GDAŃSK UNIVERSITY OF TECHNOLOGY

TASK Quarterly 29 (3) 2025

# A Comparative Analysis of Standard and Graph-Based Retrieval-Augmented Generation

**Mikita Kasiak***

Faculty of Electronics, Telecommunications and Informatics, Gdańsk University of Technology, Narutowicza 11/12, Gdańsk, 80-233, Poland
*nikitaksk1@icloud.com

https://doi.org/10.34808/tq2025/29.3/c

## Abstract

This study investigates the performance of Retrieval-Augmented Generation (RAG) systems, comparing a standard implementation with a graph-driven approach. The paper details the architectural differences between the two systems and presents a comprehensive evaluation of their performance on a diverse dataset. The results demonstrate the advantages of using knowledge graphs to capture relationships between entities, particularly for complex queries. The analysis also considers the trade-offs between performance and resource consumption, providing insights into the practical applications of each approach.

## Keywords:

Retrieval-Augmented Generation, RAG, GraphRAG, knowledge graphs, information retrieval, performance evaluation

# 1. Introduction

Large Language Models are now widely used for tasks such as generating text, answering questions, and translating languages. A significant limitation of these models, however, is that their knowledge is restricted to the data they were originally trained on. Updating this knowledge is difficult because the training process is expensive and time-consuming. This leads to inaccurate or completely wrong outputs, commonly known as hallucinations. Retrieval Augmented Generation systems were created to address this problem by providing external knowledge directly to the LLMs.

The aim of RAG is to enhance LLM outputs by first retrieving relevant information from the source and then passing this information to an LLM. This process is used to improve the accuracy and relevance of the output while allowing for source citation. The standard RAG approach relies on vector-based similarity search, which has its own limitations.

This paper aims to address these limitations by experimenting with a graph-based approach to RAG. The main objective is to implement and compare a standard RAG against a GraphRAG, which uses a knowledge graph to store the relationships between different entities.

# 2. State of the Art

The rapid evolution of Large Language Models (LLMs) has necessitated the development of robust retrieval mechanisms. While LLMs exhibit remarkable reasoning capabilities, they are hampered by static knowledge cut-offs and the propensity for hallucinations [1]. This section reviews the progression from standard retrieval methods to advanced graph-based architectures and the specific challenges of automated knowledge graph construction.

## 2.1. Evolution of Retrieval-Augmented Generation

Retrieval-Augmented Generation (RAG) was formally introduced by Lewis et al. [2] to combine parametric memory with non-parametric external indices. The standard RAG workflow relies on dense vector retrieval, where relevance is determined via cosine similarity [3].

To overcome the limitations of simple similarity search, the field has diversified into several specialized architectures:

- **Trainable Retrieval:** Systems like REALM [4] integrate the retriever into the pre-training loop, allowing the model to learn which documents are most informative for a given query.
- **Agentic & Planning RAG:** Building on the ReAct framework [5], Agentic RAG systems do not merely retrieve; they formulate a multi-step plan, utilizing tools to query external databases iteratively. This addresses tasks requiring sequential logic rather than single-step lookup.
- **Branched Reasoning:** Inspired by the Tree of Thoughts (ToT) paradigm [6], branched RAG explores multiple retrieval paths simultaneously, evaluating the coherence of different distinct "chains of thought" before converging on an answer.

Despite these advancements, vector-based approaches face the "Lost-in-the-Middle" phenomenon [7], where LLMs struggle to access relevant information buried in the middle of retrieved contexts, and they often fail to capture explicit structural relationships between entities.

## 2.2. Knowledge Graphs and LLMs

Knowledge Graphs (KGs) provide a deterministic structure $(h, r, t)$ that mitigates the probabilistic nature of vector search. Pan et al. [8] categorize the intersection of LLMs and KGs into three paradigms: KG-enhanced LLMs, LLM-augmented KGs, and Synergized models.

In RAG contexts, KGs enable structural retrieval. Instead of relying on semantic similarity, the system performs graph traversal (e.g., Breadth-First Search or Random Walk) to identify connected concepts. This allows for "multi-hop" reasoning, where the answer lies not in a single document but in the relationship between two disparate nodes—a capability that standard dense retrieval often lacks [9].

## 2.3. Automated Graph Construction

A critical challenge in adopting GraphRAG is the cost of graph construction. Traditionally, KGs were built via supervised relation extraction pipelines requiring labeled data. Recently, LLM-based construction has emerged as a viable alternative.

In this paradigm, an LLM acts as an "Open Information Extraction" (OpenIE) system, parsing unstructured text to identify entities and relationships dynamically. While powerful, this approach introduces challenges regarding resolution and consistency. Cohen et al. [10] note that LLM-generated graphs often suffer from schema proliferation (e.g., creating synonyms like "worked_at" and "employed_by" for the same relation), which complicates traversal.

## 2.4. GraphRAG Architectures

Current GraphRAG implementations generally fall into two categories:

**Global Summarization:** Microsoft's GraphRAG [11] focuses on "Sensemaking" over massive corpora. It utilizes community detection (Leiden algorithm [12]) to generate hierarchical summaries of entity clusters. This is ideal for broad queries (e.g., "What are the major themes?") but can be computationally expensive for targeted fact retrieval.

**Local Traversal & Planning:** Systems like KG-RAG [13] and DRAGIN [14] focus on precise, query-driven traversal. DRAGIN employs a dynamic mechanism where the LLM decides *when* to traverse the graph based on information gaps in its generation buffer.

