HLGraphRAG: A Graph Retrieval Augmentation System for Human-like Synthesis of Information

Mike Zeng <mzeng5@uiowa.edu>

University of Iowa

Abstract

Modern LLMs such as OpenAI's GPT-4 and Meta's LLaMA 3.2 are limited by their context length of 128k tokens. This hinders their ability to function as character AI systems. These systems require the ability to recall information across large contexts, such as storing every event in a human life. Attempts to extend context length aim to solve this issue. However, such attempts have diminishing returns and do not translate to real-world recall accuracy. Retrieval-augmentation methods, such as LangChain, address context limitations. However, they fall short in accurately simulating human memory in question-answering tasks across multiple domains. Recently, GraphRAG was introduced, offering significant memory performance improvements for global retrieval tasks. This raises the question of how well it performs specifically in character AI systems? In this paper, I explore the performance of GraphRAG compared to other systems in the scope of character AI systems. I also introduce event-based context routing. This ensures that causality and context are remembered. It approximates human memory to a greater extent, achieving superior retrieval performance for human-like information synthesis. This paper demonstrates how GraphRAG and HLGraphRAG significantly outperform traditional RAG systems in memory performance for user acceptance in character AI systems.

Table of Contents

1. Introduction

Character AIs are systems designed to simulate human behavior, often incorporating memory and personality. There are a few primary examples of this form of application and service such as c.ai, sakura.fm, and research being published in this domain such as AI NPC systems for games (Zhu *et al*., 2023); all of this to create an immersive world mimicking the real world. Unfortunately, prior work has shown a main problem with AI systems is memory. Native APIs such as Gemini have a context length limit of 128K tokens (Anil *et al*., 2023), which poses challenges for maintaining coherent conversations and handling large knowledge bases, often leading to broken immersion and inaccurate responses (context limitations, hallucinations) that come with using context length (Massarelli *et al*., 2019). For comparison, human autobiographical memory spans hundreds of thousands of words, far exceeding the typical context length of modern AI models. Additionally, APIs charge per input token, making long-context AI interactions costly and inefficient without efficient memory systems. These challenges highlight the need for a more effective and scalable memory system capable of long-term recall.

Prior research has explored different approaches to improving AI memory, such as context-length extension (Karkhanis *et al*., 2024) and Retrieval-Augmented Generation (RAG) systems (Lewis *et al*., 2020). Context-length extension adjusts the model to try to remember longer contexts, but has diminishing returns in real-world performance (Hsieh *et al*., 2024). Current RAG tools like LangChain attempt to extend AI context length through external retrieval mechanisms (Pandya and Holia, 2023), yet they often fall short in meeting user expectations in long-form interactions. One reason for this shortfall is that it fails to connect disconnected pieces of information. For example, consider the following query: “What computer was used by the daughter of the man who drank tea?” A successful RAG would generally follow these steps to answer this question:

- Identify the man who drunk tea.
- Identify the daughter of the man.
- Identify the computer used by the daughter.

LangChain attempts to directly search for phrases such as 'What computer was used by the daughter of the man who drank tea'. However, the results may include disconnected or incomplete information, such as 'all computers are used by Jane' or 'Sob drank tea', leading to insufficient data for generating a correct answer. This illustrates the challenge that LangChain faces when attempting to integrate information spread across different text fragments, resulting in incomplete or incorrect responses.

To address these limitations, recent research (Edge *et al*., 2024), has explored graph-based retrieval mechanisms, which offer a structured way to connect and recall information. GraphRAG is a system that structures retrieved knowledge as a graph rather than relying on traditional embedding-based lookups It has demonstrated improvements in knowledge synthesis, showing that graph-based retrieval mechanisms can enhance AI reasoning. However, this study has not specifically focused on long-term conversational memory, particularly in the context of Character AIs. One of my goals is to assess whether GraphRAG, when adapted with human-like memory structures, can significantly enhance conversational recall in Character AIs compared to LangChain/baseline RAG systems.

While GraphRAG has shown potential in improving AI reasoning, its current implementation falls short of mimicking the way humans store and recall information. Unlike GraphRAG, which focus mainly on entities and their relationships, human memory also incorporates events, spatial context, and causal connections, which are essential for creating a more accurate memory system (Rubin and Umanath, 2015). Therefore, I hypothesize that modifying GraphRAG to encode memory more like human cognition—by incorporating event-based retrieval and location-aware memory storage—can improve long-term recall in character AIs further.

To test this, I adapt GraphRAG to include these additional memory structures and evaluate its performance against existing solutions. Specifically, I compare the following systems:

- HLGraphRAG, the modified GraphRAG.
- Native AI models (Gemini)
- Baseline RAG models (control variable, LangChain)
- Other graph systems (GraphRAG)

I introduce key modifications to the ideas from GraphRAG, integrating:

- Event-based memory encoding: Capturing not only the content of an event but also its causality and significance.
- Location-aware memory retrieval: Storing and retrieving information based on spatial context.

To evaluate performance, I use acceptance tests, an approach from software engineering that determines whether a system meets user expectations. Acceptance tests are designed for software services to ensure that an user is satisfied, reducing the amount of complaints and hence increasing immersiveness of software systems (Ricca and Torchiano, 2019). This is a great metric for my tests because I aim to understand how well these RAG applications perform as a service for a character AI system. To measure this, I test how well the output matches the correct answer. Each of these have 3 evaluations for accuracy: unacceptable, barely acceptable, and correct. More details can be found in the **Testing** section.

