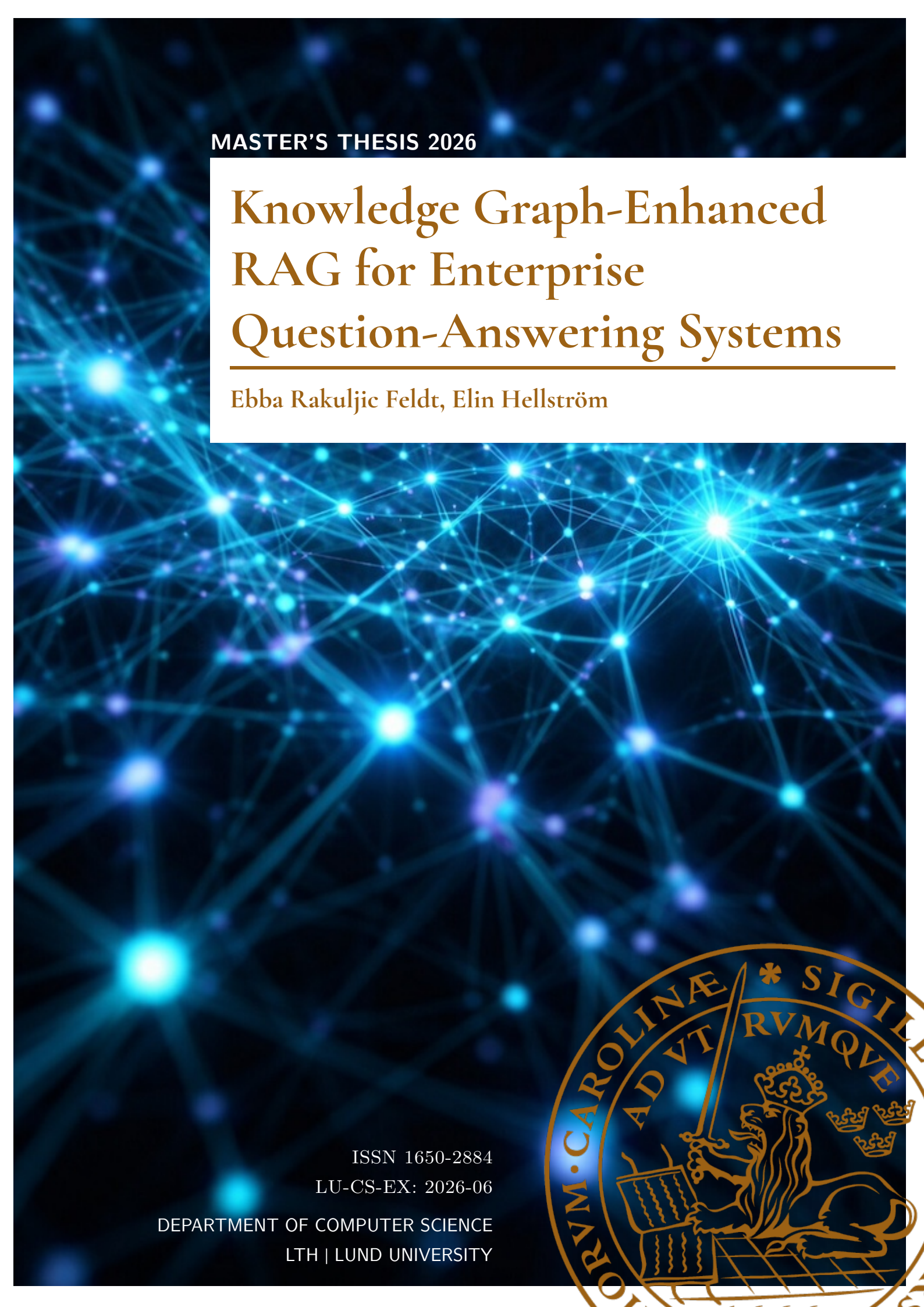
MASTER'S THESIS 2026

Knowledge Graph-Enhanced RAG for Enterprise Question-Answering Systems

Ebba Rakuljic Feldt, Elin Hellström



ISSN 1650-2884
LU-CS-EX: 2026-06

DEPARTMENT OF COMPUTER SCIENCE
LTH | LUND UNIVERSITY

EXAMENSARBETE
Datavetenskap

LU-CS-EX: 2026-06

Knowledge Graph-Enhanced RAG for Enterprise Question-Answering Systems

Användning av kunskapsgrafer i RAG för företags frågesvarssystem

Ebba Rakuljic Feldt, Elin Hellström

Knowledge Graph-Enhanced RAG for Enterprise Question-Answering Systems

(Comparative Evaluation of RAG vs GraphRAG)

Ebba Rakuljic Feldt
ebba.feldt@gmail.com

Elin Hellström
elin2001.hellstrom@gmail.com

February 26, 2026

Master's thesis work carried out at Axis Communications AB.

Supervisors: Marcus Klang, marcus.klang@cs.lth.se
Nathan Hoche, Nathan.Hoche@axis.com

Examiner: Jacek Malec, jacek.malec@cs.lth.se

Abstract

As new knowledge emerges rapidly, the knowledge base of large language models (LLMs) can quickly become outdated, requiring frequent retraining, an expensive and resource-intensive process. Retrieval-augmented generation (RAG) addresses this by providing LLMs with up-to-date or domain-specific knowledge at inference time. However, traditional RAG systems struggle with broad, summarizing queries and complex reasoning, since retrieval is primarily based on semantic similarity. Recently, an alternative approach has been proposed that combines RAG with a knowledge graph (GraphRAG) to improve logical consistency and reasoning. In this thesis, we compare the performance of GraphRAG and traditional RAG on both internal enterprise data from Axis and the GraphRAG-Bench benchmark. For automatic construction of the knowledge graph, we use LLMs to extract entities and relations. To evaluate the enterprise data, we generate a synthetic question-answer dataset using LLMs and use an LLM-as-a-judge to assess performance. Our results show that, under the evaluated conditions, traditional RAG achieves higher performance than GraphRAG on many QA tasks, both on the benchmark and in the LLM-based evaluation of internal data.

Keywords: knowledge graphs, retrieval-augmented generation (RAG), large language model (LLM), question answering (QA), LLM-as-a-judge

Acknowledgements

We would like to thank our supervisor Marcus Klang at Lund University for his continuous guidance, invaluable feedback and assistance in shaping this thesis. We also extend our gratitude to Axis Communications for providing the opportunity to conduct this thesis and to Nathan Hoche for his support and encouragement throughout the project.

Contents

Chapter 1

Introduction

Large Language Models (LLMs) have emerged as a powerful tool for natural language tasks. They are trained on massive amounts of publicly available text data, allowing them to perform well across a wide range of tasks (Minaee et al., 2025; Brown et al., 2020). As a result, companies increasingly rely on LLMs to assist in daily operations, particularly in question-answering systems. However, limitations related to data staleness, proprietary data access, and retraining costs hinder organizations from fully leveraging LLMs (Yu et al., 2025b).

The first limitation concerns the static nature of an LLM's knowledge, which is fixed at training time. Because real-world information is constantly evolving, LLMs may quickly become outdated (Gao et al., 2024). Although retraining can be used to incorporate new knowledge, it is often a cumbersome process that is costly, energy-intensive, and time-consuming (Miller, 2023; Patterson et al., 2021). In addition, companies typically possess large amounts of internal and proprietary data that cannot be shared outside the organization. As a result, LLMs trained on open-source data lack access to this information. To address this, companies can retrain or fine-tune models with internal data, enabling them to answer questions grounded in proprietary knowledge. However, current state-of-the-art models are often closed-source, meaning their parameters and architectures are not publicly accessible, primarily for competitive and commercial reasons (Manchanda et al., 2025). This makes further training unfeasible (Zhang et al., 2025) and even when open-source models are used, retraining and hosting large models locally can be expensive due to the need for powerful GPU resources (Bendi-Ouis et al., 2025).

To solve these issues, Lewis et al. (2020) proposed an alternative approach called Retrieval Augmented Generation (RAG). In RAG systems, external knowledge is provided to the LLM at inference time by retrieving context relevant to the user query. Rather than altering the model itself, RAG acts as a wrapper that supplies the LLM with additional context. This makes it easy to ingest new data and keep the system up to date (Lewis et al., 2020). It

can also be used to improve question-answering on domain-specific knowledge by grounding LLM responses in retrieved information, thereby reducing hallucinations (Lewis et al., 2020). Additionally, RAG increases traceability as it allows generated answers to reference supporting sources.

1.1 Motivation

Despite their advantages, traditional RAG systems have several limitations. Because retrieval is primarily based on semantic similarity, they may fail to capture more complex relationships within the data (Peng et al., 2025). Additionally, RAG systems struggle with queries that require summarization or multi-step reasoning over larger subsets of the corpus (Edge et al., 2025; Peng et al., 2025), particularly when the relevant information is scattered across the knowledge base.

These limitations call for alternative knowledge bases that more effectively capture relationships and logical structure within the data. Knowledge graphs address this by providing a structured representation of data in which entities are represented as nodes and relationships are represented as edges (Hogan et al., 2021). This enables connections between data to be captured rather than isolated facts. They also serve as an effective way to organize and visualize information from multiple data sources. Integrating knowledge graphs with LLMs can enhance retrieval and reasoning in RAG systems, providing structured context for improved question-answering (Zhang et al., 2025).

However, constructing high-quality knowledge graphs can be challenging. Recent work has explored using LLMs to automate knowledge graph construction, reducing the need for manual graph creation and human supervision (Zhu et al., 2024). Another challenge of using knowledge graphs in real-world applications, particularly in enterprise settings, is that a ground truth graph or a gold standard question-answering (QA) dataset rarely exists (Li et al., 2024d). For large graphs, human evaluation becomes extremely time-consuming or even infeasible, making it difficult to assess both graph correctness and answer accuracy when used in a RAG system. To address this, emerging approaches leverage LLMs themselves as evaluators, referred to as "LLM-as-a-judge" (Li et al., 2024a; Zheng et al., 2023). Similarly, LLMs can generate synthetic evaluation datasets from raw data, reducing the need for manually curated benchmarks for specific domains.

1.2 Objectives and research questions

This thesis investigates how large language models (LLMs) can be used to automatically construct knowledge graphs and how these graphs can be integrated into retrieval-augmented generation (RAG) systems to improve the quality and reliability of question-answering (QA). We compare the performance of knowledge graph-enhanced RAG (GraphRAG) with that of traditional RAG, analyzing their respective strengths and weaknesses. In addition, we examine synthetic question generation and LLM-as-a-judge as alternative methods for evaluation.

High computational resources and frequent LLM calls can create significant bottlenecks, lim-

iting the broader adoption of GraphRAG (Min et al., 2025; Xiao et al., 2025). Additionally, much of the existing research on GraphRAG relies on large, resource-intensive models (Han et al., 2025b). This thesis explores whether comparable results can be achieved using cheaper, open-source models and a simplified pipeline.

This study aims to answer the following research questions:

RQ1 How does standard RAG compare to GraphRAG in answering (a) fact retrieval and complex reasoning questions, and (b) summarization and creative generation tasks?

RQ2 How reliable are synthetic question generation and LLM-as-a-judge methods for evaluating QA systems?

RQ3 How does incorporating a knowledge graph into RAG improve an LLM's performance in enterprise-specific QA?

RQ4 What are the computational and cost trade-offs between standard RAG and GraphRAG?

1.3 Related work

Retrieval-augmented generation (RAG) has emerged as a common approach to incorporate external knowledge into large language models (Lewis et al., 2020). Recent advances in RAG have introduced graph-based representations to support multi-hop reasoning and improve retrieval accuracy. Among these approaches, Edge et al. (2025) proposed one of the earliest GraphRAG frameworks.

Evaluating these systems remains an active challenge. While various benchmarks exist for evaluating traditional RAG systems, fewer are specifically tailored for GraphRAG. Many existing evaluations treat the pipeline as a black box, performing end-to-end evaluation in which only the generated output is assessed (Xiang et al., 2025). In response to these limitations, GraphRAG-Bench (Xiang et al., 2025) was specifically developed to evaluate GraphRAG systems, assessing multiple stages of the pipeline, including end-to-end output, retrieval correctness, and meta-level graph properties. The benchmark includes datasets and an evaluation framework that combines statistical metrics with LLM-as-a-judge methodologies. This reflects a broader trend in generative AI evaluation, as modern evaluation frameworks increasingly rely on LLM-as-a-judge to automatically assess model outputs (Zheng et al., 2023; Li et al., 2024a; Gu et al., 2026). However, evaluation practices for GraphRAG remain fragmented and lack standardization (Peng et al., 2025).

1.4 Limitations

Combining RAG with knowledge graphs is a relatively new research area, and novel approaches are continuously being developed. A large proportion of the relevant literature consists of recent preprints published within the past year, with many being released during

the course of this thesis. As a result, there is currently no established consensus on which GraphRAG approaches perform best or how they should be evaluated.

The thesis was also constrained by the models accessible through the company, which made it difficult to replicate and compare benchmark results. In addition, deployment constraints such as strict rate limits and request timeouts required us to downscale tests and datasets to a feasible size.

1.5 Development tools and assistance

Generative AI tools (including ChatGPT) were used for spell-checking and text refinement. Claude Code was used for code assistance and debugging.

1.6 Division of work

The majority of the work for this master's thesis was carried out collaboratively, and the workload was divided equally. The project began with individual literature studies, after which the practical work was done through collaborative software development using a shared Git repository. Ebba focused on the LLM API calls and response handling, data preprocessing, and triple extraction, while Elin worked on chunking, community generation and summarization, and the retrievers for GraphRAG and traditional RAG. Both authors collaborated on the entity resolution module. During the evaluation phase, Ebba contributed to synthetic question generation, development of the LLM-as-a-judge system, and testing on private Axis data, whereas Elin adapted and integrated the benchmark into our system and executed the corresponding evaluations. All major design decisions and result analysis were carried out jointly. The writing of the thesis was also done collaboratively, with all sections reviewed by both authors.

1.7 Thesis disposition

- **Chapter 1: Introduction** introduces the motivation and objectives of the thesis, formulates the research questions, reviews related work, outlines limitations, and describes the division of work and overall thesis structure.
- **Chapter 2: Theoretical Background** presents the fundamental concepts relevant to this thesis, including large language models, question-answering, knowledge graphs, entity resolution, retrieval-augmented generation (RAG and GraphRAG), and evaluation methodologies.
- **Chapter 3: Method** describes the system design and methodology, including RAG and GraphRAG pipelines, datasets, language models, knowledge graph construction, retrieval strategies, and the experimental and qualitative evaluation setup.

- **Chapter 4: Results and Analysis** presents the experimental results, including benchmark evaluations, retrieval and generation performance, graph-level analysis, and qualitative results on enterprise data.
- **Chapter 5: Discussion** discusses the results in depth, including benchmark and enterprise data findings, limitations, ethical and societal considerations, and directions for future work.
- **Chapter 6: Conclusion** summarizes the main contributions of the thesis and answers the research questions.
- **Appendices** provide supplementary material including nomenclature, prompts, and example data objects used in the experiments.

Chapter 2

Theoretical Background

2.1 Large Language Models

Large Language Models (LLMs) are neural network models trained on massive text corpora to learn the statistical structure of natural language (Liu et al., 2024b). The term *large* refers to both the scale of the training data and the number of learnable parameters, enabling generalization across many domains. A *language model* predicts the next token based on previous tokens in a sequence, capturing syntactic and semantic patterns present in text. The knowledge learned during training is stored implicitly in the model parameters, often referred to as the model's *parametric knowledge* or *parametric memory*. This knowledge is fixed at inference time and cannot be updated without retraining.

Modern state-of-the-art LLMs are commonly built upon the transformer architecture introduced by Vaswani et al. (2017), which employs multi-head self-attention to compute sequence representations. These models are trained through self-supervised learning, typically by masking or removing tokens and asking the model to predict them using vast amounts of unlabeled text (Liu et al., 2024b). Although such models are not inherently generative or stochastic, they can be made generative at inference time by selecting the most likely next token (deterministic decoding) or by sampling from the model's predicted next token distribution.

We distinguish between three categories of LLM-based systems. A *base LLM* is the initial pretrained model, trained solely to model language rather than to follow instructions. To better align these models with human preferences and task-specific instructions, they are often further fine-tuned using supervised fine-tuning and reinforcement learning from human feedback (Zhao et al., 2025). The resulting *instruction-tuned LLMs* are capable of following natural-language instructions and performing downstream tasks based on user prompts.

Finally, *conversational agents* embed an instruction-tuned LLM within a broader software system that manages dialogue context, conversational history, and sometimes tool usage or external actions. These systems are deployed in user-facing applications such as chat interfaces or assistants. Additional inference-time mechanisms such as temperature or top-p sampling can be applied to control output randomness and variation in generated text (Minaee et al., 2025).

2.1.1 Prompting

Prompting strongly influences the responses generated by LLMs. Prior work shows that even changes to prompt formatting, while preserving meaning, can lead to unpredictable variation in model accuracy (Sclar et al., 2024), raising concerns about the reliability and robustness of LLM-generated outputs.

Some baseline prompting techniques that are commonly used are *zero-shot*, *one-shot* and *few-shot prompting*. Zero-shot prompting means the LLM is only given a task but no examples, relying entirely on its prior training. One- and few-shot prompting means the LLM is given one or a few examples of input-output pairs, to give the LLM more guidance on the desired output format. Supplying a few examples generally improves the accuracy, as the LLM is given more context (Brown et al., 2020). Assigning roles to the LLM, so-called role-play prompting, has also been shown to improve zero-shot question-answering (Kong et al., 2024). Furthermore, recent studies suggest that for sufficiently large models, prompting alone can achieve performance competitive with fine-tuning across a range of tasks (Brown et al., 2020).

The context window specifies the maximum number of tokens an LLM can process at once. Inputs exceeding this limit are truncated, resulting in loss of information. Although modern models offer increasingly large context windows, studies show that performance can degrade when relevant information appears in the middle of long prompts (Liu et al., 2024a), a phenomenon known as the *lost in the middle* dilemma. Models typically attend more strongly to information presented at the beginning or end of the prompt. Although longer inputs can provide more context, LLMs may struggle to utilize this information, suggesting that shorter and more focused prompts can improve robustness. In this thesis, we define the *system prompt* as the component that sets the model's role, behavior, and core instructions, and the *user prompt* as the input to which the LLM responds.

2.1.2 Reproducibility and deterministic generation

Temperature is a parameter which controls the randomness of the LLMs response. As previously stated, LLMs generate text token-by-token, choosing the next token following some probability distribution given by the networks weights. A zero temperature makes the LLM produce more deterministic responses, meaning the LLM performs greedy decoding, always choosing the most probable token. If the temperature is set to larger than zero, the LLM will not necessarily choose the most probable next token, but rather randomly sample a token given the probability distribution. The higher the temperature, the more random and "creative" we allow the model to behave (Blackwell et al., 2025).

