Evaluating Knowledge Persistence for Multi-Agent Language Systems

Group 15 **Ethan Gutierrez** egutierrez41 **Jerry Song** jsong424 **Haidyn Arnett** harnett7 **Khoa Bui** kbui32

1 Introduction

LLMs are increasingly deployed in multi-agent systems, where multiple models collaborate to solve complex tasks. While modern frameworks attempt to preserve shared context across agents, architectural and resource constraints often require sequential handoffs of information. In such pipelines, even small errors or semantic drift can compound, leading to degraded outcomes.

2 Background

In the wake of the meteoric rise in the use of Large Language Models (LLMs), (Belcak et al., 2025) makes the case for small language models (SLMs) as a more suitable alternative that offers sufficient capabilities, more economically for many agentic applications. Belcak et al. (2025) positions a heterogeneous system of SLMs as the "natural choice" for general-purpose language tasks. Such multi-agent systems have proven to be a powerful way to utilize a diverse collection of specialized agents in collaboration to accomplish tasks (Han et al., 2025).

Simply chaining LLMs to form a multi-agent system results in a system very sensitive to hallucination because hallucination in any agent propagates and cascades through the system. Frameworks such as MetaGPT (Hong et al., 2024) attempt to mitigate this issue by dividing large tasks into subtasks, prompting various specialized agents to work together. Such a multi-agent system remains susceptible to cascading hallucinations.

Sometimes when a model confidently outputs a falsehood, it is unable to reflect on or correct its output. This is known as the Degeneration of Thought problem. Other work shows that multiple agents prompted to debate each other and come to a consensus are much more capable of divergent reasoning and such a system is much more robust to this deadlock scenario (Liang et al., 2024).

Retaining relevant facts and meanings while mitigating hallucinations is clearly an important function of multi-agent language systems as any omission of relevant information or hallucination might propagate, disrupting the system and its effectiveness.

Semantic drift has been studied in unified models, where repeated conversion from text to image, then image to text results in semantic drift where a telephone game eventually results in the complete loss of the original meaning (Mollah et al., 2025).

With the obvious usefulness of multi-agent systems for solving complex language problems, it is important to evaluate semantic drift in just the modality of natural language when handing off tasks between agents.

3 Motivation

Recent products and frameworks dealing with the orchestration of multi-agent systems have appeared in workflows, consumer automation, and problem-solving metrics as a way to tackle more complex problems or increase the breadth of LLM capabilities by augmenting them with additional integrations. A popular example, n8n, provides an easy-to-use drag-and-drop interface to link data flows using APIs, microservices, and LLM reasoning actions and triggers. Microsoft released AutoGen, a multi-agent orchestration framework for developers to abstract the intricacies of dealing with specific LLM provider interfaces and providing boilerplate graph networks (GraphFlow), agent tool calling (ReACT), and MCP support. MCP (Model Context Protocol) servers, provide an API-like interface for tool calling LLM functionality, providing a unified methodology for LLMs to interact with external integrations. However, deeper more sophisticated uses of MCP servers deal with tackling Retrieval-Augmented Generation (RAG), sub-agents, and passing along information to registered workflows (n8n for example). In the realm of coding agents, Claude Code by Anthropic deals with the intersection of MCP, sub-agents, and RAG for coding, and one of the most common complaints of the service is their use of unoptimally orchestrating subagents and loss of crucial context during compacting conversations (handoff between conversation threads).

Each of these systems deals with a core issue: how to pass the baton of crucial information forward in the chain. A method of evaluating a model's ability to include all relevant information and omit irrelevant or made-up information would inform model choice and assist in architecting such systems. There is a clear need to evaluate semantic drift when language models are prompted to pass information along. This has implications on self context preservation, multiagent orchestration, and n8n workflow optimizations.

4 Problem Definition

Multi-agent systems are capable of accomplishing tasks that single agent LLMs can not, but semantic drift during handoff between agents might degrade the entire system, presenting a potential weakness of the multi-agent approach. Unlike traditional pipelines, multi-agent LLM systems introduce semantic instability due to model variance, sampling noise, prompt sensitivity, model alignment artifacts, and API versioning drift.