## 2.5. Gap Analysis

While significant research exists on novel RAG architectures, there is a limited comparative analysis focusing on the trade-offs between *construction cost*, *latency*, and *factual accuracy* in controlled environments. Most studies evaluate on standard benchmarks (e.g., HotpotQA), often obscuring the specific utility of graphs for "unknown" documents. This study aims to address this gap by evaluating Standard vs. Graph RAG on a dataset specifically designed to test knowledge boundaries.

# 3. Implementation

## 3.1. Overview

The system is built on a modular architecture defined by abstract base classes: **Indexer**, **Datastore**, **Retriever**, **ResponseGenerator**, and **Evaluator**. This design ensures a consistent API, enabling the central pipeline to orchestrate the process from data ingestion to response generation.

## 3.2. Standard RAG

The standard RAG pipeline, as illustrated in Figure 1, follows a linear process that serves as a baseline for our comparison.

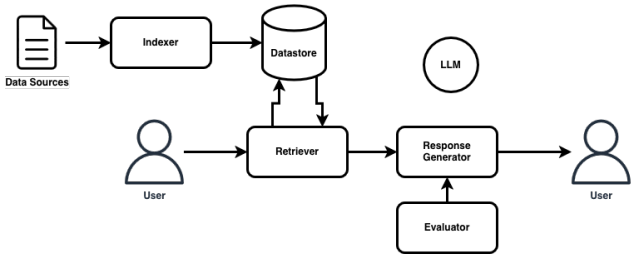


**Figure 1:** The architecture of the standard RAG pipeline, illustrating the linear flow from document indexing and retrieval to response generation.

### 3.2.1 Indexing

Documents are parsed and split into smaller text chunks using the `docling` library. Each chunk's content is then converted into a high-dimensional vector embedding and stored in a `LanceDB` table.

### 3.2.2 Retrieval and Reranking

For a given query, a vector similarity search is performed to find the top-k relevant document chunks. These chunks are then reranked using a model to improve the quality and relevance of the context provided to the language model.

### 3.2.3 Generation

The final, reranked context and the original query are formatted into a prompt for a Large Language Model (LLM), which synthesizes the information to produce a grounded answer.

## 3.3. GraphRAG

The GraphRAG pipeline, depicted in Figure 2, enhances the standard approach by constructing and leveraging a knowledge graph to capture relationships between entities within the documents.

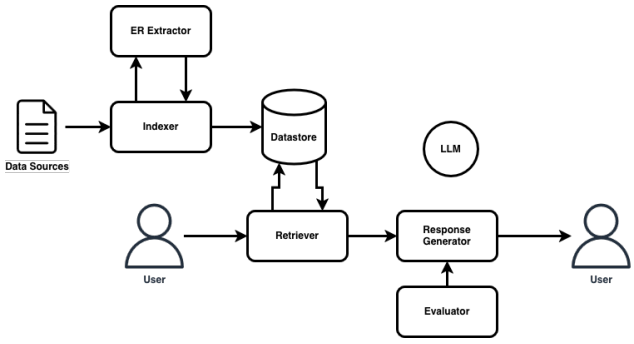


**Figure 2:** The architecture of the GraphRAG pipeline, highlighting the graph building process and the two-stage retrieval mechanism involving graph traversal.

### 3.3.1 Graph Construction and Schema

The graph-building process begins with chunking, where each text segment becomes a Chunk node. To extract structured knowledge, we employ a Large Language Model (LLM) prompted to identify entities and relationships within each chunk. The extraction is governed by a strict JSON schema requiring source, target, and label fields. This is the exact system prompt used to extract relationships from chunks:

```
You are an AI expert in knowledge graph
    extraction. Your task is to identify
    key entities and their relationships
    from the provided text.
Respond with ONLY a single, clean JSON
    object. Do not add any conversational
    text or explanations before or after
    the JSON.
The JSON object must have two keys: "
    entities" and "relationships".

- "entities" is a list of objects, where
    each object has:
  - "id": The name of the entity (e.g., "
      Tim Berners-Lee").
  - "type": The category of the entity (e.g
      ., "Person", "Organization", "Project
      ", "Concept").

- "relationships" is a list of objects,
    where each object has:
  - "source": The "id" of the source entity
      .
  - "target": The "id" of the target entity
      .
  - "label": A descriptive label for the
      relationship (e.g., "WORKS_AT", "
      PROPOSED", "BUILT").
```

The extracted data is persisted in LanceDB using a relational schema optimized for both vector search and graph traversal. Table 1 details the storage structure.

**Table 1:** Knowledge Graph Storage Schema in LanceDB

| Table | Field | Description |
|---|---|---|
| **Nodes** | id | Unique identifier (Entity Name) |
| | type | Entity category (e.g., Person, Event) |
| | vector | 384-dim dense embedding of the entity description |
| | source_id | Reference to the original Chunk ID |
| **Edges** | source | ID of the starting node |
| | target | ID of the ending node |
| | label | Semantic relationship (e.g., WORKS_AT) |

This structured output is used to build the graph. Unique entity nodes are created and connected to their source chunks and to each other, forming a comprehensive knowledge graph. The entire graph is stored in LanceDB using a two-table schema: a **nodes table** and an **edges table**. The nodes table stores the id, content, type, and a vector embedding for each node. The edges table stores the relationship with source, target, and label fields, creating a clear and efficient structure for graph traversal.

### 3.3.2 Retrieval and Reranking

Unlike standard vector retrieval, which relies solely on Cosine Similarity ($S_{cos}$), our GraphRAG implementation utilizes a hybrid scoring mechanism. The retrieval function $f(q,G)$ operates in three steps:

**1. Entry Point Identification:** We first identify a set of entry nodes $N_{entry}$ by embedding the query $q$ into a vector $v_q$ and performing a Nearest Neighbor search against the node embeddings $V_n$:

$$N_{entry} = \{n \in V \mid S_{cos}(v_q, v_n) > \theta\} \tag{1}$$

where $\theta$ is a similarity threshold.