Changing temperatures from one to zero has shown to not have a significant impact on the LLMs ability to perform problem-solving tasks (Renze, 2024). Unless creative generation and novelty is the goal of the generation, setting the temperature to zero reduces uncertainty in the generation, and thereby contributes to more deterministic and reproducible results (Blackwell et al., 2025). A random seed can also be set to fixate the randomness in the LLMs sampling process, so that the results can be reproduced.

Providing the input is the exact same, and either a random seed is used or temperature is set to zero, the LLM should in theory generate deterministic responses, yet, deterministic behavior can not be strictly guaranteed. Hardware set-up and runtime configurations, such as hardware versions, number of GPUs, or how many requests are processed in parallel during inference, can still introduce randomness, resulting in inconsistent results (Yuan et al., 2025). Online models can be set to update automatically, making it difficult to ensure the exact same model is used in the background, also hindering reproducibility. This issue of reproducibility related to LLM research is a well known challenge within the field (Siddiq et al., 2025). Still, by setting temperature to zero and specifying exact prompts, parameters and models used, researchers can maximize the likelihood of obtaining consistent outputs, even if perfect reproducibility cannot be guaranteed.

2.1.3 Limitations

Although LLMs are transforming how we work in many different industries by automating and facilitating otherwise heavy manual tasks, they have some notable limitations (Raza et al., 2025). LLMs sometimes produce inaccurate or nonsensical content, also known as hallucinations. The nature of hallucinations is complex and their root causes are many. One cause of hallucinations is that LLMs' knowledge is limited to what they learned during pretraining (Huang et al., 2025). Lack off, incorrect, or misleading information in the training data can lead to hallucinations. Because their knowledge is static, the models can become outdated, increasing the risk of inaccurate answers. To remain relevant, they have to be retrained frequently which is resource-intensive (Procko and Ochoa, 2024).

Today, many companies have adopted AI to improve efficiency across various areas and professions (Liang et al., 2025b; Statistikmyndigheten SCB, 2025). Yet, using LLMs in an enterprise environment comes with additional challenges. Most LLMs are trained on general datasets that include encyclopedic sources (e.g., Wikipedia), web crawl data, digitized books, and news or academic articles, giving them broad coverage and enabling them to answer questions across many domains. However, they often struggle with specialized, domain-specific questions that require detailed expertise (Peng et al., 2025). This makes general-purpose LLMs insufficient for certain use cases, particularly in niche sectors or within individual companies. To overcome this, LLMs can be fine-tuned on domain-specific data to specialize in areas not covered by general datasets, such as internal documentation, product manuals, or customer support knowledge bases. However, fine-tuning is costly, requires expertise, and depends on large amounts of high-quality data, making it impractical to implement for most companies. Additionally, enterprise data is often confidential or sensitive, which can limit the use of closed-source models. As a result, organizations may rely on private APIs or deploy open-source models on-premise. In both cases, additional mechanisms are needed to make

internal knowledge available to the LLM.

2.1.4 Deployment

In this thesis, we refer to *on-premise models* as those hosted locally within Axis, and *online models* as cloud-hosted models accessed via APIs, typically on a pay-per-token basis. Online models are often closed-source, meaning their architectures and weights are not publicly available. In contrast, open-source models make their code publicly available and can be deployed on-premise. Some are also open-weight, meaning their trained weights can be used locally even if the full training process or architecture is not disclosed.

2.1.5 Model families

Large language models can be divided into model families. Models within a family are typically developed by the same organization or research group, and share a common underlying architecture and training methodology. They can still differ in scale (e.g., number of parameters) or in whether they have been fine-tuned for specific tasks. Models in the same family are commonly pretrained on overlapping data sources, with newer versions trained on expanded or updated corpora. Families often evolve over multiple generations, with each iteration improving in performance or efficiency. In practice, families are also commonly released in multiple parameter variants (e.g., 7B/13B/70B) to support different trade-offs between capability and compute resources or per-call costs.

GPT-family

The GPT family, developed by OpenAI, represents one of the earliest series of transformer-based LLMs. GPT models have been released across multiple generations (GPT-1 to GPT-5), with early models open-source and later generations closed-source and accessible only through APIs (Minaee et al., 2025).

In this thesis, the benchmark we compare our results to uses GPT-4o-mini, a small variant of the GPT-4o family trading some reasoning ability for low-latency and low-cost inference (OpenAI, 2025). The model is proprietary so its parameter count is not publicly disclosed. It supports a context window of 128 000 tokens (128k).

Qwen-family

The Qwen family is developed by Alibaba Cloud and includes both base and instruction-tuned variants, with recent models targeting coding tasks and multilingual usage. In this thesis we use Qwen3-Coder-480B-A35B-Instruct (Team, 2025). This model is open-source and specifically targeted for coding tasks. The Qwen3-Coder-480B-A35B-Instruct is a large model of 480 billion parameters, with context capabilities of 256k tokens (Team, 2025).

Mistral-family

The Mistral family is developed by Mistral AI and emphasizes open and efficient models, often released in multiple parameter variants for deployment flexibility. In this thesis we use Mistral Medium 3 (Mistral AI, 2025b). This is a proprietary model, but it can be deployed on-premise under an enterprise licensing agreement. According to Mistral AI, the model is specifically designed for enterprise use cases (Mistral AI, 2025a) . The exact parameter count for Mistral Medium 3 has not been disclosed, but the model supports a context window of 128k tokens.

Gemini-family

The Gemini family is developed by Google DeepMind and features multimodal capabilities, with various sizes for cloud and on-device usage. In this thesis we use Gemini-2.5-flash-lite, a proprietary lightweight model optimized for fast and low-cost inference (Google Cloud, 2025). The parameter count is not disclosed, but the model has a large context window of ~ 1 million tokens.

2.2 Question-answering

In natural language processing, *question-answering* (QA) refers to the task of automatically responding to users' natural language questions (Hirschman and Gaizauskas, 2001). Within QA, a distinction is made between knowledge base question-answering (KBQA) and generative QA. KBQA retrieves specific facts or entities from a structured knowledge base, where evaluation often relies on exact matches, since there is typically only one correct answer. Generative QA on the other hand, produces free-form text answers, which are open-ended and may be paraphrased, making evaluation more challenging. Question-answering (QA) has become a widely used application of LLMs, particularly in chatbots and conversational agents.

2.3 Knowledge Graphs

Following Hogan et al. (2021), a *knowledge graph* is a form of knowledge representation that organizes unstructured data into a graph structure that captures both facts and the connections between them. Knowledge graphs are directed graphs composed of nodes and edges, or as they will be referred to more frequently in this study; *entities* and *relations*. Entities can be anything from an object, person, company or event to an abstract concept. The relations describes how two entities relate to each other. In figure 2.1 a close-up example of a knowledge graph is presented, and in figure 2.2 a more high-level example. Note that in our definition of a knowledge graph, multiple edges can exist between the same pair of entities.

Knowledge graphs differ from graph databases and plain graphs in the sense that they are often semantically enriched (Hogan et al., 2021). This means that they include semantic de-

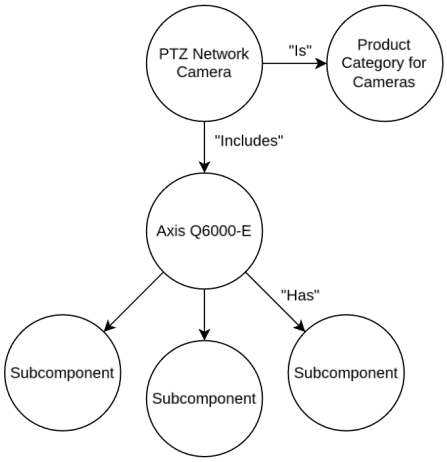


Figure 2.1: Abstract example of a knowledge graph with entities and relations.

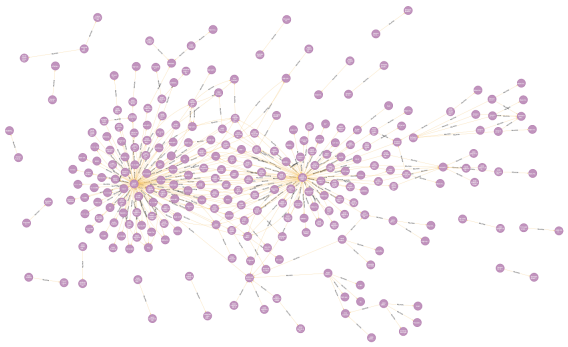


Figure 2.2: Example of one of the smaller knowledge graphs built during our thesis, rendered in Neo4j. The graph was built based on web scraped data from Axis product specification websites, the two main clusters each represents an Axis camera

scriptions of their entities and relations (Ji et al., 2022). The graphs can also include metadata, which relates each entity and relation to its origin in the source data. In our definition of a knowledge graph, no explicit ontology or schema is required, and the entities and relations do not need to have a specific type or label attached apart from their semantic description. This semantic layer makes the graph more machine-interpretable, especially for LLMs, which excel at semantic data and can query, reason, and infer new knowledge from the graph (Peng et al., 2023). At the same time, knowledge graphs can help humans to navigate and visualize complex data. They can also reveal hidden patterns and relationships that are more difficult to recognize in traditional relational databases. This can be seen in applications such as financial fraud detection and security systems (Zhou et al., 2024; Mao et al., 2022). Knowledge graphs are widely adopted in industry as enterprise knowledge graphs, with areas of application such as recommendation systems, search engines, social networks and commerce (Hogan et al., 2021). Although enterprise knowledge graphs are commonly private, there also exist open knowledge graphs that are accessible for public use, with prominent examples including Wikidata (Vrandečić and Krötzsch, 2014) and DBpedia (Lehmann et al., 2015). In later years, knowledge graphs have also proven useful for question-answering systems and information retrieval (Huang et al., 2019; Wang et al., 2024b; Yasunaga et al., 2021). Knowledge graphs can thus be used for several downstream tasks. In this thesis, we focus on their application in question-answering.

We define a triple as an ordered group of three elements of the form (subject, predicate, object). The triple represents a knowledge fact, as an example: (Albert Einstein, loved, physics). The subject and object can represent nodes, or entities, of a graph and the predicate can represent an edge, or connection, of a graph. Thus equivalently, a knowledge graph can be defined by a list of triples, which can then be transformed, and rendered, into a graph structure (Ji et al., 2022; Hogan et al., 2021). This is why *knowledge base* is sometimes used interchangeably with knowledge graph, although it is a slightly different term, referring to a set of knowledge facts or rules (Ehrlinger and Wöß, 2016). The triples, or facts, are not necessarily symmetric, hence the edges become directed in the knowledge graph (Peng et al., 2023).

While knowledge graphs have been a promising approach to tackle various knowledge retrieval tasks for a while now, there are some limitations that have hindered their adoption. Constructing knowledge graphs manually is a cumbersome process, being both complex and time-consuming (Kau et al., 2024; Zhang et al., 2025). Knowledge graphs are also specific to a domain, so generally new graphs have to be built for each area of application. They are also at risk of becoming outdated if not updated over time, so maintaining them is crucial, but doing so manually is inconvenient and can become time-consuming. This has urged researchers to find ways to automate the construction of knowledge graphs. Earlier approaches focused on entity and relation extraction using either rule-based or NLP approaches (Han et al., 2025b; Hogan et al., 2021). Different named entity recognition (NER) techniques can be used to extract entities from text, followed by entity linking to correctly map the extracted entities to existing nodes in the target graph. In addition to entity extraction, relation extraction methods are applied to identify connections between entities. However, modern approaches instead uses LLMs for NER, entity linking and relation extraction tasks, leveraging the power of LLMs to extract relations and entities directly from raw text data (Li et al., 2022; Zhang and Soh, 2024; Zhu et al., 2024). As LLMs are context-aware and highly adaptable, they are able to handle ambiguous text, infer missing information, and adapt to new domains with minimal labeled data in knowledge graph construction (Diaz-Garcia and Lopez, 2025; Hu et al., 2026). These capabilities are particularly important in enterprise contexts, where labeled data is often scarce.

2.4 Entity resolution

Entity resolution (ER) refers to finding and linking similar entities that actually refer to the same object, product, person or concept. This ensures the graph does not contain any duplicates which, if present, yields redundancy and noise in the graph. Merging similar nodes also ensures the graph is well-connected and dense, which can improve reasoning and support downstream tasks such as multi-hop reasoning.

Entity resolution is based on early work in *record linkage* (Fellegi and Sunter, 1969), which traditionally relies on rule-based methods to determine whether records refer to the same real-world entity. When machine learning models became available, they could be used to enhance ER performance by making probabilistic match decisions (Li et al., 2024b). Modern approaches rely on semantic matching and clustering methods that group entities based on meaning and filter out unlikely matches, as well as pre-trained classifier models which often require large amounts of labeled data (Li et al., 2024b).

More recently, studies have explored the possibility of using LLMs for entity matching (Li et al., 2024c; Peeters et al., 2024). This approach involves prompting the LLM to determine whether two entities correspond to the same real-world entity. Since querying each pair of entities can be both costly and time-consuming, alternative strategies have been proposed that combine the use of LLMs with established ER techniques. Semantic clustering can be used to efficiently filter candidate pairs for comparison, reducing computational cost before applying an LLM to determine whether to merge entity-pairs or not (Li et al., 2024b).

2.5 Retrieval-augmented generation

Retrieval-augmented generation (RAG) means retrieving external context to improve an LLM's responses. In this way, the LLM's knowledge can be extended to new domains. A retriever component searches a knowledge base for information relevant to the query, while a generator produces an answer using the retrieved information together with the query. These principles apply to both RAG and GraphRAG; however, standard RAG uses a document or vector index as its knowledge base, whereas GraphRAG uses a knowledge graph.

By leveraging external knowledge in this way, RAG addresses the limitations of LLMs' parametric knowledge (Lewis et al., 2020). It also allows dynamic updates of the knowledge base without retraining, and supports domain-specific applications, letting organizations integrate their internal data with LLMs. RAG also improves reliability and helps reduce hallucinations, ultimately enhancing overall LLM performance (Gao et al., 2024). The following sections first present RAG, then GraphRAG, and discuss their similarities and differences.

2.5.1 Traditional RAG

Traditional RAG (Lewis et al., 2020) involves three essential processes; *indexing*, *retrieval* and *generation* (Gao et al., 2024). The first stage of the indexing is to apply necessary prepossessing of source data to transform them into plain text data. Then the text data is split into chunks, either based on a predefined word or token length, or in a way that preserves semantic coherence. These chunks are then transformed into numerical representations as high-dimensional vector embeddings. The index is finally created by storing the embeddings in a vector database. The next stage of RAG pipeline is the retrieval process. When a user sends a query to the LLM, the query is transformed into a vector using the same embedding model as for the chunks. To find relevant information, semantic similarity search is applied between the query embedding and the chunk embeddings. The top-k most similar embeddings are identified and their corresponding chunks are retrieved. In the generation stage, the retrieved chunks are used to augment the input prompt which is then passed on to the LLM. Finally, the LLM generates a well-grounded output to the query using the chunks as supporting data.

With the help of RAG, LLMs can effectively answer focused, purely factual questions, but as queries become more challenging these systems also experience hallucinations (Huang et al., 2025). Attempts have thus been made to advance the standard approach to RAG and today there are many variations of RAG. Some developments improves the RAG's ability to handle long-form generation, summarization and reasoning through different iterative retrieval processes (Shao et al., 2023; Jiang et al., 2023). Other include integrating re-ranking to allow more information to be examined while dismissing irrelevant context (Catanzaro et al., 2024). Even though these advancements have been successful, models still struggle when it comes to broader summative questions, that require multi-step reasoning and making connections across data (Guo et al., 2025a). RAG system often fail in understanding how information in different parts of the data relate (Peng et al., 2025; Guo et al., 2025a). Because chunks are retrieved solely based on semantic similarity, RAG runs the risk of neglecting relevant infor-

mation that is not present in the input query (Peng et al., 2025). Retrieval based on semantic similarity may also result in highly similar or overlapping chunks, introducing redundancy in the retrieved context (Peng et al., 2025). This reduces the diversity of retrieved evidence and may cause other important information to be missed. Consequently, even with RAG, LLMs may have difficulty answering questions that require drawing new conclusions based on less obvious connections in the source data.

RAG systems can also be sensitive to large corpora with irrelevant noisy data (Procko and Ochoa, 2024). Companies often have a lot of data that they have collected over the years, much of which may be outdated or irrelevant in many cases. Then, RAG may not be beneficial for comprehensive usage throughout the company.

2.5.2 GraphRAG

One solution for the problems with traditional RAG is to use a different kind of knowledge base that better captures relationships in data. In recent years, knowledge graphs have been explored for this purpose.

Graph-enhanced retrieval-augmented generation (GraphRAG) is an application of RAG that uses knowledge graphs to structure and retrieve external information in order to enhance the contextual awareness and reasoning of LLMs. One of the earlier approaches to GraphRAG, was introduced by Edge et al. (2025), we will henceforth refer to their model as MS-GraphRAG, to distinguish it from the term GraphRAG which we use as umbrella term. Since then, numerous other GraphRAG frameworks have been developed to improve LLMs' performance in various tasks. Different types of KGs can be used for GraphRAG, including open knowledge graphs and enterprise knowledge graphs. However, this thesis focuses primarily on RAG systems without an existing graph, where KG construction is part of the pipeline. Existing GraphRAG frameworks differ in when and how these graphs are constructed. Some systems, e.g. MS-GraphRAG (Edge et al., 2025), HippoRAG (Gutiérrez et al., 2025) and KAG (Liang et al., 2025a), assemble graphs prior to inference, while others, e.g. G-Retriever (He et al., 2024) and LazyGraphRAG (Potts, 2024), construct graphs at query time. Regardless, these systems typically require extensive processing of the source data so that information can be represented in the nodes and edges of a graph.

As an initial step, text extracted from the source data is divided into consumable chunks (Zhang et al., 2025). Different types of information extraction approaches, e.g. named entity recognition (NER) and open information extraction, are then used to extract entities, relations or triples from these chunks. Many of today's frameworks apply these techniques using LLMs as a tool (Edge et al., 2025; Gutiérrez et al., 2025; Liang et al., 2025a; Gutiérrez et al., 2025). The extracted information is then stored, often in a graph database or as tuples in a triple store (Sarmah et al., 2024).

Different strategies are used to make graphs more organized and semantically rich. Iterative extraction techniques can be used to capture more entities and relations to ensure that no information is left out and improve graph density (Edge et al., 2025). Additionally, the LLM can be asked to extract more context from the chunks such as high-level themes or entity and relation descriptions. This makes the graph more context aware and removes ambigu-

ity. Entity resolution (see section 2.4) can be used to reduce redundancies and merge nodes to improve connectivity. Some frameworks, including MS-GraphRAG (Edge et al., 2025), ArchRAG (Wang et al., 2025) and Med-GraphRAG (Wu et al., 2024), use clustering methods and generate descriptive summaries over graph segments to create higher-level abstractions of the graph. Adding metadata, including timestamps, URLs, document titles and keywords, can also be used to enrich a graph, since it allows traceability and verification of retrieved information (Mukherjee et al., 2025; Zhang et al., 2025).

