Saarland University

Department of Computational Linguistics

Masters Thesis

Submitted as part of the degree requirements of the
MSc in Language Science and Technology, Saarland University

"Wait, who the hell am I again?" The effect of different LTM models on role and memory reinforcement in LLMs

Author:
Jack Weyen
Matriculation: 7023732

Supervisors:
Dr. Dietrich Klakow
Xenia Klinge

May 14th, 2025

Abstract

Recent advances in generative AI have resulted in a boom of interest and research in large language model (LLM) powered chatbots, particularly for applications involving long-term conversations. Two particularly productive areas of research involve the reinforcement of a chatbot's memory and assigned identity, by means of retrieval augmented generation (RAG) — the use of an external database which is updated across conversational sessions, from which relevant information is retrieved, summarized, and then used as context for generation.

However, while this general schema stays the same between approaches, the pipelines themselves can vary widely: common differences include the storage format, the type of information that is stored, the memory retrieval method, and the use of a reflection step. Moreover, many of these pipeline features have not yet been tested against one another, or if so, not in a long-term conversational context. Furthermore, many of these features that have yet to be tested in the same pipeline.

This thesis surveys these various implementations of conversational RAG, and then tests their performance on 1) enhancing a model's long-term memory and self-identity and 2) handling adversarial questions. Each variation of each pipeline feature (memory type, storage format, retrieval method, reflection) is tested individually and in combination with each other against a baseline with no long-term memory using the conversations from the LoCoMo dataset (Maharana et al. (2024)).

Several significant results are reported. Knowledge graph models are found to perform worse than knowledge bases and even the baseline, and cosine similarity outperforms topic overlap as a method of retrieval, contradicting earlier studies. Other results reaffirm previous findings, with models that store observations and turns producing better results than those that store summaries, and reflection having an overall positive impart on F1 scores. Significant interactions are identified both between pipeline features and between features and question types, but always in a way which reinforces the selection those features which are already individually the best performing.

Contents

1 Introduction

In the last three years there has been a massive boom in generative AI capabilities, research, and applications, in a large part due to the massive success of easily accessible chatbot models, such as ChatGPT. One area of particular commercial and scholarly interest is the use of such generative AI chatbots in roleplay.

Roleplay, in the context of chatbots, is the act of acting according to a persona (a character defined by a personality and biographical traits) with a human user, who might or might not also be roleplaying. Commercial applications of roleplaying chatbots have appeared in the customer support and entertainment sectors, where large language model (LLM) chatbots are asked to take on the role of customer support agents and various characters, respectively. As of 2023, a wide range of companies have supplemented or replaced their rule-based customer support chatbots with LLM alternatives, including Amazon, Zillow, Tesla, T-Mobile, and Bank of America (Pachipulusu (2023)). In the entertainment industry, LLM chatbots are being used to roleplay fictional characters and real-life personalities in a variety of formats. Online, Character.ai hosts pre- and user-made LLM chatbots for users to converse with (Character.ai (2024)), while video games have seen an influx of AI-powered NPCs, both in mods of preexisting games like Skyrim (art-from-the machine (2023)), and bespoke games like Vaudville (Steam (2023)). Finally, LLM chatbots have become characters in their own right, as exemplified by Neuro-sama, a popular AI streamer who converses with both the stream chat and human guests (vedal987 (2024)).

In such use cases, conversational history can regularly outgrow a LLM's context window. Therefore, much research has been devoted to finding solutions around this limitation, including longer contexts or fine-tuning the model itself (Fatehkia et al. (2024)). However, these methods are expensive to implement (Fatehkia et al. (2024)) and require access to a model's source code (Wang et al. (2024)), restricting their usage to relatively weak open-source models. Given the superior performance and general convenience of closed-source models, and to get around their high API costs, many researchers have instead begun to investigate the augmentation of LLMs with external long-term memories (Wang et al. (2024)). Most of such models utilize retrieval-augmented generation (RAG), in which some form of the current dialogue is stored in a long-term memory, and later retrieved when relevant to the current query, added as context for generation.

However, while the general RAG pipeline structure stays mostly the same between published models, the exact implementation can vary widely in terms of what is stored, how it is stored, how it is retrieved, and how it is processed. Moreover, while some comparative tests have been performed, most of these variant implementations of pipeline features have been not been tested against one another. Furthermore, existing studies differ in the LLMs used in testing, which complicates the generalization of any insights. As can been seen, a systemic evaluation of these different methods is necessary.

The present thesis intends to fill this need, investigating each documented variant of each part of the RAG pipeline, and evaluating them against one another. Specifically, I shall evaluate the different methods used for four features of this pipeline: memory storage format (knowledge bases and knowledge graphs), memory unit types (turns, observations, and summaries), retrieval method (semantic and topic), and reflection (whether or not it occurs). Altogether, these feature variants can be combined to produce 24 separate models. Each of these are evaluated using a battery of questions testing self- and event-knowledge and the ability to handle adversarial attempts to mislead. Both questions and conversational data are pulled from the LoCoMo dataset, a corpus of multi-session, large-scale conversations (Maharana et al. (2024)).

The results show several significant trends. Features which rely on the exact matching of text, such as knowledge graphs and topic overlap retrieval are very strongly outperformed by their alternatives. This is particularly interesting in the case of topic overlap, given that previous studies have shown it to be superior to cosine similarity (Li et al. (2024)). The other results reaffirm previous findings, with models that store observations and turns producing better results than those that store summaries, and reflection having an overall positive impart on F1 scores. While significant interactions are identified both between pipeline features and between features and question types, they always reinforce the selection of those features which are already individually the best performing: for example, while turns and reflection have a larger effect on F1 scores when knowledge bases are used to store memories, knowledge bases are already shown to be the better performing memory storage format, and thus these advantages are moot.

This thesis is structured as follows: it begins with a description of the features of the general RAG pipeline, followed by an overview on the current research on conversational models employing RAG for long-term memory. Then, based on these papers, a taxonomy is created detailing every major feature of the pipeline, and the various ways that feature has been implemented. Given this taxonomy I formulate the main research questions and describe the evaluation process. Following that is an in-depth presentation and analysis of the results, where each pipeline feature is treated individually and in conjunction with the others, as well as with different question types. As a conclusion, the implications of the data are reviewed and future research proposed.

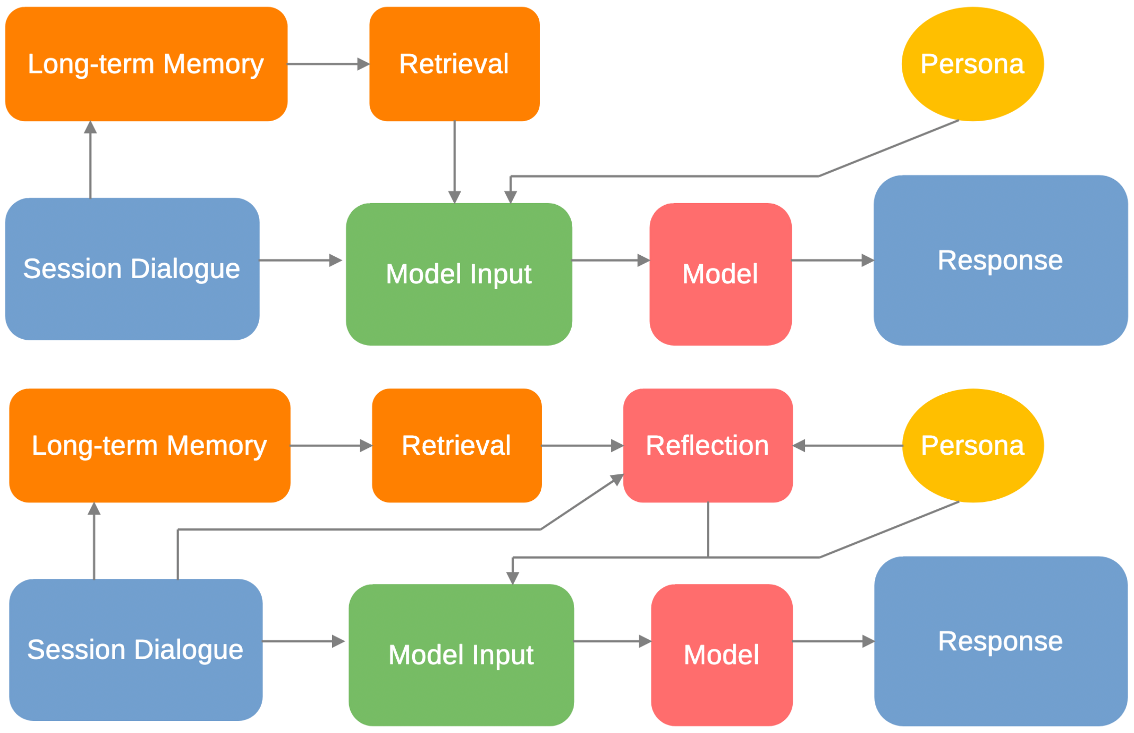

Figure 1. The general conversational RAG pipeline, both with and without the reflection module

2 General Conversational RAG Pipeline

Before we can begin the literature review, it is useful to become familiarized with the RAG pipeline as it generally appears in conversational contexts. Finer details, such as explanations of the various types of memory unit, storage formats, etc., will come later during the taxonomy. A visualization of the pipeline, and its variant with reflection, can be seen in Figure 1.

2.1 Memory Units and Storage Format

The pipeline begins with the session dialogue, the log of the current conversation at the present time. From this dialogue, a certain unit of information is taken, which can range from an individual turn (Park et al. (2023)), a summary of the entire conversational session (Li et al. (2024)), or even secondary information extracted from the dialogue, such as an observation (Maharana et al. (2024)). These units of memory are then stored in the long-term memory module, which is typically a vector database (Hatalis et al. (2024)), but may also be a knowledge base (Sanmartín (2024)).

2.2 Retrieval and Generation

With every new turn by the user, the model then retrieves the most relevant memories from the long-term memory. Given the aforementioned prevalence of vector databases as a memory format, most retrieval methods consist of calculating the cosine similarity (or some other similarity metric) of each memory with that of the user's newly input turn, and retrieving the top-k results (Hatalis et al. (2024)). However, other methods of retrieval, most notably based on topic, have been tested (Li et al. (2024)). The retrieved memories are then included along with the conversational persona (a prompt defining the personality the chatbot is to roleplay as) and session dialogue as context for generation.

2.3 Optional Step: Reflection

Some models include an additional step between retrieval and generation, termed "reflection" (Maharana et al. (2024), Park et al. (2023)). Here, the model is asked to reflect on the retrieved memories, i.e. synthesize from them an observation or insight, given the most recent user turn. The resulting reflection is then submitted to the model input in lieu of the raw memories.

3 Literature Review

Due to the novelty of conversational RAG, there are only a few papers, all published in the last two years, covering the topic. Accordingly, this literature review will cover six exemplary papers, surveying the models and techniques they employ and the results they have reported. Then, in the next section, the information will be synthesized to create a general taxonomy for conversational RAG models.

3.1 Memory Matters: The Need to Improve Long-Term Memory in LLM Agents (Hatalis et al., 2024)

Hatalis et al. (2024) is a survey of memory management approaches in LLMs, with a focus on long-term memory. They propose that conversational agents could have long-term memories modelled after the common model of cognition, which divides human memory into four subdivisions: procedural memory, which contains current goals, plans, skills; semantic memory, which records facts about the world and user; episodic memory, which focuses on events that have occurred; and working memory, which temporarily holds the most recent memories. According to them, all types of long-term memory can be implemented as a single cohesive unit in a LLM and the context window can act as the working memory.

They recognize that because of the fixed-size of context windows in LLMs, long-term memory solutions have become necessary. According to them, currently, most proposed and implemented solutions make use of knowledge bases, specifically vector databases. From these, memories are usually retrieved using similarity matching, in which the top-k memories are returned according to cosine similarity, euclidean distance, or dot product.

They find, however, that the use of vector databases have limitations. First is that different types of long-term memories, as described by the common model of cognition, could interfere with one another if they are not separated. For example, the semantic memory that the earth is round is treated the same as an episodic memory of someone claiming that the earth is flat. For a model that does not distinguish such memories, the earth could be erroneously remembered to be flat, if the similarity score is higher for the latter memory. Second, performance issues can occur as the long-term memory grows and the vector databases get larger. The authors propose a forgetting mechanism as a solution, to keep the long-term memory within a manageable size.

3.2 KG-RAG: Bridging the Gap Between Knowledge and Creativity (Sanmartín, 2024)

In contrast to the models mentioned in the previous paper, Sanmartín (2024) presents a LLM that utilizes a knowledge graph, instead of a knowledge base, to store information.

Knowledge graphs are an alternative way of storing information using structured representations of entities and the relationships between them. This typically takes the form of triples, containing a pair of nodes, usually representing entities, and their relationship: for example, "The dog likes bones" would be rendered as the triple

($n1$: dog, r: like, $n2$: bone)

where the nodes $n1$ ("dog") and $n2$ ("bone") are linked by the relation $r1$ ("like"). Then, as new information is processed, additional nodes may be attached to the previous two: given the sentence "I have a dog", the node $n3$ ("I") would be linked to $n2$ by the relation $r2$ ("have").

Sanmartín's model, KG-RAG, constructs a knowledge graph from unstructured text and then retrieves relevant info for use in generation, in hopes of mitigating hallucinations, issues with processing long contexts, and catastrophic forgetting (i.e. new information causes model to forget old information), a prospect that has been recently explored by other researchers (Guan et al. (2023), Yang et al. (2024)). He argues that knowledge graphs have an advantage over knowledge bases because information retrieval by vector similarity, which is what most knowledge bases use, is not granular enough: it returns too much information, of which not all might be relevant.

Knowledge graphs, in contrast, allow for more efficient information retrieval methods, since the information therein is much more structured and efficient, allowing information to be retrieved based on the entities involved, rather than mere sentence similarity. This difference is neatly summarized by Singhal Amit, in introducing Google's Knowledge Graph: knowledge graphs look for "things, not strings" (Singhal (2012)). Another advantage is that knowledge graphs evolve with new knowledge, whereas knowledge bases retain the old, incorrect knowledge which might be retrieved if found to be more similar to the query: if a white dog was painted blue between conversations, for example, and the question is the dog's color, a knowledge base might retrieve a memory from before the dog was painted and answer white; a knowledge graph, on the other hand, would have updated the color to blue, making it impossible to answer otherwise.