**2. Graph Traversal (1-Hop Expansion):** For each entry node $n_i \in N_{entry}$, we expand the context to include its immediate neighbors. Let $\mathcal{N}(n_i)$ denote the set of neighbors of $n_i$. The retrieved subgraph $G_{ret}$ is defined as:

$$G_{ret} = \bigcup_{n_i \in N_{entry}} (\{n_i\} \cup \mathcal{N}(n_i)) \tag{2}$$

This expansion captures the immediate relational context (e.g., *Subject* $\rightarrow$ *Predicate* $\rightarrow$ *Object*), which is often lost in chunk-based retrieval.

**3. Context Linearization:** The retrieved subgraph is serialized into a textual format for the LLM. Each edge $(h,r,t)$ in $G_{ret}$ is converted into a natural language sentence. To prevent context window overflow, we apply a token budget limit $L_{max}$, prioritizing edges based on the similarity score of their source node.

### 3.3.3 Generation

The ResponseGenerator uses the top-ranked, contextually rich information from the graph retrieval process, combined with the user's query, to generate a final answer that is more accurate and comprehensive.

## 3.4. Technical Stack

The implementation relies on the following open-source technologies:

- **Core Language:** Python 3.12.2
- **ML/AI Frameworks:** PyTorch 2.8.0, Accelerate 1.10.1
- **Large Language Models:** OpenAI (GPT-4o), Google Gemini (2 Pro), Llama.cpp, LM Studio
- **Embeddings & Reranking:** SentenceTransformers 4.56.1 (all-MiniLM-L6-v2 for embeddings,

ms-marco-MiniLM-L-6-v2 for reranking)
- **Vector Database:** LanceDB 0.25.0
- **Data Handling:** Apache Arrow 21.0.0, Pandas 2.3.2, NumPy 2.2.6
- **Document Processing:** Docling 2.52.0

# 4. Experiments

This section details the experimental methodology and presents the results obtained from evaluating our RAG implementations. The primary objective of these experiments was to rigorously compare the performance of the proposed GraphRAG pipeline against a standard RAG approach and a baseline Large Language Model without retrieval augmentation. We aimed to assess their capabilities across diverse document types and varying question complexities, while also analyzing the associated resource consumption, specifically token usage and execution time. A key focus was to determine the optimal conditions for evaluating RAG systems, particularly regarding the LLM's prior knowledge of the source documents, and to highlight the scenarios where the GraphRAG approach offers significant advantages.

## 4.1. Experimental Setup

**Dataset:** The dataset consists of four documents, each selected to represent a different type of content and relationship with the LLM's:

- **Known Human-Written Text:** "An Occurrence at Owl Creek Bridge" by Ambrose Bierce. This document is presumed to be within the LLM's training data. Written by a human, well-structured.
- **Unknown Factual Text:** A transcript from the Apple WWDC 2025 conference, containing connected factual information unknown to the selected LLM.
- **Unknown AI-Generated Text (Polish):** A short novel generated by ChatGPT in Polish, created specifically for this study and not part of the LLM's training data. The quality and coherence of this document are variable.
- **Unknown Human-Written Novel:** "Love and Deaths," a novel by "Alien Writes" published on Archive of Our Own on June 12, 2025, and not part of the LLM's training data.

To ensure that the "unknown" documents were genuinely outside the LLM's training data, several verification steps were taken. First, the creation and publication dates of these documents were intentionally set after the knowledge cut-off date of the models used in the evaluation (May 31, 2024). Additionally, as a control test, the baseline LLM was prompted to summarize the content of each "unknown" document before the main experiment. The model's inability to provide a coherent or accurate summary confirmed its lack of prior knowledge regarding these texts.

One of the goals of this paper was to determine which documents are more viable for testing and evaluating RAG systems. The experiments show that it is crucial for the AI not to know the documents to properly evaluate the augmentation process.

The source documents in the dataset were standardized in Markdown (.md) format. This allowed for the inclusion of more complex elements, such as mathematical formulas (LaTeX) lists etc., ensuring sufficient structural variety to evaluate the system across diverse scenarios.

**Questions:** The evaluation dataset consisted of 326 questions serialized in JSON format. Each entry was structured to provide as much context as needed for proper evaluation. Each entry had:

- `question`: The input query text.
- `answer`: The ground-truth response used as the baseline for evaluating generation accuracy.
- `relevant_contexts`: A list of gold-standard text snippets containing the information necessary to answer the question, used to calculate retrieval metrics.
- `doc_name`: The specific Markdown filename from which the answer is derived, used to ensure source tracking.
- `type`: The classification of the question (e.g., single-hop, multi-hop, summarization), allowing for performance analysis across different retrieval complexities.