There are more retrieval techniques for GraphRAG systems than can be covered in this thesis, but they all make use of the relational nature of graphs. Approaches differ in what part or level of the graphs are retrieved, ranging from nodes to community summaries and subgraphs (Peng et al., 2025). In general, there are some strategies that are often used for the retrieval process. When a query is sent in to the system, similarity search is applied to identify segments of the graph, such as nodes, triples, subgraphs or community summaries, that relate to the input query. Different techniques for graph traversal are often used to retrieve additional context. Furthermore, LLMs can be used throughout the process to reason what context is relevant and when sufficient information has been found to answer the query.

The different frameworks are often designed for a specific purpose, use case or domain, which is why approaches may look very different. For example, HippoRAG (Gutiérrez et al., 2025) was designed to mimic the function of the hippocampus in the human brain, while Fast-GraphRAG (Microsoft, 2026) was developed as a more cost-effective and faster alternative to MS-GraphRAG by minimizing the use of LLMs throughout the pipeline and instead leveraging NLP methods.

GraphRAG has been shown to surpass traditional RAG in some use cases. MS-GraphRAG (Edge et al., 2025) enhanced the LLM's performance in global sense-making questions, HippoRAG (Gutiérrez et al., 2025) and KAG (Liang et al., 2025a) demonstrated improved retrieval for multi-hop question-answering, and G-reasoner (Luo et al., 2025), a recently published framework, made further advancements for complex reasoning tasks. Furthermore, Med-GraphRAG (Wu et al., 2024) achieved better results than both traditional RAG and GraphRAG on 9 different benchmarks for medical QA and also outperformed some domain-trained LLMs.

Although graphs may enable better reasoning in RAG systems, they are more complex to design than traditional RAG, and often require high runtime overhead and an extensive amount of LLM calls (Zhang et al., 2025). Furthermore, the use of LLMs throughout the systems might introduce latency at query-time which might make them insufficient to replace some RAG systems. Some studies indicate that RAG may still outperform GraphRAG for simple retrieval and single-hop QA (Han et al., 2025a; Xiang et al., 2025).

MS-GraphRAG

The first step of the MS-GraphRAG (Edge et al., 2025) is the chunking process, where data is split into text chunks. Then an LLM is used to extract entities and relationships while also generating descriptions and factual claims as attributes. To deduplicate entities that had been extracted multiple times during this process, exact string matching is used. Matching entities are merged into a single unique node in the graph, and their descriptions and claims are

combined. Similarly, duplicate relationships between two nodes are identified and merged into a single edge with a weight representing the number of merged relationships (Edge et al., 2025).

Next, the *Leiden algorithm* (Traag et al., 2019) is used to detect communities which represent clusters of closely connected nodes within the graph. It is a recursive process that results in a hierarchical structure with multiple community levels of different granularity (Edge et al., 2025). As the Leiden algorithm is also used in our GraphRAG system, a more detailed description of how it works is provided in subsection 3.4.5. Finally, for each community, an LLM is prompted to generate a summary. For the lowest level communities, the summaries are based on their respective nodes and edges, including related descriptions and claims. For higher level communities, summaries are generated from the aggregated properties and previously generated summaries of their corresponding lower level sub-communities.

Edge et al. (2025) presents a retrieval process that utilizes the hierarchical structure of the graph and aims to answer questions that require global understanding. When a user query is sent in to the system, the community summaries are randomly split into chunks to increase the likelihood that relevant information is included in every chunk. Multiple LLMs are run to generate intermediate answers for these chunks and simultaneously assign a score to each answer. The scores are then used to rank the intermediate answers by their usefulness. The most useful intermediate answers that can fit within a context window are provided to an LLM, which finally generates a global answer to the query. This illustrates that MS-GraphRAG requires a large number of LLM calls, making its performance and efficiency strongly dependent on the underlying LLM.

2.6 Evaluation

The evolution of RAG and GraphRAG systems has resulted in models that differ in retrieval strategies and are designed for specific applications and domains, making them challenging to evaluate (Yu et al., 2025a). In this section, we present some of the current evaluation strategies including specific metrics, benchmarks and emerging LLM-based evaluation methods.

2.6.1 Datasets

We define a *QA dataset* as a collection of question-answer pairs, often grounded in a specific knowledge base or corpus. They are often used to evaluate a system's ability to retrieve information and generate correct answers. Specifically, the reference answer, also called the ground truth, is used to compare against the model's generated answer. QA datasets can further be classified based on their reasoning complexity. Single-fact (or single-hop) QA requires only one fact to answer a query, while multi-hop QA involves reasoning over multiple segments of the knowledge base (Yang et al., 2018; Ji et al., 2022). Some datasets provide additional information, such as supporting evidence of the answer, which can be used to assess retrieval capabilities.

For evaluating RAG systems, numerous pre-constructed QA datasets are available. Both

single-hop QA datasets, such as Natural Questions (NQ) (Kwiatkowski et al., 2019), SQuAD (Rajpurkar et al., 2016) and TriviaQA (Joshi et al., 2017), and multi-hop QA datasets such as HotPotQA (Yang et al., 2018), 2WikiMultiHopQA (Ho et al., 2020), and MultiHop-RAG (Tang and Yang, 2024) are frequently applied (Gao et al., 2024). However, finding an appropriate QA dataset for evaluation remains a challenge. Many existing QA datasets are created from publicly available data, such as Wikipedia data or news articles (Xiang et al., 2025). This data can overlap with the training corpora of the LLM, making it difficult to determine whether a generated answer comes from the retrieval rather than the model's parametric knowledge. Using datasets that are likely unseen by the LLM, or domain-specific datasets, may therefore lead to a more reliable evaluation.

Still, for some organizations, suitable QA datasets may not exist within their specific domain. Companies might want to evaluate system performance on company-specific questions to ensure quality. Therefore, some organizations manually create their own QA datasets to assess their system's performance.

2.6.2 Synthetic data generation

The process of manually constructing a QA dataset can be time-consuming and resource-intensive (Wissler et al., 2014; Schmidt et al., 2024), especially when large datasets are required for complete evaluation. To overcome these challenges, alternative approaches have emerged using LLMs to synthetically generate QA pairs (Lu et al., 2025; Puri et al., 2020; Schmidt et al., 2024). In NLP research, automatically or semi-automatically generated datasets are often referred to as silver standard, while manually annotated datasets are referred to as gold standard and can be considered highly reliable and accurate (Wissler et al., 2014). Acquiring a gold standard QA dataset can also be especially challenging in closed-domain applications, such as enterprise settings (Li et al., 2024d). Manually annotating question–answer pairs might require involvement of multiple teams or become a tedious process of repeatedly checking source documents. When time, resources, or domain expertise are limited, synthetically generated datasets can be a practical alternative.

2.6.3 Evaluation metrics

Given a QA dataset, RAG systems can be evaluated either using automatic metrics or through human and LLM-as-a-judge assessment (Brown et al., 2025). Although human evaluation is often considered reliable, it can be time-consuming and resource-intensive, making it impractical for evaluation at larger scales. Therefore, we focus on automatic evaluation methods, first discussing common statistical metrics and then describing LLM-as-a-judge evaluation.

Early evaluation methods for RAG systems often focused on end-to-end performance, evaluating only the quality of the generated answers. However, this approach neglects the complexity of RAG systems, since the final output depends not only on the generation component but also on the retrieval component and the quality of the knowledge base. When relying solely on end-to-end evaluation, it is difficult to identify potential flaws or bottlenecks. To

address this, the evaluation is often separated into two parts: *retrieval evaluation* and *generation evaluation* (Yu et al., 2025a; Peng et al., 2025). For GraphRAG systems, the evaluation can be further extended to include *graph evaluation*. In the following sections, we describe some common metrics used in these evaluations.

Generation evaluation

The generation is typically evaluated based on the accuracy, relevance and faithfulness of the generated answers. Accuracy and relevance can be assessed by comparing the generated and the ground truth answers using metrics such as *Exact Match (EM)* or *F1 score* (Brown et al., 2025). *Semantic similarity* between the answers can also be measured, using embedding based metrics.

Other metrics evaluate lexical overlap, including *ROUGE-L* (Lin, 2004) which denotes the longest common subsequence between two sequences. A score of 1 indicates that the answers are identical, while a score of 0 indicates no lexical overlap.

Faithfulness measures the extent to which the generated answer is grounded in the retrieved information. Low faithfulness suggests that the LLM introduced information based on its parametric knowledge, which may indicate hallucination. Faithfulness can be evaluated using semantic similarity metrics such as *BERTScore* (Zhang et al., 2020).

Retrieval evaluation

The retriever is commonly evaluated based on how effectively the system retrieves relevant information.

$$\text{Recall@k} = \frac{\text{relevant retrieved components in top-k}}{\text{all relevant components}} \tag{2.1}$$

Additionally, *Precision@k* can be used to measure the proportion of retrieved components in the top-k that are actually relevant to the query (Brown et al., 2025). Recall@k reveals the completeness of the retrieval while Precision@k measures the accuracy of the retrieved results.

Graph evaluation

The quality of the graph is evaluated through the properties of the graph structure that reflect the coverage, connectivity, and coherence of the knowledge graph. Node and edge count is a typical metric of the graph. Higher node counts can indicate broader domain coverage of the knowledge base, while high edge count imply connectivity in the graph (Xiang et al., 2025). Connectivity can also be measured by the *Average degree* which refers to the average number of edges per node:

$$\text{Average Degree} = \frac{1}{|V|} \sum_{v \in V} \deg(v), \tag{2.2}$$

where V is the set of nodes in the graph and $\deg(v)$ is the number of edges of node v (Xiang et al., 2025).

The coherence of the graph can be measured by calculating the *Average clustering coefficient* which is the probability that two neighbors of a node are also connected to each other. When two neighbors of a node are connected, the three nodes form a triangle.

For a given node v, the local clustering coefficient $C(v)$ is the ratio between the number of edges that exist between its neighbors and the number of edges that could exist among them. It is defined as:

$$C(v) = \frac{2 \cdot T(v)}{\deg(v) \cdot (\deg(v) - 1)}, \tag{2.3}$$

where $T(v)$ is the number of triangles centered at node v, and $\deg(v)$ is its total number of edges (Xiang et al., 2025).

Averaging $C(v)$ over all nodes V is then:

$$\text{Average Clustering Coefficient} = \frac{1}{|V|} \sum_{v \in V} C(v), \tag{2.4}$$

which gives the average clustering coefficient of the graph (Xiang et al., 2025).

2.6.4 LLM-as-a-judge evaluation

Traditional automatic metrics provide standardized and objective ways to evaluate RAG systems but are often insufficient for evaluating open-ended generative tasks, where lexical overlap does not reliably reflect accuracy (Li et al., 2024a). Human evaluation can provide richer judgments but is costly and time-consuming at larger scales (Gu et al., 2026; Li et al., 2024a; Zheng et al., 2023). It is also susceptible to inconsistencies, as results can vary between annotators (Gu et al., 2026). To address these limitations, recent work leverages large language models (LLMs) for evaluation (Es et al., 2024; Tang and Yang, 2024; Xiang et al., 2025). This paradigm, known as *LLM-as-a-judge*, involves using LLMs to assess answers and assign scores or semantic judgments based on prompts or examples, potentially complementing or replacing human judges and statistical metrics (Li et al., 2024a; Zheng et al., 2023). LLM-based evaluation offers a scalable alternative that balances human judgment and automatic evaluation metrics.

LLM-as-a-judge approaches are typically categorized as *reference-based* or *reference-free*. Reference-based evaluation provides the judge with a reference answer, while reference-free evaluation relies on the model's parametric knowledge or externally supplied context (Li et al., 2024a).

Reference-free approaches are better suited for open-ended or dynamic tasks, whereas reference-based methods are appropriate when objective answers exist. However, reference-based evaluation can introduce bias if the reference answer quality is low or if the judge adheres too strongly to the reference, a behavior highly influenced by prompting (Li et al., 2024a).

LLM-based judges show agreement with human evaluators at levels comparable to human–human agreement (Zheng et al., 2023). They are particularly useful for open-ended and complex evaluation tasks, where they can leverage contextual understanding to provide more nuanced and comprehensive assessments than traditional automatic metrics (Gu et al., 2026). LLMs can be used to evaluate both the retrieval and the generation components of RAG systems. For instance, they can assess the context relevance by scoring whether the retrieved information is relevant to the query. For generation evaluation, it can similarly assess accuracy and faithfulness (Es et al., 2024).

Limitations

While LLM-based assessment offers significant advantages, the reliability of LLM-based assessment is still challenged by hallucinations, prompt sensitivity, and systematic biases. Minor prompt changes can substantially affect evaluation outcomes, limiting reproducibility and robustness (Sclar et al., 2024). LLMs may also produce confident but incorrect judgments, undermining trustworthiness (Gu et al., 2026). Common biases include verbosity bias, which favors longer responses, and position bias, which favors answers based on their placement rather than content (Zheng et al., 2023). However, position bias can be mitigated by randomizing answer order across evaluations (Zheng et al., 2023).

2.6.5 Benchmarks

A benchmark is a resource that can be used to evaluate and compare the performance of models under consistent conditions (Koch et al., 2021). They usually consist of a dataset and a set of evaluation metrics, allowing researchers to track progress in the field, identify common weaknesses, and compare models accurately.

Some benchmarks focus only on end-to-end evaluation, while other more comprehensive benchmarks, such as MultiHop-RAG (Tang and Yang, 2024), assess the retriever and generation components separately. Traditional automatic metrics are commonly used, however recently, some benchmarks such as GraphRAG-Bench (Xiang et al., 2025), RAGAs (Es et al., 2024) and ARES (Saad-Falcon et al., 2024) have adopted LLM-as-a-judge.

For GraphRAG systems, most evaluations still rely on conventional RAG benchmarks, as standardized benchmarking for GraphRAG is limited (Peng et al., 2025). Particularly multi-hop datasets are often used, as they better highlight the advantages of graph-based reasoning over traditional RAG. In recent years, a few benchmarks that specifically target GraphRAG systems have emerged, for example GraphRAG-Bench (Xiang et al., 2025). Peng et al. (2025) emphasize that establishing a standard benchmark is crucial to provide a consistent framework for comparing different GraphRAG systems and to make advancements in the field.

Chapter 3

Method

3.1 RAG system design

Figure 3.1 shows an overview of our RAG system. Given a user query, a retriever searches the knowledge base for relevant documents. The retrieved context is then combined with the query and passed to the LLM, which generates the final answer returned to the user.

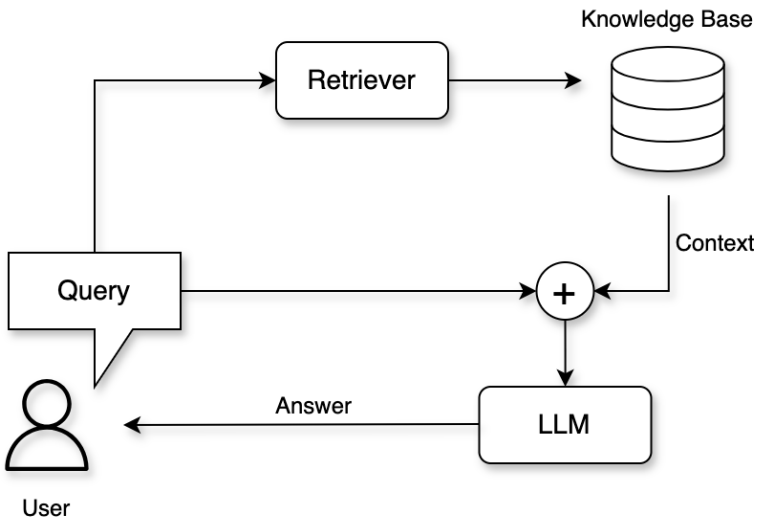


Figure 3.1: Overview of a RAG system.

For this thesis, we built both a RAG and a GraphRAG pipeline. This system architecture is

shared by both pipelines, however, the type of knowledge base and the retrieval strategy differ. The RAG pipeline uses a vector database as its knowledge base, whereas GraphRAG uses a knowledge graph. The following section presents an overview of the RAG and GraphRAG pipelines and their knowledge base construction.

3.1.1 Overview: RAG pipeline

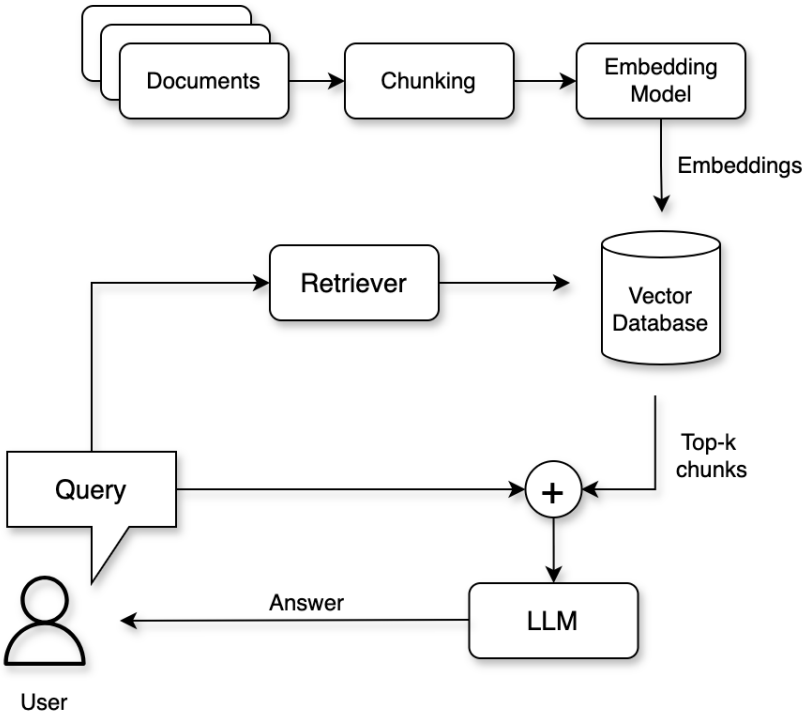


Figure 3.2: Overview of our RAG pipeline.

An overview of our RAG pipeline is provided in figure 3.2. The documents here represent the data source, consisting of unstructured text data. This data is chunked into smaller sections, which are embedded and stored in a vector database. When a user submits a query to the LLM, the query is also embedded, and similarity search is performed over the stored chunks. The top-k chunks that matches the query is retrieved and provided to the LLM as context together with the original query. The LLM then generates a final response based on both the retrieved content and the query.

3.1.2 Overview: GraphRAG pipeline

The different stages of the GraphRAG pipeline are outlined in figure 3.4, describing how raw text data is transformed into a structured knowledge graph and subsequently used for RAG with an LLM.