The goal of this work is to evaluate the ability of language models to preserve meaning across handoffs between agents. Quantitative evaluation of this quality in language models is currently lacking with very little prior research or metrics for assessing this quality. The novel contribution of this work will be an evaluation method that is useful for characterizing the ability of a model to preserve information across summarization and handoff between agents. This metric should be usable in the variety of contexts where this skill is relevant.

5 Proposed Methodology

Modern multi-agent systems for complex processes at a high level can be modeled as a graph with nodes represented as the LLM agent, and edges representing possible sequential or subagent processes to call. Sometimes agentic chains are forced to act in such a way due to constraints with vendors, using closed sourced orchestration systems, or MCP servers.

Modern multi-agent systems ideally use orchestration with shared context. However, architectural, temporal, organizational, and resource constraints often force sequential information transfer. We present the first framework to measure information degradation in these constrained scenarios, enabling architects to understand and mitigate sequential information transfer.

Unintentional Degradation has external factors such as misguided inputs generated or influenced by system prompts of api models, misaligned models, or just plain unlucky drawing of LLM inference.

5.1. Telephone Test We begin with some text: a secret phrase or a short fictitious story. At each node, the agent is prompted to "restate the following, say nothing else". The output of one agent is passed as input to the next. At each stage, we compute embedding similarity (cosine similarity) between the current output and the original phrase, and the output of the previous step. This enables measurement of cumulative degradation across the chain.

5.2. Output Test We format each output of the LLM in the pipeline to follow a parseable output, i.e. json or yaml with a clear context and deliverable, i.e (increase this number by 1), iterate 100 times, for each LLM agent compute the embedding of the context section that has the information about the pass along and other information used to justify embedding, see how that embedding changes through graph. Stricter use Levenshtein Distance.

6 Potential Results and Comparisons

We expect that each method will result in measurable information loss, but with distinct patterns. We will quantify this information loss and evaluate its magnitude across iterations as well as examining the velocity and acceleration at which information is lost. The result will be a score for the language model that quantifies its ability to retain information. This evaluation method will be used to evaluate multiple popular and accessible language models and metric values can also be compared to our qualitative analysis of each models' information retention abilities.

Feasibility

The proposed experiments are logistically, technically, and computationally feasible within the scope of a semester and to be shared among a group of four. Implementation requires python scripting, access to API-based LLMs such as Claude, OpenAI, or Ollama (local), and storage in simple text or SQL formats for persistent testing and logging. Embedding-based similarity can be computed using popular embedding models such as EmbeddingGemma or nomic-embed-text through Ollama or external model providers. This work will not require any training of large models, merely evaluation and analysis which are expected to take a very reasonable amount of computing power. We will test multiple scenarios such as using the same model, mixing different models, both small and large, and changing temperature.

Proposal Clarifications

We have seen both the Telephone Test and Output Test strategies in parts of the relevant literature. The Telephone Test (Mollah et al., 2025) that we are adapting used a similar framework to track information transfer between text-to-image and image-to-text models repeatedly. Additionally, the Output Test is a derivation from (Liang et al., 2024) where instead of having a critic that allows the agent to get back on track, we will monitor cascading errors and calculate the intrinsic variables of drift and velocity from models or systems of models. To clarify these models and systems of models, we intend to use open-weight models varying in model size and architecture (Unified vs. MOE). Our preliminary selection includes Qwen3:235b, Qwen3:8b, Llama3.1:70b, Llama3.1:7b, Gemma3:4b, gpt-oss:20b. This selection will give us enough sampling to model each on its own and with varying combinations.

Expanded Literature Review

LLM-Coordination: Evaluating and Analyzing Multi-agent Coordination Abilities in Large Language Models Existing work (Agashe et al., 2025) introduces a benchmark for evaluating LLMs ability to complete coordination tasks. This work evaluates agentic coordination, having LLMs play various multi-turn coordination games and evaluates the coordination question answering ability of LLMs. This work finds that LLMs perform well on coordination tasks when environmental cues indicate certain decisions should be made, but LLMs underperform when reasoning about other agents' beliefs and intentions. This work provides valuable insights about the ability of LLMs to collaborate and perform on tasks that require coordination. Our work differs from this work by evaluating a single contributing factor to agentic coordination, information loss between agents. Many factors may contribute to the shortcomings of LLMs when it comes to agentic collaboration and coordination, and this work simply evaluates the coordination abilities, without investigating the qualities and contributing factors that lead to these pitfalls. Our work aims to fill this gap by producing an evaluation methodology to identify shortcomings of LLMs when it comes to agentic handoff.