Sanmartín tests this KG-RAG model against a knowledge base and human baselines using the ComplexWebQuestions dataset, a collection of nearly 35,000 questions, recording exact matches and the corresponding F1 score. The results show that KG-RAG is worse than knowledge base RAG in terms of F1 and accuracy; however, in terms of mitigating hallucinations, the paper's primary area of interest, KG-RAG does indeed outperform the knowledge base baseline.

These results show that knowledge graphs could be a viable substitute for knowledge bases, but carry their own drawbacks in terms of answer accuracy. However, it must be noted that this line of research has only focused on knowledge graph RAG in the context of question-answering tasks, so their performance in conversational contexts, which require a wider set of skills outside of question-answering, remains to be seen.

3.3 Evaluating Very Long-Term Conversational Memory of LLM Agents (Maharana et al., 2024)

In this paper, Maharana et al. (2024) state that despite many advancements in LLMs for extended contexts and with RAG, none have truly been tested over extremely long contexts, and indeed, no sufficiently long conversational datasets currently exist. They correct this oversight by creating LoCoMo, a dataset of 10 very long-term dialogues, each consisting of 300 turns and 9K tokens on average and representing an average of 19 separate conversational sessions, which they use to test the memory capabilities of conversational models.

To create the LoCoMo dataset, and for the experiments run upon it, the researchers used LLM-based conversational models equipped with character personas (commonly called "generative agents") to generate the conversations. Each model was equipped with: a knowledge base for a long-term memory, from which memories were retrieved by semantic similarity; a persona; and a reflection mechanism. Unlike other models in the conversational RAG pipeline, in addition to the long-term memory and the conversational history (which is typically considered the short-term memory), the authors include an additional short-term memory consisting of the summary of most recent conversational session.

The memory units stored in the long-term memory require special attention. For generating the dataset, the researchers store observations generated from by a separate instance of the model which has been given the current conversational history. After each new conversational session the model extracts relevant facts about the persona and the user, and stores them in the long-term memory. With these generative agents, the authors generate the conversations that comprise the dataset, which they then refine using human annotators to correct any long-term inconsistencies to produce the final product, LoCoMo.

Using LoCoMo, the authors test generative agents on three types of tasks: question-answering, graph summarization, and multi-modal dialog generation.

The question-answering tasks consist of a battery of questions falling into five reasoning types: 1) single-hop (answered by remembering information from a single session); 2) multi-hop (answered by remembering information from multiple sessions); 3) temporal reasoning (answered by understanding time-related cues); 4) commonsense or world knowledge (answered by understanding basic facts about the world); and 5) adversarial (answered by understanding the user is presenting the model with a misleading or unanswerable question). For these tasks, the generative agent model was tested against a baseline LLM and an LLM with a long context window (16K tokens), as well as human participants. Furthermore, three versions of the generative agents model were tested, each differing in the type of unit stored in the long-term memory: observations, individual turns, and summaries of past conversational sessions.

The graph summarization task measures causal and temporal relations by having models extract graphs based on text, and comparing them to an ideal graph. Finally, the multi-modal dialog generation task investigates if models can generate narrative-consistent responses based on past context. The results of these two tasks, however, are not relevant for the current thesis proposal, and will be ignored.

The results for the question-answering task found that, on average, all generative agent models outperformed the baseline LLM, and the ones storing observations and turns outperformed the long-context model. The generative agents were found to be much better at answering temporal and adversarial questions than long-context models. However, all models performed poorly in contrast to human participants. Also, observations were found to be the best overall performing type of memory unit, followed by turns and then summaries. Finally, all models performed better on single-hop questions than multi-hop ones, while models storing summaries actually performed better with answering adversarial questions than any other type of question, and even outperformed turn-storing models at that task.

3.4 Generative Agents: Interactive Simulacra of Human Behavior (Park et al., 2023)

The modern concept of "generative agents", conversational RAG LLMs, comes from Park et al. (2023). In this bombshell paper, the authors assert that in order for LLMs to be able to handle long-term coherence and interaction, they require three abilities: 1) to remember (retrieve) relevant events, 2) to reflect on these memories given the situation, and 3) to plan and react in the moment, and in the long term, based on these reflections and the current situation. The resulting models, generative agents, are then evaluated with interviews, ablation studies, and most famously, placed in a virtual town environment and left to interact with one another with minimal outside interference.

The generative agent model is based upon the following architecture. First is a memory stream, the long-term memory, which records experiences (both conversational turns and actions made within the virtual environment), which are then retrieved based upon relevance, recency, and importance. Next comes a reflection stage, in which agents reflect upon their memories and then ask themselves questions, the answers to which they add to their memory. Finally, unlike other models in this literature review, these generative agents have a planning stage, where given conclusions from reflection and the current environments, agents form plans and enact them. These resulting actions then enter into memory, and the process begins again.

For evaluation, the authors interviewed agents individually with a slew of questions covering several abilities: self-knowledge, memory retrieval, plan generation, reacting to situations ("Your house is on fire!"), and reflecting on information. These were analyzed by 100 crowdsourced human participants, and ranked according to their "believability", i.e. their subjective ability to accurately mimic realistic human behavior. While model performance was found to be believable for most of these questions, the authors noted three behaviors which caused the models to fall short of human simulacrum: agents would fail to retrieve relevant memories, fabricate embellishments to their memories, and inherit overly formal speech or behavior from the language model (a hallmark of LLMs familiar to anyone who has roleplayed with one). These were done in combination with an ablation study, where the retrieval, reflection, and planning stages alternately removed before the interview. With this, the authors found that all three stages of the pipeline had almost equal effects on increasing the model's believability.

3.5 MemoryBank: Enhancing Large Language Models with Long-Term Memory (Zhong et al., 2023)

As previously mentioned, Maharana et al. (2024) individually compared observations, turns, and session summaries as units of memory to be stored in the long-term memory. Zhong et al. (2023) introduce a model, MemoryBank, which utilizes multiple types of memory unit at once. Memorybank stores the turns of a conversation in one knowledge base, and from these generates event summaries, i.e. summaries of the events which occur within a dialogue, which it stores in a separate knowledge base. Both sets of memory are then subjected to a version of the Ebbinghaus Forgetting Curve: memories are given a memory strength score S which exponentially decreases as time progresses, and is reinforced every time the memory is recalled. In addition to this long-term memory system, the model maintains a personality module, a separate memory in which stores summaries of the user's personality traits and emotions, distilled from observations the model is asked to make given the conversation dialogue.

According to a qualitative analysis, MemoryBank, when incorporated into a conversational model, was able to identify topics it had discussed before, and those it had not. Furthermore, the model was shown to make suggestions according to a user's aforementioned personality traits, which was taken to demonstrate the success of MemoryBank's personality module.

3.6 Hello Again! LLM-powered Personalized Agent for Long-term Dialogue (Li et al., 2024)

Finally, Li et al. (2024) present LD-Agent, which like Zhong et al. (2023) describes a conversational RAG model that combines a knowledge base storing dialogue event summaries with a separate module that stores persona information. Their proposed model, LD-Agent, has three principal components: an event memory perception module, a persona extraction module, and a response generation module.

The event memory perception module consists of the long-term memory and short-term memory, the former storing an impersonal (not influenced by model persona) summary of the events of the previous conversational session, and the latter consisting of the dialogue for the current session, stored in a cache that gets transferred to the long-term memory after a sufficient delay in the conversation. Of particular interest is the method by which stored session summaries are retrieved: unlike any other knowledge base model presented, and like knowledge graph models, LD-Agent retrieves long-term memories by incorporating topic overlap. Alongside more the standard measures of semantic relevance and temporal recency, the model assigns scores to stored summaries based on the average proportion of common nouns in the current user turn and the summary.

In the persona extraction module, personality traits for both the user and persona are extracted from the dialogue at each turn and stored in respective persona banks. This is done in the hopes of maintaining persona consistency in the responses of the model for both participants. The final component, the response generation module, simply generates responses based on the current user turn, the relevant long-term memories, the short-term memory, and the user and agent personas.

Models incorporating LD-Agent were tested against baseline models lacking it, as well as against HAHT, a previous state-of-the-art long-term dialogue model, on two multi-session datasets of around 5 sessions each, each session consisting of around 50 turns. Each model was tested using BLEU-N, ROUGE-L, and METEOR to measure response quality, and also evaluated by human participants for coherence, fluency, and engagingness. LD-Agent was shown to outperform all baselines in long-term dialogue tasks, and ablation studies found that all three modules positively influenced performance. Of these, event memory had the biggest effect, offering more stable performance as the number of sessions increased, while the other modules suffered. Human evaluation found that topic based retrieval significantly outperforms direct semantic retrieval in accuracy and recall, and the inclusion of LD-Agent led to more coherent, fluent, and engaging dialogues.

4 General Conversational RAG Taxonomy

Having presented the relevant papers, we can now begin piece together a taxonomy of the different approaches currently being employed in the developing field of conversational RAG. From the literature we can identify five different aspects of the conversational RAG pipeline that have been approached with different techniques. These are presented below, the varying techniques presented alongside any insights into their comparative performance with one another.

4.1 Memory Storage Format

Memory storage format refers to the nature of the storage system used to store long-term memories. Currently, all RAG systems employed explicitly for conversation utilize knowledge bases, as stated by Hatalis et al. (2024). However, as we have seen, research into RAG within question-answering tasks has presented knowledge graphs as a possible alternative (Sanmartín (2024), Guan et al. (2023), Yang et al. (2024)). Sanmartín (2024) in particular shows that while less accurate than knowledge bases when used in RAG, knowledge graphs are more useful in mitigating model hallucination. Given the importance of the ability to produce believable answers has in roleplay, there is strong motivation to see if knowledge graphs would provide similar benefits in the conversational domain, especially against antagonistic questioning.

4.2 Memory Unit Type

Memory unit type refers to the unit of memory that is stored in the long-term memory. Among the papers discussed, there are three main types of memory unit: first are single turns in the conversation (Park et al. (2023)), second are individual observations, i.e. facts, that can be drawn from the dialogue using a separate LLM (Maharana et al. (2024)), and third are summaries of sections of dialogue, typically sessions defined by arbitrarily long intervening gaps in time (Maharana et al. (2024), Zhong et al. (2023), Li et al. (2024)). One common type of summary, event summaries (Zhong et al. (2023), Li et al. (2024)), only summarize events and therefore could be considered a separate type of memory unit; in this taxonomy, it will not. Sometimes multiple of these memory units are employed at once, as is the case with Zhong et al. (2023). All three memory unit types are compared in Maharana et al. (2024), as previously mentioned, with observations performing best overall, turns performing better on single- and multi-hop questions, and summaries being particularly suited to adversarial questions. It remains to be seen, however, if these trends hold when these memory unit types are paired with different pipeline features.

4.3 Retrieval Method

Semantic similarity appears to be the most common metric used to determine what long-term memories should be retrieved (Hatalis et al. (2024), Maharana et al. (2024), Park et al. (2023), Zhong et al. (2023), Li et al. (2024)), being used by every model using a knowledge base. However some models utilize alternative metrics alongside this, such as temporal recency (Park et al. (2023), Li et al. (2024)), frequency of retrieval (Zhong et al. (2023)), importance (Park et al. (2023)), and topic (Li et al. (2024)). The only paper to make a direct comparison between these various methods is Li et al. (2024), who compare semantic- and topic-based retrieval and find the latter to be better quantitatively (better precision and recall) and qualitatively (more coherent, fluent and engaging dialogues). Since Li et al. (2024) only test retrieval using memories with summary-level granularity, it remains to be seen if these improvements persist with units with more granularity (turns, observations).

4.4 Reflection

Of the papers covered, only two include the optional step of summarizing or reflecting upon retrieved memories, Maharana et al. (2024) and Park et al. (2023)), and of these two, only Park et al. (2023) evaluates the method against a baseline. In the corresponding ablation study, the exclusion of reflection from the model pipeline leads to a significant decrease in overall behavioral believability. Thus, reflection allows models to synthesize their available information in order to create opinions and come to conclusions, allowing for the expression of more believable behaviors. However, while reflection has been proven against non-reflecting baselines, it has not been tested against types of memory units.

Furthermore, it appears that reflection has been implemented differently between Maharana et al. (2024) and Park et al. (2023). In the former, retrieved memories are reflected upon, and that reflection is fed directly into the model for generation. In the latter, however, reflection is much more complicated. The model is asked to reflect and come up with high-level questions given its past experiences, retrieve memories (and previous reflections) based upon these questions, and then answer them. As just now hinted, reflections are not used immediately for generation, but stored for use later. Thus, while this more complex form of reflection has been evaluated against a baseline, the more simple one has not, and the two forms have not been compared against the other.

4.5 Persona Module

A final aspect of the conversational RAG pipeline that sees varied implementation is that of a separate long-term memory storage module for memories of facts relating to the model's and user's identities, i.e. their persona (Zhong et al. (2023), Li et al. (2024)). Ablation studies from Li et al. (2024) have shown that having a module specifically for personal information positively influences long-term dialogue capabilities. However, such findings speak to a general improvement in performance, and no research has investigated how persona memory can influence specific conversational tasks; for example, one might argue that including a memory module for personas would help models better handle adversarial questioning, as the retrieved persona traits could help "remind" the model of important information.

5 Chatbot Architecture

In this section, we cover the architecture of the chatbot used in this study. This architecture consists of a central pipeline that remains unchanged across studies and handles the main chatbot loop, and the various functions that are referenced within it, including the features (memory storage format, memory unit type, retrieval method, and reflection) and memory management (loading, updating, saving, deleting). Each will be covered in turn below. The code for testing persona modules, deactivated due to time constraints, is also included, albeit only in part. Additionally, the dataloader used to pre-load long-term memories for testing will also be described.

5.1 General Pipeline

What follows is a high-level overview of the general pipeline used across tests. The chatbot is designed to operate in two main modes: a "testing" mode, where input in the form of a list of questions can be fed into the model along with information about which features are to be tested, questions which the chabot then answers with reference to a pre-uploaded long-term memory; and a "chatting" mode in which the user can chat in real time with a default persona. In-depth explanations of each feature of the pipeline will follow in the next subsections.