The types, their quantities, and examples are as follows:

- **Single-Hop (48):** Questions that can be answered from a single piece of information in one document. For example: *"What is the name of the new macOS introduced at WWDC2025?"*
- **Multi-Hop (112):** Questions that require combining information from multiple sources within the same document. For example: *"What emotions activated physical effects in the Harposphere?"*
- **Long-Hop (10):** A more complex version of multi-hop questions that require reasoning over a longer chain of information. For example: *"Why did Evelyn accept the demon's offer despite initially trying to dismiss him as a prank?"*
- **Thinking (77):** Questions that require the model to use information that is not explicitly stated in the text. For example: *"Why did Elya decide to integrate Elya.23 instead of destroying her?"*
- **Distracting (17):** Questions that include irrelevant or misleading information to test the system's ability to

focus on the relevant facts. For example: *"The stew morphed into a smiling face, but what is Evelyn's role in the castle?"*

- **Out-of-Context (27):** Questions that are not related to the content of the documents. For example: *"What are the differences between Apple Vision Pro and Meta Quest 3?"*. Conceptually, the right answer must state that the information is not located in the text.
- **Comparing (25):** Questions that require comparing and contrasting information from different parts of a document or across multiple documents. For example: *"Compare Dr. Elya Meren's and V'rath Zenthari's initial attitudes towards emotions as shown in Chapter 1."*
- **Multi-Doc (10):** Questions that require information from multiple documents to be answered. For example: *"Compare the nature and purpose of artificial intelligence as represented by Nokari in the AI novel and Apple Intelligence in the WWDC summary."*

**Evaluation Metrics:** While traditional Information Retrieval (IR) metrics such as *Normalized Discounted Cumulative Gain* (NDCG), *Mean Reciprocal Rank* (MRR), and *Precision@k* are standard for evaluating search ranking quality, they were not the primary focus of this study. These metrics assess the retrieval component in isolation. However, in RAG systems, the ultimate measure of utility is the factual correctness of the generated answer, regardless of the rank position of the supporting context.

Similarly, while *Human Evaluation* remains the gold standard for nuances, it is unscalable for large datasets (326 questions × 3 systems). Therefore, we selected **Binary Factual Accuracy** as our primary metric. This approach:

1. Focuses on the end-to-end system performance (Generation quality) rather than just the intermediate retrieval step.
2. Penalizes hallucinations strictly (an answer is either factually supported or it is not).
3. Allows for reproducible, large-scale evaluation using an LLM-as-a-Judge proxy, which has been shown to correlate closely with human judgment for factual tasks [15].

Consequently, our reported metrics are:

- **Factual Accuracy:** A binary metric where 1 indicates the generated answer contains the core truth required by the ground truth, and 0 indicates hallucination, incompleteness, or refusal to answer.
- **Token Usage:** The total count of input and output tokens, serving as a proxy for cost.
- **Execution Time:** The end-to-end latency from query submission to final response generation.

The system prompt used for the LLM judge to determine factual accuracy is as follows:

```
You are a system that evaluates the
    correctness of a response to a question.
The question will be provided in <question
    >...</question> tags.
The response will be provided in <response
    >...</response> tags.
The expected answer will be provided in <
    expected_answer>...</expected_answer>
    tags.

The response doesn't have to exactly match
    all the words/context the expected answer
    . It just needs to be right about
the answer to the actual question itself.

Evaluate whether the response is correct or
    not, and return your reasoning in <
    reasoning>...</reasoning> tags.
Then return the result in <result>...</result
    > tags --- either as 'true' or 'false'.
```

The OpenAI GPT-4.1 Nano model was used for our measurements due to its small size, which does not impact its ability to deliver exceptional performance on targeted, high-frequency tasks such as classification, autocompletion, and strict instruction following. [16]

## 4.2. Factual Accuracy Comparison

The overall factual accuracy of the three systems is presented in Figure 3. To ensure the reliability of our findings, each experiment was conducted multiple times, and the reported scores represent the average factual accuracy. The error bars in the figure indicate the standard deviation across these runs.

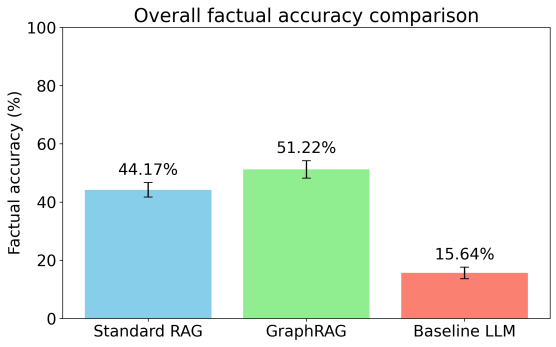


**Figure 3:** Comparison of overall factual accuracy scores across the three systems: Baseline LLM, Standard RAG, and GraphRAG.

The results show a clear performance leader. The GraphRAG system outperforms both the Standard RAG and the baseline LLM. On average, the baseline LLM answered 51 questions correctly. Notably, an average of 26 of these correct answers were from the known document (accounting for 59% of correct answers related to the known document), while on average only 25 (8% of all questions related to unknown documents) were from the

unknown documents. This supports the hypothesis that to accurately evaluate a RAG system's performance, the evaluation dataset must contain information unknown to the LLM.

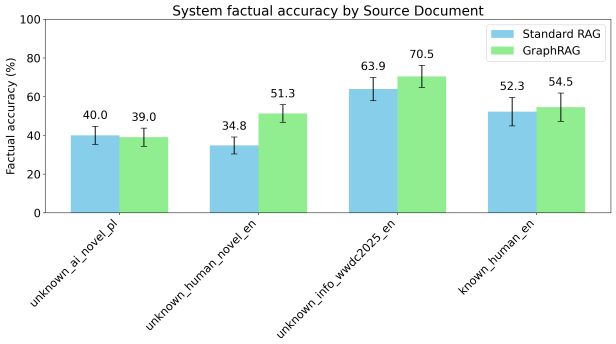


**Figure 4:** Factual accuracy scores broken down by document type, comparing the performance of Standard RAG and GraphRAG systems.