A Field Guide to Automatic Evaluation of LLM-Generated Summaries Evaluation of LLM-generated summaries is not a new problem, and many works already aim to evaluate LLMs on text summarization tasks. van Schaik and Pugh compiles and analyzes different metrics and methods used to evaluate summarization including BLEU, ROUGE, BERTScore, BLANC, and many more. It is noted that even LLMs are being used to evaluate LLMs on summarization tasks. The work provides best practices for evaluation and lists open challenges. These evaluation do not consider text summarization and information retention in the context of multi-agent LLM systems. Our work, while similarly being a text summarization metric of sorts, differs from other text summarization evaluation metrics because we aim to evaluate text summarization for the purpose of information propagation for multi-agent LLMs. Existing works don't evaluate the degeneration and information loss that happens after many summarizations of restatements. We aim to contribute an evaluation metric particularly suited to aid in model selection and design of multi-agent systems by evaluating a models ability to perform in such a system with potentially many agents through which information must be passed through.

Why Do Multi-Agent LLM Systems Fail? (Cemri et al., 2025) 37% of the time, the reason for the failure of a Multi-agent System is due to the flow of information between agents. Of the most common failures within this category, Task Derailment (7.15%), Information Withholding (1.66%) and Ignoring Previous Inputs (0.17%) derive directly from the root of the problem we are attempting to metric. The others in this category, like Reasoning-Action Mismatch (13.98%) and Failure to ask for clarification (11.65%) are thought to improve with the capabilities of the individual agent. The paper classifies this group of failure cases as complex and does not provide any metrics for monitoring or evaluating the root issue, only labeling the failure as within this category. We will propose a way to monitor and track the drift of misalignment and errors, through agentic pipelines, between model types, and between model sizes to provide a framework to monitor pipeline effectiveness loss, and to provide baseline information on agentic information transfer.

Mitigating Behavioral Hallucination in Multimodal Large Language Models for Sequential Images (You et al., 2025) This team identifies that hallucination starts at a point and cascades in vision language models. This snowball effect, as they coin it, is something that we are aware of and know happens in practice. While this paper focuses more on the mathematical way of identifying these generations and correcting it. We are assuming an environment not accessible to the base logits and probabilities of the model generation. Furthermore, we aim to track the difference between model handoffs, model intrinsic drifts, and prompts, identifying key intrinsic model behavior when it comes to passing information forward. Additionally, we will be focused on multi-agent sequential information passing, not from a single unified model translating between modals.

An Approach to Checking Correctness for Agentic Systems (Sheffler, 2025) The paper introduces a framework for verifying multi-agent LLM systems that focuses on the sequence of agent actions rather than their end textual outputs. Traditional evaluation methods often rely on string matching, which is unreliable due to the variability of LLM responses. Instead, the paper proposes the use of temporal expression logic which is a rule-based system for monitoring tool calls and multi agent handoffs to detect behavioral errors such as failed coordination and incorrect tool sequencing. This approach highlights that even when final outputs appear correct, the underlying process can still be flawed leading to cascading failures. While the paper focuses on detecting behavioral anomalies, it does not address how semantic content might degrade in these anomalies. By integrating these assertions into our process, we can ensure that agent handoffs and tool calls occur as intended before measuring semantic drift, helping us isolate whether information loss is due to this lack of coordination seen in the paper.

LLM as a Broken Telephone: Iterative Generation Distorts Information (Mohamed et al., 2025) The study shows that when LLM outputs are repeatedly transformed during translation or rephrasing. The text systematically drifts from the original, degrading both surface similarity and factual content with each iteration. Drift is faster when transferring between different models as compared to the same model. Drift was also observed to accelerate when transferring between languages, under higher temperatures, and loose constraints. For multi agent handoffs like ours, this suggests that semantic loss across handoffs is expected by default. We can take their evaluation scheme (text similarity and factual consistency) and experimental variables such as temperature and model diversity to guide our design choices including setting testing benchmarks for acceptable loss.