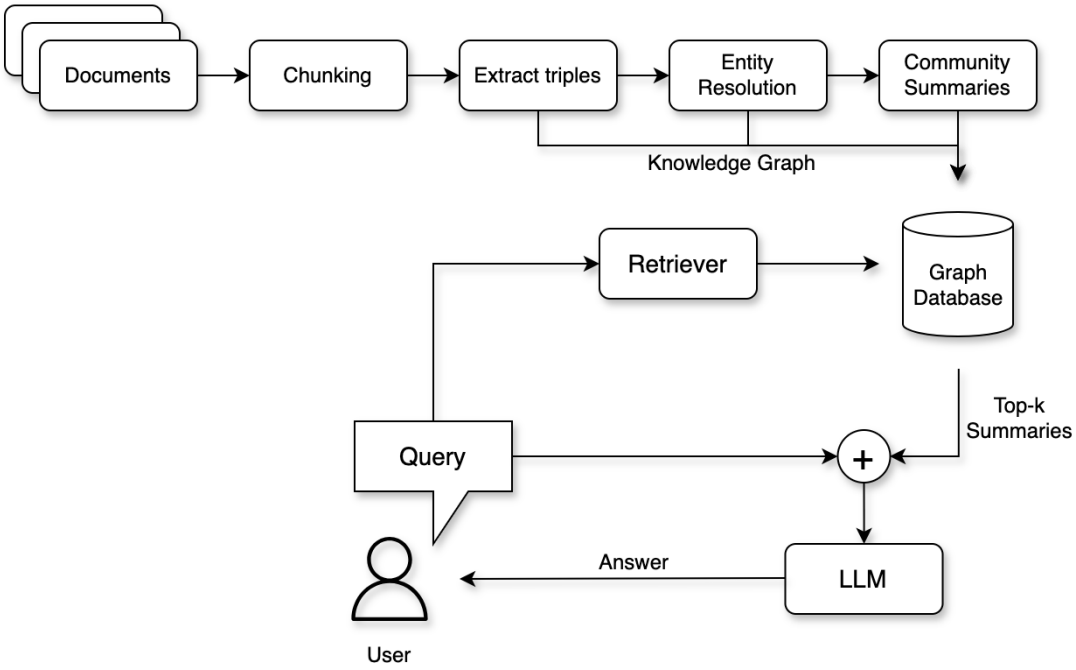


Figure 3.3: Overview of our GraphRAG pipeline.

First, documents of text data are divided into smaller chunks. From each chunk, semantic triples are extracted and uploaded as nodes and edges into a graph database. Triples are statements of the form (subject, predicate, object), such as ('Axis', 'sells', 'cameras'). Entity merging is then performed to remove duplicates and create a more coherent and densely connected graph. Finally, community summaries are generated. A community is a cluster of nodes and edges in the graph. Its community summary is a natural-language description generated by an LLM that synthesizes the main entities, relationships, and textual descriptions within the cluster. Communities exist at different hierarchical levels: lower-level summaries describe individual clusters, while higher-level summaries aggregate other community summaries, providing a broader, high-level explanation of the graph. This process moves from fine-grained detail to increasingly abstract, zoomed-out descriptions of the graph. Both the triple extraction and community generation steps are performed using LLMs.

At this point, the text data has been transformed into a knowledge graph representing entities and and relations within the data. The RAG system then leverages this graph as a knowledge base for the LLM. When a user submits a query, the retriever searches the graph for relevant communities, and the top-k summaries are combined with the original query and fed into the LLM. In the same way as for RAG, the LLM then generates a response based on both the retrieved content and the query.

3.2 Datasets

This section presents the datasets explored during this thesis. Only a subset was used for evaluation, while the remaining datasets were used during pipeline development. Most datasets consisted of both a corpus and a QA dataset. However, the Axis dataset did not contain a QA dataset, which was addressed by synthetically generating one.

MultiHop-RAG and podcast transcripts Edge et al. (2025) evaluated their GraphRAG framework on two different datasets in the 1 million tokens range. One of them was MultiHop-RAG which was produced by Tang and Yang (2024) and consisted of news articles. The other dataset was built from podcast transcripts and can be found on Microsoft (2026). Edge et al. (2025) used an LLM to generate "global sense making questions" which they use to evaluate their RAG systems.

HotpotQA HotpotQA is among the most commonly used datasets for multi-hop question answering (Yang et al., 2018). It included a corpus based on Wikipedia articles and QA pairs that require complex reasoning over multiple documents. It was meant for benchmarking QA systems in general and not specifically RAG systems.

GraphRAG-Bench datasets GraphRAG-Bench introduced two new datasets which they call the *Medical dataset* and the *Novel dataset*. The Medical dataset was constructed from the National Comprehensive Cancer Networks (NCCN) clinical guidelines, including text describing standardized treatment protocols, drug interaction hierarchies, and diagnostic criteria (Xiang et al., 2025). The Novel dataset consisted of pre-20th-century novels from the Gutenberg library (Xiang et al., 2025). Xiang et al. (2025) argued that many existing datasets, often based on publicly available Wikipedia or news articles, are not optimal for testing GraphRAG systems. To support this, they conducted a statistical analysis of three popular benchmarks, including MultiHop-RAG and HotpotQA. From this analysis, they concluded that their own two datasets provided denser and richer graphs, making them more suitable for testing GraphRAG systems (See Appendix E in Xiang et al. (2025)). According to the paper, previous datasets do not fully test the core strength of GraphRAG systems, which lies in their ability to leverage domain hierarchies (Xiang et al., 2025). By 'leverage domain hierarchies', we mean using the structured relationships between concepts in a domain to improve retrieval and multi-hop reasoning.

Because our GraphRAG pipeline is inspired by the implementation of Edge et al. (2025), we initially planned to use the same benchmark and developed large parts of our pipeline using their datasets. However, a few months into the project we decided to revise our evaluation strategy. Since most LLMs are trained on general knowledge and publicly available data, evaluating RAG systems on domain-specific datasets can potentially provide a more accurate assessment of retrieval performance. They also better reflect enterprise use cases, as their data is often private and proprietary, not previously seen by LLMs (Wang et al., 2024a). These findings led us to switch course and instead use the GraphRAG-Bench dataset presented in

this paper (Xiang et al., 2025). For these reasons, we decided on adopting the GraphRAG-Bench for our experiments.

3.2.1 Evaluation datasets

Public Dataset

As our public dataset, we used the Medical dataset from the GraphRAG-Bench (Xiang et al., 2025). We chose the Medical dataset since it was smaller than the Novel dataset, which allowed for faster graph construction and iteration. The Medical text corpus contained approximately 220k tokens, while the Novel text corpus contained approximately 1.1M tokens. The size of the corpora were not stated in the original benchmark paper (Xiang et al., 2025), so we estimated the corpus size using the TikToken tokenizer (OpenAI, 2025). The Medical dataset contained both a text corpus as well as a QA dataset.

Private Dataset

The dataset provided by Axis consisted of web-scraped text data from their internal HR-related websites. The data was provided in markdown files (.md), which included textual content on topics such as employee benefits or internal company guidelines. This dataset was relatively small, consisting of around 23k tokens. We chose a smaller dataset for the Axis data to perform a qualitative analysis of the responses and generated QA dataset, allowing us to manually review all the data and judge the model's outputs ourselves. This dataset did not have any attached questions and answers, consequently the QA dataset was generated synthetically in a later stage.

We were also given access to product data for the cameras, as well as camera support data. However, these datasets were also web-scraped and of lower quality as the markdown files were more difficult to interpret, even for a human. We ran tests on this data, but the LLM performed poorly. Both datasets would require more preprocessing to make them sensible for the LLM, so we chose to focus on the HR dataset instead. This decision was also guided by the fact that the HR data was easier to understand and more suitable for human evaluation.

3.2.2 Synthetic QA data Generation

We wanted to experiment with some different types of question-answer generation, so we generated three different QA datasets using three different prompting strategies:

1. **Answer given**
2. **No answer given, single fact**
3. **No answer given, multiple facts**

Answer given

Some synthetic question generation approaches use an "answer-given" strategy where entities are extracted from the source text and supplied to the LLM as answers, prompting the model to generate corresponding questions (Puri et al., 2020; Schmidt et al., 2024). Inspired by this, we used the spaCy library's named entity recognition (NER) (Explosion AI, 2023) to extract key entities from the source documents. This approach was chosen for its simplicity, as it requires no task-specific training. Duplicate entities were removed, and for each document, five randomly selected entities were used to generate questions. The HR-data corpus contained 35 documents, and the LLM was prompted to generate a maximum of three questions per document to limit the number of questions and reduce run-time.

No answer given, single fact

For the single-fact questions, the LLM was zero-shot prompted to generate three QA pairs per document without being provided an answer. The prompt required the questions to be specific and answerable using the source text. This zero-shot prompting approach was inspired by (Takahashi et al., 2023).

No answer given, multiple facts

Finally, we generated questions requiring reasoning over multiple facts. The LLM was prompted to create three QA pairs per document, each combining at least two different facts from the source text.

All prompts can be found in Appendix B.

3.3 Language Models

We use LLMs in several parts of the pipeline. First in the triple extraction, then in the entity resolution step, then for generating the community summaries, and lastly in the evaluation step for LLM-as-a-judge assessment. We had access to both online and on-premise models from Axis, which could be accessed using a web-API. The on-premise API endpoint follows the OpenAI API format. The two available on-premise models were Mistral Medium 3 (Mistral AI, 2025b) and Qwen3-Coder-480B-A35B-Instruct (Team, 2025). As we experimented a lot during the process of building the pipeline, we decided not to use online models, since the ones we had access to employed a pay per-call pricing model, and could therefore quickly become expensive. This was also due to our pipeline being call-intensive and we wanted to be able to freely experiment and not accidentally cause too much expenses. Comparing the two on-premise models was a bit tricky, as the latter one is intended for coding-specific tasks, making it mostly evaluated on coding specific benchmarks. We were unable to find any scientific papers that compares these two models on the same benchmark. However, we found two online sources indicating that Qwen3-Coder generally scores higher across several benchmarks (LLM Stats, 2026; Qwen Team, 2025; Mistral AI, 2025c), hence we decided to use the Qwen3-Coder model for our RAG pipeline. Qwen3-coder was also preferred because the Mistral model was more widely used at the company and under a higher load, making the

Qwen3 API a more stable choice. For the LLM-as-a-judge evaluation, three different LLM models were used, see subsection 3.9.1.

To ensure reproducibility and allow for consistent measurement of model improvements, the temperature of the LLM was set to 0 across all experiments. During benchmark testing, however, a temperature of 0.7 was used for response generation to replicate the setup described in the benchmark study.

The LLM API enforced a rate limit of 20 requests per minute and a 60 second timeout. These were very restrictive considering the fact that our pipeline required thousands of API calls across the triple extraction, entity resolution, community summarization and generation step alone. When evaluating our systems, these limitations were even more difficult to comply with since the evaluation framework provided by the benchmark required an even higher number of requests. Some of the evaluation queries were also more demanding, leading to frequent API timeouts.

3.4 Constructing the knowledge graph

The following sections will explain the knowledge graph construction process in detail. An overview of the different stages can be seen in figure 3.4.

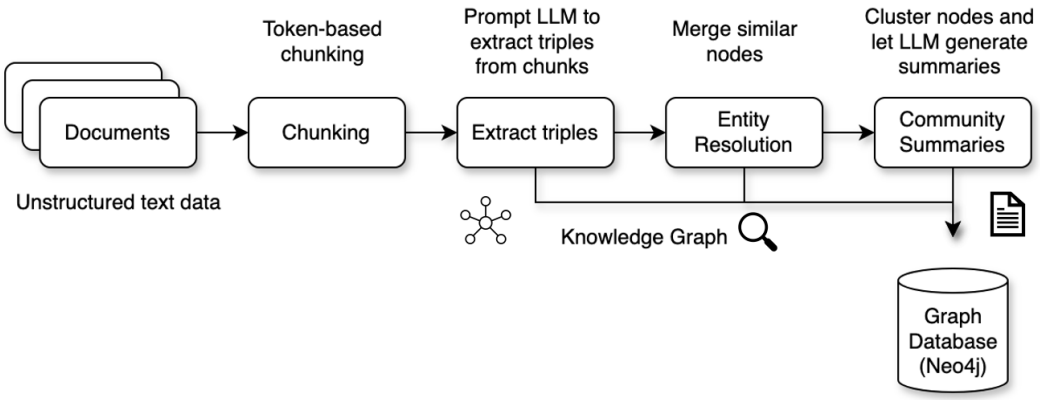


Figure 3.4: Stages of constructing the knowledge graph.

3.4.1 Text Chunking

The first step of the pipeline is the chunking mechanism. We experimented with several chunking methods including word-based chunking, token-based chunking and semantic-based chunking. The text corpus in the medical dataset was provided as a single unsegmented text sequence. Some GraphRAG frameworks that use semantic chunking first divide the text based on document structure and then they use a language model to decide if a chunk is semantically coherent (Wu et al., 2024; Liang et al., 2025a). In our case, due to the lack of clear document, section, or paragraph delimiters and to minimize overall LLM usage, we decided

to avoid semantic chunking. We chose to use token-based chunking as it is the method used in Edge et al. (2025) and provides more consistent chunk sizes.

The text corpus was split into smaller chunks of an equal number of tokens including an overlap. We initially chose to use a chunk size of 600 tokens with an overlap of 100 tokens, inspired by (Edge et al., 2025). Each chunk was indexed and enriched with metadata, such as chunk and document indices when available, to ensure traceability.

3.4.2 Entity and Relation Extraction

After the data has been chunked, each chunk is sent into an LLM which is prompted to extract triples from the chunk at hand. The full system and user prompt can be found in Appendix B. The LLM is instructed to return the triples in a list of json objects on the format $\{s$: subject, r : relation, o : object, s_{desc} : opt. desc., r_{desc} : opt. desc., o_{desc} : opt. desc.$\}$. The LLM is prompted to add descriptions of the subject, relation or object, if it deems it to be necessary. The response from the LLM is parsed as json objects, and each extracted triple is enriched with metadata of which chunk and document it was extracted from. An example triple can be seen in Appendix C.

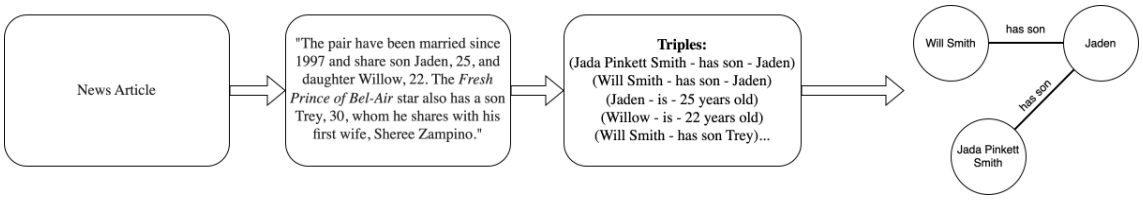


Figure 3.5: Example of how triples are extracted from data chunks and constructed into a subgraph of nodes and edges.

3.4.3 Upload triples to the graph database

A list of triples can be regarded as a graph structure in itself. However, for efficient querying, modification, and visualization, we stored the triples in a graph database. We used Neo4j Community Edition (version 2025.09.0), which is a free, self-hosted version of Neo4j. We did not use any commercial or enterprise extensions. The Neo4j Graph Data Science (GDS) library extension was used for community detection and graph analytics, allowing these steps to run directly inside Neo4j. The database was hosted locally during development.

The triples were inserted into Neo4j using Cypher queries. Nodes and relationships were assigned labels, descriptions (optional), and internal IDs, and additional metadata was stored as node properties to retain provenance from the source documents. Neo4j automatically assigns an internal ID to each node and relationship.

Metadata properties varied across datasets, but chunk_id, document_id, and a within-document chunk_index were always included. Depending on the dataset, additional fields such as URLs, file names, or article titles were also stored. Nodes also stored an embedding property and a community property, containing a list of community IDs indicating which communities the

node belonged to. Community nodes were assigned a summary, an ID, a level indicator, and an embedding property.

3.4.4 Entity resolution

To reduce redundancy and improve graph coherence, we applied entity resolution (ER). This was crucial since the LLM-extracted entities showed large variation in spelling, abbreviations, and naming conventions, creating ambiguity about which mentions referred to the same real-world entity.

We considered two main approaches for identifying mergeable node pairs: (1) comparing their semantic similarity, and (2) prompting an LLM to determine whether two nodes refer to the same entity. Comparing all node pairs grows quadratically, so using an LLM for pairwise comparison does not scale due to the large number of time-consuming LLM calls required (Papadakis et al., 2020). To reduce the number of candidate pairs, we applied filtering to discard semantically dissimilar pairs prior to LLM comparisons.

As a first filtering step, we generated an embedding for each node and stored it as a node property. We then applied a k-Nearest Neighbor (k-NN) search based on cosine similarity. For each node, the algorithm compared its embedding to all others; pairs with a similarity above the threshold (0.95) were retained as candidate matches, while isolated nodes were discarded. This acts as an efficient pre-filter, reducing the candidate space before applying more computationally expensive validation steps.

From these similarity-based candidate pairs, we applied a weakly connected components (WCC) algorithm to capture transitive similarities. In this step, nodes connected through similarity chains (e.g., $A \sim B \sim C$) are placed in the same component, allowing us to also consider transitive pairs such as $A \sim C$ as candidates, even if they did not exceed the similarity threshold directly. This prevents relevant pairs from being missed due to the strict similarity threshold.

As a second filtering step, we applied word-level filtering by computing the Levenshtein distance between all candidate pairs. Levenshtein distance measures the edit distance between two strings, defined as the minimum number of single-character insertions, deletions, or substitutions required to transform one string into another (Miller et al., 2009). Pairs with a distance greater than three were discarded.

This idea of filtering the number of comparisons in ER is inspired by prior work proposing blocking and clustering techniques, often combined with embeddings, to avoid unnecessary entity comparisons (Zheng et al., 2025; Papadakis et al., 2020; Christophides et al., 2020). We also drew inspiration from Neo4j's MS-GraphRAG implementation (Bratanič, 2024b,a).

Entity merging

The remaining candidate pairs were then evaluated by an LLM, which determined whether two nodes should be treated as duplicates and merged into a single node. This can be seen as an LLM-based disambiguation step, where the LLM links entities by determining whether

two nodes refer to the same real-world entity. When nodes were merged, a single entity label was retained, and the surviving node inherited all relations from the removed node, with duplicate relations omitted when they referred to the same relation label and target node. This approach of using LLMs for entity merging was inspired by prior work on LLM-based entity matching (Peeters et al., 2024; Li et al., 2024b).

Note on duplicate relations

We did not perform any record linkage or merging of relations in the graph. The reason for this is that, in later stages of the pipeline, the graph is clustered using the Leiden algorithm, which primarily groups nodes based on their connectivity. Each relation receives a weight, so a higher number of relations between two nodes represents a stronger connection. If the same or similar relation appears multiple times across different chunks or in slightly different forms, this possibly indicates higher significance, and we therefore decided to retain these repetitions to preserve the weighting structure during clustering.