Figure 4 presents a breakdown of the RAG systems' performance by document. The error bars represent the standard error of the mean across multiple evaluation runs. The GraphRAG system shows a significant improvement in factual accuracy, especially in more structured documents. These results strongly support the hypothesis that documents used for evaluating should not be known by the LLM used for testing, because the gap in results is much less visible on the documents that the LLM was trained on. This aligns with our hypothesis that RAG evaluation requires documents unknown to the model. The performance gap is significantly narrower for known documents because the LLM can rely on its parametric memory rather than the retrieved context. Consequently, the 'Unknown' document categories provide a more accurate measure of the retrieval system's effectiveness.

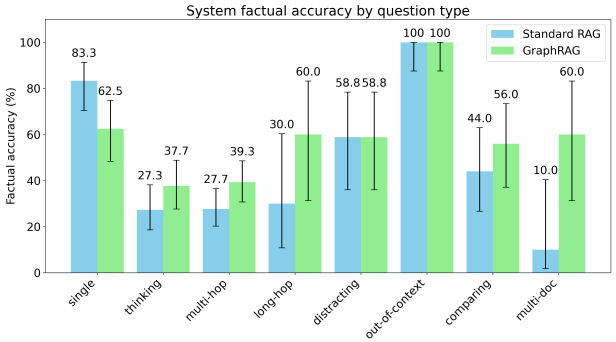


**Figure 5:** Factual accuracy scores broken down by question type, comparing the performance of Standard RAG and GraphRAG systems.

Figure 5 illustrates the factual accuracy of each system by question type. The error bars indicate the standard error, highlighting the consistency of performance for each question category. The GraphRAG system's ability to consistently provide context from multiple documents is a key advantage, resulting in the largest performance gap in the "Multi-Doc" category. Furthermore, the graph architecture allows it to outperform the Standard RAG in all multi-hop question types, while lagging slightly in contextually small questions.

## 4.3. Resource Usage

In addition to factual accuracy, the resource usage of each system was measured in terms of token usage and execution time. The values shown in Figures 6 and 7 are averages from multiple runs, with error bars representing the standard deviation to indicate the variability of resource consumption.

A significant portion of the GraphRAG's cost is attributed to the initial graph construction. This process took an average of 554 seconds and consumed 140,848 tokens to create a graph of 1,457 nodes and 3,101 edges from the source documents. It is important to note that this is a one-time, upfront cost for a given dataset. However, the quality of this generated graph is a critical factor. A manual analysis of a subset of the graph revealed that approximately 15% of the extracted relationships were either factually incorrect or irrelevant, which can impact the quality of the retrieved context.

The subsequent question evaluation phase also showed significant differences. The Standard RAG system's evaluation phase consumed an average of 816,106 tokens. In contrast, the GraphRAG system's evaluation phase consumed a significantly higher average of 4.7m tokens, excluding the initial graph building costs.

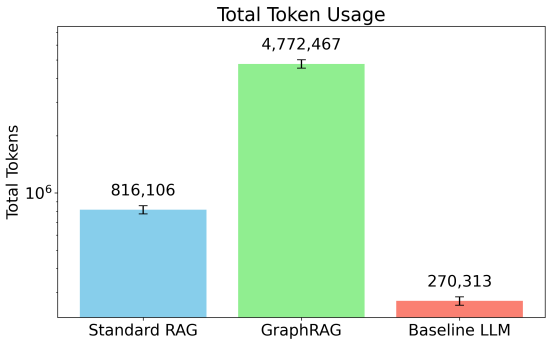


**Figure 6:** Total token consumption for a single evaluation run, comparing the three systems.

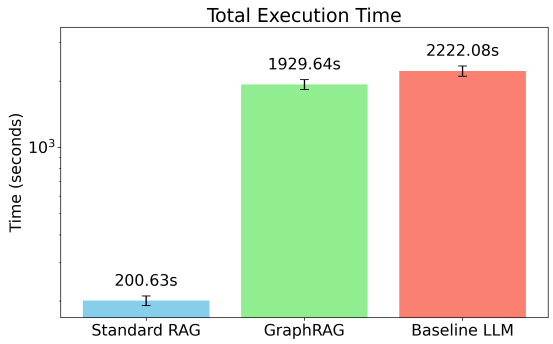


**Figure 7:** Total execution time for a single evaluation run, comparing the three systems.

As shown in Figures 6 and 7, the GraphRAG system exhibits higher token usage and execution time compared to the other systems. This is an expected trade-off for the increased accuracy, as the GraphRAG system has the additional overhead of constructing and querying the knowledge graph. The significant increase in token usage for GraphRAG led to API rate limiting during evaluation, which in turn increased the overall execution time. In contrast, the Standard RAG's more precise and strictly related context resulted in lower token usage, allowing for parallel evaluation in 8 threads without encountering rate limits. It is also worth noting that the LLM without RAG took a significantly longer time to attempt to answer questions without context.

# 5. Analysis and Limitations

While the experimental results demonstrate the superior accuracy of GraphRAG, particularly for multi-hop reasoning, a comprehensive evaluation must consider the operational trade-offs. This section analyzes the computational costs, the impact of graph quality on retrieval, and the latency implications for real-time deployment.

## 5.1. Cost-Benefit Analysis

The implementation of GraphRAG introduces a two-fold cost structure: the one-time cost of graph construction and the recurring cost of graph traversal during inference.

**Construction Overhead:** As detailed in Section 4.3, transforming the source documents into a knowledge graph is resource-intensive. The construction phase alone consumed approximately 140,848 tokens and required 554 seconds for a relatively small corpus. Unlike vector indexing, which scales linearly with text length, graph construction involves complex entity extraction and resolution steps that can scale quadratically depending on the density of relationships.

**Inference Costs:** A direct comparison of the resource consumption during the evaluation phase (answering 326 questions) reveals a significant disparity between the two systems. Table 2 summarizes the total resource usage.