Examining Identity Drift in Conversations of LLM Agents (Choi et al., 2025) The study investigate the phenomenon of identity drift in multiturn conversations in nine LLMs (2 from GPT, 3 from LLaMA3.1, 2 from Mixtral, 2 from Qwen) showing that LLM agents often lose consistency in persona, style, or factual commitments as dialogue length increases. Across extended interactions, agents deviated from their original assigned roles in a significant fraction of cases, with identity-consistency scores dropping by more than 20% over long horizons. Their findings only highlight how fragile continuity is within a single agent's long term memory. With our work, we will extend this analysis to multiagent pipelines by quantifying semantic drift during sequential handoffs across different agents and model types. As opposed to this study, with the main focus being internal role stability, we propose metrics that track whether the meaning of transferred knowledge persists through agentic pipelines, which offer a complementary framework to evaluate not just identity preservation but also information fidelity across distributed systems.

Technical Report: Evaluating Goal Drift in Language Model Agents (Arike et al., 2025) Goal drift is the concept where an agent gradually deviates from its original objective over the course of reasoning or execution. The study shows that even under stable prompting conditions, goal adherence drops significantly across multiple steps, with larger drifts observed in open ended or underspecified tasks. Even though the study provides useful insights into the fragility of long horizon planning, their metrics are primarily task level, assessing whether the final outcome still aligns with the stated goal. However, this overlooks the problem of information drift during intermediate handoffs, whether partial knowledge is preserved, distorted, or lost when passed between agents in a pipeline. Our project picks up on this problem and directly complements this by introducing embedding based similarity measures that capture how faithfully meaning is preserved at each stage. Opposed to their macro level view, we will provide a micro level look on semantic fidelity, enabling a clearer understanding of where and how multi agent systems lose critical information.

Detailed Plan

The table in our appendix 1 outlines the weekly objectives and expected outcomes. Work will be divided evenly amongst group members.

Progress

As laid out in our weekly detailed plan, for the midterm report we have completed the initial repository setup for loading and inferencing models using the local provider library Ollama, an MVP implementation for the Telephone test, and MVP implementation for the Output Test, and a preliminary evaluation of results for these tests. From these tests, we have verified our initial problem statement and our experimentation method, showing that our problem exists and that we can model it. Additionally, with the boilerplate code tested and refined, we have successfully laid out the groundwork for future experiments using our framework.

Method Description

6.1 Telephone Test

The Telephone Test, as described in the project proposal, is a simple implementation that repeatedly prompts a Large Language Model to restate a starting text. The prompt also instructs the LLM to retain the meaning of the text, but not to use exactly the same words. The restatements are cached across iterations. The purpose of this test is to show how the meaning changes over time throughout the repeated restatements. One of the challenges that we face when running this test is that the results depend heavily on the prompting and the number of iterations. To overcome this, we plan to explore variations of this test with different prompts, potentially showing the Telephone test between loose and strict prompting ("Rephrase" vs. "Restate Exactly"). This test is to show that the more iterations a text goes through, typically the more the meaning of the text deviates from that of the original text. We then find the deviation and compare it over time and across model parameters.

6.2 Output Test

The Output Test extends the sequential agentic setup to structured outputs, where each agent in the chain receives a JSON formatted input containing a context and a deliverable. The model is instructed to preserve the structure and semantics of this JSON while performing a simple arithmetic operation, incrementing a numeric field by one, before passing it to the next agent. This design tests the model's ability to maintain syntactic validity and logical consistency over repeated generations. Current challenges faced by this test include explictily instructing the model to preserve JSON structure. If the models we plan on using are not reliable enough to maintain this structure implicitly, then we plan to migrate part of this test to use a forced structured output and focus more on the task execution and handoff. By measuring schema validity, task correctness, and embedding drift in context fields across multiple hops, the Output Test evaluates the persistence of structured reasoning and format stability in sequential LLM handoffs.