My paper aims to have 3 main contributions:

- Demonstrate that graph-based memory systems improve performance over existing RAG-based systems and native AI APIs for human conversation recall.
- Analyze how event encoding and retrieval influence AI performance.
- Advance character-based AI systems toward real-world usability by evaluating their performance through acceptance tests.

2. Background

My modified graphRAG system will rely on three concepts: RAG, LangChain, GraphRAG.

Retrieval Augmented Generation. Retrieval Augmented Generation is a method introduced by Facebook in which data retrieval is separated from the model's context, leveraging the use of an external retrieval structure rather than attention (Lewis *et al*., 2020). It is highly beneficial compared to context lengths due to the fact that it is capable of storing documents of much larger sizes. In addition, it solves the problem of vanishing gradients, also known as the problem that occurs when the input size gets too large and it forgets something earlier, by filtering out and disabling data pieces that it deems as non-needed. In fact, RAG has been shown (Zhang *et al*., 2025) to be more accurate than long context in turn-based conversations, visualizing the importance of information refreshes- the idea of placing important information near the top of the model input.

RAG Illustration Diagram

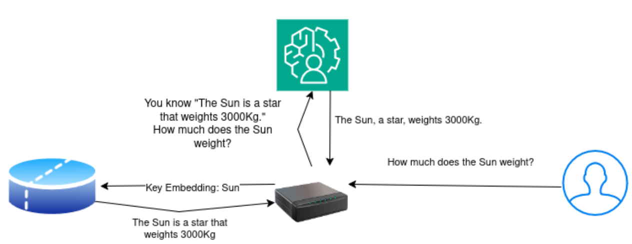

Figure 1.1: A Basic RAG System for QA where a LLM answers how much the sun weights without hallucinating.

LangChain. LangChain is an extremely popular RAG system. There are two parts of LangChain's RAG: encoding and retrieval (LangChain, 2024). During encoding, I aim to store the data in a manner that is easy to retrieve. In LangChain, the document is separated into text fragments, which are then used to generate text embeddings. Embeddings are representations of text in a vector space first introduced by Google (Mikolov *et al*., 2013), allowing for word meaning similarity comparisons between strings. For example, the similar phrases "the cow chewed the lawn" and "the cow ate grass" have very different characters, but would be very close when represented as a text embedding. These come into play during retrieval, so when "cow" or "eating" are searched, it matches and returns both the phrases.

GraphRAG. GraphRAG is a recent RAG system published by Microsoft. Like LangChain, there are two phases: encoding (referenced as indexing) and retrieval (referenced as query). During encoding, I first break down a passage to text chunks. Each chunk is then searched to find entities and their descriptions such as name, type, text. Relationships are then identified as source, target, and description of relationship. To build the graph, GraphRAG uses the Leiden technique to identify communities and build a hierarchical structure representation of knowledge. I move to the retrieval phase. GraphRAG extracts chunks of relevant community clusters, ranks them, and returns back the output.

3. Design

To test my idea of a graph retrieval and how each part affects performance, I implemented a directed, weighted graph (AssociationGraph) using Python. The nodes are stored in an adjacency list as an embedding. There are two parts: encoding and retrieval. To do this, I implemented subject-relation query, event-context-routing, directional weights, and ranked retrieval.

HLGraphRAG

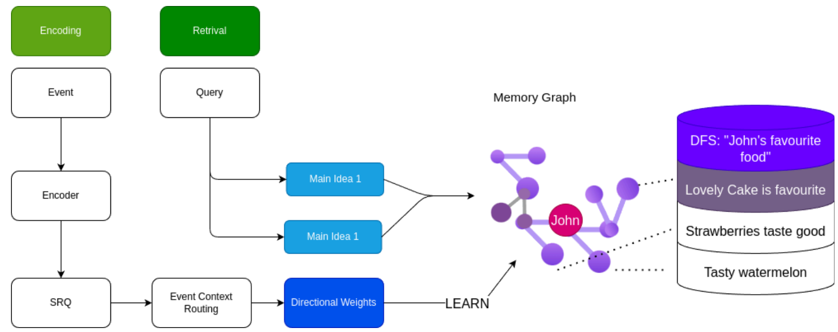

Figure 2.1: Encoding and Retrieval. This diagram highlights the process undergone when information is encoded and retrieved.

Encoding. For the Encoder, I take an input of data to store it into the graph. With inspiration from GraphRAG, I have the model prompt for subjects to implement subject-relation query. However, I also have the model look for events.

Subject-Relation Query (SRQ) and Context Routing (CR). After this, I start generating relationships using zero-shot prompting + reasoning steps (Kojima *et al*., 2022) to GeminiAPI on Gemini Pro 1.5. Using the model, I have it examine the prompt for any relationships between the subjects. From its finding, it generates a list of (meaning, object 1, object 2). Then, I examine the events and form relationships in the form of (event, object 1, object 2, why), with a general time frame attached to each.

SRQ & CR Implementation

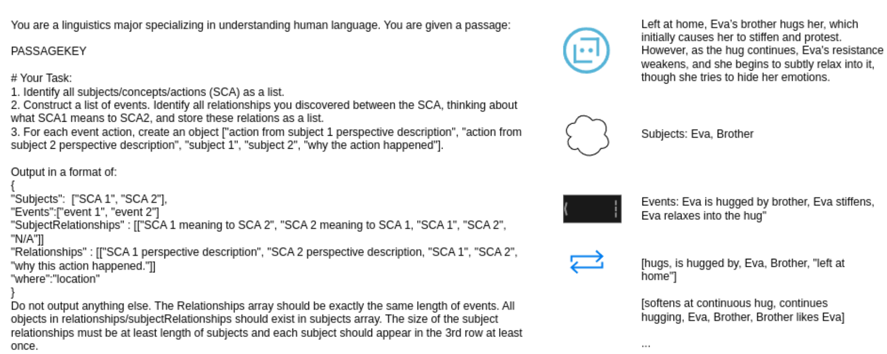

Figure 2.2: A reference of the prompt and an example input-to-output.