3.4.5 Creating communities and summaries

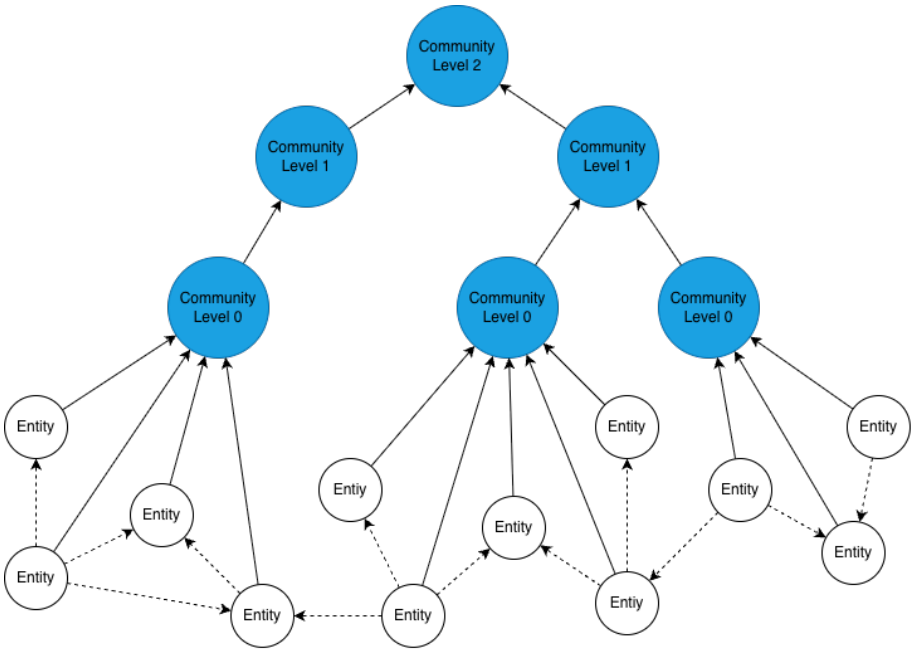


Figure 3.6: Conceptual figure of community hierarchy in our GraphRAG pipeline.

Similar to MS-GraphRAG, we created a hierarchy of communities for our knowledge graph. As we use Neo4j as our graph database, we could utilize its Graph Data Science (GDS) library, which contains the necessary functions to create a hierarchical community network. First, we made a projection of the graph, where all relationships between two nodes were merged

into one single edge with a weight based on the number of relationships between those nodes. These weights were then taken into consideration when creating the communites.

Leiden algorithm

In alignment with MS-GraphRAG, we used the Leiden algorithm (Traag et al., 2019) to detect clusters, also called communities, within the knowledge graph. The algorithm iteratively partitions the graph into communities based on node connectivity using the edge weights. The resulting communities form the lowest level communities in the hierarchy. These are then aggregated to create community nodes which are again partitioned into communities to create the next level of the hierarchy. This process is repeated recursively until no more community partitions can be made. The resulting structure is a hierarchical network of communities.

We then added the communities into the original graph as nodes with an ID and their level as properties. Each entity node was connected via an "IN_COMMUNITY" edge to the lowest level community to which it belonged. The hierarchy of the communities was also represented with "IN_COMMUNITY" edges directed from lower-level communities to higher-level communities.

Creating community summaries

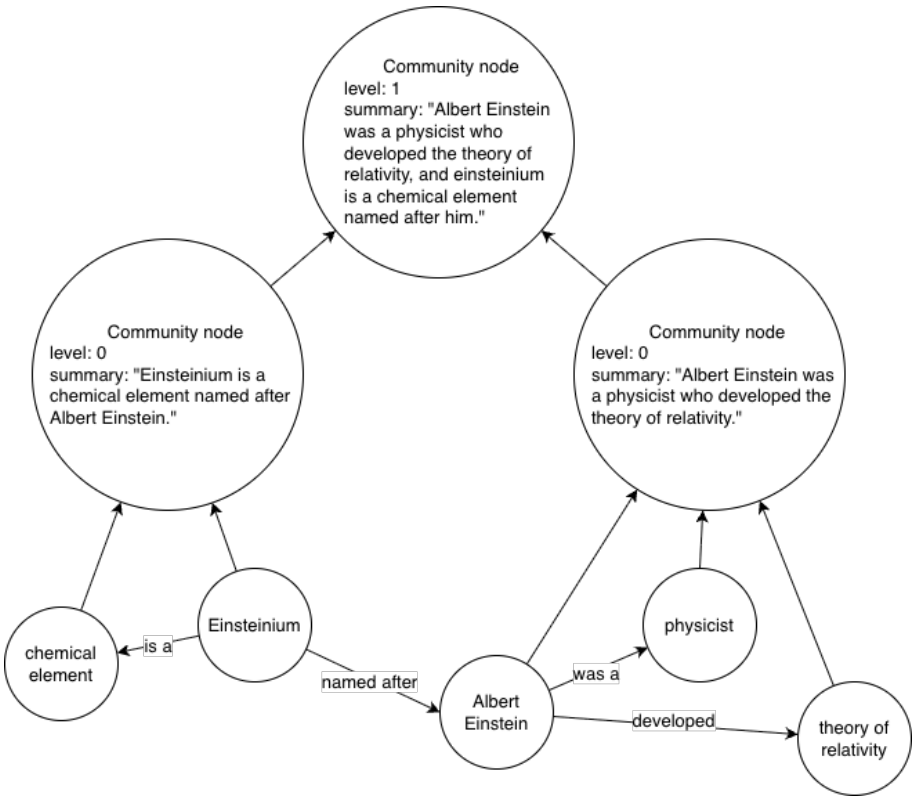


Figure 3.7: Example of how community summaries are created.

Once the communities had been generated, we created a summary for each community with the help of an LLM. Figure 3.7 illustrates this process on a small subgraph, where triples are first summarized into level-0 community nodes, which are then recursively summarized into higher-level community nodes. We constructed a prompt containing task instructions as well as the information on which the summary would be based. For the lowest level communities, summaries were generated based on the triples (including entity and relationship descriptions) of their connected entity nodes. These summaries were then used as grounding information for the next higher level community summaries. This process continued for all hierarchy levels. If a community had only one connection to a lower-level community, the summary was inherited since no additional information had been added.

3.5 RAG

We implemented a traditional RAG pipeline according to our definition. The same chunking method we did for the GraphRAG was also used for RAG, i.e. token-based chunking. This also aligned with the chunking they used for the RAG system in the benchmark. We used the same chunk size as for GraphRAG of 600 tokens and 100 token overlap, to replicate the benchmarks configuration we also tested with 256 tokens per chunk and no overlap. To embed the chunks we initially used Google Cloud's Vertex AI text-embeddings-004 embedding model (Google Cloud, 2024), but also tested with bge-large-en-v1.5 for reproducing the benchmark (Beijing Academy of Artificial Intelligence, 2024). To save our embeddings we use Chroma as a local vector database (LangChain Community, 2025). For the retrieval step of our RAG pipeline, the query was encoded into a vector embedding, in the same way as the chunk embeddings. Then we perform cosine similarity search between the query embedding and the chunk embeddings in the vector database. We retrieve the top-5 chunks which are added to a prompt with the input query. Finally, the prompt is sent to the LLM to generate an answer.

3.6 Graph-Retrieval

We experimented with different methods for the GraphRAG retriever. Initially, before the community summaries was added to the graph, we simply embedded the triples and the query and retrieved the top-k triple matches. We also tried out a different approach of extracting keywords from the prompt, and then searching for them in the graph, retrieving the matching entity as well as its neighboring nodes and relations. However, with both of these approaches, we saw that the retriever performed way worse then the traditional RAG did, according to our own implemented LLM-as-a-judge evaluation script.

When we later applied hierarchical communities to the graph, we implemented a retriever based on community summaries. Since we wanted the retriever to be as efficient as the retriever in traditional RAG, we wanted to limit the API call per query to one. This was achieved by first embedding the community summaries on all levels, as well as the query, and then performing similarity search and retrieving the top-5 most relevant community sum-

maries. We did the similarity search over all community levels since summaries on different levels might capture different relationships. The top-5 community summaries were given as context to the LLM which could then generate an answer to the query.

3.7 Evaluation overview

We evaluated our RAG systems in two different settings. First, we assessed our models' performance using GraphRAG-Bench (Xiang et al., 2025), enabling comparison against other leading GraphRAG models. Secondly, we evaluated our systems on private Axis data using LLM-as-a-judge, followed by a qualitative analysis, assessing the accuracy of the judges' responses and the quality of the synthetically generated QA datasets. The qualitative analysis aimed to validate the reliability of our LLM-based assessment and synthetic QA data generation, which we identified as important components for deploying closed-domain RAG systems at scale in corporate settings.

3.8 Benchmark evaluation

We evaluated our system using the GraphRAG-Bench framework (Xiang et al., 2025), covering generation, retrieval, and graph evaluations. Generation evaluation measures end-to-end answer accuracy, retrieval evaluation assesses how well the retrieved knowledge matches the query, and graph evaluation examines structural properties of the graph such as connectivity and size. Collectively, these evaluations cover multiple components of the GraphRAG pipeline. We conducted two runs with different embedding models and hyperparameter settings, referred to as Experiment 1 and Experiment 2 (see Table 3.1 for exact configurations). Experiment 2 uses the same configuration as the benchmark, whereas Experiment 1 uses a different embedding model and hyper-parameters. Benchmarking was performed using the GraphRAG-Benchmark GitHub repository (commit 5ec02ce, retrieved 2025-12-09), with minor modifications to enable execution (GraphRAG-Bench, 2025).

3.8.1 Benchmark tasks

The benchmark consists of four types of tasks (questions) of increasing complexity (Xiang et al., 2025):

- *Fact Retrieval* is used to evaluate the systems performance on purely factual questions, whose answers can be found in a single passage of the text data.
- *Complex Reasoning* tests the ability to handle difficult questions that require the system to make connections over different parts of the data and use multi-hop reasoning.
- *Contextual Summarization* requires synthesizing hierarchical relationships from context to generate a coherent summary.

- *Creative Generation* assesses the system's ability to generate lengthy and insightful content for a specific hypothetical situation. In some cases, the systems are also asked to generate content in a specified style or perspective.

All four tasks are covered in the medical QA dataset provided by the benchmark. It contains 2062 questions in total; 1098 for fact retrieval, 509 for complex reasoning, 289 for contextual summarization and 166 for creative generation. The questions are accompanied by a question ID, question type, ground-truth answer, and supporting evidence sentences, which is extracted factual statements from the source data which can be used to evaluate retrieval. The questions, answers and supporting evidence are automatically generated using an LLM from the medical dataset, and validated to ensure that the evidence supports the answer.

3.8.2 Evaluation metrics

The benchmark provides an evaluation framework which we used to evaluate our GraphRAG system across retrieval, generation, and graph quality. Most of the metrics were computed with the assistance of an LLM in a few-shot prompting setup to determine relevance or support. Specifically, for generation metrics that require factual assessment, the LLM extracts "statements" from both generated and ground-truth answers, which are then classified to compute metrics such as answer accuracy, evidence coverage, and faithfulness.

Generation evaluation

ROUGE-L is used to measure the lexical overlap between the generated and the ground-truth answer, as described in subsection 2.6.3.

Answer accuracy (Acc) combines semantic similarity and factual precision. An LLM is used to classify statements from the generated answer as true positive (TP), false positive (FP) or false negative (FN) with respect to the ground truth. The factual correctness is then computed as the F1 score of these:

$$F_1 = 2 \cdot \frac{TP}{TP + FP + FN} \tag{3.1}$$

The final answer accuracy is calculated as a weighted combination of the two scores:

$$\text{Acc} = \alpha \cdot F_1 + (1 - \alpha) \cdot SS \tag{3.2}$$

where $SS = \cos(f_i, c_j)$ is the semantic similarity between the embeddings of the generated answer f_i and the ground truth answer c_j. $\alpha = 0.5$ is the applied weight.

Evidence coverage (Cov) reflects how much of the ground truth answer is covered in the generated answer. An LLM determines whether the factual statement in the ground-truth answer is covered by the generated answer. The final evidence coverage is computed as:

$$\text{Cov} = \frac{\text{number of covered statements}}{\text{total number of statements}} \tag{3.3}$$

Faithfulness (FS) assesses whether the generated answer is supported by the retrieved information. An LLM classifies statements from the generated answer as supported (1) and not supported (0) by the retrieved information. The faithfulness score is then computed as:

$$\text{FS} = \frac{\text{number of supported statements}}{\text{total number of statements}} \tag{3.4}$$

Retrieval evaluation

Evidence recall measures the fraction of relevant information retrieved. An LLM is used to classify each evidence sentence as 0 (not supported) or 1 (supported) with respect to the retrieved information. Recall@k is then computed as discussed in subsection 2.6.3.

Context relevance measures how relevant the retrieved information is to the input query. Each retrieved component (chunk or community summary) is scored by an LLM (0 = irrelevant, 1 = partially relevant 2 = fully relevant). This is repeated twice and normalized, then averaged over the number of retrieved instances.

Graph evaluation

The quality of the graph is evaluated through the properties of the graph structure, including the number of nodes and edges, the average degree, and average clustering coefficient. These are measured as descibed in subsection 2.6.3.

3.8.3 Experiments

Following the benchmark's guidelines, we constructed a knowledge graph from the medical dataset and ran our GraphRAG system on its corresponding QA dataset. The evaluation metrics were then computed using the benchmark's scripts. For comparison, we did the same for our RAG system. Since GPT-4o-mini, used in the original benchmark, was not accessible, we used Qwen3-Coder-480B-A35B-Instruct as our LLM.

Configuration set-up

We conducted two experiments using different hyperparameters and embedding models. All three evaluations were performed for both experiments.

Table 3.1: Configurations for the two experiments.

Parameter	Experiment 1	Experiment 2	
	RAG / GraphRAG	RAG	GraphRAG
Chunk size	600	256	1000
Chunk overlap	100	0	100
Retrieved top-k	5	5	5
Embedding model	text-embeddings-004	bge-large-en-v1.5	bge-large-en-v1.5
Generation temperature	0.0	0.7	0.7
Other temperatures	0.0	0.0	0.0
Seed	42	42	42

Generation temperature refers to the answer generation stage. Experiment 2 followed the benchmark configuration settings. Since only the generation temperature was specified in the temperature, we assumed 0 for the rest of the pipeline.

3.9 Qualitative evaluation

We performed a qualitative analysis, by examining the judges scoring and generated QA pairs ourself. As the corpus with HR-data was on the smaller size, reading through the whole corpus was feasible. We randomly selected 10 QA pairs from each of the three different QA datasets, and verified the quality and relevance of the question and reference answer, the generated answer, and the LLM-as-a-judge's scoring from 1-5. This was done for both the RAG and GraphRAG answers, so in total we looked at 20 data points per QA dataset.

3.9.1 LLM-as-a-judge

Zheng et al. (2023) argue that while few-shot prompting improves judgment consistency, this does not necessarily translate into higher accuracy and may even risk introducing bias. To avoid the risk that a small number of examples would overly influence the LLM's judgments, we adopted a zero-shot prompting strategy, meaning no example judgments were provided in the system prompt. The temperature was set to zero for all our judges for consistency, as this was prevalent, or explicitly recommended, in several papers (Gu et al., 2026; Wei et al., 2025; Salinas et al., 2025). One study states that the temperature choice did not have a huge impact on the accuracy (Wei et al., 2025), whilst another even claims higher temperatures worsen performance (Salinas et al., 2025). Both agree that lower temperatures increase consistency, hence we set the temperature to zero. Inspired by Verga et al. (2024)'s methodology, we employed a multi-judge system with three different LLMs and absolute scoring on a scale of 1–5. Our judge panel consisted of one online model gemini-2.5-flash-lite, and two on-premise models Mistral Medium 3 and Qwen3-Coder-480B-A35B-Instruct. As the judges produced absolute scores rather than binary decisions, we computed the average score across

the three judges instead of majority voting. The choice of LLM model strongly impacts judge performance, where large models (>10B parameters) are generally reported to have better performance (Salinas et al., 2025). Following the categorization proposed in this work, all models in our panel substantially exceed 10 billion parameters. We did not ask the LLM to supply any explanation for its answer, as this could possibly hurt performance (Salinas et al., 2025). The LLM was supplied with the query, reference answer, and the produced answer. The full prompt can be found in Appendix B, and an example of a QA pair with judge scoring can be found in Appendix C.

Many studies show that comparative, or pair-wise, assessment is more robust and often outperforms absolute rating (Liusie et al., 2024), however it is best used for large scale benchmarking settings where multiple answer candidates can be compared. Since we had a relatively small dataset and therefore few QA pairs, absolute scoring was a better fit for our use-case.

Chapter 4

Results and Analysis

This chapter presents the results of the benchmark evaluations on GraphRAG-Bench (Xiang et al., 2025) and LLM-as-a-judge and QA dataset evaluation on the Axis data.

4.1 Benchmark Evaluation

4.1.1 Generation evaluation

The results of the generation evaluation for our RAG and GraphRAG systems are displayed in Table 4.1. We observe that for both experiments our RAG model scored slightly higher than GraphRAG on all scores except for faithfulness (FS) in creative generation tasks. From the results in Table 4.1 we can conclude that the RAG model overall performs slightly better in experiment 1 than 2, most notably for evidence coverage (Cov), where experiment 1 scores considerably higher. This could be due to the decreased chunk size and the removal of overlap in experiment 2. The GraphRAG model also performed slightly better in the first experiment with a difference of $\sim 1\%$ for all scores. We could therefore conclude that overall the change of embedding model and hyper-parameters between experiment 1 and 2 did not effect the generation significantly. Unfortunately, we were not able to generate accuracy results for the Creative Generation tasks. Generating outputs for the creative tasks took significantly longer than for other tasks, causing LLM calls to hit our strict timeout limit. As a result, the accuracy evaluation, which required a three-stage LLM-as-a-judge process, was often too complex to complete within the available API timeout (even with retries). Therefore, the accuracy scores for our models are left out of the results. Fortunately, accuracy is assessed across the other tasks, enabling an overall evaluation of accuracy.

Table 4.1: Generation evaluation results for our RAG model and GraphRAG model. The highest score between the models is underlined for each metric.

Experiment	Model	Fact Retrieval		Complex Reasoning		Contextual Summarize		Creative Generation		
		ACC	ROUGE-L	ACC	ROUGE-L	ACC	Cov	ACC	FS	Cov
1	RAG	<u>71.64</u>	<u>34.87</u>	<u>69.94</u>	<u>22.03</u>	<u>72.99</u>	<u>82.27</u>	–	39.36	<u>67.32</u>
	GraphRAG	66.29	30.37	67.07	20.20	66.80	61.90	–	<u>40.50</u>	55.63
2	RAG	<u>70.10</u>	<u>33.18</u>	<u>67.33</u>	<u>21.10</u>	<u>72.11</u>	<u>77.00</u>	–	36.29	<u>57.97</u>
	GraphRAG	65.67	30.38	66.16	19.07	66.42	61.67	–	<u>39.31</u>	54.19

A comparison between our RAG and GraphRAG generation results and those reported for other RAG and GraphRAG models in the benchmark paper is presented in Table 4.2. Our models showed comparable performance to the existing models. The table display that our GraphRAG model (Exp. 2) had good answer accuracy (ACC) and coverage (Cov) compared to the other graph-enhanced models. Only HippoRAG2 (Gutiérrez et al., 2025) and RAPTOR (Sarthi et al., 2024) had more coverage of the ground truth in their answers than our model did. On the measurement for the longest subsequence in common with the ground truth answer (ROUGE-L), our answers were ranked in the middle among the other models. Finally, even though our GraphRAG model outperformed our RAG model in faithfulness, it still achieved a lower faithfulness score than most of the other models in the benchmark.

