Conversational Multi-agent Symposium; Addressing Model Sycophancy and User Perception

Liam Houston

ACSHF, Michigan Technological University

CS 5090: Discourse Dynamics

Dr. Leo Ureel

December 10, 2025

Abstract

Large Language Models often prioritize user satisfaction over factual accuracy, leading to sycophancy and hallucinations that degrade the reliability of interactive simulations. This paper presents a prototype multi-agent system, developed as a graduate capstone project at Michigan Technological University, designed to mitigate these issues through architectural constraints rather than standard prompt engineering. By utilizing a novel “Two-Pass” grounding mechanism and “Granular Retrieval-Augmented Generation,” the system decomposes narrative authority into distinct, knowledge-restricted personas that must validate user input against a rigid database before responding. The resulting prototype demonstrates that distributing agency across multiple constrained agents effectively enforces information asymmetry and significantly reduces model agreeableness, offering a viable framework for creating more objective and grounded conversational AI tools.

Generative AI Transparency Statement

Gemini 3 Pro was used in the writing of this paper to draft the wording for sections requireing the use of technical wording. This primarily involved the section on system architecture. All AI informed content was evaluated and edited before being used. The same model was used to generate the Abstract for the paper. All other content is the product of human authorship.

The goal with almost all conversational applications of Large Language Models (LLM) is to have a conversational partner that can engage with the user while also meeting additional requirements of being factually accurate and achieve user end goals. Within many applications of these LLMs the issues of model sycophancy and bias remain prevalent. These issues can result in unsatisfactory outcomes for users seeking an accurate and objective tool.

This paper details the development of a prototype system designed to mitigate model sycophancy and provide the user with more agency through a multi-agent system. The central hypothesis that drove the development of this system is that a conversation with multiple agents with differing personas would result in less model sycophancy, and less conversational bias, than a single-agent system.

Through the creation of distinct digital personas and restricting their access to specific subsets of a Retrieval Augmented Generation (RAG) vector database, this system provides the opportunity for users to engage with multiple answers to the same prompt. This framework for the system helps to avoid undetectable bias through providing the user with multiple, parallel outcomes. Agents in this system are designed not to please the user, but to adhere to their defined roles and available data.

This project also serves as an exploration of practical applications of custom AI system design principles within a graduate course at Michigan Technological University. Beyond the theoretical application to mitigate sycophancy, the development of the symposium system focused on developing skills and understanding of LLMs through the use of custom python scripting and the Ollama commercial program.

Background and Motivation

The concept of multi-agent systems already have some backing in academic literature. This literature served as the intellectual basis for the design of the symposium system. A study by Gamage et al. (2024) introduced a multi-agent RAG chatbot designed to support human agents in net-zero emissions energy systems. The application demonstrated in this paper proved the effectiveness of specialized agents in facilitating efficient information retrieval in complex contexts.

Another study by Ding et al. (2023) detailed the design of an expert system referred to as DesignGPT. This particular system used multiple agents to simulate different roles in a conversation,

and allowed for human designers to collaborate naturally within a larger conversation with multiple perspectives informing their interactions. The results of this implementation were positive, and show the potential benefit in using such a collaborative tool over a single-agent tool for improving user performance and perceived value.

Both of these studies support the merit in developing a multi-agent conversational model for certain applications. From these papers, the basic structure and desired function of the symposium system were derived. The mechanisms to solve issues with sycophancy and bias issues were expanded upon the initial framework.

Methodology and System Architecture

The system framework operates as a local, offline application to take advantage of the Ollama program and make a custom system without concern of potential barriers to development like API costs. The core architecture relies on Python 3.10 for the orchestration code, ChromaDB to achieve the RAG knowledge base, and Ollama for serving the Llama 3 model. The system design consists of a main orchestrator code that calls agents from a separate file—which defines agent behavioral logic, temperature, and sets up the "Two-Pass" grounding mechanism. Agents call to a separate knowledge base for reference information. Two primary mechanisms characterize the current system: the "Two-Pass" grounding architecture and the implementation of metadata-based granular permissions.