Directional Weights. Finally, I implement directional weights, in which case there are two versions of the event/meaning for object 1 -> object 2 and object 2 -> object 1. This allows my model to make better connections between objects. For example, “Selena was pushed by John” would result in a relationship between “Selena” and “John” nodes both being “pushed by”. However, John was not pushed by Selena, so this will cause errors during retrieval. So, I label the edges Selena->John as “was pushed by” and John->Selena as “pushes”. This allows the graph to capture the true relationship between objects.

In my AssociationGraph, I have text-embedding nodes and edges with a list of weights. The nodes are text-embedded to lower precision requirements in model searches, and the edges are a list of weights so that objects can have multiple relationships with each other. These subjects are then used to create a text embedding using all-MiniLM-L6-v2 (Reimers and Gurevych, 2019). before being appended to the graph as a node. Each edge is appended to the list.

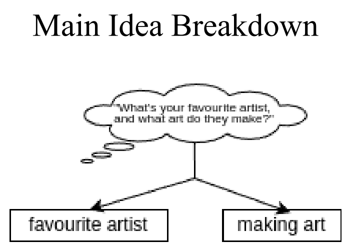

Figure 2.3: The process of how a prompt is turned into search-queries.

Retrieval system. Instead of a community-cluster system ranking, I implement a guided depth-first transversal. This was decided to keep the retrieval system simple, easy to maintain, and optimize it for 100% acceptance rate. To retrieve, I start with the user request. The prompt is broken down into main ideas using Gemini, which are then queried against the graph. The graph finds the node with the highest embedding similarity, and then starts from there. It then does a depth-first-search, and collects nodes up to a set max amount. I used a max amount of 5 nodes.

Ranked Retrieval. To make the DFS more precise, I implemented ranked retrieval. This is a sophisticated method in which, instead of my standard DFS during node collection, my graph instead takes another parameter, and ranks the edges to other nodes by search_term2. It then prioritizes the nodes and edge labels with the highest similarity to walk. Basically, I iterate through a DFS as normal, but before the node transverses, I sort the children based on similarity and then pick the highest unvisited node.

4. Testing

During development, I needed a way to measure improvement of my model. I considered manual, human-review of output, but this method is too time expensive and error prone (that can result in bias). Therefore, I constructed a standard testing service that can be used with OpenAI, CharacterAI, etc. The idea: **How similar is a conversation to a real one given context?** To understand, I will define a few terms, create a structure for the tests, and implement a testing driver that runs autonomously.

Definitions. To evaluate the model's performance, I categorize errors into three groups:

- Choke: A severe deviation from expected behavior, where the response contradicts fundamental knowledge. For example, if the model recalls that John's family owns a bakery but later insists they are doctors, this constitutes a choke.
- Inaccuracy: A minor deviation that does not significantly alter the intended meaning. For instance, if John was expected to say "my family's bakery" but instead says "a bakery," the core meaning remains intact, though the response is not fully precise.
- Correct: An accurate and contextually appropriate response.

Due to the substantial disparity in severity between chokes and Inaccuracies, my primary evaluation metric prioritizes reducing chokes while acknowledging that minimizing Inaccuracies remains desirable. I define this approach as an acceptance test, where chokes indicate failure to meet an acceptable standard, whereas Inaccuracies, while suboptimal, do not necessarily disqualify a response.

Comparisons. I compared the following groups to LangChain (control) to understand how each factor impacts acceptance performance:

- Native Gemini API
- GraphRAG
- GraphRAG + SRQ + CR
- GraphRAG + SRQ + CR + DW
- GraphRAG + SRQ + CR + DW + RR

Choke, Inaccurate, or Correct Analysis

You are a psychology major specialized in analyzing memory impairments. Your job is assessing memory retention by categorizing responses as choke (significant memory failure), Inaccurate (minor deviations that do not imply severe memory impairment), or Correct (accurate recall). For example, given the context that "John knows that [free-spirited traveler named Serena] had differing life goals, .John knows that [differing life goals] caused heartbreaking separation." The stranger asks "Alright! Well then, let's get this show started. Have you ever been in love? Who?"? The correct answer is "I loved someone named Selena". A choke answer would be one such as "No, I have never been in love". A inaccurate answer would be "I have been in love". A correct answer would be "I liked a person named Selena" due to how it doesn't deviate anything. Now, given [content], when an [entity] says [dialogue 1] and [entity 2] responds with [dialogue 2] when the correct answer is [correct], is this a choke, inaccurate, or correct?

Figure 4.1: The prompt for determining Choke, Inaccurate, or Correct.

Test Structure. With my definitions out of the way, I can now talk about my tests. I selected the CMU Document Grounded Conversations Dataset due to its conversation-like nature and particularly for its focus for testing long-term memory, information retrieval, and reasoning over multiple turns with its documents in a conversation-like setting (Zhou, Prabhumoye, 2018) and selected the top 1219 JSON data tests sorted by name. During conversation simulation, I used few-shot prompting (Brown *et al*., 2020) on Gemini API to determine if an output is a choke, inaccuracy, or correct. Prior to the next turn in the conversation, I swapped the output of the model with the real answer to prevent the model from accumulating unnecessary errors due to one error it made earlier.