**Table 2:** Comparative Resource Consumption (Evaluation Phase)

| Metric | Std. RAG | GraphRAG | Factor Inc. |
|---|---|---|---|
| Input Tokens | $\approx$ 650,000 | $\approx$ 3,800,000 | 5.8× |
| Output Tokens | $\approx$ 160,000 | $\approx$ 800,000 | 5.0× |
| Total Tokens | 816,106 | 4,631,619 | 5.7× |
| Avg. Execution Time | 200s | 1,629s | 8.1× |

The GraphRAG system consumed nearly 6× more tokens than the Standard RAG. This increase is attributed to the "context window flooding" caused by graph traversal. While Standard RAG retrieves a fixed number of chunks (top-$k$), GraphRAG's traversal often retrieves a subgraph that includes the central node, its edges, and neighboring nodes. While this rich context improves accuracy, it significantly inflates the prompt size sent to the LLM.

## 5.2. Graph Quality and Hallucinations

A critical limitation of GraphRAG is its dependence on the quality of the underlying Knowledge Graph (KG). Since the KG is constructed automatically by an LLM, it is susceptible to "extraction hallucinations." Unlike vector search, where low-relevance chunks can be filtered out via similarity thresholds, a hallucinated edge in a graph creates a valid traversal path. If the extraction step incorrectly links Entity A to Entity B, the traversal algorithm treats this connection as ground truth, potentially retrieving explicit misinformation.

**Invented Relationships:** During our manual audit of the constructed graph, we observed that approximately 15% of the extracted relationships were either redundant or contextually inaccurate. If the extraction model hallucinates a relationship (e.g., incorrectly linking *Entity A* to *Entity B* due to ambiguous syntax), this error is treated as ground truth by the retrieval system. Unlike vector search, where noise is often filtered out by low similarity scores, a hallucinated edge in a graph provides a "hard" path that the traversal algorithm will confidently follow.

**Error Propagation:** This creates a propagation effect. If a query relies on a hallucinated "bridge" node to connect two concepts, the GraphRAG system will generate an answer based on false premises, even if the generation LLM is capable of reasoning. Furthermore, inconsistencies in entity resolution (e.g., creating separate nodes for "Steve Jobs" and "Mr. Jobs") can fracture the graph, preventing the retrieval algorithm from finding the complete

context.

## 5.3. Latency and Real-Time Viability

The execution time analysis highlights a major bottleneck for real-time applications. The GraphRAG system exhibited an average execution time over 8× slower than Standard RAG.

**Time to First Token (TTFT):** In a Standard RAG setup, the latency is dominated by the vector search ($O(\log N)$) and the generation. In GraphRAG, the "Time to First Token" is significantly delayed by the traversal step. The system must:

1. Identify entry nodes via vector search.
2. Load the adjacency list.
3. Traverse edges (often requiring multiple database lookups).
4. Rerank the expanded subgraph.

This multi-stage process introduces a latency floor that may be unacceptable for interactive chat applications (e.g., customer support bots) where sub-second response times are expected. However, for asynchronous tasks such as analytical report generation, complex legal discovery, or scientific literature review, the increased latency is a justifiable trade-off for the demonstrated gain in factual accuracy and reasoning depth.

## 5.4. Error Analysis

Upon a detailed manual inspection of the system outputs and the underlying graph structures, we identified three primary categories of errors that significantly impact performance: topological noise in the graph, citation inaccuracies, and sensitivity issues within the automated evaluation pipeline.

### 5.4.1 Entity Extraction and Resolution Failures

Our analysis indicates that entity extraction via LLMs introduces systematic error rates of approximately 15–20% during knowledge graph construction. These errors propagate throughout the pipeline, fundamentally altering the graph topology and degrading downstream retrieval quality.

The failures can be categorized into three distinct architectural limitations:

- **Naive Name-Based Resolution (False Mergers):** Most GraphRAG implementations predominantly rely on exact string matching for resolution. If two entities share an identical name, they are automatically merged. This creates "False Mergers," where distinct entities (e.g., two hospitals in the same city with the same name) are collapsed into one node. This topological error physically connects unrelated subgraphs, causing the retrieval algorithm to "hallucinate" connections between unrelated contexts.
- **Entity Fragmentation (Surface Form Variations):** Conversely, the system frequently fails to unify different surface forms of the same entity. Variations such as "John" versus "J. Smith," or "The Agency" versus "CIA," are treated as entirely separate nodes. This fragmentation disperses the knowledge about a single entity across multiple disconnected nodes, preventing the retrieval algorithm from aggregating all relevant context in a single pass.

A recurrent issue was the contamination of the knowledge graph with generic pronoun nodes, such as `(PERSON, "He")` or `(PERSON, "She")`. This creates "super-nodes" that are densely connected to unrelated entities, degrading the quality of graph traversal.

**Reasons and Assumptions:**

- **Lack of Coreference Resolution:** This is the primary cause. Our system processes documents in independent chunks (e.g., 600 tokens). If a chunk begins with *"He walked into the room..."*, the LLM lacks the context from the previous chunk where "He" was identified as "John". Without an external coreference resolution step prior to chunking, the extraction model treats "He" as a standalone entity. This might result in cross-document edges that will reduce both the performance and the accuracy of the system.
- **Capitalization Bias:** LLMs exhibit a statistical bias to treat capitalized words at the start of sentences as proper nouns. This is exacerbated if the extraction prompt instructions are not sufficiently strict regarding pronoun exclusion.

### 5.4.2 Citation and Attribution Failures

While the system often retrieved correct information, it frequently failed to provide accurate document sources or answer specific attribution questions (e.g., *"Who said..."*).

