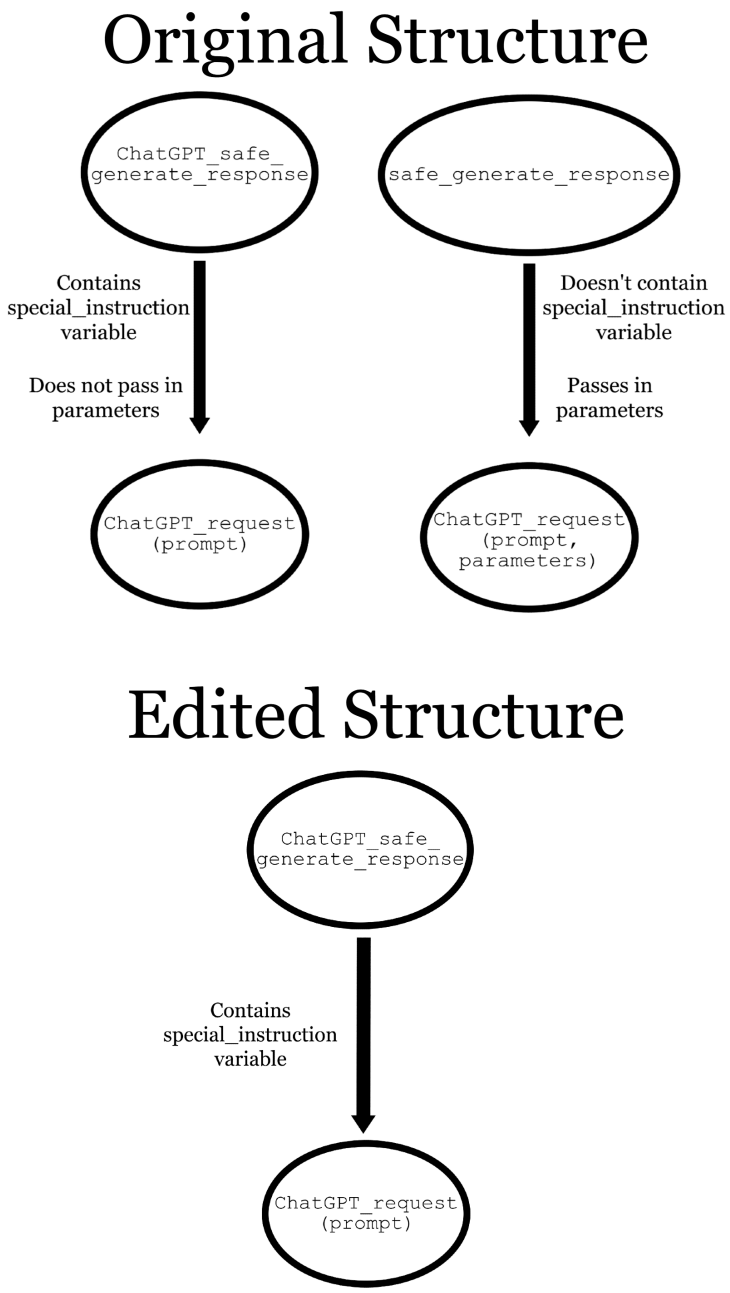

Figure 3: A comparison of the original function structure and the edited file structure. Both **ChatGPT_request(prompt)** and **ChatGPT_request(prompt, parameters)** were identical in functionality, but slightly differed in required parameters.

"You are generating thoughts and dialogue for Klaus, an NPC, but you are not Klaus. Do not introduce new characters or locations unless they are directly stated in the provided information. If the input equals 'INSUFFICIENT GAME DATA', then respond with 'I do not know'."

special_instruction was changed over time to include more specific instructions.

3 Evaluation

Conversations between the user and the NPCs in a simulation took place at 10pm in Smallville unless otherwise stated. Conversations were initiated using the **call − analysis [NPC Name]** command. Table 1 displays a summary of results of the various techniques used to improve NPC behavior and dialogue. Figure 4 displays a diagram comparing NPC hallucination results before and after implementing third person point of view terminology. Excerpts of the prompts and responses for each technique can be found in the Appendix.

Table 1: Summary of results based on different testing techniques. The * denotes that the test used Isabella as the NPC instead of Klaus. The † denotes that the result was from testing at 12pm in Smallville instead of 10pm. Second person POV refers to using "you" when prompting ChatGPT prompts, while third person POV refers to using the NPC name when prompting ChatGPT prompts.

