Evaluating Knowledge Persistence for Multi-Agent Language Systems

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 succeptible 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 reg-

istered 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 models 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.

References

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.

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.

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.