CMU DoG Dataset Examples

Example Conversation
user1: "hello" user2: "hello there, I have not seen this movie so im going to take a minute to look it over :)" user1: "Alright that is fine. What is the movie?" user2: "The movie is The Social Network" user1: "I have not seen that one either.", user1: "It appears to be like a biographical movie about Mark Zuckerberg and when he creates Facebook" user2: "Ohh yes I remember hearing about that movie, but it was a long time ago it feels like." user1: "Rotten Tomatoes has rated it 96% so that's decent, and it is a movie from 2010" user2: "may be worth watching!" user1: "Yes I think it could be, although I personally do not like facebook as a company." user2: "the movie portrays the founding of social networking website Facebook and the resulting lawsuits. It even has Justin Timberlake in it, I don't think I've ever seen him act."
Document-Grounded
In October 2003, 19-year-old Harvard University student Mark Zuckerberg is dumped by his girlfriend Erica Albright. Returning to his dorm, Zuckerberg writes an insulting entry about Albright on his LiveJournal blog and then creates a campus website called Facemash by hacking into college databases to steal photos of female students, then allowing site visitors to rate their attractiveness. After traffic to

the site crashes parts of Harvard's computer network, Zuckerberg is given six months of academic probation. However, Facemash's popularity attracts the attention of Harvard upperclassmen and twins Cameron and Tyler Winklevoss and their business partner Divya Narendra. The trio invites Zuckerberg to work on Harvard Connection, a social network featuring the exclusive nature of Harvard students and aimed at dating. ",
Saverin and Zuckerberg meet fellow student Christy Lee, who asks them to 'Facebook me', a phrase which impresses both of them. As Thefacebook grows in popularity, Zuckerberg extends the network to Yale University, Columbia University and Stanford University. Lee arranges for Saverin and Zuckerberg to meet Napster co-founder Sean Parker, who presents a 'billion dollar' vision for the company that impresses Zuckerberg. He also suggests dropping 'The' from Thefacebook, just calling it Facebook. At Parker's suggestion, the company moves to Palo Alto, with Saverin remaining in New York to work on business development. After Parker promises to expand Facebook to two continents, Zuckerberg invites him to live at the house he is using as company headquarters. "...

Figure 4.2: An example of a test case involving a conversation and a wiki taken from the dataset.

Autonomous Testing Driver. I will now talk about the testing driver, or the autonomous implementation of these tests. The testing driver simulates the conversation on APIs by doing the following:

1. Knowledge Integration: Relevant Wikipedia documents are encoded into the Retrieval-Augmented Generation (RAG) system.
2. Model Initialization: The conversation begins with a structured chatbot prompt (e.g., “You are a chatbot.”). The model confirms its role.
3. Information Retrieval: Before each user message, the RAG system retrieves relevant knowledge.
4. Response Generation: The system templates the response with contextual knowledge (e.g., “You are Bob, and you know [retrieved knowledge]”). The model acknowledges this and generates a response.
5. Evaluation: A secondary model (Gemini API) categorizes the response as Choke, Inaccurate, or Correct using few-shot learning. The results are logged.
6. Error Correction: The correct response replaces any inaccuracies to prevent error propagation.
7. Iteration: Steps 3–6 repeat until the conversation concludes.
8. Performance Analysis: The total counts of chokes and inaccuracies are recorded.

For Gemini, it simply appends the entire history to every message as if there was a RAG that printed everything. For LangChain, it follows their documentation for populating the RAG/retrieving values. In all my models, I include a message that says “Do not use any information not included in the context” to prevent the model from using its pre-trained data to answer the question.

Now, I will analyze my testing method by discussing various other ways I could have run the tests and the limitations of my testing approach.

Analysis of Tests. Other methods were considered to test the model. In my autonomous testing, I considered text embeddings to measure chokes, inaccuracies, or correct, but soon found out that it was unreliable due to the fact that output could be very short. I also considered simply accumulating loss from semantic differences in output, but determined this was not optimal because in my situation, my goal is acceptance, so one model that has 150 inaccuracies is better than a model that has 75 chokes and 0 inaccuracies since 75 chokes would receive more complaints. As mentioned earlier, human evaluations were too expensive due to the large nature of data, but I opted to include a few pieces of outputs, provided in **Results**. My test is more rigorous than a human because a model should fully satisfy the end user's memory recall if it contains exactly the necessary information—no more, no less. Since the distinction between inaccuracies and correct information can be ambiguous, a human judge may be unable to make a definitive assessment.

Limitations. During testing, I noticed a lot of variation in Gemini performance. Occasionally, the model produced suboptimal responses despite having access to all necessary information. This expresses concerns in how different RAG systems need to feed it data. To minimize the impact this had on my result, I tweaked a few model parameters, but found they did not change results significantly. As a result, I ran tests for 15 times to minimize the impact of this while also noting that since I measure real life performance, ease of integration with APIs is also important. The last thing to note is that while I attempted to prevent the model from using pre-trained data, it likely could have affected the results. To see the effects of this, I include the base comparison of LangChain vs GraphRAG from GraphRAG's findings.

Finally, my Gemini judge may not reflect the opinion of end users. By approximating an user for judging, it may still be the case that end user acceptance results can vary. However, I manually reviewed various cases of sample output and included my findings in the results to evaluate if inaccuracies existed in my judge.