First the history, long-term memory, past session summary, and persona modules are uploaded from memory files; if they do not exist, empty ones are made. In chatting mode, memory files can be deleted if desired, to allow the user to begin conversing with a blank chatbot memory. Following this is the main chatbot loop, which in testing mode runs until the final question has been answered, and in chatting model ends when the user types "Quit". Within this loop, the query is taken either from a list of questions (when testing) or from the user input (when chatting). Whatever the input, it is then added to the conversation history.

Next, if it is not empty, the long-term memory is consulted. The top-k highest scoring memories are retrieved according to the specified retrieval method; also retrieved are the exact session and turn(s) each of those memories originally came from (called the "memory evidence"). The model then reflects on these memories (if so desired), and a reflection is obtained.

Following this, the input for the LLM is assembled. First are rules which dictate that the LLM act as a person, rather than as an assistant, followed by the model persona. The persona for testing mode gives the LLM the role of someone answering questions regarding a conversation between two friends, while the persona for chatting mode gives it the role of a stereotypical anime girl named "Hiyuki". To this, either the user query is added, in the case of testing, or the entire conversation history is added, in the case of chatting. The reason the entire history is not used when testing is to prevent the previous questions and answers from influencing the model's answer, since the goal is to test its long-term memory. Next, if memories were successfully retrieved, either the reflection, or, in the case reflection is not used, then the retrieved memories themselves are added to the input along with an explanatory prompt. Finally, the past session summary and persona modules for both speakers are added, if required. The chatbot then generates a response based on this input using Llama-3-8B-8192, accessed by means of a Groq client. This response is then added to the conversation history, and additionally printed to the terminal in chatting mode.

The final actions of the loop differ depending upon the mode. In testing mode, the loop proceeds to the next question until the final question has been answered, at which point the answers and the memory evidence are returned. In chatting mode, the history is instead converted to the appropriate type of memory unit and added to the long-term memory. For turns and observations, this occurs within the chatbot loop, whereas for summaries this occurs once the conversational session has been concluded, since they are generated based on the entire session. At the end of the session, the persona modules are also updated with facts drawn from the session for both speakers, and a session summary is made. Finally, these updated long-term memories, persona modules, and past session summary are saved to file.

5.2 Dataloader

As previously mentioned, during testing the chatbot uses pre-loaded long-term memories that have been saved to various files. This was done to ensure that long-term memories wouldn't need to be recreated every time a pipeline was tested, saving on runtime. However, since these testing-specific long-term memories are meant to store data taken from LoCoMo, and since it was unfeasible to modify the the general pipeline to do so, a separate dataloader was made to load the LoCoMo conversations into long-term memory files. The structure of the dataloader is similar to the general pipeline, utilizing many of the same functions. For each conversation and session in LoCoMo, turns are processed two at a time, as the query and response would be in the general pipeline. These turns are converted into memories according to unit type, and added to the long-term memory according to the specified format. Because LoCoMo provides the observations and summaries for each session alongside the session transcript, the option was included to either use these or generate new ones on the fly. Like in the general pipeline, turns and observations are stored every two turns, and summaries at the end of each session. Then, once all sessions of a conversation have been processed, the long-term memory is saved to a unique file. Since the long-term memories vary according to memory storage format and memory unit type, six different types of long-term memory were produced for each conversation in LoCoMo, resulting in 60 separate long-term memories. In addition to these, the summary of the last session for each conversation was stored, again with the option to use the ready made one found in the dataset, or to generate one anew.

5.3 Long-term Memory Formats

Below are the two types of memory storage format.

5.3.1 Knowledge Base

The knowledge base used is a simple list consisting of individual memories. Each memory consists of the memory text (either a turn, an observation, or a summary), the topics, the sentence embedding, and the memory evidence.

5.3.2 Knowledge Graph

The knowledge graph used is a NetworkX MultiDiGraph, a directed graph with parallel edges that stores a source, its target, and the directed relationship between, them alongside other data. To update the knowledge graph with a memory, all relationship triples (tuples containing the source, relationship, and target) are extracted from the memory's text. The method of relationship extraction used differs in three fundamental ways from most traditional methods. First, typical models define the source, relationship, and target as the subject, verb, and direct object respectively; in the present model, however, targets have been expanded to include predicate adjectives ("Guilliman is detail-oriented"), prepositional phrases ("The Emperor resides on Terra"), and sentence clauses ("Yarrick said he killed the ork"), allowing for more information to be stored.

Second, rather than selecting just the heads of phrases, the entire phrase is selected: for example, the noun phrase "the imperial feudal world" is reduced to "world" using other methods, in this one it remains unchanged. While this admittedly makes sources and targets more specific, and therefore less likely to be retrieved, it is hoped that this drawback would be balanced by an increase in more relevant results: if someone is talking about space marines, it would be preferable that only information about space marines is retrieved, rather than information about marines of any type. Finally, while most relationship extraction methods rely on dependancy trees, the method used in this thesis relies on constituency trees (extracted using the Berkeley Neural Parser). Unlike dependency trees, which force the user to account for every possible combination of modifiers in order to extract a phrase, constituency trees allow phrases to be selected as a single unit, which offers a considerable advantage.

From the constituency tree, all clauses are extracted, and for each, conjunctions are resolved such that additional copies of the clause are created for each member of the conjunction. For example, the sentence

"I like canids and felinids."

would result in three separate clauses — the original and an additional for each animal:

"I like canids and felinids."

"I like canids."

"I like felinids."

Then, from each clause the source, target, and relationship are extracted. The source is the first noun phrase in the sentence, whereas the target is the first phrase in the predicate that is 1) a noun phrase or sentence phrase, 2) a noun phrase in a prepositional phrase, or 3) a predicate adjective, in that order. Finally, the relationship is defined as all text that occurs between the source and target. When no target was found, "N/A" was used as a default, and the entire predicate becomes the relationship. Given the above example, the extracted relationship triples would be:

("I", "like", "canids and felinids")

("I", "like", "canids")

("I", "like", "felinids")

Then for each extracted relationship triple the source, relationship, and target and joined into a single text, any speaker names are masked (discussed in detail in Section 5.5.1), and a set of topics and a sentence embedding is produced. Finally, from these an edge is made consisting of the source, target, relationship, topics, embedding, and memory evidence.

The knowledge graph used in this study utilizes a simplified version of the "Chain of Explorations" method presented in Sanmartín (2024). Given a query, the knowledge graph will extract the sources and targets and return all unique relationship triples which use them as nodes. These triples are then scored for relevancy against the query, and the top five are saved. When the next round begins, each top-scoring memory from the previous round is used as the query for the knowledge graph, and the process repeats until three rounds have been completed. Then, the top-k best scoring memories are selected from the relevant memories collected across all three arounds, and returned. The result is branching pattern, in which the knowledge graph is explored according to a chain of the most relevant memories.

As previously mentioned, this version of Chain of Explorations differs from that described in Sanmartín (2024). The original has a planning section, in which an LLM produces a multi-step plan for responding to the query. After each round of exploration, the returned triples are then evaluated by another LLM according to whether the information required by the current step has been found, and the decision is made whether or not to proceed to the next step. An LLM is also used to determine how relevant a triple is, as a supplement to cosine similarity. Due to the complexity and the increase in runtime time that this would entail, these LLM portions were omitted. Finally, the original algorithm allowed exploration based on matching relationships in addition sources and targets; only the latter two were allowed in the present version, given the predominance of semantically "light" relationships such as forms of "to be" and "to go to".

5.4 Memory Creation

Memories are created from the chatbot history at the end of each exchange of turns, or in the case of summaries, at the end of each session. For each type of memory unit, an important preprocessing step is performed: all first- and second-person pronouns are exchanged with the names of the speakers, converting the sentences into the third person and, importantly, removing any confusion as to which pronouns refer to who. Next, the text is processed according to memory unit type. Turns are taken from the final two turns of the history, which correspond to the most recent query and the chatbot response. The same is done for observations, with an additional step which extracts an observation from the query and response separately.[1] For summaries, the entire session history is utilized to produce a multi-sentence summary. Finally, once the memory units are created the topics are extracted and a sentence embedding is produced. These are then combined into a single memory consisting of the memory text, the topics, the embedding, and the memory evidence.

5.5 Retrieval

The retrieval system is comprised of two parts: the creation of the inputs used to score relevancy (topics and sentence embeddings) and the relevancy scoring methods (topic overlap and cosine similarity).

5.5.1 Relevancy Scoring Input Creation

As mentioned in the pervious section, topics and extracted and sentence embeddings are generated during the memory creation step. Immediately preceding this is a speaker name masking step: initial tests found that speaker names exert a major bias on relevancy results, with the highest scoring memories being those which mention a speaker's name the most; only when speaker names are masked do the expected results appear, with relevancy being based upon subject matter. After masking, the topics are extracted.

Topics may be defined in two ways: as a noun, or as a noun phrase. When defining topics as a noun phrase, non-possessive pronouns and determiners are ignored, such that "the sneaky purple ork" results in "sneaky purple ork", but "my boltgun" remains unchanged. In addition to this, to handle more general references to the same topic, for each noun phrase a new topic is made by removing a modifier until the head of the phrase is reached: thus, returning to the above examples, the topics "purple ork", "ork",

[1]An important note must be made here: this design, that observations are drawn from each turn, is modelled after the description given in section 3.3 in Maharana et al. (2024). However, it appears that this description is incorrect, for in the appendix for that paper it is clearly shown that observations are drawn from the entire session, not from individual turns. This mistake was only recognized after the decision was made to forgo generating new observations and use those available in LoCoMo, and thus the original observation generator design has remained unchanged, and is being described as it currently exists in the codebase.

and "boltgun" are also extracted. The intended result of this is to reward specificity: the more specific the matching phrase, the more topics which overlap and therefore the higher relevancy score. After the topics are extracted, a sentence embedding is generated using Sentence Transformers model all-MiniLM-L6-v2.

5.5.2 Relevancy Scoring

Topic overlap scoring was identical to that presented in Li et al. (2024), where the memory recall and query recall are calculated and averaged:

$$\text{Topic Overlap Score} = \frac{1}{2}\left(\frac{\text{Shared Topics}}{\text{Query Topics}} + \frac{\text{Shared Topics}}{\text{Memory Topics}}\right)$$

Cosine similarity was then calculated between the sentences, and the top-k scoring memories, along with the memory evidence, were returned.

5.6 Reflection

The reflection step is simple: given the memories retrieved from the long-term memory and the conversation history, a persona-less LLM is asked to reflect and write down its thoughts in three or four sentences. Similar to the general pipeline, in testing mode only the most recent entry in the history, i.e. current question, is provided. The rest of the history is disregarded to prevent the previous questions and answers from influencing the model's answer by providing context clues. When testing with no reflection, this step is skipped.

6 Experiments

As we have seen, there exists a great diversity of approaches to implementing a conversational RAG pipeline. Yet, probably due to the fact this area of research is in its infancy, only a few of these approaches have been compared against one another. Even when differing approaches are compared, it is done without any consideration to the effects of the other parts of the pipeline, or tests general performance that overlooks the nuances of particular tasks, like multi-hop or adversarial question-answering.

Thus, a comprehensive test of these differing approaches is justified. The proposed thesis is tasked with answering the following question: which combinations of memory storage format, memory unit, retrieval method, and reflection technique lead to better performance in answering questions involving self- and event-knowledge and adversarial manipulation. This is done following the methodology of Maharana et al. (2024), using the conversational data and questions from the LoCoMo dataset to record the F1 score and recall for each combination of pipeline features.

6.1 Pipeline Variants

The experiments focused on the following variants in the conversational RAG pipeline:

1. Storing long-term memories in a knowledge base or a knowledge graph
2. Populating the long-term memory with individual turns, observations, or summaries
3. Retrieving memories based on similarity or topic
4. Reflecting upon retrieved relevant memories, or not

Rather than each part of the pipeline being tested individually, each was tested in combination with one another, with the intent to capture any possible interactions that have been overlooked by pervious studies, such as the performance of reflection on different types of memory unit, or the effect of using a knowledge graph on downstream parts of the pipeline.

In total, this amounted to 24 pipelines as well as a baseline lacking a long-term memory but still having access to the most recent session summary.

6.2 Dataset

The dataset used is the LoCoMo dataset from Maharana et al. (2024), which consists of 10 long-term conversations. Each conversation involves a different pair of speakers, and is split up into multiple sessions. Each conversation is accompanied by several sets of additional data:

1) The conversation transcript

2) A question-answering dataset consisting of a question, the question type, the expected answer, and the evidence for that expected answer

3) The time and date each session took place on

4) Summaries of the events that take place in the lives of each speaker

5) Observations about the speakers that are drawn from each session

6) A summary for each session

Of these, the conversation transcript, question-answering dataset, observations, and session summaries were utilized by the dataloader to create long-term memory files to be utilized during testing. As mentioned in the dataloader description, with two memory storage formats, three memory unit types, and ten conversations in the dataset, this resulted in 60 separate long-term memory files. In addition, for each long-term memory file a past-session summary file was saved — these remain the same across memory storage formats and unit types.

6.3 Evaluation

Testing was conducted on each of the 24 pipelines using the question-answering data, with each pipeline answering questions for each of the 10 LoCoMo conversations. Of the five question types present in the dataset, only three were utilized in testing: single-hop, multi-hop, and adversarial. The remaining two, temporal reasoning and commonsense/world knowledge, where excluded as the former was more of a test of the ability for the chatbot to remember timestamps rather than personal information, and the latter because it can be argued that commonsense/world knowledge questions moreso test the pretraining dataset of the underlying LLM, as opposed to than any ability of the RAG pipeline itself.

In these tests, k was set to 10, such that the top 10 memories would be retrieved for each response. This is a departure from Maharana et al. (2024), which tests several different k values. While other k values could have been tested, this would have multiplied the amount of tests that would need to be conducted, which was deemed unfeasible given the time constraints and the limited scope of this thesis.

Scoring followed the procedure laid out by Maharana et al. (2024): for each question, an F1 similarity score was calculated between the chatbot's response and the expected answer, and the recall was calculated from the proportion of correct memories found in the memory evidence. These will be discussed in further detail below.