Table 4.2: Generation evaluation results for our RAG model and GraphRAG model compared with results for other models provided by the benchmark.

Category	Model	Fact Retrieval		Complex Reasoning		Contextual Summarize		Creative Generation		
		ACC	ROUGE-L	ACC	ROUGE-L	ACC	Cov	ACC	FS	Cov
	Medical Dataset									
RAG	**RAG (Exp. 1)**	**71.64**	**34.87**	**69.94**	**22.03**	**72.99**	**82.27**	–	**39.36**	**67.32**
	RAG (Exp. 2)	**70.10**	**33.18**	**67.33**	**21.10**	**72.11**	**77.00**	–	**36.29**	**57.97**
	RAG (w/o rerank)	63.72	29.21	57.61	13.98	63.72	77.34	58.94	35.88	57.87
GraphRAG	**GraphRAG (Exp. 1)**	**66.29**	**30.37**	**67.07**	**20.20**	**66.80**	**61.90**	–	**40.50**	**55.63**
	GraphRAG (Exp. 2)	**65.67**	**30.38**	**66.16**	**19.07**	**66.42**	**61.67**	–	**39.31**	**54.19**
	MS-GraphRAG (Edge et al., 2024)	38.63	26.80	47.04	21.99	41.87	22.98	53.11	32.65	39.42
	HippoRAG (Gutiérrez et al., 2024)	56.14	20.95	55.87	13.57	59.86	62.73	64.43	69.21	65.56
	HippoRAG2 (Gutiérrez et al., 2025)	66.28	36.69	61.98	36.97	63.08	46.13	68.05	58.78	51.54
	LightRAG (Guo et al., 2024)	63.32	37.19	61.32	24.98	63.14	51.16	67.91	78.76	51.58
	Fast-GraphRAG (CircleMind-AI, 2024)	60.93	31.04	61.73	21.37	67.88	52.07	65.93	56.07	44.73
	RAPTOR (Sarthi et al., 2024)	54.07	17.93	53.20	11.73	58.73	78.28	62.38	59.98	63.63
	Lazy-GraphRAG (Darren Edge, 2024)	60.25	31.66	47.82	22.68	57.28	55.92	62.22	30.95	43.79

4.1.2 Retrieval evaluation

The results for the retrieval evaluation are shown in Table 4.3. In Experiment 1, our RAG model outperformed our GraphRAG model in both information relevance and evidence recall across all difficulty. When we switched embedding model and decreased the chunking size and removed overlap for the RAG model (Exp. 2), the difference between the models' performance decreased and the GraphRAG model scored higher for retrieval in creative generation tasks. Both the RAG and the GraphRAG model performed slightly worse for all tasks in Experiment 2 than in Experiment 1.

Table 4.3: Retrieval evaluation results for our RAG model and GraphRAG model. The highest score between the models is underlined for each metric.

Experiment	Model	Fact Retrieval		Complex Reasoning		Contextual Summarize		Creative Generation	
		Recall	Relevance	Recall	Relevance	Recall	Relevance	Recall	Relevance
1	RAG	93.97	96.95	89.55	95.09	90.20	96.63	82.97	93.22
	GraphRAG	77.63	89.05	71.59	88.36	63.42	90.14	65.50	73.95
2	RAG	88.74	94.76	82.58	91.89	83.60	95.24	67.14	70.93
	GraphRAG	75.01	88.84	71.02	88.46	66.49	92.21	73.74	75.90

Table 4.4 presents a comparison between our RAG and GraphRAG retrieval results and those reported for RAG and multiple GraphRAG models in the benchmark paper.

The recall for our RAG model (Exp. 2) stay relatively consistent with the recall results for then benchmarks RAG (w/o rerank), except in creative generation tasks where it scored much higher. For the relevance metric, our RAG model scored significantly higher than the benchmarks RAG (w/o rerank) model, across all different tasks.

Analyzing our GraphRAG model (Exp 2), it placed itself in the lower half for evidence recall across all tasks except for creative generation. Our Graph model's information relevance where evaluated higher than most of the other models.

Table 4.4: Retrieval evaluation results for our own RAG and GraphRAG model compared with results for other models provided by the benchmark.

Category	Model	Fact Retrieval		Complex Reasoning		Contextual Summarize		Creative Generation	
		Recall	Relevance	Recall	Relevance	Recall	Relevance	Recall	Relevance
Medical Dataset									
RAG	**RAG (Exp. 1)**	**93.97**	**96.95**	**89.55**	**95.09**	**90.20**	**96.63**	**82.97**	**93.22**
	RAG (Exp. 2)	**88.74**	**94.76**	**82.58**	**91.89**	**83.60**	**95.24**	**67.14**	**70.93**
	RAG (w/o rerank)	86.24	63.71	84.97	84.11	84.14	89.94	44.88	58.73
GraphRAG	**GraphRAG (Exp. 1)**	**77.63**	**89.05**	**71.59**	**88.36**	**63.42**	**90.14**	**65.50**	**73.95**
	GraphRAG (Exp. 2)	**75.01**	**88.84**	**71.02**	**88.46**	**66.49**	**92.21**	**73.74**	**75.90**
	MS-GraphRAG (Edge et al., 2024)	38.06	05.67	61.32	04.25	59.66	05.24	66.59	02.76
	HippoRAG (Gutiérrez et al., 2024)	87.25	52.44	83.80	42.19	83.46	49.13	81.66	45.03
	HippoRAG2 (Gutiérrez et al., 2025)	78.70	87.96	77.00	80.94	77.40	86.85	61.12	78.64
	LightRAG (Guo et al., 2024)	80.32	41.27	82.91	42.79	85.71	43.11	81.34	45.17
	Fast-GraphRAG (CircleMind-AI, 2024)	66.82	45.86	74.93	38.80	77.27	47.58	62.99	25.15
	RAPTOR (Sarthi et al., 2024)	85.40	69.38	89.70	53.20	88.86	58.73	72.70	52.71
	Lazy-GraphRAG (Darren Edge, 2024)	74.29	19.90	78.65	17.50	78.72	21.35	83.41	15.09

4.1.3 Graph evaluation

The graph evaluation was performed on the two graphs that were built in Experiment 1 and Experiment 2. A comparison between our graph results and those reported for other GraphRAG models in the benchmark paper is presented in Table 4.5. Both our graph and the MS-GraphRAG model's graph have summary nodes, and the benchmark paper does not specify whether these should be included in the node count. We decided to exclude them, as summary nodes do not represent graph entities but rather acts as a meta abstraction layer.

As shown in Table 4.5, our graphs contain substantially more nodes and edges than those reported in the benchmark results. The magnitude of the difference is notable, and the underlying cause is not fully clear. Although part of the discrepancy may be due to our method extracting more entities, the reported node counts of the benchmark still appear unexpectedly low given the size of the data set. One possibility is that the benchmark normalizes

counts (e.g., per 1,000 tokens), although this is not explicitly stated in their paper. Node and relation counts were computed using the code supplied by the benchmark, so our calculations are expected to be consistent with the benchmark's methodology. However, the repository has been updated since the paper's release, so discrepancies may occur. We have verified that no double counting occurred, and our results appear consistent from our side. We have attempted to contact the authors of the paper via email to get clarity on this, but we are yet to receive response at the time of writing. Therefore, we can only acknowledge the possibility that there is a discrepancy in how the paper carried out its calculations. Nonetheless, we have made all reasonable efforts to replicate and understand their methodology.

However, assuming these results are accurate, the higher node counts could indicate that our method achieves broader coverage of the underlying text. On the other hand, it could also indicate redundancy caused by insufficient entity resolution. The average degree of our graphs was more on par with the other models', even higher than both MS-GraphRAG (Edge et al., 2025) and HippoRAG (Gutiérrez et al., 2025), but still much lower than HippoRAG2 (Gutiérrez et al., 2025), which had the highest average of 13.31 connections per node. Additionally, the average clustering coefficient was also much lower than most of the other models'.

Table 4.5: Index evaluation results for our model compared to results for MS-GraphRAG (Edge et al., 2025), HippoRAG2 (Gutiérrez et al., 2025), LightRAG (Guo et al., 2025b), Fast-GraphRAG (Microsoft, 2026) and HippoRAG (Gutiérrez et al., 2025) provided in the benchmark.

Model	Number of nodes	Number of edges	Average Degree	Avg. Clust. Coeff
Medical Dataset				
GraphRAG Exp. 1	9207	9563	2.08	0.0362
GraphRAG Exp. 2	9778	10561	2.16	0.032
MS-GraphRAG	182	166	1.82	0.300
HippoRAG2	598	3979	13.31	0.497
LightRAG	115	149	2.58	0.139
Fast-GraphRAG	127	350	5.50	0.347
HippoRAG	207	213	2.06	0.087

4.2 Evaluation on Axis data

4.2.1 LLM-as-a-judge scores

The results in table 4.6 and figure 4.1 indicate that RAG consistently outperforms GraphRAG across all three synthetic datasets, when evaluated using our own LLM-as-a-judge scoring. Specifically, RAG achieves mean scores of 4.03, 4.42, and 3.97 on the answer given, no answer given (single fact) and no answer given (multiple facts) datasets. While GraphRAG scores lower at 3.05, 3.23, and 2.41 respectively. The differences range from 0.98 to 1.57, with the largest gap observed on the multiple facts dataset. The error bars in 4.1 indicate the standard deviation, showing some variability in individual scores.

Figure 4.2 shows the distribution of LLM-as-a-judge scores for RAG and GraphRAG across

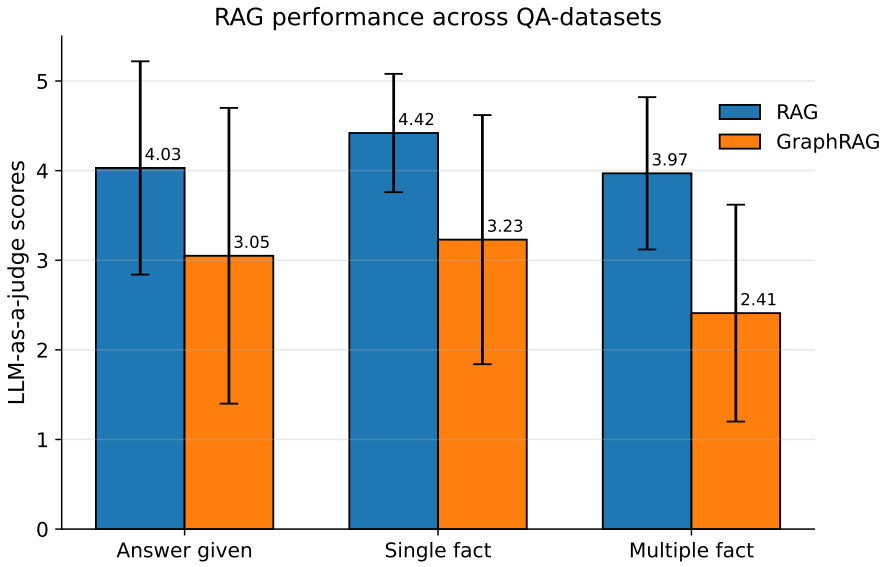


Figure 4.1: Aggregated LLM-as-a-judge scores for RAG and GraphRAG across the three datasets. Single fact and multiple facts refers to the no answer given datasets. Error bars indicate standard deviation.

Dataset	RAG	GraphRAG	Difference
Answer given	4.03 ± 1.19	3.05 ± 1.65	-0.98
Single fact	4.42 ± 0.66	3.23 ± 1.39	-1.19
Multiple fact	3.97 ± 0.85	2.41 ± 1.21	-1.57

Table 4.6: LLM-as-a-judge mean scores (± std) for RAG vs GraphRAG across datasets. Single fact and multiple facts refers to the no answer given datasets.

the three datasets, with each curve corresponding to one of three LLM judges: Mistral Medium 3 (Judge 1), Qwen3-Coder (Judge 2), and Gemini-2.5-Flash-Lite (Judge 3). Solid lines indicate RAG, while dashed lines indicate GraphRAG. The curves show normal distributions fitted using the mean and standard deviation of the aggregated scores from 105 QA pairs per dataset.

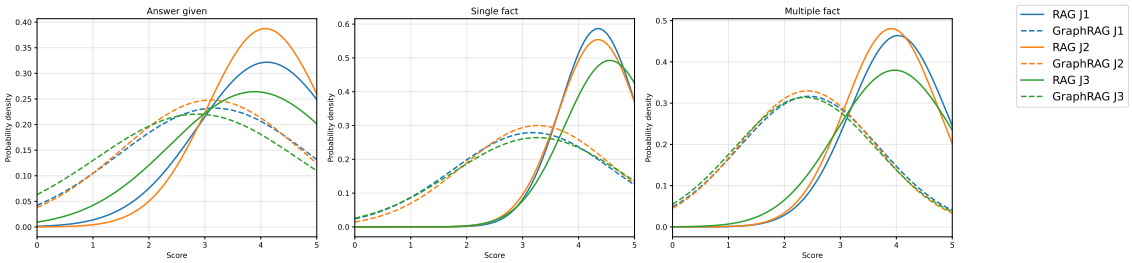


Figure 4.2: Normal distribution of evaluation scores across the three datasets for RAG and GraphRAG, separated by judge 1–3. Solid lines represent standard RAG and dashed lines represent GraphRAG. The scores are aggregated over 105 QA pairs, for each dataset. Single fact and multiple facts refers to the no answer given datasets.

Across all datasets, GraphRAG exhibits wider distributions, reflecting higher variability in the LLM-judge's scorings, whereas RAG produces narrower distributions, indicating more consistent evaluations. As observed earlier, RAG generally achieves higher mean scores for all three judges. There do not appear to be substantial differences in the mean or variance of scores between the judges, nor any evidence of systematic bias by any judge toward more restrictive or more lenient scoring.

4.2.2 Qualitative Analysis

For each dataset, 10 QA pairs were randomly sampled for human evaluation along with their generated answers and LLM scores. Since each pair included answers from both RAG and GraphRAG, 20 judged answers were analyzed per dataset. Each QA pair was labeled as correct or incorrect, and incorrect cases were assigned an error category (incorrect answer, incorrect question, or missing information). For the multiple-fact dataset, an additional category was introduced for questions that were incorrect because they did not require multi-fact reasoning. The results are shown in Table 4.7.

Overall, single-fact prompting yielded **90%** correct QA pairs. By contrast, the answer-given dataset performed poorly, likely due to entity extraction producing unsuitable answer candidates, forcing the LLM to form questions around single words or isolated entities and resulting in low-quality questions. We also found that nearly half of the multiple-fact questions did not actually require reasoning over multiple facts, indicating that zero-shot prompting is not a reliable approach for constructing multi-fact QA datasets.

Evaluate QA-pair	Answer given	Single fact	Multiple facts
Correct QA-pair	50%	90%	30%
Incorrect QA-pair	50%	10%	70%
Incorrect question	20%	0%	0%
Incorrect answer	40%	0%	29%
Answer missing information	40%	100%	29%
Not multi-fact	–	–	43%

Table 4.7: Human evaluation of synthetically generated QA-pairs across the three different datasets. Single fact and multiple facts refers to the no answer given datasets. If the QA-pair was considered incorrect, its classified as either; incorrect answer, incorrect question, or answer missing information. Multi-hop questions could also be classified as 'Not Multi-fact', if they were not considered requiring any multi-hop reasoning.

In Table 4.8, the results from the judge evaluation are presented. Each LLM score was classified as correct or incorrect, and incorrect cases were categorized as either too strict or too lenient. Note that this is not an evaluation of the correctness of the generated answer itself, but rather the correctness of the judge's scoring.

Overall, the judge tended to be overly strict, assigning the highest score only when the produced answer almost exactly matched the reference answer. Even when the content closely matched the reference, answers with different phrasing or word order received lower scores. For the multiple-fact dataset, the reference answers introduced additional issues. These questions were generally more open-ended, with several acceptable formulations. However, because the judge relied heavily on the provided reference answers, it assigned very low scores to both GraphRAG- and RAG-generated answers. This suggests that reference-based scoring can exert too much influence on the judge's assessment, although a different prompting strategy for the LLM-judge could potentially mitigate this.

Evaluate judge	Answer given	Single fact	Multiple facts
RAG			
Correct scoring	90%	40%	60%
Incorrect scoring	10%	60%	40%
Too strict	100%	100%	100%
Too lenient	0%	0%	0%
Graph RAG			
Correct scoring	80%	70%	90%
Incorrect scoring	20%	30%	10%
Too strict	100%	100%	100%
Too lenient	0%	0%	0%

Table 4.8: Human evaluation of judge scoring on RAG and GraphRAG answering across the three different datasets. Single fact and multiple facts refers to the no answer given datasets. Each incorrect scoring was classified as either 'Too strict' or 'Too lenient' in its scoring.

Chapter 5

Discussion

This chapter discusses our results and their implications. It reviews limitations and sources of error, outlines future work, and addresses relevant societal and ethical considerations.

5.1 Benchmark discussion

Our benchmark results indicate that, overall, RAG still outperforms GraphRAG across multiple tasks. This is consistent with the benchmark paper, which reports higher performance for RAG than for existing GraphRAG models (Xiang et al., 2025). Notably, the GraphRAG models achieve higher accuracy (ACC) and faithfulness (FS) than RAG on Creative Generation tasks, indicating that graph-based retrieval may offer advantages for more open-ended and creative generation scenarios. Our GraphRAG implementation showed competitive performance to other currently established GraphRAG models, showcasing the highest relevance score of all models on the retrieval evaluation, and scored above or close too the highest accuracy score on the generation evaluation. However, it should be noted that different LLMs were used in our evaluation compared to the benchmarks, which may limit comparability to some extent. As described in section 3.8.3, we employed a different LLM as we did not have access to GPT-models. Finding reliable sources, such as academic papers, that compared their model (GPT-4o-mini) to ours (Qwen3-Coder-480B-A35B-Instruct) was difficult, since Qwen is a coding-oriented model that is typically evaluated on coding benchmarks rather than general language benchmarks. Consequently, the impact of using a different LLM is difficult to quantify.

To estimate the impact of substituting a different LLM, one would ideally reproduce the benchmark results using Qwen instead of GPT-4o-mini. However, we were not able to reproduce the benchmark results for any of the GraphRAG models (see section 5.3 for further

details). We did, however, obtain comparable results for their RAG baseline. The benchmark paper (Xiang et al., 2025) does not provide the exact implementation of their RAG (w/o rerank) model, so it cannot be guaranteed that our implementation fully matches theirs. Nevertheless, as standard RAG has relatively few moving components, we can assume that our implementation is roughly the same. Under this assumption, using Qwen as the LLM appears to have had a positive effect on performance, as RAG (Exp. 2) generally scored higher than the benchmark's RAG (w/o rerank). However, because the LLM is used for evaluation as well, the choice of LLM may influence scoring. In fact, ROUGE-L is the only metric of the benchmark that does not rely on LLM-as-judge. LLMs can vary significantly in their responses depending on their size and pretraining data. In our evaluation, the judge LLM was restricted to binary or ternary scoring and the evaluation instructions were highly constrained. Even so, it cannot be ruled out that the results might have been different if another LLM had been used. Moreover, whether the same positive impact of the Qwen model would extend to the GraphRAG systems is uncertain, as these models also incorporate LLMs throughout the construction of the knowledge base itself. Most GraphRAG models, including ours, integrate LLMs throughout multiple stages of the pipeline. Consequently, the choice of LLM is likely to have a larger impact on these models than on traditional RAG systems. For this reason, the cross-benchmark comparisons presented in section 3.8 should be interpreted with some caution.