5. Results

I ran the test 15 times for a total time of 21.1 hours on all 1219 test cases. The numbers were averaged to form the numbers in my table and rounded to the nearest whole number.

Numerical Data on Model Performance

	Chokes	Inaccuracies	Correct
Native	219	459	541
LangChain Baseline	180	391	648
GraphRAG	130	301	788
HLGraphRAG [SRQ, ECR]	67	132	1020
HLGraphRAG + DW	65	121	1033
HLGraphRAG + DW + RR	33	80	1106

Figure 5.1: A table representing choke count, inaccuracies, and corrects.

I made notes of this table (the decreasing amounts of chokes) and made a graph. I ensure to merge inaccuracies with correct to a single column to denote user acceptance rate (the value I care about).

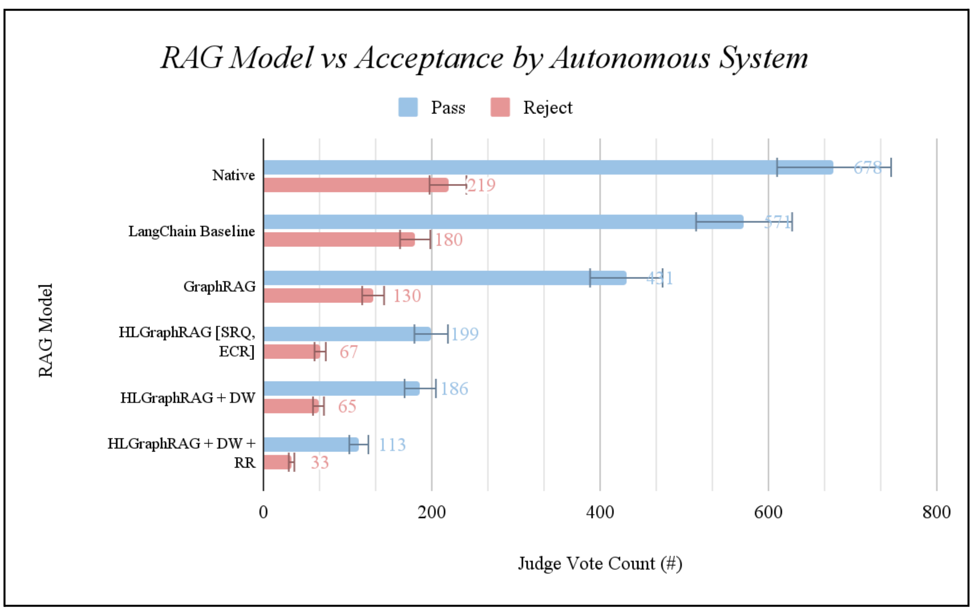

Figure 5.2: A graph formed by combining inaccuracies with correct. The error bars are set at 10%.

The above plot assures us that my data differences are statistically significant and that there trends in my data. I will now take a look at the shape of the test data (Histograms) for each run for analysis.

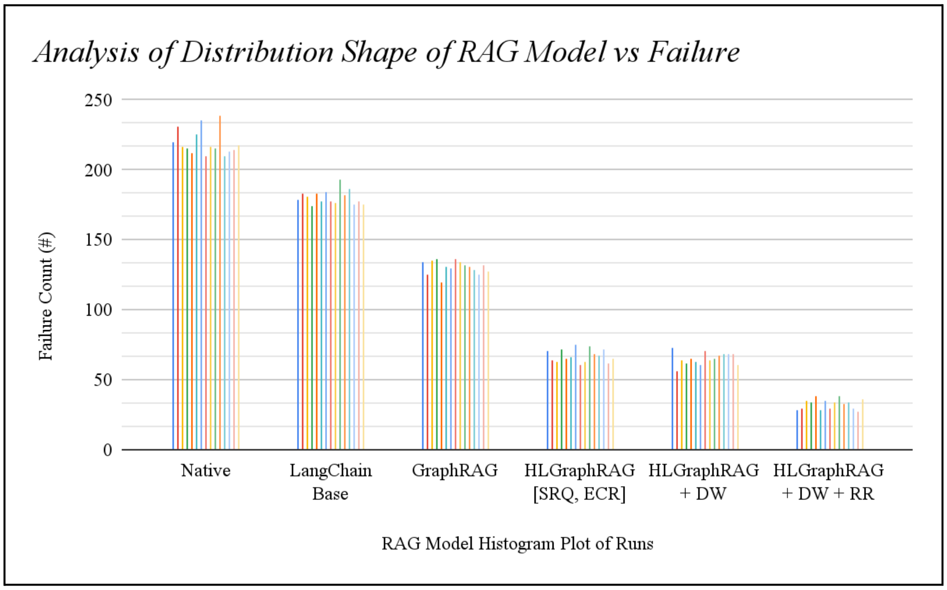

Figure 5.3: A Histogram collection visualizing each run's failure count.

In this diagram, I observe that there are not any runs that are particularly below where the performance tends to lie. I additionally see that the shape of distribution is generally the same between models, a continuous uniform distribution.

I examine the failure count (chokes) across all 15 tests. A sample of 5 is shown below; the full table may be found in the **Appendix**.

Failure Count by Model

Model Type	Run 1	Run 2	Run 3	Run 4	Run 5
Native	220	231	217	215	212
LangChain Base	179	183	181	174	183
GraphRAG	134	125	135	136	120
HLGraphRAG [SRQ, ECR]	71	64	63	72	65
HLGraphRAG + DW	58	56	64	62	65
HLGraphRAG + DW + RR	28	30	35	34	38

Figure 5.4: A table visualizing the raw number of chokes across 5 runs.