6.3.1 F1 Score

Maharana et al. (2024) used a partial-match method for calculating F1 score, which was based on identifying matching lemmas between the chatbot response and the expected answer. This however was found to come with serious drawbacks: the reliance of exact word matching resulted in an excess of zeros in the initial results, comprising roughly a third of the total F1 scores, which left the data anaemic and hard to analyze. Thus, a switch was made to BERTScore, which calculates F1 score by means of similar word embeddings and therefore eliminates zeros while offering more meaningful results that can accurately take synonyms and paraphrasing into account.

6.3.2 Recall

With each response the chatbot produces, it also returns memory evidence, i.e. a list consisting of the ultimate source(s) for each memory retrieved from the long-term memory, identified by the numbers of the session and turn they occur in. During scoring, when the chatbot response is scored against the expected response, its memory evidence is also checked against the expected memory evidence, i.e. the memory or memories that are needed to correctly answer the question. The recall, the percentage of the correct memories were retrieved by the chatbot during answering, is then calculated.

6.3.3 Adversarial Questions

A special note must be made for adversarial questions: while the expected answers for single- and multi-hop questions are gold-standard answers, for adversarial questions they represent adversarial answers: the answers the chatbot would give, if it has been successfully tricked. As a result, the F1 scores for adversarial questions must be interpreted opposite to single- and multi-hop questions, with *lower* F1 scores representing *better* performance, and vice-versa.

7 Results

The data collected, almost 40,000 datapoints, are large and complex and thus will be treated in parts. First a cursory exploration of the data will be presented. Then, each pipeline will be tested against the baseline results, first according to individual features, and then according to each individual pipeline. Following this, each part of the pipeline is examined individually and in combination with the others to identify, analyze, and explain significant differences and interactions. Finally, pipeline features are analyzed in respect to the three question types: single-hop, multi-hop, and adversarial. While two metrics — F1 score (BERTScore) and recall accuracy — were collected, all statistical tests are performed on the former, the latter only being treated in the data exploration section.

7.1 Data Exploration

The test result data, specifically the F1 scores, prove to be eccentric: their distribution is non-normal, heteroscedastic, platykurtic, and highly skewed to the right. Despite this, the data do seem to preserve a similar shape, both as a whole and when grouped by feature variant, which is exemplified in Figure 2 below. Thus, non-parametric approaches to analysis were necessary, namely Mann-Whitley U tests and ART ANOVAs; these will be covered in the following subsections. Also, because this abnormal data are not necessarily clustered around the mean, it was more appropriate to look at the medians; this will be the practice for this and future sections.

7.1.1 F1 Score

Looking at the median of each tested pipeline in Table 1, as well as the median of the baseline data, reveals some interesting trends.

First, baseline F1 score is ranked eighth out of all tests and thus divides the data in two, with seven of the LTM models performing better than the baseline, and seventeen performing worse. This division mostly corresponds memory storage format: all seven better-than-baseline models use knowledge bases, while twelve of the seventeen worse-than-baseline models use knowledge graphs. Thus, it appears that knowledge bases outperform knowledge graphs, and are the only memory-storage format that offers any improvement to having no long-term memory at all.

Additional observations can be made by looking at the knowledge base and graph models that stand out the most. The best performing knowledge graph model stores observations, retrieves them based on cosine similarity, and thereafter performs reflection; this is interesting, since it so happens that the knowledge base equivalent of this pipeline is the second best performing overall. Of the five knowledge base models that perform worse than the baseline, four of them use topic overlap retrieval, four lack a reflection step, and three store summaries. So then, its seems that pipelines involving observations, cosine similarity, and reflection appear to be some of the strongest performing, while those involving summaries, topic overlap, and no reflection some of the weakest performing.

Looking in terms of memory unit types, models storing observations and turns are more likely to perform worse than the baseline, and this tend is even more pronounced for summaries. Turning to retrieval method, a prominent trend is that the top four performing models all employ cosine similarity, while the bottom three performing models employ topic overlap. Finally, all but two better-than-baseline models have a reflection step. Thus, it appears that observations and turns perform better than summaries, cosine similarity performs better than topic overlap, and reflection performs better than no reflection.

7.1.2 Recall

Looking at the average recall, again in Table 1, one finds that the best performing models tend to have some of the highest recall: the top four performing models have recalls between 51% to 54%.

However, the absolute highest recalls belong not to these, but to models with the combination of knowledge bases and, interestingly enough, summaries: these can have recalls in the range of 70% to 76%, even though such models have much lover median F1 scores than their turn and observation counterparts. This seeming discrepancy can be explained by the relatively small size of knowledge bases that store summaries: since summaries are derived from entire sessions, and there are only 19-35 sessions in a LoCoMo conversation, the knowledge bases constructed from these summaries will only have 19 to 35 memories in total. Combined with the k-value of 10 used in retrieval, there is a 33% to 50% chance that the correct memory will be retrieved at random, which only increases when taking cosine similarity and topic overlap into account.

Another trend is that knowledge graphs appear to have over lower recall rates than knowledge bases: not one knowledge graph model has a higher recall than a knowledge base model. Indeed, the best knowledge graph model recall (0.292161) is almost ten full points behind the worst knowledge base model recall (0.383856). The reason for this difference is difficult to discern, although it might be related to the issues knowledge graphs have with information loss and retrieval, which will be covered the following subsections.

7.2 Analysis Against Baseline

Since the data is abnormally distributed and heteroscedastic, but nonetheless regular in general shape, pairwise testing was done using Mann-Whitney U tests with Holm–Bonferroni correction.

The data were compared against the baseline data in two ways. First, the data were grouped according to each variant of each feature of the pipeline (memory storage format, memory unit type, retrieval method, and reflection). In other words, each variant part of

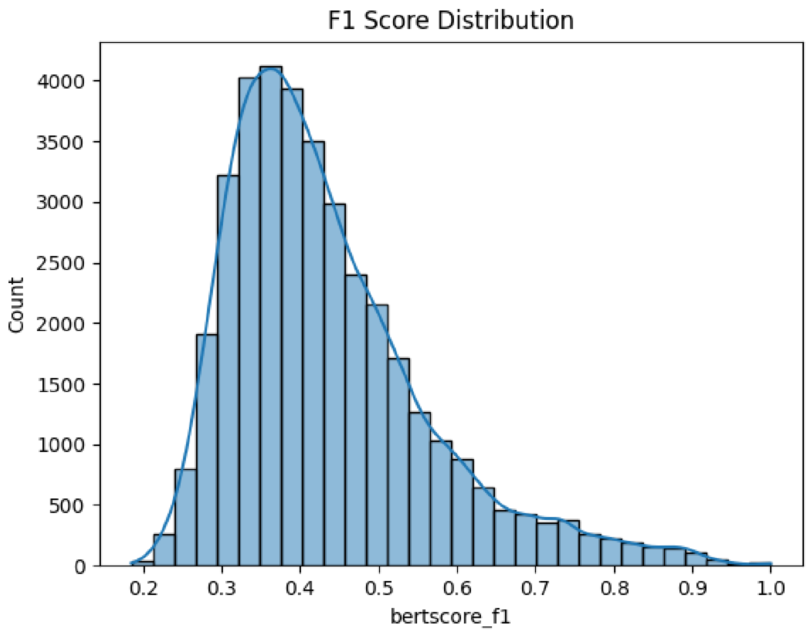

Figure 2. F1 Score Distribution of All Pipelines

memory_storage_format	memory_unit_type	retrieval_method	reflection	bertscore_f1	recall
knowledge base	turn	cosine similarity	True	0.453117	0.532216
knowledge base	observation	cosine similarity	True	0.452971	0.516694
knowledge base	observation	cosine similarity	False	0.444574	0.516694
knowledge base	turn	cosine similarity	False	0.437911	0.532216
knowledge base	turn	topic overlap	True	0.430522	0.439872
knowledge base	summary	cosine similarity	True	0.430064	0.751187
knowledge base	observation	topic overlap	True	0.428343	0.383856
Baseline	Baseline	Baseline	Baseline	0.420121	0.000000
knowledge base	summary	topic overlap	True	0.419738	0.708246
knowledge graph	observation	cosine similarity	True	0.412327	0.212505
knowledge graph	observation	cosine similarity	False	0.409076	0.212505
knowledge graph	turn	cosine similarity	True	0.404336	0.203542
knowledge graph	observation	topic overlap	True	0.403828	0.164406
knowledge base	turn	topic overlap	False	0.402905	0.439872
knowledge graph	summary	cosine similarity	True	0.401496	0.280427
knowledge base	observation	topic overlap	False	0.400507	0.383856
knowledge graph	turn	topic overlap	True	0.394122	0.153908
knowledge graph	summary	topic overlap	True	0.391807	0.292161
knowledge graph	observation	topic overlap	False	0.391188	0.164406
knowledge graph	summary	cosine similarity	False	0.390025	0.280427
knowledge base	summary	cosine similarity	False	0.387475	0.751187
knowledge graph	turn	cosine similarity	False	0.387198	0.203542
knowledge base	summary	topic overlap	False	0.385587	0.708246
knowledge graph	summary	topic overlap	False	0.376829	0.292161
knowledge graph	turn	topic overlap	False	0.374681	0.153908

Table 1. F1 Score and Recall for all RAG Pipelines and Baseline

the long-term memory pipeline was compared as a group against the baseline. Next, each individual pipeline was compared against the baseline.

7.2.1 By Feature Variants

When grouped by feature variant (Tables 2 through 5), no group of models proved to perform significantly better than the baseline; indeed, only one group, those with knowledge bases, even had a higher median F1 score than the baseline. All other groups of models are significantly outperformed by the baseline.

These results seem to imply that all tested long-term memory features are no better — and indeed worse — than simply having the language model make educated guesses. However, these exact features were proven to outperform the baseline in all other papers in the current literature (e.g. Maharana et al. (2024), Li et al. (2024)). The key difference is that those papers only tested models that use knowledge bases as a memory storage format; it follows that the inclusion of knowledge graph models might be responsible for the present discrepancy. Indeed, this appears to be the case: when rerunning the Mann-Whitney U tests with knowledge graph data excluded (Tables 6 through 8), feature variants begin to significantly outperform the baseline; namely cosine-similarity, turns, observations, and reflection. Of these, the presence of reflection appears to result in the most marked difference from the baseline (r = -0.084946). Why reflection might have such a pronounced effect size will be covered later.

7.2.2 By Pipeline

Looking at each long-term memory pipeline that was tested individually (Table 9, presented in landscape), we find which of the trends discussed in the data exploration subsection are statistically significant.

The most striking observation is that all but one knowledge graph pipelines perform significantly worse than the baseline; only the top-performing knowledge graph model, which involves observations, cosine similarity, and reflection, is not significantly worse, and only barely so (p-value = 0.061958).

Next, of the seven pipelines that performed better than the baseline, six do so significantly. These six all use knowledge bases, and store either turns or observations. Of these, the top four performing models are all highly significant, and retrieve memories using cosine similarity. The other two are only slightly significant (p-values = 0.020339, 0.010427), and retrieve memories using topic overlap. The non-significant pipeline, unlike the others, stores summaries; also, while non-significant, it is only barely so (p-value = 0.061958).

Meanwhile, of the seventeen pipelines that performed worse than the baseline, fourteen do so significantly. The sole exceptions are knowledge base models which both use topic overlap, and the best performing knowledge graph model, which was previously

Level 1	Level 2	Median 1	Median 2	Statistic (U)	Raw p-value	Corrected p-value	Rank-biserial corr.
knowledge base	none	0.421058	0.420121	15051412.500000	0.210100	0.210100	-0.019014
knowledge graph	none	0.394159	0.420121	12611947.500000	0.000000	0.000000	0.146143

Table 2. Memory Storage Format against Baseline

Level 1	Level 2	Median 1	Median 2	Statistic (U)	Raw p-value	Corrected p-value	Rank-biserial corr.
observation	none	0.416780	0.420121	9744138.500000	0.499076	0.499076	0.010450
turn	none	0.406920	0.420121	9308817.000000	0.000407	0.000814	0.054659
summary	none	0.397231	0.420121	8610404.500000	0.000000	0.000000	0.125585

Table 3. Memory Unit Type against Baseline

Level 1	Level 2	Median 1	Median 2	Statistic (U)	Raw p-value	Corrected p-value	Rank-biserial corr.
cosine similarity	none	0.414514	0.420121	14458323.500000	0.163500	0.163500	0.021140
topic overlap	none	0.398672	0.420121	13205036.500000	0.000000	0.000000	0.105990

Table 4. Retrieval Method against Baseline

Level 1	Level 2	Median 1	Median 2	Statistic (U)	Raw p-value	Corrected p-value	Rank-biserial corr.
True	none	0.416798	0.420121	14572633.000000	0.377083	0.377083	0.013401
False	none	0.395878	0.420121	13090727.000000	0.000000	0.000000	0.113729

Table 5. Reflection against Baseline

Level 1	Level 2	Median 1	Median 2	Statistic (U)	Raw p-value	Corrected p-value	Rank-biserial corr.
observation	none	0.432982	0.420121	5268068.500000	0.000018	0.000053	-0.069980
turn	none	0.431144	0.420121	5238188.500000	0.000088	0.000176	-0.063911
summary	none	0.405148	0.420121	4545155.500000	0.000002	0.000010	0.076849

Table 6. Memory Unit Type against Baseline (knowledge graphs excluded)

Level 1	Level 2	Median 1	Median 2	Statistic (U)	Raw p-value	Corrected p-value	Rank-biserial corr.
cosine similarity	none	0.432617	0.420121	7869343.000000	0.000031	0.000063	-0.065544
topic overlap	none	0.411232	0.420121	7182069.500000	0.080519	0.080519	0.027516

Table 7. Retrieval Method against Baseline (knowledge graphs excluded)

Level 1	Level 2	Median 1	Median 2	Statistic (U)	Raw p-value	Corrected p-value	Rank-biserial corr.
True	none	0.435632	0.420121	8012634.500000	0.000000	0.000000	-0.084946
False	none	0.406260	0.420121	7038778.000000	0.002882	0.002882	0.046918

Table 8. Reflection against Baseline (knowledge graphs excluded)

mentioned. Given that twenty-four pipelines were tested in total, this means that the majority of long-term memory pipelines tested are significantly worse than having no long-term memory at all. Common among these are a reliance on knowledge graphs, and among those which use knowledge bases, the use of topic overlap, summaries, and a lack of reflection.