**Reasons and Assumptions:**

- **Loss of Provenance in Summaries:** GraphRAG is architected for global understanding rather than local precision. It aggregates information from many text chunks into "summaries." Consequently, the link between a specific fact in a summary and its exact source chunk is often indirect or lost, making backtracking to a specific file or line number difficult or almost impossible.
- **Granularity Mismatch:** Attribution is inherently a *local* task, whereas GraphRAG's Global Search is optimized for *global* thematic questions. This represents

an architectural mismatch.

### 5.4.3 LLM-as-a-Judge Sensitivity

Our evaluation pipeline revealed instances where the "Judge" LLM penalized responses that were partially correct but incomplete, labeling them as entirely false. This suggests that strict binary metrics may underestimate the utility of RAG systems.

An example of this strict penalization is shown in Listing 1. The model correctly identified the "Foundation Models framework" but was marked incorrect for omitting "Xcode 26," despite the answer being factually true.

Listing 1: Example of LLM Judge rejecting a partially correct answer.

```
<question>
Which new development tools support building
    apps using on-device Apple Intelligence?
</question>

<response>
The Foundation Models framework supports
    building apps that use on-device Apple
    Intelligence.
</response>

<expected_answer>
The Foundation Models framework and large
    language model integration in Xcode 26.
</expected_answer>

<reasoning>
The response mentions that the Foundation
    Models framework supports building apps
    using on-device Apple Intelligence, which
     aligns with the expected answer...
    However, the expected answer explicitly
    mentions "large language model
    integration in Xcode 26"... whereas the
    response only references the Foundation
    Models framework... Therefore, the
    response is incomplete and does not fully
     match the expected answer.
</reasoning>

<result>false</result>
```

# 6. Conclusions

## 6.1. Summary of Findings

This paper presented a comparative study of a standard RAG system, a GraphRAG system, and a baseline LLM. The experimental results demonstrate a clear performance hierarchy, with the GraphRAG system consistently outperforming the other two approaches across all evaluation metrics. The GraphRAG system's ability to leverage the relationships between entities in the knowledge graph allows it to provide more accurate and contextually relevant answers, particularly for complex questions that require multi-hop reasoning [17].

## 6.2. Discussion

The findings of this study indicate that GraphRAG is a promising approach for enhancing the capabilities of LLMs. The main advantage of GraphRAG is its ability to understand the relationships between entities, which allows it to provide more comprehensive answers than standard RAG systems. However, this improved performance comes with several limitations. Firstly, its effectiveness is heavily reliant on the quality of entity and relationship extraction performed by the LLM during graph building. Secondly, the overall performance can be significantly affected by the complexity and structure of the input documents. Finally, graph extraction is an inherently costly process, both in terms of token usage and computational time, and can be non-deterministic, posing challenges for reproducibility and consistent quality.

A deeper analysis of GraphRAG's failures reveals challenges in tuning the graph traversal process. Finding the optimal balance for exploration was a significant engineering effort. An overly broad traversal often introduced noise by pulling in too many irrelevant neighbor nodes, diluting the context provided to the LLM. Conversely, a traversal that was too narrow often failed to retrieve sufficient context, performing no better than a standard vector search. The system's performance is therefore highly sensitive to its internal parameters, specifically the similarity threshold ($\theta$) and the maximum neighbor count ($k$). Many of the observed errors can be attributed to this tuning challenge, where the retrieval was either too wide, leading to a noisy context, or too thin, leading to an incomplete one.

## 6.3. The Necessity of Unseen Data for Evaluation

One of the most significant findings of this study extends beyond the architectural comparison of RAG versus GraphRAG. Our experiments identified a critical flaw in standard evaluation methodologies that utilize public datasets.

When evaluating on the "Known" document (*Owl Creek Bridge*), the performance gap between the Baseline LLM (no context) and the RAG systems was minimal. This indicates parametric Memory Leakage: the model was answering questions based on its pre-training weights rather than the retrieved context [18].

In stark contrast, the "Unknown" datasets (e.g., the unpublished novel and 2025 transcripts) acted as a

stress test that forced the systems to rely exclusively on retrieval. It was only under these conditions that the superior reasoning capabilities of GraphRAG became statistically apparent. Consequently, we argue that RAG benchmarks must strictly enforce a "Knowledge Cutoff" protocol, using only synthetic or post-cutoff data to prevent the LLM's internal knowledge from masking retrieval failures.

### 6.4. Future Work

There are several promising directions for future research to address the limitations identified above.

**Algorithmic Parameter Tuning:** Given the system's sensitivity to configuration, a critical area for improvement is the automation of hyperparameter selection. Future work will explore the application of statistical optimization methods—such as **linear regression analysis** or **Bayesian optimization [19]**—to dynamically tune the similarity threshold ($\theta$) and neighbor expansion limits based on query complexity and graph density. This would replace the current manual calibration process with a data-driven approach, ensuring optimal traversal depth without extensive human engineering.

**Advanced Graph Analysis:** The GraphRAG system could be further improved by incorporating more advanced graph analysis techniques. We aim to integrate community detection algorithms (e.g., Leiden [12] or Louvain) to identify clusters of related information, allowing the system to retrieve "themes" rather than just individual node paths. Additionally, the Named Entity Recognition [20] (NER) process could be refined to extract a wider range of entity types and subtler semantic relationships.

**Deterministic Extraction Models:** To mitigate the instability of LLM-based extraction, we propose replacing the generative extraction layer with specialized, deterministic models (e.g., GLiNER [21] or fine-tuned BERT-based Named Entity Recognizers). Moving away from "prompt-based" extraction to "model-based" extraction could significantly reduce hallucinated nodes and improve graph consistency while lowering the one-time indexing cost.