We structure the remainder of the benchmark discussion around three components: graph construction, retrieval, and generation.

5.1.1 Graph construction

Our LLM-based entity and relation extraction produced a graph with a substantially larger number of nodes and edges than the benchmark models had. On the one hand, this may indicate broad coverage of the source data and successful extraction of many entities. On the other hand, it may also suggest insufficient entity resolution, leading to fragmented representations and reduced connectivity. Since there is no clear reference for the expected or "correct" number of entities in the corpus, it is unclear which interpretation is more accurate.

The results of the graph evaluation (see Table 4.5) suggest that our graph construction leads to a relatively sparse graph. Although our model lies in the mid-range compared to the benchmark models in terms of average degree and clustering coefficient, the resulting graph appears relatively sparse. This is reflected in the fact that nodes only have two edge connections on average, indicating limited connectivity of the graph. This observation is somewhat unexpected, given that the benchmark authors state that the dataset was specifically chosen to be rich in hierarchical structure and relationships, which would be expected to yield a more connected graph (Xiang et al., 2025). However, graph density is strongly influenced by the underlying dataset. For example, a dataset composed of largely unrelated documents with little thematic overlap would naturally yield a sparse graph, consisting of mostly disconnected or weakly connected clusters. Since the benchmark models exhibit similar sparsity patterns, it is difficult to assess how much connectivity this dataset should reasonably produce. Furthermore, the benchmark graphs show substantial variation across models, with some exhibiting very high node counts and connectivity. As a result, it remains somewhat unclear what level

of connectivity should be considered appropriate or expected for this dataset.

The sparsity of the graph also appears to have influenced the community generation process. The Leiden algorithm assigns nodes to the same community based on direct or indirect connectivity. Given that the median connected component size was approximately two nodes, this resulted in a large number of small communities, often consisting of only two nodes. Their effectiveness as a summarizing meta-layer may be limited when they represent only a small number of nodes. Furthermore, a large number of communities increases the number of summaries that must be generated, leading to additional LLM calls during graph construction and potentially introducing noise during retrieval. Conversely, smaller and more fine-grained communities may yield more detailed summaries, which could benefit retrieval by improving granularity. Further investigation is therefore required to better understand how graph sparsity and the number of community clusters affect retrieval performance.

Improvements

The sparsity of the graph could possibly be related to limitations in the entity resolution step, which may have failed to merge equivalent entities. Entity resolution could potentially be improved by incorporating additional contextual information from the source text into the LLM's merge decisions. Specifically, for each pair of candidate entities, the occurrences of the entities could be identified in the source text, and a surrounding text window could be extracted and provided to the LLM to allow better informed merge decisions. This approach was considered and partially implemented, but could not be fully incorporated in time of the benchmark results being produced. In retrospect, the use of a Levenshtein distance threshold of three edits for candidate filtering was likely overly restrictive. This threshold will often exclude abbreviations and acronyms, and may also have been overly restrictive for valid alternative spellings or name variants exceeding the three-edit threshold.

The sparsity could also be traced back to the entity and relation extraction step. Some systems ask the LLM to assign labels to extracted entities, which can be helpful for subsequent entity resolution (Guo et al., 2025b). Others first extract entities and then pass the entities together with the original text chunks back to the LLM to retrieve relations (Edge et al., 2025). This two-step extraction process may make it easier to capture relationships and produce a more connected graph. Either of these approaches could potentially reduce the frequency of small connected components.

Another limitation was that, despite being designed to ignore isolated nodes, some still appeared in the graph. This occurred when the LLM produced incomplete triples with an empty object that were not fully filtered during entity resolution and were therefore counted as isolated nodes during benchmark evaluation. A more constrained or guided entity extraction process could mitigate this issue.

5.1.2 Retrieval

In our retrieval evaluation, our RAG outperformed our GraphRAG across all tasks, with the exception of creative generation. This was expected as the benchmark's own evaluation

showed that both RAG models achieved competitive retrieval performance to the GraphRAG models, with RAG with re-rank model gaining the highest recall score in fact retrieval and the highest relevance score in both complex reasoning and summarization tasks. This demonstrates the reliable retrieval performance of traditional RAG systems. The difference between our models ability to retrieve relevant information was rather small. Possibly because both retrieval methods were based on semantic similarity, over chunks or community summaries, which is why they might have found similar information. Despite this, the RAG model was quite a lot better at retrieving information that was actually sufficient to answer the question. Since the recall was measured by comparing the retrieved context to the evidence that was given in the QA dataset, this might reflect the fact that our GraphRAG retrieved summaries which might have left out detailed information. On another note, the difference between the models' performance decreased when the task complexity increased. This result supports the notion that traditional RAG struggles with more complex tasks, such as creative generation. The community summaries of our GraphRAG may have captured more relations within the data which could be used to guide the LLM through these tasks.

5.1.3 Generation

Our GraphRAG model was not able to rise above our RAG models performance in generating good answers. First of all, its answers were less accurate which could have been affected by the retrieval stage which retrieved summaries and not word-by-word phrasings from the text data. Additionally, the GraphRAG model's answers in the summarization and creative generation tasks did not cover as much of the ground truth. This is reflected in the recall score from the retrieval evaluation, which indicates that the model lacked sufficient information to answer the questions fully. Furthermore, the only task for which our GraphRAG scored higher than our RAG model was again creative generation, where it had higher faithfulness. As mentioned, the GraphRAG model retrieved both more relevant, and helpful, information in these types of tasks. This improved retrieval reduced the need for the LLM to compensate for missing information during generation, thereby lowering the risk of hallucinations and improving faithfulness.

5.2 Enterprise data discussion

We saw that the RAG model outperformed GraphRAG on the synthetically generated QA datasets. In our qualitative analysis we saw that a zero-shot prompting technique was promising for generating fact retrieval questions, while the answer-based prompting style was far more unstable, with half of the QA pairs being faulty in some way. However, these poor results could also be explained by the NER extraction technique being too simplified and that a more complex NLP extraction method could possibly yield better results. The LLM also struggled with generating queries requiring multi-hop reasoning, suggesting that this is a more challenging task. This difficulty in generating multi-hop questions could possibly be solved by providing the LLM with more examples, more comprehensive prompt instructions, or multi-step prompting. We can still conclude that, for simple fact-retrieval questions,

synthetic QA generation using zero-shot prompting can be a reliable alternative. However, our qualitative evaluation only analyzed a small subset of the QA datasets, and this has to be taken into consideration when interpreting our results. Generally, we saw that the LLM-judge could provide correct ratings, but saw problems with the LLM often adhering too much to the reference answer, making it too restrictive in its scoring. Reference-free evaluation may mitigate this strict matching behavior, but often requires few-shot examples or explicit evaluation criteria, as well as more deliberate prompting strategies to ensure consistent scoring. We observed substantial overlap in the scores produced by the three judges, which indicates consistency across the LLM judges. This suggests that LLM-based evaluation can be a robust means of evaluation.

The dataset we used may have constrained the performance of GraphRAG. The Axis corpus was small and based on web-scraped HR documentation containing mixed modalities (e.g., images, hyperlinks, tables, videos) and noisy scraped layout content that the LLM could not reliably interpret. While suitable for simple fact lookups, it provided limited relational structure and few repeated mentions of the same entities. Consequently, it may not have supported multi-hop or relational reasoning. This likely limited the opportunity for GraphRAG to demonstrate advantages over the RAG baseline. More generally, the dataset may not have been ideal for evaluating a graph-based retrieval method, given its small size, noise, and lack of structure.

5.3 Limitations

A considerable limitation of our project was that our pipeline required many API calls to the LLM. Although we optimized it to use fewer calls than many other GraphRAG systems, API usage remained a key constraint. The endpoint's rate limit of 20 calls per minute introduced significant latency in our pipeline, making both runtime and benchmark testing time-consuming. Combined with the limited time of our thesis, this restricted our ability to iterate on and refine the pipeline.

This also complicated benchmarking, as reproducing all results from the original paper was not feasible (Xiang et al., 2025). Ideally, we would have replicated their experiments with our LLM to validate the reliability of the benchmark and assess the impact of using a different model. In addition to the API rate limit, reproduction was further complicated by the limited availability of implementation details in the benchmark paper, and by cases where the provided documentation and instructions did not work as expected.

The rate limit also influenced our methodological decisions. We initially planned to reproduce the Edge et al. (2025) GraphRAG implementation. However, upon closer examination, we realized their approach required a large number of LLM API calls (Xiang et al., 2025). Recent work has shown that GraphRAG often results in high token costs and substantial runtime overhead (Zhang et al., 2025). From a practical standpoint, implementing such a GraphRAG system was unsustainable given our current infrastructure. As a result, we implemented a simplified retriever, as described in the method section 3.6.

Another constraint was that many commonly used datasets were too extensive for us to use, as building the graph and retrieval components would require a significant amount of time.

Yet, results become more interesting when the corpus is large enough. Small graphs are much more manageable, while larger graphs introduce ambiguity and homographs among entity names. Addressing these challenges likely requires more advanced graph structures, enriched with additional metadata and attributes.

Data contamination bias

A general limitation in evaluating RAG systems is the potential overlap between evaluation datasets and the parametric knowledge of the underlying LLMs. Since many LLMs are pre-trained on large-scale corpora that include sources such as Wikipedia, evaluating RAG or GraphRAG models on datasets derived from similar sources may introduce bias. In such cases, correct answers may stem from the model's intrinsic knowledge rather than effective use of retrieved documents, making it difficult to assess the true contribution of retrieval mechanisms. This issue, often discussed in terms of dataset contamination, has been shown to affect question-answering benchmarks (Cheng et al., 2025; Monteiro et al., 2024). Wang et al. (2024a) further raises this concern specifically in regards to RAG systems, noting that since Wikipedia is widely used in LLM pre-training, the information contained in retrieved documents may already be known to the model, making it questionable whether RAG models truly utilize the retrieved references rather than their intrinsic knowledge (Wang et al., 2024a). Evaluating RAG systems on domain-specific datasets may partially mitigate this issue by reducing overlap with LLM pre-training data.

5.4 Ethical and societal considerations

AI-based QA systems are prone to hallucinations, which in an enterprise setting can lead to incorrect decisions or misinformation being propagated internally. RAG partly reduces this risk by grounding answers in source data, but errors can still arise from misinterpreted context or outdated information. In GraphRAG, inaccuracies may also be introduced when graphs are built automatically using LLMs. When no ground truth graph exists or verification is infeasible, ensuring correctness becomes non-trivial. There is a risk of over-reliance in these systems, where users place more trust in outputs than warranted.

When deployed in customer-facing chatbots, organizations can risk legal and financial consequences if incorrect information is presented to end-users. Companies can be held responsible, as seen in cases where misleading chatbot responses led to disputes and legal actions (The Associated Press, 2025; Lifshitz and Hung, 2024). Jailbreaking and prompt manipulation further introduce security risks, as users may bypass safeguards and trigger unintended responses.

Bias is another ethical concern. LLMs can reproduce social or cultural biases from their training data, potentially reinforcing stereotypes or patterns. When LLMs are used to generate data and to evaluate results, bias may further influence outcomes, which needs to be actively addressed.

Privacy considerations arise when enterprise QA systems require access to internal knowledge sources such as documentation, tickets, or communication channels. In practice, the

distinction between personal data and company data is not always clear, and both may coexist within the same communication streams. This necessitates appropriate data governance practices to ensure that sensitive or personal information is not accidentally exposed through the QA interface.

5.5 Future work

System improvements

As mentioned in section 1.3, we initially chose to base our method on Edge et al. (2025) GraphRAG. However, as computational constraints emerged, we had to make considerable alterations to our pipeline. Due to limited time, we were unable to experiment with different approaches to the extent we had hoped. This could have involved experimenting with different retrievers or LLM models, exploring alternative graph construction strategies, or tuning hyperparameters and assessing their influence on performance, such as chunk size or retrieval top-k. For example, we could explore the effect of providing the LLM with a larger number of community summaries, thereby utilizing more of its available context window. For future work, it may be more beneficial to explore newer or less established methods that better align with practical constraints and the specific use case, rather than extensively adapting an existing framework.

There is also room for improvement in our prompting strategy. While many existing implementations rely on long and complex prompt variations adapted to different use cases, we opted for short and concise prompts containing only essential information. This simplified adjustments to the prompt when addressing unwanted outputs or dataset changes and avoided introducing unnecessary noise into the prompts. Since prompting was not identified as a primary limitation of our system, it was not further optimized. However, future work could benefit from a more systematic and extensive prompting strategy.

Dynamic graph updates

In future work, exploring how the knowledge graph can be dynamically updated with new knowledge over time is a highly relevant area of research (Peng et al., 2025). Most GraphRAG solutions today are built upon a static knowledge base, which does not reflect the needs of many real-world industry applications. As new data constantly emerges, methods for efficiently and reliably updating the graph will be highly valuable. However, real-time updates is a challenging problem, including issues such as entity linkage, versioning, and maintaining consistency of information in the graph. Despite these challenges, solving dynamic graph updates would significantly increase the relevance of GraphRAG systems for enterprise use.

Multimodal data

Another interesting challenge is how to extend GraphRAG from purely textual data to integrating multimodal information. This proved highly relevant in our own setting, as the

web-scraped Axis data often included tables, links, video tutorials, and images, which the LLM struggled with comprehending or entirely ignored. Developing methods to represent, store, and reason over multiple data modalities would substantially increase the usability and robustness of GraphRAG systems.

Future directions for GraphRAG

From our benchmarking results, we observed that our GraphRAG implementation performed on par with other GraphRAG models. However, like the other models of the benchmark, it was still outperformed by standard RAG. This is well aligned with recent studies, reporting that GraphRAG models often underperform compared to RAG in many real-world applications (Han et al., 2025a; Xiang et al., 2025). Although GraphRAG have been reported to achieve higher performance on summarization and multi-hop reasoning tasks than RAG (Zhou et al., 2025; Edge et al., 2025), an important question is raised whether its added complexity and LLM calls is justified when RAG system, for many QA tasks, can provide similar results while being much less resource-intensive. Even more interesting is whether GraphRAG will see broader adoption in enterprise applications, despite its remaining challenges with deployment, cost, and latency.

On a final note, the relative advantage of GraphRAG over traditional RAG is still widely debated. Questions remain about what types of queries are best suited for evaluating a knowledge graph, what the model's key strengths truly are, and whether it should be benchmarked using the same criteria as traditional RAG systems. In other words, does GraphRAG outperform standard RAG, or might its strengths lie in areas where standard RAG is limited, acting more as a complement than a substitute?

Chapter 6

Conclusion

In this thesis, we investigated the emerging use of knowledge graphs as an alternative knowledge base for retrieval-augmented generation systems. We examined the feasibility of building such a system using existing resources within a company. A traditional RAG system was developed as a baseline for comparison with the GraphRAG approach. The systems were evaluated on question-answering tasks of varying difficulty using the GraphRAG-Bench framework.

Furthermore, we examined the applicability of GraphRAG to internal company data. As part of this work, we synthetically generated question-answer datasets from company data and developed an LLM-as-a-judge system for automating evaluation using LLMs. A qualitative analysis of both the generated datasets and the evaluation approach was conducted to assess their reliability. Overall, our results provide insight into the potential and limitations of integrating knowledge graphs into RAG systems, including the use of LLMs for graph construction and evaluation.

Many existing GraphRAG systems are designed to achieve high performance on complex global question-answering and multi-hop reasoning tasks. However, such systems often rely on computationally expensive pipelines and extensive use of LLMs, which may limit their practicality in real-world and resource-constrained settings. Our work highlights these practical challenges. In attempting to implement an established GraphRAG approach, we found that the associated computational and operational costs made the methodology difficult to justify in practice, despite its reported performance advantage over traditional RAG on certain tasks. While GraphRAG remains a promising and evolving approach, these observations suggest that its benefits must be carefully weighed against development complexity and deployment cost, particularly when compared to the relative simplicity and robustness of standard RAG systems.

6.1 Research Questions

The research questions posed in the beginning of this thesis can be summarized and answered as follows:

RQ1 **How does standard RAG compare to GraphRAG in answering (a) fact retrieval and complex reasoning questions, and (b) summarization and creative generation tasks?**

(a) **fact retrieval and complex reasoning questions?**

Our benchmark results show that standard RAG performed better than GraphRAG on fact retrieval and complex reasoning tasks, producing more accurate answers with sufficient context.

(b) **summarization and creative generation tasks?**

The GraphRAG model's performance became more comparable to RAG as question complexity increased. It demonstrated better retrieval performance and answer faithfulness in creative generation tasks. However, on the summarization tasks, the GraphRAG model did not exceed standard RAG.

RQ2 **How reliable are synthetic question generation and LLM-as-a-judge methods for evaluating QA systems?**

Synthetic QA generation was reliable for simple fact-based questions, while multi-hop and answer-based prompting was unstable. The LLM-judge produced consistent scores but matched candidate answers too literally to the reference answer, resulting in lower scores for responses that where semantically correct but differently phrased.

RQ3 **How does incorporating a knowledge graph into RAG improve an LLM's performance in enterprise-specific QA?**

In the enterprise setting, incorporating a knowledge graph did not improve QA performance. GraphRAG received lower scores than the RAG baseline, and qualitative inspection showed that RAG produced more accurate and complete answers. However, the Axis dataset was small in size and consisted of web-scraped HR documentation, suitable for factual lookups but not strongly relational, making it difficult for GraphRAG to demonstrate advantages. As we were unable to generate true multi-hop reasoning questions for this data, the evaluation did not meaningfully probe the reasoning tasks where GraphRAG is expected to benefit.

RQ4 **What are the computational and cost trade-offs between standard RAG and GraphRAG?**

GraphRAG incurred substantially higher computational costs because graph construction and summary generation require an extensive amount of LLM calls, making it difficult to justify in enterprise deployments compared to the simpler RAG pipeline.

References

Beijing Academy of Artificial Intelligence (2024). Bge-large-en v1.5: General-purpose text embedding model. [Model] Retrieved December 7, 2025, from https://huggingface.co/BAAI/bge-large-en-v1.5.

Bendi-Ouis, Y., Dutartre, D., and Hinaut, X. (2025). Deploying open-source large language models: A performance analysis. arXiv preprint arXiv:2409.14887v4. https://doi.org/10.48550/arXiv.2409.14887.

Blackwell, R. E., Barry, J., and Cohn, A. G. (2025). Towards reproducible LLM evaluation: Quantifying uncertainty in LLM benchmark scores. arXiv preprint arXiv:2410.03492v2. https://doi.org/10.48550/arXiv.2410.03492.