7.2.3 Examining the Performance of Knowledge Graphs

Given these baseline tests, a major observation can be made: knowledge graph models not only fail to outperform a baseline which relies on educated guesses, but somehow significantly performs worse than it. This would imply, then, that knowledge graphs are not merely incapable of providing models with the correct memories, but actively prevent the model from answering correctly, even if it could have guessed the correct answer. While explanations can be found for why knowledge graphs perform worse than knowledge bases, as will be seen later, why they should be worse than the baseline is not immediately obvious.

One might speculate, however, that knowledge graphs actively misinform the chatbot due to the way it understands the task of remembering. While the chatbot might be free to make educated guesses when no memories are available, when they *are* available the chatbot is asked to make use of them in formulating its answer. This might shift the chatbot's priorities, so that it cares more about using the retrieved memories to make an answer, than whether that answer makes sense given the context. And, given the poor recall that knowledge graphs exhibit, this means that the memories the chatbot tries to make an answer from are much more likely to lack the correct answer. In other words, when the chatbot has access to memories, it becomes in a sense "overconfident" in those memories, and trusts them more than its own intuition, thus providing a confident misinformed answer instead of a tentative educated guess. However, this is only speculation, and a deeper investigation of this phenomenon would go beyond the scope of this thesis.

7.3 Tests Within Features

Having finished testing features and pipelines against the baseline, it comes time to test the variants of each feature against the others. Like with the baseline tests, pairwise testing was carried out using Mann-Whitney U tests with Holm–Bonferroni correction. Below each feature is considered in turn.

7.3.1 Memory Storage Format

Beginning with memory storage formats, turning to Table 10 we find that knowledge bases perform significantly better than knowledge graphs. This difference is also the greatest in magnitude (r = -0.139818) of those between any feature variants. Given the results presented up until now, which testifies to the stark difference in performance between

Level 1	Level 2	Median 1	Median 2	Statistic (U)	Raw p-value	Corrected p-value	Rank-biserial corr.
knowledge base, turn, cosine similarity, True	none	0.453117	0.420121	1420395.500000	0.000000	0.000000	-0.153967
knowledge base, observation, cosine similarity, True	none	0.452971	0.420121	1436346.500000	0.000000	0.000000	-0.166926
knowledge base, observation, cosine similarity, False	none	0.444574	0.420121	1368087.000000	0.000000	0.000001	-0.111470
knowledge base, turn, cosine similarity, False	none	0.437911	0.420121	1343150.500000	0.000010	0.000087	-0.091211
knowledge base, turn, topic overlap, True	none	0.430522	0.420121	1303783.500000	0.004068	0.020339	-0.059228
knowledge base, summary, cosine similarity, True	none	0.430064	0.420121	1292310.000000	0.015490	0.061958	-0.049907
knowledge base, observation, topic overlap, True	none	0.428343	0.420121	1312472.000000	0.001303	0.010427	-0.066287
knowledge base, summary, topic overlap, True	none	0.419738	0.420121	1247327.000000	0.516930	0.516930	-0.013362
knowledge graph, observation, cosine similarity, True	none	0.412327	0.420121	1170665.000000	0.017650	0.061958	0.048921
knowledge graph, observation, cosine similarity, False	none	0.409076	0.420121	1154985.500000	0.002783	0.016697	0.061659
knowledge graph, turn, cosine similarity, True	none	0.404336	0.420121	1108128.500000	0.000001	0.000014	0.099727
knowledge graph, observation, topic overlap, True	none	0.403828	0.420121	1110816.000000	0.000002	0.000022	0.097544
knowledge base, turn, topic overlap, False	none	0.402905	0.420121	1170859.000000	0.018018	0.061958	0.048763
knowledge graph, summary, cosine similarity, True	none	0.401496	0.420121	1097167.500000	0.000000	0.000002	0.108632
knowledge base, observation, topic overlap, False	none	0.400507	0.420121	1151163.000000	0.001681	0.011770	0.064765
knowledge graph, turn, topic overlap, True	none	0.394122	0.420121	1049563.500000	0.000000	0.000000	0.147307
knowledge graph, summary, topic overlap, True	none	0.391807	0.420121	1023658.000000	0.000000	0.000000	0.168353
knowledge graph, observation, topic overlap, False	none	0.391188	0.420121	1039603.500000	0.000000	0.000000	0.155399
knowledge graph, summary, cosine similarity, False	none	0.390025	0.420121	1040190.500000	0.000000	0.000000	0.154922
knowledge base, summary, cosine similarity, False	none	0.387475	0.420121	1009053.500000	0.000000	0.000000	0.180218
knowledge graph, turn, cosine similarity, False	none	0.387198	0.420121	1017843.500000	0.000000	0.000000	0.173077
knowledge base, summary, topic overlap, False	none	0.385587	0.420121	996465.000000	0.000000	0.000000	0.190445
knowledge graph, summary, topic overlap, False	none	0.376829	0.420121	904233.000000	0.000000	0.000000	0.265377
knowledge graph, turn, topic overlap, False	none	0.374681	0.420121	895093.000000	0.000000	0.000000	0.272803

Table 9. Individual Pipelines against Baseline

Level 1	Level 2	Median 1	Median 2	Statistic (U)	Raw p-value	Corrected p-value	Rank-biserial corr.
knowledge base	knowledge graph	0.421058	0.394159	202029113.500000	0.000000	0.000000	-0.139818

Table 10. Memory Storage Format Variants Comparison

Level 1	Level 2	Median 1	Median 2	Statistic (U)	Raw p-value	Corrected p-value	Rank-biserial corr.
observation	summary	0.416780	0.397231	86739698.500000	0.000000	0.000000	-0.101088
turn	observation	0.406920	0.416780	75799831.500000	0.000000	0.000000	0.037784
turn	summary	0.406920	0.397231	83599832.500000	0.000000	0.000000	-0.061230

Table 11. Memory Unit Type Variants Comparison

Level 1	Level 2	Median 1	Median 2	Statistic (U)	Raw p-value	Corrected p-value	Rank-biserial corr.
cosine similarity	topic overlap	0.414514	0.398672	190299585.500000	0.000000	0.000000	-0.073642

Table 12. Retrieval Method Variants Comparison

Level 1	Level 2	Median 1	Median 2	Statistic (U)	Raw p-value	Corrected p-value	Rank-biserial corr.
True	False	0.416798	0.395878	192449006.500000	0.000000	0.000000	-0.085769

Table 13. Reflection Variants Comparison

Level 1	Level 2	Median 1	Median 2	Statistic (U)	Raw p-value	Corrected p-value	Rank-biserial corr.
observation	summary	0.432982	0.405148	22250600.500000	0.000000	0.000000	-0.129811
turn	observation	0.431144	0.432982	19610502.000000	0.680501	0.680501	0.004244
turn	summary	0.431144	0.405148	22124596.000000	0.000000	0.000000	-0.123413

Table 14. Memory Unit Type Variants Comparison (knowledge graphs excluded)

Level 1	Level 2	Median 1	Median 2	Statistic (U)	Raw p-value	Corrected p-value	Rank-biserial corr.
cosine similarity	topic overlap	0.432617	0.411232	47667499.500000	0.000000	0.000000	-0.075732

Table 15. Retrieval Method Variants Comparison (knowledge graphs excluded)

Level 1	Level 2	Median 1	Median 2	Statistic (U)	Raw p-value	Corrected p-value	Rank-biserial corr.
True	False	0.435632	0.406260	49230046.500000	0.000000	0.000000	-0.110994

Table 16. Reflection Variants Comparison (knowledge graphs excluded)

knowledge base and knowledge graph models, this should hardly be surprising. However, the reasons for this marked difference has up until now not been explored in detail.

One explanation that has been previously alluded to is that relationship extraction, which is a prerequisite step before uploading text into knowledge graphs, results in information loss. Relationship extraction only selects specific noun, adjective, and verb phrases from a text, and thus anything passed over is missing from the resulting relationship triples. Information and nuances important to answering questions could therefore be excluded from the knowledge graph, meaning relevant memories would be both less informative, and harder to retrieve due to lower cosine similarity and topic overlap scores. For example, take the following question from LoCoMo:

> "What did Caroline research?"

The answer is "adoption agencies", which can be found in this sentence:

> "Researching adoption agencies — it's been a dream to have a family and give a loving home to kids who need it."

The relationship triplets extracted from this sentence, however, are the following:

> ('it', 's been a dream to have a family and give a loving home to kids who need it', 'N/A')
>
> ('it', 's been a dream to have a family', 'N/A')
>
> ('it', 's been a dream to give a loving home to kids who need it', 'N/A')
>
> ('it', 's been', 'a dream')
>
> ('who', 'need it', 'N/A')

As can be seen, the key piece of information required to answer the question, "researching adoption agencies", didn't match the syntactic criteria that the relationship extractor uses for selecting the source, relationship, and target and thus was omitted. The question becomes unanswerable. Lacking this vital information, these relationship triplets also have an average lower cosine similarity with the question (0.090356088429689) compared to the original text (0.17500635981559753), meaning they are less likely to be retrieved.

The above example hints at another reason for the poor knowledge graph performance: the limitations of the relationship extractor. Of the five relationship triplets extracted from the text above, only one, ('it', 's been', 'a dream'), has both a source and a target. The rest have "N/A" as their target, which is a default result which occurs when the extractor cannot find a valid target. The relationship extractor also has issues with questions:

> "What does Melanie do to destress?"

becomes

('what', 'does', 'melanie')

when the relationship triples are extracted, completely cutting off the main content of the question ("do to relax"), and rendering the question unanswerable. These limitations in the ability to correctly extract the most important relationships from a text, especially when it comes to questions, are certainly a contributing factor to poor performance.

Another flaw inherent to the specific knowledge graph implementation used in this study that can help account for the low F1 score is that knowledge graph querying is done according to sources and targets, but not according to similar relationships. Since relationships, or edges, contain all the verbal information which defines the relationship between the source and the target, the knowledge graph cannot be explored according to matching verbs. For example, given the sentence "I taught Esperanto", which would result in the triplet ('i', 'taught', 'esperanto'), it is not possible to search for all memories that contain the edge "taught", that is, all memories involving teaching.

This method of querying the knowledge graph was excluded on the grounds that the most frequent edges, such as "is", "like", and "went to", are too semantically light to be useful for querying, and so querying by edges was deemed too impractical to implement. Nonetheless, this omission might also be responsible for the poor performance of knowledge bases.

A final explanation lies not in the structure of the particular knowledge graph used in this study, but in an issue common to all knowledge graphs, again involving how they are queried. Knowledge graphs are queried and explored according to shared sources and targets, i.e. exactly matching noun phrases and predicate adjectives. This technique comes with a major drawback: unless a query contains one of the exact noun or adjective phrases stored in the knowledge graph, it is impossible for the knowledge graph to retrieve the relevant answer. In other words, even if the knowledge graph contains the correct information needed to answer a question, that information might not be retrieved because knowledge graphs have no means of handling paraphrasing and synonyms. Consider the following example. In the LoCoMo dataset, there is this question:

"What is Caroline's relationship status?"

When converted to a relationship triple, it looks like this:

('what', 'is', 'caroline s relationship status')

The answer to this question, "single", is found in two sentences:

1) "Their love and help have been so important especially after that tough breakup."

2) "It'll be tough as a single parent, but I'm up for the challenge."

When converted into relationship triples, the relevant triples are the following:

1) ('their love and help', 'have been so important especially after', 'that tough breakup')

2) ('it', 'll be tough as', 'a single parent')

As can be seen, although the phrase "caroline s relationship status" is semantically related to "that tough breakup" and "a single parent", these two memories would never be retrieved from the knowledge graph because they are not exact matches.

Note: Given the drastic difference between knowledge bases and knowledge graphs, both in nature and performance, and the apparent unfeasibility of knowledge graphs, following sections will consider the combined results data, and then the data when excluding knowledge graphs. It is hoped that this will help reveal differences between feature variants that would otherwise be masked by the poor knowledge graph data.

7.3.2 Memory Unit Type

Continuing on to memory unit types, in Table 11 we find some interesting results. Considering the data as a whole, we find that models which store observations perform significantly better than those which store turns, while those that store either type of memory unit are significantly better than those which store summaries.

When excluding the knowledge graph data (Table 14) the results are mostly the same, with one key difference: observation models no longer significantly outperform turn models, and indeed to a minor extent the reverse is true: turn models slightly outperform observation models, albeit not significantly.

These results more or less support those of Maharana et al. (2024): observation models outperform turn models, and both outperform summary models. As for why this pattern exists, there are several possible explanations. Summaries inherently involve the loss of information, and thus are less likely to contain the information needed to answer any particular question. However, this also applies to observations: the number of observations is always lower than the amount of turns in a session, with a general ratio of one observation to every three turns, meaning that information is always lost. And yet, observations are nonetheless either superior or at least on par with turns in terms of F1 scores.

The reason why storing observations produces better results than storing summaries might be a simple matter of prompting: according to the prompts used for generating the observations and summaries found in LoCoMo, the LLM is free to make as many observations from a session transcript as it it sees fit, whereas it demands this same information be condensed into a single summary. The result is that the LLM is free to give each significant piece of information its own separate observation, while for summaries it must make concessions.

As for why observations can outperform turns, which inherently involve no information loss, the answer might be a matter of information consolidation. Information that would split up between multiple turns in a turn model, including pronouns and other indirect references, would be consolidated into a single observation. In other words, while a turn model would have to retrieve two or three specific turns to get the full context needed to answer a question, an observation model would only need to return one. However, this doesn't explain why observations perform no better than turns when knowledge graph data is excluded.

7.3.3 Retrieval Method

Looking at Tables 12 and 15, there is a consistent pattern in retrieval methods: cosine similarity performs significantly better than topic overlap. However, this directly contradicts the findings of Li et al. (2024), despite the fact that the method used in the present study was taken directly from that one. Still, there are two differences between that implementation of topic overlap retrieval and the present. First, Li et al. (2024) used event summaries — short summaries of each events which occur in the course of a conversation — as memory units, rather than turns, observations, or session summaries. Second, the present model includes the entire noun phrase as topics in addition to the noun, whereas Li et al. (2024) only define topics as nouns. There are no obvious qualities that event summaries have which would favor topic overlap retrieval, and although the the inclusion of noun phrases as topics might decrease topic overlap scores (since the larger the noun phrase, the greater the total amount of topics for a given query or memory), this only occurs when comparing a less specific topic (e.g. "dogs") to a more specific one (e.g. "big blue dogs"), since this design rewards topics with similar levels of specificity; memories are still perfectly capable of being retrieved based on matching nouns alone.