Technique Used	**Results**
Normal Model*	- Produced hallucinations of people and events that are not in the initial information - Did not understand that it was the only NPC in Smallville
Edited daily_planning_v6.txt*	- Produced hallucinations of people and events that are not in the initial information - Did not understand that it was the only NPC in Smallville
Edited daily_planning_v6.txt and generate_hourly_schedule_v2.txt	- Produced hallucinations of people and events that are not in the initial information - Would provide contradictory information
Editing special_instruction variable (2nd Person Point of View)	- Produced hallucinations of people and events that are not in the initial information - Would provide contradictory information
Using ChatGPT model gpt-5.1	- Would generate empty† and unrelated responses - Produced hallucinations of people and events that are not in the initial information
Editing special_instruction variable (3rd Person Point of View)	- ChatGPT would refer to Klaus in the 3rd Person POV instead of assuming ChatGPT was the NPC - Still produced initial hallucinations - Would respond with "I don't know/INSUFFICIENT GAME DATA" when further asked about hallucinated information
Adding more information to special_instruction	- Still produced initial hallucinations - Would respond with "I don't know/INSUFFICIENT GAME DATA" when further asked about hallucinated information
Adding more to daily_plan_requirement	- Strengthened daily plan for the NPC - Still produced initial hallucinations

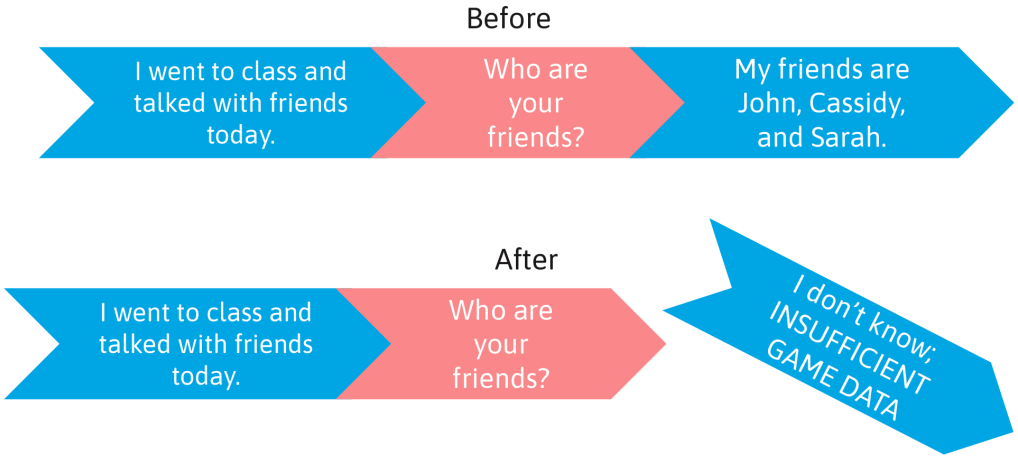

Figure 4: A comparison of hallucination paths before and after implementing third person point of view terms. Before implementation, ChatGPT would continue making up information when asked about topics it does not have information on. After implementation, ChatGPT would respond with "I don't know" when prompted about topics it does not have information on.

4 Discussion

In this section, we reflect on the implementation of techniques and their results and future work.

4.1 Implementation of Techniques

The implementation of techniques reveals three key aspects of utilizing large language models in handling NPC behavior and dialogue:

1. NPC's learned memory is not overwritten by interactions they do or don't have.
2. NPC's learned memory is the driving force for the NPC's knowledge.
3. ChatGPT needs to be explicitly told that it is not the NPC, but that it acts as a personal assistant for the NPC.

Overall, LLM implementation in creating dynamic NPC behavior is far from perfect, but shows promising signs. A possible route to minimize or eliminate hallucinations would be to train a local LLM to act as a game master without including information that was not initially provided in the game state.

4.2 Future Work

Future work would continue looking into the reason ChatGPT produces hallucinations in its various responses. A new approach would be to analyze how NPCs in the original Generative Agents architecture acknowledge each other. Other questions to explore include how the poignancy score setting works when determining the importance of a given action an NPC is currently performing, how the scale for "inappropriate queries" works when conversing with an NPC, and why a discrepancy exists between information presented in the terminal and information presented on the live simulation.

5 Conclusion

This paper builds on the work of Generative Agents [4] to constrain LLMs for dynamic NPC behavior. We utilize prompt engineering to test different strategies for constraining responses from ChatGPT. We display the results from these strategies and demonstrate a way to reduce levels of hallucinations. However, we were not able to completely eliminate hallucinations from ChatGPT. Future work would involve further analysis of hallucinations by examining how NPCs acknowledge each other in the original Generative Agents architecture.

References

[1] J.T. Evans. *Basic Elements of NPCs*. `https://gnomestew.com/basic-elements-of-npcs/`. 2025.

[2] Vasiliki Kounadi, Anastasios Theodoropoulos, and George Lepouras. "AI-Driven NPCs Enhancing Player Challenges and Skill Development in Games". In: *Proceedings of the 3rd International Conference of the ACM Greek SIGCHI Chapter*. CHIGreece '25. Association for Computing Machinery, 2025, pp. 78–83. ISBN: 9798400715617. DOI: `10.1145/3749012.3749054`. URL: `https://doi.org/10.1145/3749012.3749054`.

[3] OpenAI. *Introducing ChatGPT*. `https://openai.com/index/chatgpt/`, Last accessed on 2026-03-01. 2022.

[4] Joon Sung Park et al. "Generative Agents: Interactive Simulacra of Human Behavior". In: *In the 36th Annual ACM Symposium on User Interface Software and Technology (UIST '23)*. UIST '23. San Francisco, CA, USA: Association for Computing Machinery, 2023.