Two-Pass Grounding Mechanism

The test scenario for the system helped to uncover a significant technical hurdle. Agents—in the context of a role-playing game—had a propensity for hallucinating information that did not exist in the vector database. This issue arose from the need for the agents to generate engaging responses while also being constrained to what information actually exists in the knowledge base. The issue was solved by implementing a "Two-Pass" mechanism that splits agent processes into two sequential phases. The first phase, termed "The Detective," utilizes a call to the LLM with a 0.0 temperature. This phase analyzes the user's prompt exclusively to extract keywords, ignoring any emotional content or narrative framing. By stripping away the conversational context, the Detective functions as a deterministic search engine query generator.

The second phase, termed “The Responder,” receives the output of the database search. Crucially, the system implements a code-level override during this phase. If the database returns a flag for no result, the system injects a rigid instruction into the Responder’s context window, explicitly commanding the model to admit ignorance. This programmatic intervention physically prohibits the model from generating a hallucinated response.

Granular Data Permissions

Unlike standard RAG systems where all vectors are equally accessible, this project aimed to establish more effective conversational agents through allowing each agent private data to diversify their abilities to respond to user queries. This feature was achieved through a permission-based retrieval system. During the ingestion of textual data, files are tagged with metadata corresponding to specific agent roles. For example, a file named “tactician_data.txt” is tagged with a permission string restricting access to the Tactician agent.

When an agent initiates a search, the retrieval function performs a filtered query. The system verifies the agent’s identity and restricts the search scope to documents matching that identity or documents tagged as public. This mechanism allows an agent to provide additional and unique context for a query that involves knowledge specific to their specification. This mechanism is also used to ensure an agent cannot agree with a user’s false premise regarding secret knowledge, simply because the agent cannot “see” that knowledge. If a user asks an agent to confirm a detail about data in their private access, the retrieval system returns an empty result, forcing the agent to abstain from answering concretely and defer to other agents.

Focus Masking

To further mitigate context contamination, the system employs a technique referred to as Focus Masking. In early iterations, feeding the entire conversation history into the retrieval prompt caused agents to fixate on past events. By separating the input streams, the system ensures that the “Detective” phase sees only the immediate user prompt, preventing it from re-searching previously resolved topics. Simultaneously, the “Responder” phase sees the full history to maintain narrative continuity. This separation of concerns intends to make the system agile in knowledge searches while maintaining the

flow of the conversation.

Agents

In order to make agents easier to customize, a separate file for agent structure and definition of personas. The current upper limit of agents is unknown. Three agents were defined for the sake of evaluating the success of the code in this project. Two of these agents were active agents instructed to engage with user. The third agent was called the “Narrator”, who’s function was to summarize the outputs from the other agents. This narrator was intended to streamline the system reporting on what the player agents had decided in any response—helping the user take their next step and add the potential to develop a log of the conversation for reference if ever desired. Such a logging function was never successfully implemented, but would be desirable in future iterations as a method to create a shared memory of events for the player agents and give a summary report of entire interactions to the user for evaluation.

Implementation and Technical Challenges

The realization of this prototype faced several practical challenges. The primary challenge faced was the fact that the sole developer was rather novice in their knowledge of both python and the components of an AI model. This resulted in the project being almost entirely vibe coded through the use of Gemini 3 Pro—provided for free to Michigan Tech students. Once code had bee generated and fulfilled the basic functions desired, some minor edits were made by the developer. The quality and effectiveness of the final prototype is inherently effected by the lack of clear direction and ability to diagnose generated code throughout development.

The latency resulting from running sequential inference chains on a single device was also a major challenge. To address this, the Tkinter user interface was decoupled from the main execution thread using Python’s threading library. This was done to ensure that the computational load of the “Two-Pass” system did not freeze the application window, maintaining a responsive user experience.

Testing System Functions

With a lack of defined application setting, the symposium system was designed around playing a simple role-playing game—like Dungeons and Dragons but with significantly less structure. This role-

playing context was chosen for the purpose of driving development toward a defined use case and providing context for testing the successful implementation of key mechanisms. No formal methodology was used to direct testing. Testing consisted of several ad hoc sessions probing system agents to uncover limitations and shortcomings in mechanism functions.

Discussion

The finalized prototype proved successful at achieving the basic goals for its function. The prototypes also provides compelling qualitative evidence that a multi-agent conversational system can successfully address the issues of model sycophancy present in single-agent systems. Agents were successful in first making knowledge checks to the knowledge base before producing responses, correctly identifying and abstaining from answering questions when they did not have the information to do so accurately.