While this discrepancy with previous findings remains unexplained, an explanation for the present results can be found. As with knowledge graphs exploration, topic overlap retrieval relies on the exact matching of strings, rendering it blind to paraphrasing and synonyms. This decreases the chances of retrieving a memory which is necessary for answering a question.

7.3.4 Reflection

As with retrieval methods, the presence or absence of reflection shows the same results across both datasets (Tables 13 and 16): reflection produces significantly higher F1 scores than no reflection. This can be explained by looking at the reflections themselves. When a model (knowledge base, turns, cosine similarity) is presented the question

"What career path has Caroline decided to persue?"

the retrieval system returns ten memories, of which only three are relevant[2]:

> "Lately, Caroline has been looking into counseling and mental health as a career. Caroline want to help people who have gone through the same things as Caroline."
>
> "Caroline mentor a transgender teen just like Caroline. We has been working on building up confidence and finding positive strategies, and it's really been paying off! We had a great time at the LGBT pride event last month."
>
> "Woah, Caroline, it sounds like Caroline is doing some impressive work. It's inspiring to see Caroline's dedication to helping others. What motivated Caroline to pursue counseling?"

When all ten memories are fed into the reflection model, the resulting reflection is the following:

> "As I revisit these memories, I see that Caroline has been seeking a career path that aligns with their values of compassion and helping others. Their personal experiences, including mentoring a transgender teen and overcoming similar challenges, have motivated them to pursue counseling and mental health as a career."

As can bee seen, the reflection step acts as a filter: given a list of 10 potentially relevant memories, and the question they were retrieved to answer, the reflection model attends only to the memories that appear most relevant to the question. This reduces the amount of information fed into the chatbot during the response generation step, which makes it easier for the chatbot to utilize the information to answer the question correctly. In the case where reflection is not used, the unfiltered memories are given directly the chatbot and could act as noise, hindering the chatbot's ability to answer the question correctly.

7.4 Tests Between Features

Now it is time to turn our attention to what might be the most interesting analyses of this thesis, that of interactions between different features in the conversational RAG pipeline. As with the pairwise comparisons, a non-parametric method was utilized, in this case Aligned Rank Transform ANOVA. Alongside this, plots are presented to ascertain which specific variants of each feature are interacting, and the nature of their interaction. When possible, tentative explanations are offered for these interactions, although most of these cannot be considered definitive, and require further research to confirm or deny.

[2]As mentioned in Section 5.4, first- and second-person pronouns have been replaces with the names of the speaker and addressee, respectively.

7.4.1 Memory Storage Format × Memory Unit Type

Significant interactions were found between memory storage formats and memory unit types (Table 17, Figure 3). This makes sense, since there is an inherent, intimate relationship between a storage system, and that which is stored within it. Looking at the plot, we first find that observations and summaries draw slightly closer in distance when used with knowledge graphs, meaning that observations are particularly more effective than summaries when used with knowledge bases, and that knowledge graphs mitigate this effect. Why this is the case is not immediately clear. Looking at the same plot, however, we find that turns react to memory formats in a much more extreme way: in knowledge bases, they are equal in performance to observations, whereas in knowledge graphs they perform much worse, being equal to summaries.

Focusing on this second interaction, one explanation might be that turns, being pure spoken dialogue, are much more grammatically complex than observations, which are simple factual statements, and thus are more difficult for the relationship extractor to successfully parse, leading to worse retrieval and lower F1 scores. This would account for why turns perform much worse than observations when stored in knowledge graphs.

As for why turn-based models have similar performance to observation-based ones when stored in knowledge bases, it is harder to find an explanation. Earlier it was argued that observations should capture more referential information than turns, in order to explain the significant difference between observations and turns within the data. However, now it can be seen that this difference is only present in the knowledge graph models, and does not hold true for knowledge base ones. Thus we have no choice other than to reject this explanation.

Instead, we can conclude that turns and observations have no real difference in the quality of information they store, leading to similar scores when used in conjunction with knowledge bases.

	sum_sq	df	F	PR(>F)
C(memory_storage_format)	7764215620.759016	1.000000	66.021742	0.000000
C(memory_unit_type)	2877075893.649273	2.000000	12.232373	0.000005
C(memory_storage_format):C(memory_unit_type)	11288676962.470539	2.000000	47.995712	0.000000
Residual	4427673492927.104492	37650.000000	NaN	NaN

Table 17. ART ANOVA for Memory Storage Format × Memory Unit Type

7.4.2 Memory Storage Format × Retrieval Method

Only a very slight significant interaction was found between memory storage formats and retrieval methods (Table 18, Figure 4); however, the plotted lines for cosine similarity and topic overlap remain more or less parallel between memory storage formats, only ever so slightly narrowing when used with knowledge graphs. Thus, while significant, this interaction is not worth investigating.

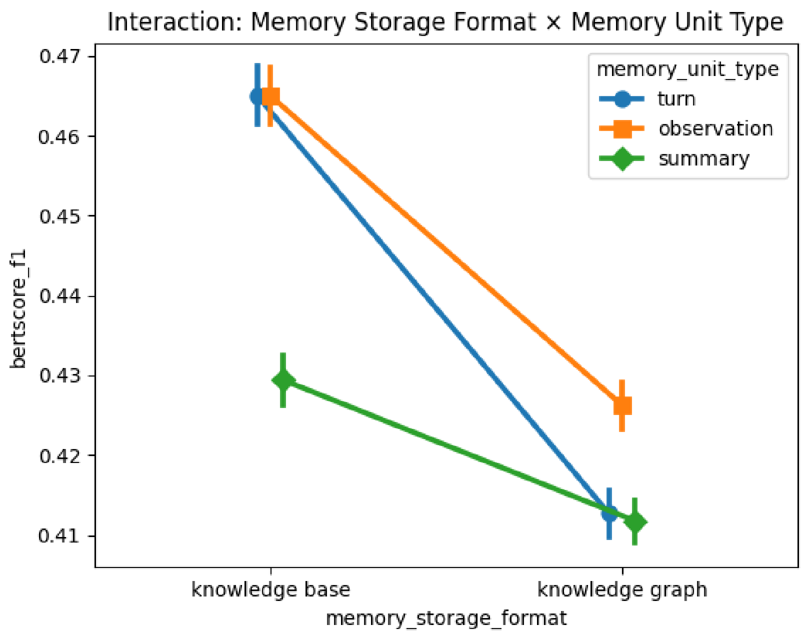

Figure 3. Interactions between Memory Storage Format and Memory Unit Type

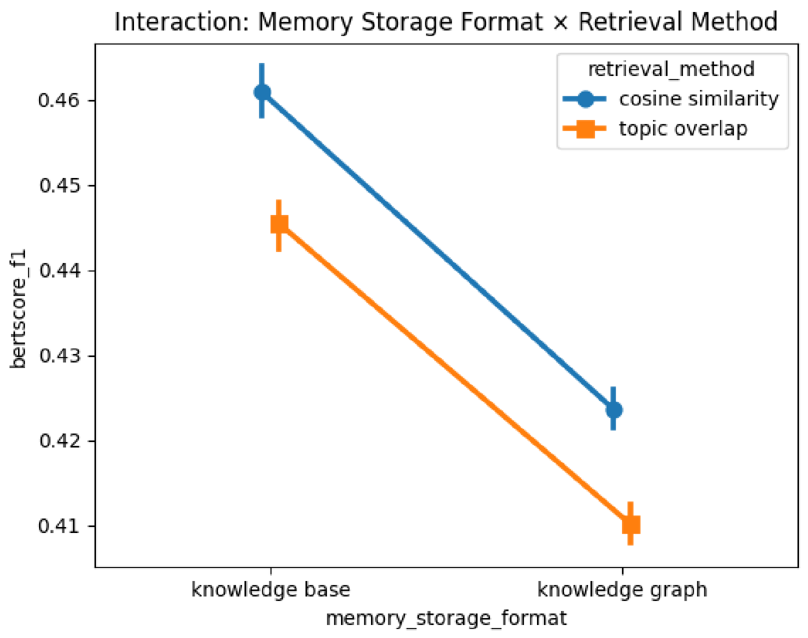

Figure 4. Interactions between Memory Storage Format and Retrieval Method

	sum_sq	df	F	PR(>F)
C(memory_storage_format)	8461821515.733070	1.000000	71.749897	0.000000
C(retrieval_method)	16801832.237121	1.000000	0.142467	0.705843
C(memory_storage_format):C(retrieval_method)	637581400.465838	1.000000	5.406212	0.020070
Residual	4440487255435.046875	37652.000000	NaN	NaN

Table 18. ART ANOVA for Memory Storage Format × Retrieval Method

7.4.3 Memory Storage Format × Reflection

Significant interacts were found between memory storage formats and reflection, with reflection having a much larger effect on F1 scores when used in conjunction with knowledge bases (Table 19, Figure 5). One possible explanation for this is the lower recall displayed by knowledge graphs: reflection has been shown to work as a filter which selects which memory out those retrieved are relevant to the question, and since knowledge graphs are much worse at retrieving the correct memories, reflection will be less effective with them.

	sum_sq	df	F	PR(>F)
C(memory_storage_format)	8507983728.165408	1.000000	72.217997	0.000000
C(reflection)	103393458.241963	1.000000	0.877631	0.348857
C(memory_storage_format):C(reflection)	5219596560.485344	1.000000	44.305304	0.000000
Residual	4435772485741.586914	37652.000000	NaN	NaN

Table 19. ART ANOVA for Memory Storage Format × Reflection

7.4.4 Memory Unit Type × Retrieval Method

Memory unit type was shown to have a significant effect on retrieval method, specifically when it comes to summaries (Table 20, Figure 6). When summaries are the unit of memory being stored, the difference between cosine similarity and topic overlap is much less than with the other two memory unit types: cosine similarity displays much less of an advantage. This might be the influence of unit length on the effectiveness of cosine similarity:. Summaries are on average 650 characters long, whereas turns and observations are only 120 and 90 characters long on average, making summaries roughly six times longer. Having to store six times the information as other embeddings, and in the same number of dimensions, summary embeddings probably suffer from a generally more muted, less distinctive semantic profile, which would lead to less stark differences in cosine similarity scores during retrieval, making the retrieval method less effective.

	sum_sq	df	F	PR(>F)
C(memory_unit_type)	3874279975.711880	2.000000	16.411560	0.000000
C(retrieval_method)	26471068.961635	1.000000	0.224264	0.635812
C(memory_unit_type):C(retrieval_method)	1681647019.948506	2.000000	7.123505	0.000807
Residual	4444021063775.873047	37650.000000	NaN	NaN

Table 20. ART ANOVA for Memory Unit Type × Retrieval Method

7.4.5 Memory Unit Type × Reflection

Two significant interactions can be identified between memory unit types and reflection (Table 21, Figure 7). First, models storing observations are less effected by a lack of reflection compared to other models. This might be attributed to the previously discussed consolidation and synthesis of information inherent in the creation of observations, which could anticipate the consolidation and synthesis that occurs in reflection: observations have already gone through a form of reflection, and thus the lack of an additional reflection

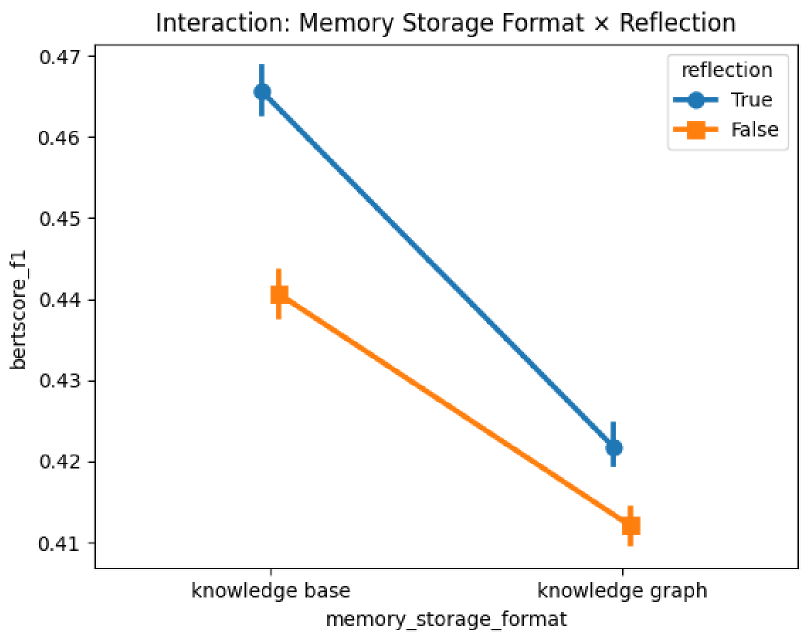

Figure 5. Interactions between Memory Storage Format and Reflection

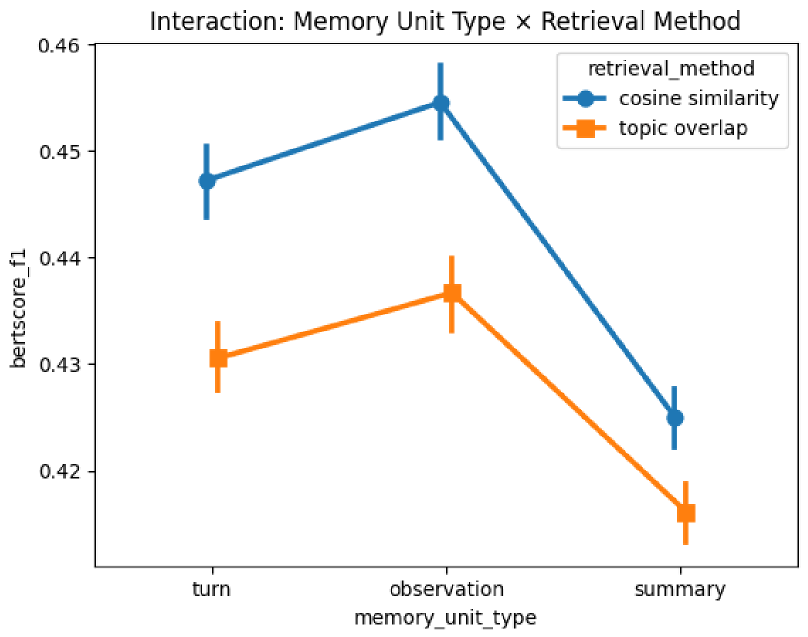

Figure 6. Interactions between Memory Unit Type and Retrieval Method

step is less consequential. However, as we have seen in Section 7.4.1, there is evidence against this hypothesis.