**Broader Evaluation:** Finally, the evaluation could be extended to include a more diverse range of document types (e.g., legal contracts, technical manuals) and a broader set of question types. Applying the system to new domains will help verify its generalizability and further quantify the trade-off between the high computational cost of graph construction and the gains in reasoning accuracy.

## Availability of Data and Materials

To facilitate reproducibility and further research, the source code for the GraphRAG implementation, along with the evaluation datasets and the specific prompts used in this study, are openly available on GitHub at: `https://github.com/Nikitaksk/RAG_vs_GRAPHRAG`.

## References


[1] A. Alansari and H. Luqman, "Large language models hallucination: A comprehensive survey," 2025. https://arxiv.org/abs/2510.06265.

[2] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W. tau Yih, T. Rocktäschel, S. Riedel, and D. Kiela, "Retrieval-augmented generation for knowledge-intensive nlp tasks," 2021. https://arxiv.org/abs/2005.11401.

[3] V. Karpukhin, B. Oğuz, S. Min, P. Lewis, L. Wu, S. Edunov, D. Chen, and W. tau Yih, "Dense passage retrieval for open-domain question answering," 2020. https://arxiv.org/abs/2004.04906.

[4] K. Guu, K. Lee, Z. Tung, P. Pasupat, and M.-W. Chang, "Realm: Retrieval-augmented language model pre-training," 2020. https://arxiv.org/abs/2002.08909.

[5] S. Yao, J. Zhao, D. Yu, N. Du, I. Shafran, K. Narasimhan, and Y. Cao, "React: Synergizing reasoning and acting in language models," 2023. https://arxiv.org/abs/2210.03629.

[6] S. Yao, D. Yu, J. Zhao, I. Shafran, T. L. Griffiths, Y. Cao, and K. Narasimhan, "Tree of thoughts: Deliberate problem solving with large language models," 2023. https://arxiv.org/abs/2305.10601.

[7] N. F. Liu, K. Lin, J. Hewitt, A. Paranjape, M. Bevilacqua, F. Petroni, and P. Liang, "Lost in the middle: How language models use long contexts," 2023. https://arxiv.org/abs/2307.03172.

[8] S. Pan, L. Luo, Y. Wang, C. Chen, J. Wang, and X. Wu, "Unifying large language models and knowledge graphs: A roadmap," *IEEE Transactions on Knowledge and Data Engineering*, vol. 36, p. 3580–3599, July 2024. http://dx.doi.org/10.1109/TKDE.2024.3352100.

[9] H. Trivedi, N. Balasubramanian, T. Khot, and A. Sabharwal, "Interleaving retrieval with chain-of-thought reasoning for knowledge-intensive multi-step questions," 2023. https://arxiv.org/abs/2212.10509.

[10] R. Cohen, M. Geva, J. Berant, and A. Globerson, "Crawling the internal knowledge-base of language models," 2023. https://arxiv.org/abs/2301.12810.

[11] D. Edge, H. Trinh, N. Cheng, J. Bradley, A. Chao, A. Mody, S. Truitt, D. Metropolitansky, R. O. Ness, and J. Larson, "From local to global: A graph rag approach to query-focused summarization," 2025. https://arxiv.org/abs/2404.16130.

[12] V. A. Traag, L. Waltman, and N. J. van Eck, "From louvain to leiden: guaranteeing well-connected communities," *Scientific Reports*, vol. 9, Mar. 2019. http://dx.doi.org/10.1038/s41598-019-41695-z.

[13] P. Jiang, L. Cao, R. Zhu, M. Jiang, Y. Zhang, J. Shen, J. Sun, and J. Han, "Ras: Retrieval-and-structuring for knowledge-intensive llm generation," 2026. https://arxiv.org/abs/2502.10996.

[14] W. Su, Y. Tang, Q. Ai, Z. Wu, and Y. Liu, "Dragin: Dynamic retrieval augmented generation based on the information needs of large language models," 2024. https://arxiv.org/abs/2403.10081.

[15] L. Zheng, W.-L. Chiang, Y. Sheng, S. Zhuang, Z. Wu, Y. Zhuang, Z. Lin, Z. Li, D. Li, E. P. Xing, H. Zhang, J. E. Gonzalez, and I. Stoica, "Judging llm-as-a-judge with mt-bench and chatbot arena," 2023. https://arxiv.org/abs/2306.05685.

[16] Z. Li, Y. Zhang, M. Li, Y. Ji, Y. Zeng, N. Cheng, Y. Zhu, Y. Wang, S. Wang, J. Xiao, and D. He, "Rethinking llm-as-a-judge: Representation-as-a-judge with small language models via semantic capacity asymmetry," 2026.

[17] X. Wang, J. Wei, D. Schuurmans, Q. Le, E. Chi, and D. Zhou, "Rationale-augmented ensembles in language models," 2022. https://arxiv.org/abs/2207.00747.

[18] A. Mallen, A. Asai, V. Zhong, R. Das, D. Khashabi, and H. Hajishirzi, "When not to trust language models: Investigating effectiveness of parametric and non-parametric memories," 2023. https://arxiv.org/abs/2212.10511.

[19] G. Onorato, "Bayesian optimization for hyperparameters tuning in neural networks," 2024. https://arxiv.org/abs/2410.21886.

[20] K. Pakhale, "Comprehensive overview of named entity recognition: Models, domain-specific applications and challenges," 2023. https://arxiv.org/abs/2309.14084.

[21] U. Zaratiana, N. Tomeh, P. Holat, and T. Charnois, "Gliner: Generalist model for named entity recognition using bidirectional transformer," 2023. https://arxiv.org/abs/2311.08526.