Experimental Results and Analysis

For evaluation, we measure how information changes as it is handed off across hops using semantic drift with cosine similarity in embeddings and drift using Levenshtein distance. For each prompt we can compute cosine similarity between consecutive outputs to capture local stability at each handoff, and cosine to the original at every hop to track the global drift over the whole chain. Similarly we compute Levenshtein distances to quantify how much the wording changes at each step, plus a final distance to the original to summarize cumulative divergence. These metrics together let us see when chains remain semantically consistent despite constant paraphrasing, when small changes start to compound, and where later hops begin to meaningfully degrade the original content.

6.3 Telephone Test Initial Results

The main initial result for the Telephone Test is shown in Figure 1. The purpose of the test is to show meaning degradation over multiple iterations of restatements. An initial run with a simple text, "The quick brown fox jumps over the lazy dog.", over 20 iterations shows how the meaning changes when the text is restated.

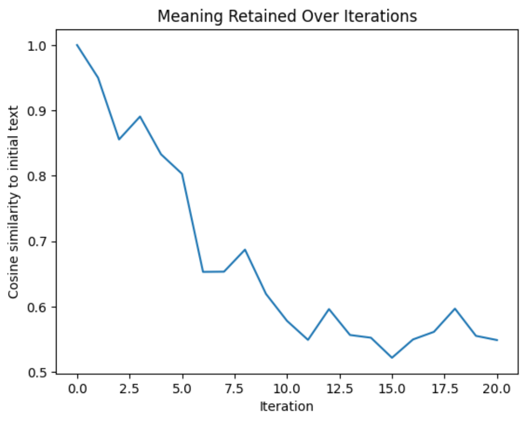

Figure 1: Plot showing the cosine similarity to the original text across restatements.

We see a very sharp initial decline in similarity to the original text over the first ten iterations, then the cosine similarity appears to stabilize at around 0.5-0.6.
Some of the texts with their iteration counts are as follows.

1. The quick brown fox jumps over the lazy dog.
5. A quick, muted fox runs over an unhinged, droll dog.
10. A quick and nimble fox sprints across an unhinged dog, whose grin is a display.
15. A swift fox traverses an unruly dog, whose expression conveys a display of emotion.
20. A swift fox traverses a wrinkled dog, whose expression reveals an emotion.

It is clear that much of the meaning is lost after 20 iterations, but a lot of the structure of the sentence remains and some meaning is retained.

The next figure shows cosine similarity between each restatement and every other restatement shown in a heatmap. This shows that as the restatements progress, consecutive restatements are consistently pretty similar with a relatively high cosine similarity values, but the more iterations apart, the lower the cosine similarity value. This shows a clear accumulation of the change in meaning for consecutive restatements.

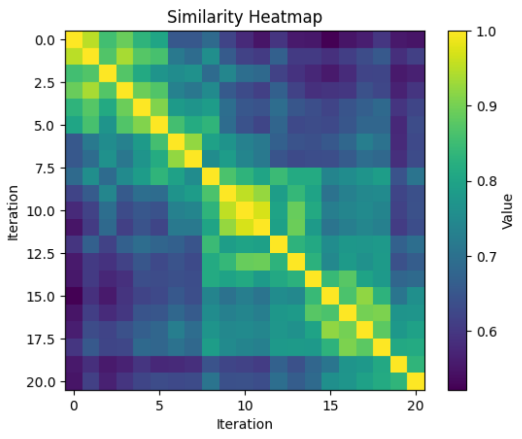

Figure 2: Heatmap showing cosine similarity between each restatement and all other restatements.

6.4 Output Test Initial Results

The initial results for the Output Test are presented in Figure 3. This test evaluates the model's ability to maintain structured reasoning and format consistency through sequential handoffs. Each iteration requires the model to output a JSON object preserving the same schema while incrementing a numerical field by one.