Statistical Analysis. I selected the Mann-Whitney U test for statistical analysis due to my data graph having similar shapes but small sample size (15) and same-shape data. I conduct the following comparisons on my table to check for significance.

- LangChain vs GraphRAG
- GraphRAG vs HL GraphRAG [SRQ, ECR]
- HLGraphRAG [SRQ, ECR] vs HLGraphRAG [SRQ, ECR, DW]
- HLGraphRAG [SRQ, ECR, DW] vs HLGraphRAG [SRQ, ECR, DW, RR]

I use a significance value of $p = 0.05$ for testing, $n = 15$, and I let the *set1* be the chokes of model 1, and *set2* be the chokes of model 2. I use **SciPy** for the Mann-Whitney U test. The values are displayed in the following table. Refer to the **Appendix** to view code for computation.

Statistical Significance for Mann-Whitney U Test of RAG 1 vs RAG 2

Comparison	***p* value**
LangChain vs GraphRAG	0.00
GraphRAG vs HLGraphRAG [SRQ, ECR]	0.00
HLGraphRAG [SRQ, ECR] vs HLGraphRAG + DW	0.15
HLGraphRAG + DW vs HLGraphRAG + DW + RR	0.00

Figure 5.5: Comparisons between models. 0.00 *p* values were rounded down.

Observations. All forms of GraphRAG perform a statistically significant smaller amount of fails compared to LangChain or native (as shown by error bars). Furthermore, the addition of Subject-Relation-Query (SRQ) and Event-Context-Routing (ECR) improves a statistical improvement over standard GraphRAG. Directional weighting (DW) does not prove to improve the model significantly, but ranked-retrieval (RR) does. I did not observe any outliers.

Alongside my statistics, I will examine a few selected pairs of output from the listed systems. I selected one about an event and one Wiki-based. HLGraphRAG is shortened to HLGR in my table. In addition, instead of using the exact output, I use *Native* as the baseline system as the Wiki document in these examples (not always the case) can fit comfortably into the context length of Gemini, providing us the best results.

Randomized output [1] from LangChain, GraphRAG, HLGraphRAG (HLGR)

Question	**"How does the storyline go?"**
LangChain (inaccurate)	The main cast is hired and recruited to perform corporate spying.
GraphRAG (inaccurate)	The main characters are hired to do corporate spying by entering people's dreams. But in this mission, instead of stealing an idea, they have to plant one in the person's mind.
HLGR [SRQ, ECR] (correct)	The main characters are hired to spy on a company by entering people's dreams. But the big mission in the movie is not to steal an idea but to plant one in someone's mind through their dreams.
HLGR + DW (correct)	The main characters are hired to spy on companies by going into people's dreams. But in this movie, their mission is different—they have to plant an idea in someone's mind instead of stealing one.
HLGR + DW + RR (correct)	The main characters are hired to steal secrets by going into people's dreams. But in this job, instead of stealing an idea, they have to plant one in someone's mind.
Native	"To leave out spoilers the Main cast is hired or more so recruited to perform corporate espionage through entering dreams. However the focal point of the movie is a job where rather than taking an idea out is leaving an idea with the person whose dreams they are within."

Figure 5.6: Example output regarding Wiki descriptions.

Randomized output [2] from LangChain, GraphRAG, HLGraphRAG (HLGR)

Question	**"What was the plot of the movie? Can you give a brief summary of what the movie said was the dispute?"**
LangChain (choke)	The plot was about how facebook was created. The movie argued about a dispute between Zuckerburg and Saverin.
GraphRAG (inaccurate)	The movie is about Mark Zuckerberg creating Facebook while at Harvard. He starts by making a site called Facemash, which gets him in trouble but catches the attention of some students who ask him to work on their social network idea. Instead, he makes Facebook. His friend Eduardo Saverin helps with funding but later gets pushed out, and Napster co-founder Sean Parker influences Zuckerberg. The movie shows lawsuits from the Winklevoss twins and from Saverin.
HLGR [SRQ, ECR] (inaccurate)	The movie is about Mark Zuckerberg's creation of Facebook and the disputes that arise. It shows how Zuckerberg made a campus website, got into trouble, and then expanded it into Facebook with the help of others like Eduardo Saverin and Sean Parker. The disputes in the movie revolve around claims by the Winklevoss twins, who say Zuckerberg stole their idea, and by Saverin.
HLGR + DW (inaccurate)	A Harvard student, Mark Zuckerberg, gets dumped and creates a website that rates female students' attractiveness. This gets him in

	trouble but also attracts attention from others who want to build a social network. Facebook grows, and Mark meets Sean Parker, who influences him. However, a dispute arises and the movie shows lawsuits where the Winklevoss twins say Zuckerberg stole their idea, and Eduardo Saverin sues Zuckerberg as well.
HLGR + DW + RR (correct)	The movie The Social Network (2010) follows the rise of Facebook and the legal battles surrounding its creation. It portrays Mark Zuckerberg, a Harvard student, who creates a social networking site called Facemash, which gains attention and leads to the development of Thefacebook (later Facebook). The Winklevoss twins and Divya Narendra claim that Zuckerberg stole their idea for Harvard Connection, while Eduardo Saverin, Zuckerberg's co-founder and initial financier, later sues him for unfairly diluting his shares. The film is intercut with legal depositions from these lawsuits, culminating in a settlement with Saverin.
Native	The movie The Social Network portrays the founding of Facebook and the legal battles that followed. It focuses on Mark Zuckerberg's creation of Facebook while at Harvard, and the subsequent lawsuits filed by the Winklevoss twins and Eduardo Saverin. The Winklevoss twins claim Zuckerberg stole their idea for a social networking site, Harvard Connection, which they had proposed to him. Saverin, Zuckerberg's former business partner and co-founder, sues after claiming that Zuckerberg unfairly diluted his shares in the company when Facebook incorporated. These legal disputes are interwoven with the film's narrative, highlighting Zuckerberg's ambition, personal relationships, and the tensions that arise as Facebook grows.