The other significant interaction is that between the lack of reflection and summaries, with the former having a much larger impact on the latter than in other models. As mentioned earlier, summaries are roughly six times larger on average than observations; thus, it might be that chatbots have increased difficulty filtering out the correct information from what is essentially a wall of text, and therefore benefit more from a filtering step like reflection.

	sum_sq	df	F	PR(>F)
C(memory_unit_type)	4270639549.551753	2.000000	18.092730	0.000000
C(reflection)	224861641.568090	1.000000	1.905270	0.167498
C(memory_unit_type):C(reflection)	1622205664.088950	2.000000	6.872537	0.001037
Residual	4443485754737.792969	37650.000000	NaN	NaN

Table 21. ART ANOVA for Memory Unit Type × Reflection

7.4.6 Retrieval Method × Reflection

Finally, a slightly significant interaction is found between retrieval method and reflection, with the use of reflection having a slightly larger effect when paired with topic overlap (Table 22, Figure 8). An explanation for why this might be the case, however, doesn't offer itself.

sum_sq	df	F	PR(>F)
60247520.018752	1.000000	0.509907	0.475183
181967688.749743	1.000000	1.540091	0.214612
632206247.051770	1.000000	5.350703	0.020719
4448729038964.180664	37652.000000	NaN	NaN

Table 22. ART ANOVA for Retrieval Method × Reflection

7.5 Tests Between Features and Question Types

The final stage of analysis will now look for interactions between each feature and each of the three question types: single-hop, multi-hop, and antagonistic. Before this, however, we shall look at the major trends that question types exhibit across all features.

7.5.1 General Trends Exhibited by Question Types

While the interactions between question types and pipeline features can vary widely, as will be seen below, two trends remain constant.

First, single-hop questions always produce the highest F1 scores, followed by multi-hop questions, and then adversarial questions. Single-hop questions leading to better performance than multi-hop questions supports the findings of Maharana et al. (2024). Since multi-hop questions require a chatbot to retrieve multiple memories, and thus are inherently harder to answer, this pattern is to be expected.

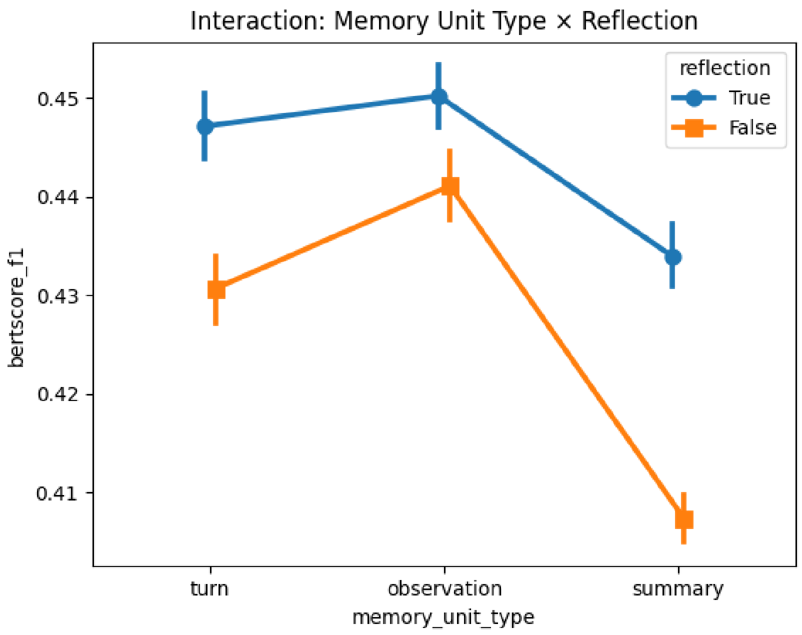

Figure 7. Interactions between Memory Unit Type and Reflection

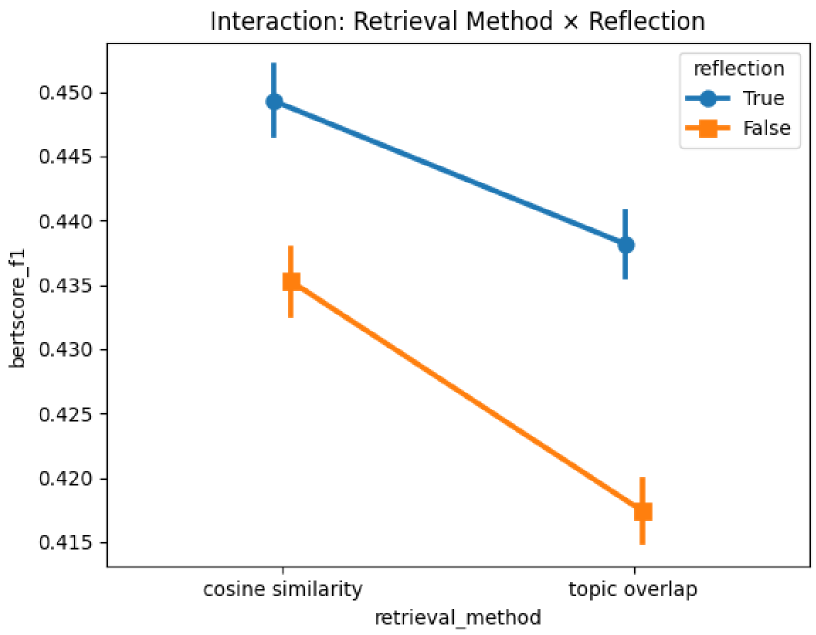

Figure 8. Interactions between Retrieval Method and Reflection

Second, adversarial questions always produce the lowest, and therefore best, F1 scores for feature variants which have been shown to be the worst performing: knowledge graphs, summaries, topic overlap, and no reflection. This is curious, as it appears unlikely that such poorly performing feature variants would all happen to be predisposed to be better at handling adversarial questions. On the other hand, these results are partially supported by Maharana et al. (2024), which show that summary models, otherwise the worst performing, actually excel at adversarial questions. Still, it seems more likely that these models are not receiving the correct memories, and responding that they cannot answer the question because they don't know the answer, rather than retrieving the correct information and identifying the question as adversarial. This would result in a bad BERTScore with the adversarial answer, and thus a deceptively low F1 score.

7.5.2 Memory Storage Format × Question Type

There is a very significant interaction between the memory storage format and single-hop questions: knowledge base models are much better at answering single-hop questions than the other two types, but for knowledge graph models this performance with single-hop questions is much less pronounced, with F1 scores essentially equal to those of multi-hop questions (Table 23, Figure 9).

Where does this drop off in the ability of knowledge graph models to answer single-hop questions come from? And why do they treat them similarly to multi-hop questions? One possible explanation is that since relationship extraction can convert a single text into multiple discrete triplets of information, the answer to a question which would be contained by a single turn, observation, or summary in a knowledge base would be split into several separate relationship triplets in a knowledge graph. This would mean that single-hop questions would actually become no different than multi-hop questions for knowledge graph models, as multiple relevant triples would need to be returned to answer the question fully. This would explain both why single-hop questions produced a worse performance, and why their performance appears to be the same as multi-hop questions.

	sum_sq	df	F	PR(>F)
C(memory_storage_format)	6865893552.409551	1.000000	58.559864	0.000000
C(question_type)	7350892592.852382	2.000000	31.348234	0.000000
C(memory_storage_format):C(question_type)	21085226023.871746	2.000000	89.918957	0.000000
Residual	4414301445925.313477	37650.000000	NaN	NaN

Table 23. ART ANOVA for Memory Storage Format × Question Type

7.5.3 Memory Unit Type × Question Type

Two significant interactions can be observed between memory unit types and question types (Table 24, Figure 10).

First, while single- and multi-hop questions produce a constant trend for turns and observations, with the former leading to better results than the latter, for summaries single-hop questions lose their advantage, and perform no better than multi-hop questions. This is quite surprising, as this contradicts the results of Maharana et al. (2024) which show that summary models perform more than twice as well on single-hop questions than multi-hop ones. Why this might be the case, is unknown.

Second, unlike the other two question types, which produce higher F1 scores with observation models than with turn models, adversarial questions produce the opposite results: turn models have the higher F1 scores. As theorized in Section 7.5.1, it's possible that the F1 scores for adversarial questions don't accurately represent how well a model performed, and that poor memory retrieval could result in low F1 scores. Because of this, an explanation of this trend is not possible.

	sum_sq	df	F	PR(>F)
C(memory_unit_type)	3336394640.014003	2.000000	14.181104	0.000001
C(question_type)	8531410536.839378	2.000000	36.262143	0.000000
C(memory_unit_type):C(question_type)	9122442742.766777	4.000000	19.387141	0.000000
Residual	4428613212876.884766	37647.000000	NaN	NaN

Table 24. ART ANOVA for Memory Unit Type × Question Type

7.5.4 Retrieval Method × Question Type

When looking at retrieval methods and question types, a significant interaction was found: multi-hop questions resulted in significantly worse F1 scores for topic overlap models (Table 25, Figure 11). More specifically, cosine similarity models handled multi-hop questions nearly as well as single-hop ones, both with much higher F1 scores than adversarial questions, while it was the reverse for topic overlap models, whose F1 scores for multi-hop questions resembled those for adversarial questions. Why this might be the case, however, is not immediately clear.

	sum_sq	df	F	PR(>F)
C(retrieval_method)	25682262.971798	1.000000	0.218082	0.640508
C(question_type)	9272628090.726065	2.000000	39.369433	0.000000
C(retrieval_method):C(question_type)	6478976517.943932	2.000000	27.508235	0.000000
Residual	4433826172184.858398	37650.000000	NaN	NaN

Table 25. ART ANOVA for Retrieval Method × Question Type

7.5.5 Reflection × Question Type

A final interaction was found between reflection and question type, with multi-hop questions performing significantly worse with models lacking reflection (Table 26, Figure 12). This is to be expected, since the reflection step involves synthesizing information from retrieved memories, which is important for multi-hop questions. When a model lacks a reflection step and is tasked with answering a multi-hop question, then, the correct answer is not fed to the model already pieced together, meaning it would have to do so during

	sum_sq	df	F	PR(>F)
C(reflection)	216668286.657645	1.000000	1.838047	0.175188
C(question_type)	9477839064.107950	2.000000	40.201340	0.000000
C(reflection):C(question_type)	1740505254.709300	2.000000	7.382552	0.000623
Residual	4438168445741.525391	37650.000000	NaN	NaN