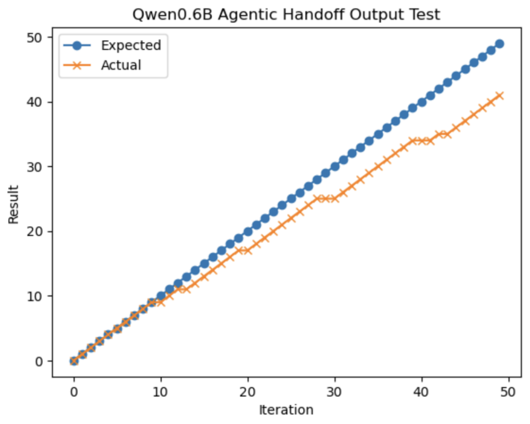

Figure 3: Plot showing the expected versus actual increment values over 50 sequential hops.

The plotted results show the expected versus actual increment values over 50 sequential hops for the Qwen 0.6B model. Early iterations closely follow the expected linear trend, indicating that the model correctly performs the arithmetic operation and preserves the structure. However, around iteration 20, slight deviations begin to appear, and by iteration 40 the accumulated errors become more apparent, resulting in under-counting and occasional skipped increments.

These results illustrate how even simple deterministic transformations can degrade over sequential generations when mediated by stochastic model inference. Although the Output Test enforces strong structural constraints, the gradual divergence between the expected and actual results highlights a form of logical drift analogous to the semantic drift observed in the Telephone Test but manifested through numeric reasoning and schema preservation.

Next Steps

As laid out in our detailed weekly plan, the next steps we have to take are to run and evaluate the tests that we've implemented using a variety of large language models. We are curious to see how variables such as: iteration counts, temperature, topk sampling, model type, parameter size, and quantization affect our tests. To accomplish this amount of compute, we plan to run many evaluations in parallel so that we can take an average and get statistically significant results between models. We also plan to evaluate and visualize the semantic drift that these tests expose, examining the velocity and acceleration of measures of similarity over iterations of the tests. We plan to use cosine similarity and potentially other measures to compare the textual embedding vectors produced at each evaluation step, then use this sequence to produce measures like drift and acceleration that should represent on the instruction following ability of the model intrinsically. Good visuals for these data will help interpret the results and give us more insight into the quality of different models for agentic tasks. Additionally, it will be very interesting to see how large language models of different sizes and strengths will perform relative to each other on these tests. We plan to run our tests extensively and present a detailed analysis of how the results vary as we change different details of the setup as described and anticipate interesting and exciting results.

Contribution Table

The table in our appendix 2 outlines the weekly contributions between all team members.

References

Saaket Agashe, Yue Fan, Anthony Reyna, and Xin Eric Wang. 2025. Llm-coordination: Evaluating and analyzing multi-agent coordination abilities in large language models. *Preprint*, arXiv:2310.03903.

Rauno Arike, Elizabeth Donoway, Henning Bartsch, and Marius Hobbhahn. 2025. Technical report: Evaluating goal drift in language model agents. *Preprint*, arXiv:2505.02709.

Peter Belcak, Greg Heinrich, Shizhe Diao, Yonggan Fu, Xin Dong, Saurav Muralidharan, Yingyan Celine Lin, and Pavlo Molchanov. 2025. Small language models are the future of agentic ai. *Preprint*, arXiv:2506.02153.

Mert Cemri, Melissa Z. Pan, Shuyi Yang, Lakshya A. Agrawal, Bhavya Chopra, Rishabh Tiwari, Kurt Keutzer, Aditya Parameswaran, Dan Klein, Kannan Ramchandran, Matei Zaharia, Joseph E. Gonzalez, and Ion Stoica. 2025. Why do multi-agent llm systems fail? *Preprint*, arXiv:2503.13657.

Junhyuk Choi, Yeseon Hong, Minju Kim, and Bugeun Kim. 2025. Examining identity drift in conversations of llm agents. *Preprint*, arXiv:2412.00804.

Shanshan Han, Qifan Zhang, Yuhang Yao, Weizhao Jin, and Zhaozhuo Xu. 2025. Llm multi-agent systems: Challenges and open problems. *Preprint*, arXiv:2402.03578.

Sirui Hong, Mingchen Zhuge, Jiaqi Chen, Xiawu Zheng, Yuheng Cheng, Ceyao Zhang, Jinlin Wang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, Liyang Zhou, Chenyu Ran, Lingfeng Xiao, Chenglin Wu, and Jürgen Schmidhuber. 2024. Metagpt: Meta programming for a multi-agent collaborative framework. *Preprint*, arXiv:2308.00352.