Figure 5.7: Example output regarding conversations.

Observations. HLGraphRAG performs similarly to GraphRAG in terms of conversation retrieval. Both of these graph-like architectures utilize the information significantly better than LangChain such as how the reason for corporate espionage is mentioned. When the document is a Wiki article, there are minimal benefits of using HLGraphRAG.

However, when history is an event, HLGraphRAG provide substantial benefits compared to GraphRAG due to understanding the event better (its focus on ensuring the recording of why the event happened). This is seen as GraphRAG does retrieve why the Winklevoss twins sue Zuckerburg.

Directed weights is marginally the same as HLGraphRAG, but ranked retrieval does provide a benefit. It was able to retrieve why Saverin sued Zuckerberg, likely due to the fact that ranked retrieval prioritizes relevant queries to prevent the collection limit from being unnecessarily hit.

I did not observe any outliers through my reviews were HLGraphRAG performed worse than GraphRAG, or GraphRAG performed worse than LangChain. I also deemed that generally, the Gemini Judge was able to mark the output generally well, such as when it marked LangChain from example 2 as a choke for missing too much details.

6. Discussion

From my data, I made a few conclusions.

Takeaway 1. The implementation of GraphRag does indeed provide significant benefits to LangChain. Failure rate is significantly reduced as shown by my first observation. This suggests that this model fits my acceptance criteria by decreasing the failure rate. Furthermore, by switching to HLGraphRAG, I were able to decrease my failure rate further, suggesting while both GraphRAG systems provide meaningful improvements compared to other solutions in acceptance tests, HLGraphRAG provides a meaningful improvement over normal GraphRAG..

Takeaway 2. The output visualizes the limitations of each RAG. As you can see in the output of Gemini, due to the problem of vanishing gradients, it fails to retrieve enough information to answer the question correctly. Using LangChain, I see that the model is able to retrieve some information from the documents. However, on some cases, it is failing the conversation due to the multi-domain aspect of querying. GraphRAG handles this with ease, moving through what is important. HLGraphRAG performs similarly to GraphRAG from observations on Wiki-like documents; However, on data types that revolve around documents that are based on events, I see HLGraphRAG outperforms GraphRAG on queries that may involve complex reasoning of events from taking time to understand the events.

Takeaway 3. Putting together these takeaways, I make some general notes about GraphRAG and HLGraphRAG.

1. GraphRAG systems outperforms standard RAG such as LangChain for complex wiki articles and event-based articles.
2. Event-based encoding helps significantly for event-centered tasks through reasoning, but not for general knowledge QA.
3. I can reduce error rates significantly with the use of GraphRAG and its extension HLGraphRAG in character AI systems.

Related Work. GraphRAG was advantageous 80% of the time compared to baseline RAG (Microsoft) in previous work. In my work, LangChain failed 25% more than GraphRAG. These numbers are not directly comparable, but I can assume that LangChain loses to GraphRAG in at least 25% of tests, which would get us close to the loss of 20% from previous work despite the differences in test setup and LangChain use. This suggests my testing method relating to human conversations works well with GraphRAG as I were able to produce similar results.

Limitations. There were a few limitations that I noticed during research. The first few ones were documented in the Testing section, but there are a few limitations with my results as well. Primarily, the dataset I selected could not have been diverse enough to fully understand how my model performs. Due to costs limitations, I used a significant but limited amount (1219) pieces of data. Though 1219 is a large amount of data, it is still possible that it may result in a bias in the data so my model performs better. In addition, my dataset revolves around talking about concepts and details contained inside a Wiki in an online chat where all messages were short. It would be useful it I tested this model with other scenarios where the user messages were not just short strings of text and understand how different my models could perform.

Future Work

The testing framework presents multiple opportunities for refinement. Conducting additional human evaluations for acceptance testing would provide more authentic data on user reception. Furthermore, integrating HLGraphRAG and GraphRAG into deployed applications such as Character AI or Sakura FM could offer deeper insights into the practical implications of enhanced RAG memory in real-world systems. Expanding the range of datasets used in evaluation could help identify limitations within my RAG system. Additionally, incorporating a broader set of RAG implementations as independent variables would enable a more comprehensive assessment of whether alternative approaches can achieve effective memory acceptance in character AI applications.

7. Conclusion

I presented the introduction of event-based encoding and retrieval to GraphRAG while evaluating it with other RAG systems in the domain of turn-based conversations. My tests and results suggest substantial improvements from using HLGraphRAG to other RAG systems in human conversation performance as well as the significance of Graph structures in modeling human memory. For applications requiring human recall such as character AI services, event-based graph structures provides leading performance in memory, showcasing a significant improvement in user acceptance compared to other structures. By implementing event-based graph structures for RAG in character AI services, hosts can improve their services for end-users while improving user satisfaction.