Informal walkthroughs were able to prove the success of the “Two-Pass” mechanism as well as the secret knowledge function within the knowledge base in the current version of the prototype. Agents in the model were moderately reliable at reporting when they did not have information to answer a question. There were some cases where hallucinating information that did not exist still occurred—although these cases were consistent to one persona and could not be consistently reproduced.

These interactions highlighted the core value of the developed system. Through removing the conversational model from direct interaction with the user, and defining additional logical constraints within a multi-agent system, the intended-as-helpful behavior that results in model bias and sycophancy is redefined.

Limitations

This prototype is not without limitations. The latency introduced by multiple sequential calls results in a real-time interaction that is always slower than a standard chat bot. The issues of hallucination and sycophancy were also not entirely resolved through this implementation. While the occurrence of these issues was better mitigated through hard set rules placed on the model, hallucination is still possible since there is no distinction as to what information is allowed to be improvised upon when absent from the knowledge base. This is a problem potentially resulting from

the role-play game setting used. The choice of the Llama 3 model also presented specific challenges. The prototype has been built to use this model and would likely be incompatible with a different model without some alterations.

Despite these limitations, the prototype system demonstrates that the guardrails used to place strict constraints on an AI model for the purpose of creating a conversational symposium is a viable method. The prototype is also able to achieve its intended results with some degree of reliability despite these limitations.

Possible Applications and Next Steps

The test scenario of a role play game served it purpose as a structure behind development, but is not the intended application setting for this digital symposium. The primarily vibe-coded application which was developed is also potentially flawed in the logic it applies to achieve desired functions. A proper diagnosis and redevelopment of the code under a concretely defined methodology, with a clear approach and defined outcomes, is required to actualize the intent of the digital symposium correctly.

The intended application of a digital symposium as an aid to diminish risks of single model bias and sycophancy and to foster additional user agency through the discourse of multiple agents should be the eventual goal of future iterations of the symposium that come out of this prototype.

Conclusion

This project explored the practical application of agentic workflows to address the inherent limitations of single-agent Large Language Models, specifically the prevalence of sycophancy and conversational bias. By developing a local, multi-agent prototype, the research validated the hypothesis that architectural guardrails are more effective than simple prompting in maintaining narrative and factual consistency. The implementation of “Granular Data Permissions” ensured that agents could not agree with false premises outside their knowledge scope, while the “Two-Pass” grounding mechanism provided a robust defense against hallucination by programmatically forcing the admission of ignorance when data was absent.

The development process was heavily influenced by the educational context of the project. Starting with limited prior experience in Python and machine learning architecture, the realization of this

system relied on iterative experimentation and AI-assisted coding. Despite the resulting inefficiencies and the lack of a rigorous testing methodology, the functional prototype serves as a successful proof-of-concept. It demonstrates that complex behaviors can be engineered by orchestrating smaller, local models with strict logical constraints.

Ultimately, this work highlights that the path to more reliable AI systems may not lie in larger models, but in better systems design. Future iterations of this framework, developed with more robust engineering practices, hold significant potential for applications beyond role-playing games—specifically in domains where objective analysis and diverse perspectives are critical to decision-making. The project stands as a testament to the value of hands-on application in bridging the gap between theoretical AI concepts and functional software engineering.

References

Chase, H. (2022). LangChain: Building applications with LLMs through composability. Software. https://github.com/langchain-ai/langchain

Chroma. (2023). ChromaDB: The AI-native open-source embedding database. Software. https://www.trychroma.com/

Ding, S., Chen, X., Fang, Y., Liu, W., Qiu, Y., & Chai, C. (2023). DesignGPT: Multi-Agent Collaboration in Design. 2023 16th International Symposium on Computational Intelligence and Design (ISCID), 204-208.

Gamage, G., Mills, N., Silva, D.D., Manic, M., Moraliyage, H., Jennings, A., & Alahakoon, D. (2024). Multi-Agent RAG Chatbot Architecture for Decision Support in Net-Zero Emission Energy Systems. 2024 IEEE International Conference on Industrial Technology (ICIT), 1-6.

Ollama. (2023). Ollama: Get up and running with large language models locally. Software. https://ollama.com/