Table 26. ART ANOVA for Reflection × Question Type

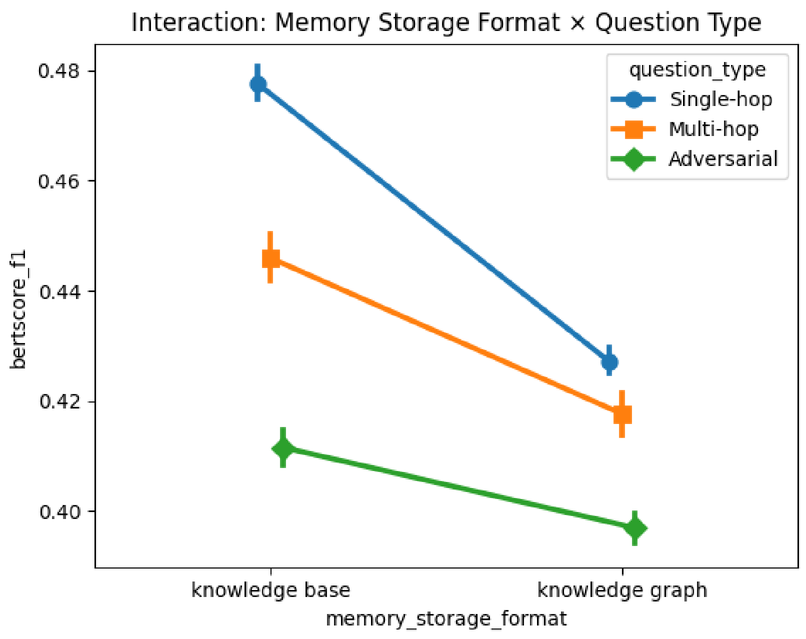

Figure 9. Interactions between Memory Storage Format and Question Type

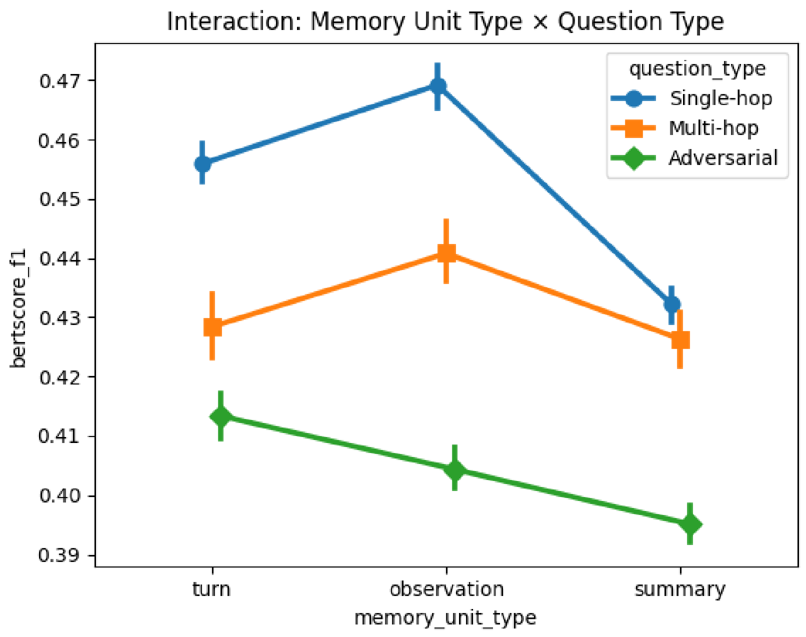

Figure 10. Interactions between Memory Unit Type and Question Type

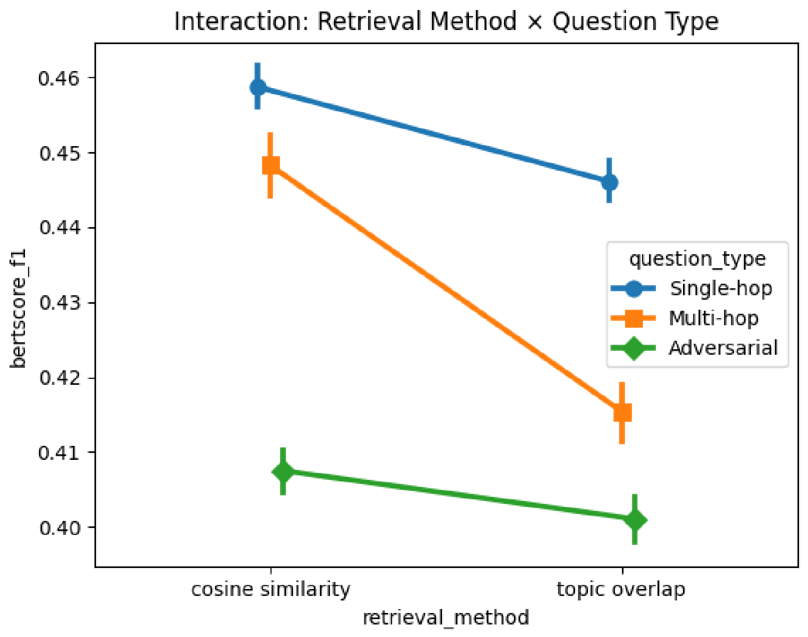

Figure 11. Interactions between Retrieval Method and Question Type

the more complex chatbot generation step, which would lead to more errors and a lower F1 score.

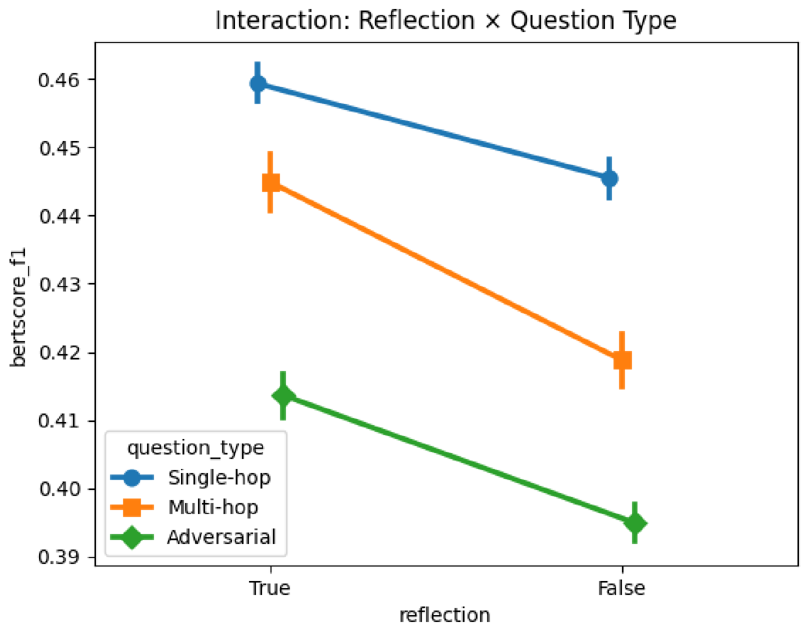

Figure 12. Interactions between Reflection and Question Type

8 Conclusion

The goal of this thesis, as stated in the introduction, was to test every combination of the various techniques that have been used for each part of the RAG conversational chatbot pipeline, and discover if general trends held for untested combinations of features, and if any significant interactions existed between them, as well as between them and the various types of questions they were used to answer. Testing proved that such trends and interactions do in fact exist, and explanations for each were put forward, when possible. In the following subsections, the major contributions of this study are summarized and conclusions are drawn for each. This is followed by a section suggesting topics for future research.

8.1 Contributions

Reviewing the data presented above, we find and identify several significant trends and interactions, both expected and otherwise.

8.1.1 Failure of Knowledge Graphs

The most significant findings, and the ones not tested in the previous literature, involve knowledge graphs. Overall, knowledge graphs proved to be a much worse format for storing memories than knowledge bases, indeed performing even worse than a baseline where the chatbots simply made educated guesses. This reinforces the findings of Sanmartín (2024), which also show knowledge graphs to be inferior to knowledge bases, albeit to a much more extreme degree. This poor performance is attributed to 1) the limitations of the form of relationship extraction utilized, which can only extract information from specific syntactic positions — a difficult task given the complex structure of conversational language — and 2) the reliance on exact text matches for querying, which causes the chatbot to overlook paraphrases, synonyms, and other semantically similar or relevant phrases.

It is thus recommended that alternative methods of extracting sources, targets, and relationships be tested before knowledge graphs be dismissed as unfeasible for storing conversational memories. Langchain's Conversation Knowledge Graph (Langchain (2024)) would be of especial interest for future research, since it utilizes neural models to extract relationships and therefore should avoid at least one the aforementioned drawbacks.

8.1.2 Ineffectiveness of Exact Matching Techniques

As mentioned above, part of the reason knowledge graphs failed to perform well is due to their reliance of exact text matching. The same issue is found with pipelines featuring topic overlap, which produce significantly worse results than cosine similarity and likewise rely on exactly matching topics. Based on these findings, we argue that text-matching techniques are not worthwhile for features of the conversational RAG pipeline which are involved in memory retrieval, given how prevalent paraphrasing and synonyms are in conversations.

8.1.3 Inconsistency Regarding Topic Overlap Performance

The present results regarding the performance of topic overlap conflict with those of Li et al. (2024), which found topic overlap to perform better than cosine similarity. While two differences exist between the implementation of topic overlap in that study and the present one, specifically differences in the memory units being retrieved and the inclusion of noun phrases at topics, they are not considered sufficient to explain these conflicting findings.

8.1.4 Other Features Perform Predictably

The memory unit type findings are uncontroversial, matching those of previous studies when the problematic knowledge graph data are removed: observations produce the best F1 scores, followed closely by turns, and then those two followed distantly by summaries. Likewise, reflection outperforms no reflection, apparently acting as a filter that ignores irrelevant memories and synthesizes the relevant ones into a cohesive answer.

8.1.5 Interactions can be Ignored

Turning to interactions, significant interactions between different parts of the pipelines abounded. However, while significant, any importance these interactions might have in informing pipeline design are for the most part nullified by stark contrasts in performance between feature variants. While turns and reflection are significantly better when used with knowledge bases, these advantages are moot considering that knowledge graphs perform so poorly that knowledge bases are the only feasible choice. Similarly, storing summaries reduces the effectiveness of cosine similarity and models which do so are more sensitive to a lack of reflection, but turns and observations already outperform summaries by such a wide margin that any further motivation to avoid summaries is superfluous. Observations are more resilient to a lack of reflection, but models lacking reflection never outperform those with it, so there's no reason reflection would ever be omitted. And finally, reflection works better when paired with topic overlap, but topic overlap is already beat in every way by cosine similarity, even with reflection. In other words, when considering what features to include in a conversational RAG pipeline, no interaction is strong enough that it would lead one to pick a specific feature over the overall best performing one.

This conclusion, however, assumes that the features used are designed the same way as in this study; knowledge graphs with neural-network powered relationship extraction, or an improved form of topic overlap utilizing word embeddings, might indeed produce interactions what have actually meaningful impacts on performance worth consideration.

8.1.6 Interactions with Questions

When considering how different features perform with different question types, the situation is the same as above, at least for single- and multi-hop questions: the overall best performing feature variant is also the best performing feature variant for each type of question, and thus any and all interactions between features and these question types can be safely ignored. As for adversarial questions, however, conclusions are more difficult to draw, since it is uncertain whether low F1 scores represent resistance to being tricked, or simply that the model is bad at retrieving memories. Assuming the second case to be true, then the conclusions are the same as with the other two questions.

8.2 Future Research

Despite the large scope of this study, or indeed because of it, and certainly because of its many limitations, there is much research to be done that can expand upon the current results.

8.2.1 Extra Feature Variants and Features

First, there are many variants for each feature of the pipeline that are yet to be tested. Already mentioned are LLM- and embedding-powered knowledge graphs, such as Langchain's Conversation Knowledge Graph. In addition, event summaries can be considered as a memory unit type separate from summaries; alternative retrieval methods, such as BERTScore and time decay; and the form of reflection used by generative agents in Park et al. (2023), where every few turns the chatbot reflects upon the conversation history, asks questions, answers them, and adds these answers to its long-term memory. Additional pipeline features, such as persona modules (which were discussed, but left untested in this study) might also be tested.

8.2.2 Combinations of Feature Variants

As well as new variants, future studies can explore models which employ different variants of the same feature at once, like in Park et al. (2023) and Zhong et al. (2023), where scores from different retrieval methods are weighted and combined into a single score. It is also possible for long-term memories to consist of multiple types of memory unit, such as observations, summaries, and generative agent reflections. Combinations like these offer fertile ground for exploration.

8.2.3 Additional Test Methodology

In addition to what is tested, there is room to improve on the tests themselves. First, instead of scoring adversarial questions based on how similar answers are to the adversarial answer, which can produce similar F1 scores both when the chatbot is performing well (the chatbot states it is being tricked) and poorly (the chatbot states it cannot remember anything), it might be better to simply count the number of answers in which chatbot mentions it has been tricked. Moving on, one interesting metric which was omitted due to time constrains was a form of LLM-powered evaluation, which would have labelled generated answers as "correct", "incorrect", or "non-answer" — the last reserved for instances where the chatbot admits it cannot remember the answer to the question (i.e. a failure in memory retrieval). This would provide a great point of comparison against the BERTScore F1 data, cannot necessarily distinguish between an answer that is factually incorrect, and one that is gives no actual answer. Another opportunity for a follow-up study can be found in the unused LoCoMo question types, namely those questions dealing with temporal reasoning and open-domain knowledge. Finally, since this study limited itself to exploring interactions between two features at a time, or a feature and the different question types, there is an opportunity to explore higher levels of interactions, such as interactions between three features, or two or more features against the different question types.

8.2.4 Alternate Data Processing

The current study could be rerun with changes to how the textual data is processed. As it stands, single unaltered turns are tested, meaning that dividing turns by sentences, or storing turns in groups of two or more, are left for future work to explore. There is also the possibility of adding pronoun resolution and other steps to the session history before memories are made and saved, which might improve turn-based model scores.

References

art-from-the machine (2023). Mantella. https://github.com/art-from-the-machine/Mantella. [Online; accessed: 2024-07-12].

Character.ai (2024). Character.ai. https://character.ai/. [Online; accessed 2024-07-12].

Fatehkia, M., Lucas, J. K., and Chawla, S. (2024). T-RAG: Lessons from the LLM Trenches. https://arxiv.org/abs/2402.07483.

Guan, X., Liu, Y., Lin, H., Lu, Y., He, B., Han, X., and Sun, L. (2023). Mitigating large language model hallucinations via autonomous knowledge graph-based retrofitting. https://arxiv.org/abs/2311.13314.

Hatalis, K., Christou, D., Myers, J., Jones, S., Lambert, K., Amos-Binks, A., Dannenhauer, Z., and Dannenhauer, D. (2024). Memory matters: The need to improve long-term memory in llm-agents. *Proceedings of the AAAI Symposium Series*, 2:277–280.

Langchain (2024). Conversation knowledge graph. https://python.langchain.com/v0.1/docs/modules/memory/types/kg/. [Online; accessed: 2025-05-12].

Li, H., Yang, C., Zhang, A., Deng, Y., Wang, X., and Chua, T.-S. (2024). Hello again! llm-powered personalized agent for long-term dialogue. https://arxiv.org/abs/2406.05925.

Maharana, A., Lee, D.-H., Tulyakov, S., Bansal, M., Barbieri, F., and Fang, Y. (2024). Evaluating very long-term conversational memory of llm agents. https://arxiv.org/abs/2402.17753.

Pachipulusu, B. T. (2023). 16 Leading Companies Using Chatbots for Customer Service. https://sitegpt.ai/blog/companies-using-chatbots-for-customer-service. [Online; accessed: 2024-07-10].

Park, J. S., O'Brien, J. C., Cai, C. J., Morris, M. R., Liang, P., and Bernstein, M. S. (2023). Generative agents: Interactive simulacra of human behavior. https://arxiv.org/abs/2304.03442.

Sanmartín, D. (2024). Kg-rag: Bridging the gap between knowledge and creativity. https://arxiv.org/abs/2405.12035.

Singhal, A. (2012). Introducing the knowledge graph: things, not strings. https://blog.google/products/search/introducing-knowledge-graph-things-not. [Online; accessed: 2024-07-26].

Steam (2023). Vaudeville. https://store.steampowered.com/app/2240920/Vaudeville/. [Online; accessed: 2024-07-12].

vedal987 (2024). vedal987. https://www.twitch.tv/vedal987/. [Online; accessed: 2024-07-12].

Wang, Z. M., Peng, Z., Que, H., Liu, J., Zhou, W., Wu, Y., Guo, H., Gan, R., Ni, Z., Yang, J., Zhang, M., Zhang, Z., Ouyang, W., Xu, K., Huang, S. W., Fu, J., and Peng, J. (2024). Rolellm: Benchmarking, eliciting, and enhancing role-playing abilities of large language models.

Yang, L., Chen, H., Li, Z., Ding, X., and Wu, X. (2024). Give us the facts: Enhancing large language models with knowledge graphs for fact-aware language modeling. https://arxiv.org/abs/2306.11489.

Zhong, W., Guo, L., Gao, Q., Ye, H., and Wang, Y. (2023). Memorybank: Enhancing large language models with long-term memory. https://arxiv.org/abs/2305.10250.