8. References

Anil, R., Borgeaud, S., Wu, Y., Alayrac, J.-B., Yu, J., Soricut, R., Schalkwyk, J., Dai, A. M., Hauth, A., et al. (2023). Gemini: A family of highly capable multimodal models. *arXiv preprint* arXiv:2312.11805. https://arxiv.org/abs/2312.11805

Reimers, N., Gurevych, I. (2019). Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP) (pp. 3982–3992). Association for Computational Linguistics. https://doi.org/10.18653/v1/D19-1410

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., ... Amodei, D. (2020). Language models are few-shot learners. In *Advances in Neural Information Processing Systems (NeurIPS 2020)*.

Edge, D., Trinh, H., Cheng, N., Bradley, J., Chao, A., Mody, A., Truitt, S., & Larson, J. (2024). From local to global: A graph RAG approach to query-focused summarization (*arXiv*:2404.16130). *arXiv*.

Hsieh, C., Wang, Y., & Liu, X. (2024). Why does the effective context length of LLMs fall short? In *Proceedings of the International Conference on Learning Representations (ICLR 2025)*.

Kang, X., Wang, Z., Jin, X., Wang, W., Huang, K., & Wang, Q. (2024). Template-driven LLM-paraphrased framework for tabular math word problem generation. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI 2025)*.

Kojima, T., Gu, S. S., Reid, M., Matsuo, Y., & Iwasawa, Y. (2022). Large language models are zero-shot reasoners. In *Proceedings of the 36th Conference on Neural Information Processing Systems (NeurIPS 2022)*.

LangChain. (2024). Build a Retrieval Augmented Generation (RAG) App: Part 1. https://python.langchain.com/docs/tutorials/rag/

Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W., Rocktäschel, T., Riedel, S., & Kiela, D. (2020). Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Advances in Neural Information Processing Systems (NeurIPS 2020)*.

Massarelli, L., Petroni, F., Piktus, A., Ott, M., Rocktäschel, T., Plachouras, V., Silvestri, F., & Riedel, S. (2020). How decoding strategies affect the verifiability of generated text. In *Findings of the Association for Computational Linguistics: EMNLP 2020* (pp. 223–235). Association for Computational Linguistics.

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient estimation of word representations in vector space. In *Proceedings of the International Conference on Learning Representations (ICLR 2013)*.

Pal, A., Karkhanis, D., Roberts, M., Dooley, S., Sundararajan, A., & Naidu, S. (2023). Giraffe: Adventures in expanding context lengths in LLMs. *arXiv preprint* arXiv:2308.10882. https://doi.org/10.48550/arXiv.2308.10882

Pandya, K., & Holia, M. (2023). Automating customer service using LangChain: Building custom open-source GPT chatbot for organizations (*arXiv*:2310.05421). *arXiv*.

Ricca, F., Torchiano, M., Di Penta, M., Ceccato, M., & Tonella, P. (2009). Using acceptance tests as a support for clarifying requirements: A series of experiments. *Information and Software Technology, 51*(2), 270–283. https://doi.org/10.1016/j.infsof.2008.01.007

Rubin, D. C., & Umanath, S. (2015). Event memory: A theory of memory for laboratory, autobiographical, and fictional events. *Psychological Review, 122*(1), 1–23. https://doi.org/10.1037/a0037907

Zhang, Y., Li, X., Wang, S., & Liu, J. (2025). Long context vs. RAG for LLMs: An evaluation and revisits. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing (EMNLP 2025)*. Association for Computational Linguistics.

Zhou, K., Prabhumoye, S., & Black, A. W. (2018). A dataset for document grounded conversations. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP 2018)* (pp. 708–713). Association for Computational Linguistics.

Zhu, Z., Li, Z., Li, S., & Lu, Y. (2023). Learning to prompt for vision-language models. *arXiv preprint* arXiv:2304.03442. https://doi.org/10.48550/arXiv.2304.0344

Appendix

Full Length Data [HL = HLGraphRAG, Rx = Run x]

Model Type	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	R11	R12	R13	R14	R15
Native	220	231	217	215	212	225	235	210	217	215	239	210	213	214	218
LangChain Base	179	183	181	174	183	178	184	178	176	193	182	186	175	178	175
GraphRAG	134	125	135	136	120	131	130	136	134	132	131	129	125	132	127
HL [SRQ, ECR]	71	64	63	72	65	66	75	61	63	74	68	67	72	62	65
HL + DW	58	56	64	62	65	59	61	59	64	65	58	63	63	62	61
HL + DW + RR	28	30	35	34	38	28	35	29	34	38	33	34	29	27	36

This is the data we used for statistical analysis. The averages to the 1st decimal place are as follows:

- 219.4, 180.3, 130.5, 67.2, 61.3, 32.5

Mann-Whitney U Test in Python

```
import numpy as np
from scipy.stats import mannwhitneyu

# Sample data (two independent groups)
set1 = np.array([71, 64, 63, 72, 65, 66, 75, 61, 63, 74, 68, 67, 72, 62, 65])
set2 = np.array([28, 30, 35, 34, 38, 28, 35, 29, 34, 38, 33, 34, 29, 27, 36])

# Perform one-tailed Mann-Whitney U test (test if set1 > set2)
stat, p = mannwhitneyu(set1, set2, alternative='greater')

# Print results
print(f"U statistic: {stat}")
print(f"P-value: {p}")
```

In my code, I modified *set1* and *set2* to be the data I am comparing. This code finds the p value that set2 < set1.