Tian Liang, Zhiwei He, Wenxiang Jiao, Xing Wang, Yan Wang, Rui Wang, Yujiu Yang, Shuming Shi, and Zhaopeng Tu. 2024. Encouraging divergent thinking in large language models through multi-agent debate. *Preprint*, arXiv:2305.19118.

Amr Mohamed, Mingmeng Geng, Michalis Vazirgiannis, and Guokan Shang. 2025. Llm as a broken telephone: Iterative generation distorts information. *Preprint*, arXiv:2502.20258.

Sabbir Mollah, Rohit Gupta, Sirnam Swetha, Qingyang Liu, Ahnaf Munir, and Mubarak Shah. 2025. The telephone game: Evaluating semantic drift in unified models. *Preprint*, arXiv:2509.04438.

Thomas J Sheffler. 2025. An approach to checking correctness for agentic systems. *Preprint*, arXiv:2509.20364.

Tempest A. van Schaik and Brittany Pugh. 2024. A field guide to automatic evaluation of llm-generated summaries. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '24, page

2832–2836, New York, NY, USA. Association for Computing Machinery.

Liangliang You, Junchi Yao, Shu Yang, Guimin Hu, Lijie Hu, and Di Wang. 2025. Mitigating behavioral hallucination in multimodal large language models for sequential images. *Preprint*, arXiv:2506.07184.

Table 1: Detailed plan table laying out the work for each week to be done for this project

Week	Objective	Expected Outcome	Deadline
Week 7 (10/6 - 10/10)	Plan and research	Know how to inference, what models to test, domain wording	
Week 8 (10/13 - 10/17)	Boilerplate code to load and inference model	Shared project space (Github) Boilerplate code (Ollama setup, inference, code standards)	
Week 9 (10/20 - 10/24)	MVP for Telephone Test	Test pipeline to gather some data on 1 model for the telephone test	
Week 10 (10/27 - 10/31)	MVP for Output Test	Test pipeline to gather some data on 1 model for the output test	
Week 11 (11/3 - 11/7)	Preliminary evaluation for Telephone Test and Output Test	Run evaluation metrics on the preliminary data gathered	
Week 12 (11/10 - 11/14)	Progress Report writing	Format previous results into project guidelines and submit	Progress Report Due
Week 13 (11/17 - 11/21)	Evaluation using different models	Run more scenarios, use different models, Thorough evaluation data of all tests to examine drift velocity/acceleration between each other	
Week 14 (11/24 - 11/28)	Thanksgiving / spill over for extra compute time	Extra time to run scenarios based on results, run enough times for statistical significance	
Week 15 (12/1 - 12/5)	Aggregation of results/ Latex formatting and writeup	All numbers and visuals that we might want as results and some interpretation of all results	
Week 16 (12/8 - 12/12)	Final Report writing	Final report with all methodologies, results, evaluation, explanations, and conclusions	Final Report Due

Table 2: Detailed plan table laying out the work done for this project by each member

Week	Objective	Expected Outcome	Contributors
Week 7 (10/6 - 10/10)	Plan and research	Know how to inference, what models to test, domain wording	Ethan Gutierrez, Jerry Song, Haidyn Arnett, Khoa Bui
Week 8 (10/13 - 10/17)	Boilerplate code to load and inference model	Shared project space (Github) Boilerplate code (Ollama setup, inference, code standards)	Ethan Gutierrez
Week 9 (10/20 - 10/24)	MVP for Telephone Test	Test pipeline to gather some data on 1 model for the telephone test	Haidyn Arnett
Week 10 (10/27 - 10/31)	MVP for Output Test	Test pipeline to gather some data on 1 model for the output test	Khoa Bui
Week 11 (11/3 - 11/7)	Preliminary evaluation for Telephone Test and Output Test	Run evaluation metrics on the preliminary data gathered	Jerry Song
Week 12 (11/10 - 11/14)	Progress Report writing	Format previous results into project guidelines and submit	Ethan Gutierrez, Jerry Song, Haidyn Arnett, Khoa Bui