Bratanič, T. (2024a). Global GraphRAG notebook. [Jupyter notebook]. GitHub repository. Retrieved January 21, 2026, from https://github.com/tomasonjo/blogs/blob/master/llm/ms_graphrag.ipynb.

Bratanič, T. (2024b). Implementing 'from local to global' GraphRAG with Neo4j and LangChain: Constructing the graph. [Blog post]. Retrieved January 21, 2026 from https://neo4j.com/blog/developer/global-graphrag-neo4j-langchain/.

Brown, A., Roman, M., and Devereux, B. (2025). A systematic literature review of retrieval-augmented generation: Techniques, metrics, and challenges. *Big Data and Cognitive Computing*, 9(12):320. https://doi.org/10.3390/bdcc9120320.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., 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., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., and Amodei, D. (2020). Language models are few-shot learners. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H., editors, *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran

Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf.

Catanzaro, B., Liu, Z., Ping, W., Shoeybi, M., Wang, B., You, J., Yu, Y., and Zhang, C. (2024). RankRAG: Unifying context ranking with retrieval-augmented generation in LLMs. In *Advances in Neural Information Processing Systems*, volume 37, pages 121156–121184. https://doi.org/10.52202/079017-3850.

Cheng, Y., Chang, Y., and Wu, Y. (2025). A survey on data contamination for large language models. arXiv preprint arXiv:2502.14425v2. https://doi.org/10.48550/arXiv.2502.14425.

Christophides, V., Efthymiou, V., Palpanas, T., Papadakis, G., and Stefanidis, K. (2020). An overview of end-to-end entity resolution for big data. *ACM Computing Surveys*, 53(6):1–40. https://doi.org/10.1145/3418896.

Diaz-Garcia, J. and Lopez, J. (2025). A survey on cutting-edge relation extraction techniques based on language models. *Artificial Intelligence Review*, 58:287. https://doi.org/10.1007/s10462-025-11280-0.

Edge, D., Trinh, H., Cheng, N., Bradley, J., Chao, A., Mody, A., Truitt, S., Metropolitansky, D., Ness, R. O., and Larson, J. (2025). From local to global: A GraphRAG approach to query-focused summarization. arXiv preprint arXiv:2404.16130v2. https://doi.org/10.48550/arXiv.2404.16130.

Ehrlinger, L. and Wöß, W. (2016). Towards a definition of knowledge graphs. In *International Conference on Semantic Systems*. https://api.semanticscholar.org/CorpusID:8536105.

Es, S., James, J., Espinosa-Anke, L., and Schockaert, S. (2024). RAGAs: Automated evaluation of retrieval augmented generation. In *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics: System Demonstrations*, pages 150–158, St. Julians, Malta. Association for Computational Linguistics. https://doi.org/10.18653/v1/2024.eacl-demo.16.

Explosion AI (2023). spacy: Industrial-strength natural language processing. [Software library] Retrieved January 5, 2026, from https://github.com/explosion/spaCy.

Fellegi, I. P. and Sunter, A. B. (1969). A theory for record linkage. *Journal of the American Statistical Association*, 64(328):1183–1210. https://doi.org/10.1080/01621459.1969.10501049.

Gao, Y., Xiong, Y., Gao, X., Jia, K., Pan, J., Bi, Y., Dai, Y., Sun, J., Wang, M., and Wang, H. (2024). Retrieval-augmented generation for large language models: A survey. arXiv preprint arXiv:2312.10997v5. https://doi.org/10.48550/arXiv.2312.10997.

Google Cloud (2024). Get text embeddings with vertex ai. [Documentation] Retrieved October 1, 2025, from https://docs.cloud.google.com/vertex-ai/generative-ai/docs/embeddings/get-text-embeddings.

Google Cloud (2025). Gemini 2.5 flash-lite. Retrieved January 21, 2026, from https://docs.cloud.google.com/vertex-ai/generative-ai/docs/models/gemini/2-5-flash-lite.

GraphRAG-Bench (2025). Graphrag-benchmark (version 5ec02ce) [Source code]. Retrieved December 9, 2025, from https://github.com/GraphRAG-Bench/GraphRAG-Benchmark.

Gu, J., Jiang, X., Shi, Z., Tan, H., Zhai, X., Xu, C., Li, W., Shen, Y., Ma, S., Liu, H., Wang, S., Zhang, K., Lin, Z., Zhang, B., Ni, L., Gao, W., Wang, Y., and Guo, J. (2026). A survey on LLM-as-a-judge. *The Innovation*. https://doi.org/10.1016/j.xinn.2025.101253.

Guo, K., Dai, X., Zeng, S., Shomer, H., Han, H., Wang, Y., and Tang, J. (2025a). Beyond static retrieval: Opportunities and pitfalls of iterative retrieval in GraphRAG. arXiv preprint arXiv:2509.25530v1. https://doi.org/10.48550/arXiv.2509.25530.

Guo, Z., Xia, L., Yu, Y., Ao, T., and Huang, C. (2025b). LightRAG: Simple and fast retrieval-augmented generation. arXiv preprint arXiv:2410.05779v3. https://doi.org/10.48550/arXiv.2410.05779.

Gutiérrez, B. J., Shu, Y., Gu, Y., Yasunaga, M., and Su, Y. (2025). HippoRAG: Neurobiologically inspired long-term memory for large language models. In *Advances in Neural Information Processing Systems*, volume 37, pages 59532–59569. Curran Associates, Inc. https://doi.org/10.52202/079017-1902.

Gutiérrez, B. J., Shu, Y., Qi, W., Zhou, S., and Su, Y. (2025). From RAG to memory: Non-parametric continual learning for large language models. arXiv preprint arXiv:2502.14802v2. https://doi.org/10.48550/arXiv.2502.14802.

Han, H., Ma, L., Shomer, H., Wang, Y., Lei, Y., Guo, K., Hua, Z., Long, B., Liu, H., Aggarwal, C. C., and Tang, J. (2025a). RAG vs. GraphRAG: A systematic evaluation and key insights. arXiv preprint arXiv:2502.11371v2. https://doi.org/10.48550/arXiv.2502.11371.

Han, H., Wang, Y., Shomer, H., Guo, K., Ding, J., Lei, Y., Halappanavar, M., Rossi, R. A., Mukherjee, S., Tang, X., He, Q., Hua, Z., Long, B., Zhao, T., Shah, N., Javari, A., Xia, Y., and Tang, J. (2025b). Retrieval-Augmented Generation with Graphs (GraphRAG). arXiv preprint arXiv:2501.00309v2. https://doi.org/10.48550/arXiv.2501.00309.

He, X., Tian, Y., Sun, Y., Chawla, N. V., Laurent, T., LeCun, Y., Bresson, X., and Hooi, B. (2024). G-retriever: Retrieval-augmented generation for textual graph understanding and question answering. In *Advances in Neural Information Processing Systems*, volume 37, pages 132876–132907. Curran Associates, Inc. https://doi.org/10.52202/079017-4224.

Hirschman, L. and Gaizauskas, R. (2001). Natural language question answering: The view from here. *Natural Language Engineering*, 7(4):275–300. https://doi.org/10.1017/S1351324901002807.

Ho, X., Duong Nguyen, A.-K., Sugawara, S., and Aizawa, A. (2020). Constructing a multi-hop QA dataset for comprehensive evaluation of reasoning steps. In *Proceedings of the*

28th International Conference on Computational Linguistics, pages 6609–6625. International Committee on Computational Linguistics. https://doi.org/10.18653/v1/2020.coling-main.580.

Hogan, A., Blomqvist, E., Cochez, M., D'amato, C., Melo, G. D., Gutierrez, C., Kirrane, S., Gayo, J. E. L., Navigli, R., Neumaier, S., Ngomo, A.-C. N., Polleres, A., Rashid, S. M., Rula, A., Schmelzeisen, L., Sequeda, J., Staab, S., and Zimmermann, A. (2021). Knowledge graphs. *ACM Computing Surveys*, 54(4):1–37. http://dx.doi.org/10.1145/3447772.

Hu, Y., Zuo, X., Zhou, Y., Peng, X., Huang, J., Keloth, V. K., Zhang, V. J., Weng, R.-L., Shyr, C., Chen, Q., Jiang, X., Roberts, K. E., and Xu, H. (2026). Information extraction from clinical notes: Are we ready to switch to large language models? *Journal of the American Medical Informatics Association*. Advance online publication. https://doi.org/10.1093/jamia/ocaf213.

Huang, L. et al. (2025). A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions. *ACM Transactions on Information Systems*, 43(2):55. https://doi.org/10.1145/3703155.

Huang, X., Zhang, J., Li, D., and Li, P. (2019). Knowledge graph embedding based question answering. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*, page 105–113, New York, NY, USA. Association for Computing Machinery. https://doi.org/10.1145/3289600.3290956.

Ji, S., Pan, S., Cambria, E., Marttinen, P., and Yu, P. S. (2022). A survey on knowledge graphs: Representation, acquisition, and applications. *IEEE Transactions on Neural Networks and Learning Systems*, 33(2):494–514. http://dx.doi.org/10.1109/TNNLS.2021.3070843.

Jiang, Z., Xu, F., Gao, L., Sun, Z., Liu, Q., Dwivedi-Yu, J., Yang, Y., Callan, J., and Neubig, G. (2023). Active retrieval augmented generation. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 7969–7992, Singapore. Association for Computational Linguistics. https://doi.org/10.18653/v1/2023.emnlp-main.495.

Joshi, M., Choi, E., Weld, D., and Zettlemoyer, L. (2017). TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1601–1611, Vancouver, Canada. Association for Computational Linguistics. https://doi.org/10.18653/v1/P17-1147.

Kau, A., He, X., Nambissan, A., Astudillo, A., Yin, H., and Aryani, A. (2024). Combining knowledge graphs and large language models. arXiv preprint arXiv:2112.01716v1. https://doi.org/10.48550/arXiv.2407.06564.

Koch, B., Denton, E., Hanna, A., and Foster, J. G. (2021). Reduced, reused and recycled: The life of a dataset in machine learning research. arXiv preprint arXiv:2112.01716v1. https://doi.org/10.48550/arXiv.2112.01716.

Kong, A., Zhao, S., Chen, H., Li, Q., Qin, Y., Sun, R., Zhou, X., Wang, E., and Dong, X. (2024). Better zero-shot reasoning with role-play prompting. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 4099–4113, Mexico City, Mexico. Association for Computational Linguistics. https://doi.org/10.18653/v1/2024.naacl-long.228.

Kwiatkowski, T., Palomaki, J., Redfield, O., Collins, M., Parikh, A., Alberti, C., Epstein, D., Polosukhin, I., Devlin, J., Lee, K., Toutanova, K., Jones, L., Kelcey, M., Chang, M.-W., Dai, A. M., Uszkoreit, J., Le, Q., and Petrov, S. (2019). Natural questions: A benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:453–466. https://doi.org/10.1162/tacl_a_00276.

LangChain Community (2025). langchain-community (version 0.3.x) [software]. Retrieved November 27, 2025, from https://github.com/langchain-ai/langchain-community.

Lehmann, J., Isele, R., Jakob, M., Jentzsch, A., Kontokostas, D., Mendes, P. N., Hellmann, S., Morsey, M., van Kleef, P., Auer, S., and Bizer, C. (2015). DBpedia – a large-scale, multilingual knowledge base extracted from wikipedia. *Semantic Web*, 6(2):167–195. https://doi.org/10.3233/SW-140134.

Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W.-t., Rocktäschel, T., Riedel, S., and Kiela, D. (2020). Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Advances in Neural Information Processing Systems*, volume 33, pages 9459–9474. Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2020/file/6b493230205f780e1bc26945df7481e5-Paper.pdf.

Li, H., Dong, Q., Chen, J., Su, H., Zhou, Y., Ai, Q., Ye, Z., and Liu, Y. (2024a). Llms-as-judges: A comprehensive survey on llm-based evaluation methods. arXiv preprint arXiv:2412.05579v2. https://doi.org/10.48550/arXiv.2412.05579.

Li, H., Feng, L., Li, S., Hao, F., Zhang, C. J., and Song, Y. (2024b). On leveraging large language models for enhancing entity resolution: A cost-efficient approach. arXiv preprint arXiv:2401.03426v2. https://doi.org/10.48550/arXiv.2401.03426.

Li, H., Li, S., Hao, F., Zhang, C. J., Song, Y., and Chen, L. (2024c). BoostER: Leveraging large language models for enhancing entity resolution. In *Companion Proceedings of the ACM Web Conference 2024*, WWW '24, page 1043–1046, New York, NY, USA. Association for Computing Machinery. https://doi.org/10.1145/3589335.3651245.

Li, J., Sun, A., Han, J., and Li, C. (2022). A survey on deep learning for named entity recognition. *IEEE Transactions on Knowledge and Data Engineering*, 34(1):50–70. https://doi.org/10.1109/TKDE.2020.2981314.

Li, X., Liu, M., and Gao, S. (2024d). GRAMMAR: Grounded and modular methodology for assessment of closed-domain retrieval-augmented language model. arXiv preprint arXiv:2404.19232v7. https://doi.org/10.48550/arXiv.2404.19232.

Liang, L., Bo, Z., Gui, Z., Zhu, Z., Zhong, L., Zhao, P., Sun, M., Zhang, Z., Zhou, J., Chen, W., Zhang, W., and Chen, H. (2025a). KAG: Boosting LLMs in professional domains via knowledge augmented generation. In *Companion Proceedings of the ACM on Web Conference 2025*, WWW '25, page 334–343, New York, NY, USA. Association for Computing Machinery. https://doi.org/10.1145/3701716.3715240.

Liang, W., Zhang, Y., Codreanu, M., Wang, J., Cao, H., and Zou, J. (2025b). The widespread adoption of large language model-assisted writing across society. *Patterns*, 6(12):101366. https://doi.org/10.1016/j.patter.2025.101366.

Lifshitz, L. R. and Hung, R. (2024). BC tribunal confirms companies remain liable for AI chatbot-created information. Retrieved January 28, 2026, from https://www.torkin.com/insights/publication/bc-tribunal-confirms-companies-remain-liable-for-ai-chatbot-created-information.

Lin, C.-Y. (2004). ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics. https://aclanthology.org/W04-1013/.

Liu, N. F., Lin, K., Hewitt, J., Paranjape, A., Bevilacqua, M., Petroni, F., and Liang, P. (2024a). Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, 12:157–173. https://doi.org/10.1162/tacl_a_00638.

Liu, Y., He, H., Han, T., Zhang, X., Liu, M., Tian, J., Zhang, Y., Wang, J., Gao, X., Zhong, T., Pan, Y., Xu, S., Wu, Z., Liu, Z., Zhang, X., Zhang, S., Hu, X., Zhang, T., Qiang, N., Liu, T., and Ge, B. (2024b). Understanding LLMs: A comprehensive overview from training to inference. *Neurocomputing*, 620:129190. https://doi.org/10.1016/j.neucom.2024.129190.

Liusie, A., Manakul, P., and Gales, M. (2024). LLM comparative assessment: Zero-shot NLG evaluation through pairwise comparisons using large language models. In *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 139–151, St. Julian's, Malta. Association for Computational Linguistics. https://doi.org/10.18653/v1/2024.eacl-long.8.

LLM Stats (2026). LLM stats: AI leaderboards and model comparison. Retrieved January 21, 2026, from https://llm-stats.com/.

Lu, Y., Chen, L., Zhang, Y., Shen, M., Wang, H., Wang, X., van Rechem, C., Fu, T., and Wei, W. (2025). Machine learning for synthetic data generation: A review. arXiv preprint arXiv:2302.04062v10 https://doi.org/10.48550/arXiv.2302.04062.

Luo, L., Zhao, Z., Liu, J., Qiu, Z., Dong, J., Panev, S., Gong, C., Vu, T.-T., Haffari, G., Phung, D., Liew, A. W.-C., and Pan, S. (2025). G-reasoner: Foundation models for unified reasoning over graph-structured knowledge. arXiv preprint arXiv:2509.24276v1. https://doi.org/10.48550/arXiv.2509.24276.

Manchanda, J., Boettcher, L., Westphalen, M., and Jasser, J. (2025). The open source advantage in large language models (LLMs). arXiv preprint arXiv:2412.12004v3. https://doi.org/10.48550/arXiv.2412.12004.

Mao, X., Sun, H., Zhu, X., and Li, J. (2022). Financial fraud detection using the related-party transaction knowledge graph. *Procedia Computer Science*, 199:733–740. https://doi.org/10.1016/j.procs.2022.01.091.

Microsoft (2026). Graphrag-benchmarking-datasets/data at main (commit bc187ac) [GitHub repository]. Retrieved January 18, 2026, form https://github.com/microsoft/graphrag-benchmarking-datasets/tree/main/data.

Microsoft (2026). Methods – GraphRAG. Retrieved January 8, 2026, from https://microsoft.github.io/graphrag/index/methods/.

Miller, F. P., Vandome, A. F., and McBrewster, J. (2009). Levenshtein distance: Information theory, computer science, string (computer science), string metric, damerau–levenshtein distance, spell checker, hamming distance. page 68. Alpha Press. https://api.semanticscholar.org/CorpusID:64130157.

Miller, K. (2023). How do we fix and update large language models? Retrieved 5 January, 2026, from https://engineering.stanford.edu/news/how-do-we-fix-and-update-large-language-models.

Min, C., Bansal, S., Pan, J., Keshavarzi, A., Mathew, R., and Kannan, A. V. (2025). Towards practical GraphRAG: Efficient knowledge graph construction and hybrid retrieval at scale. arXiv preprint arXiv:2507.03226v3. https://doi.org/10.48550/arXiv.2507.03226.

Minaee, S., Mikolov, T., Nikzad, N., Chenaghlu, M., Socher, R., Amatriain, X., and Gao, J. (2025). Large language models: A survey. arXiv preprint arXiv:2402.06196v3. https://doi.org/10.48550/arXiv.2402.06196.

Mistral AI (2025a). Medium is the new large. Retrieved January 21, 2026, from https://mistral.ai/news/mistral-medium-3/.

Mistral AI (2025b). Mistral medium 3 (model identifier: mistral-medium-2505). Retrieved January 10, 2026, from https://docs.mistral.ai/models/mistral-medium-3-25-05.

Mistral AI (2025c). Mistral medium 3.1. Frontier-class multimodal language model. https://artificialanalysis.ai/models/mistral-medium-3-1.

Monteiro, J. a., Noël, P.-A., Marcotte, E., Rajeswar, S., Zantedeschi, V., Vázquez, D., Chapados, N., Pal, C., and Taslakian, P. (2024). RepLiQA: A question-answering dataset for benchmarking LLMs on unseen reference content. https://proceedings.neurips.cc/paper_files/paper/2024/file/2b23626015b6311369e95a70735cbb72-Paper-Datasets_and_Benchmarks_Track.pdf.

Mukherjee, M., Kim, S., Chen, X., Luo, D., Yu, T., and Mai, T. (2025). From documents to dialogue: Building KG-RAG enhanced ai assistants. arXiv preprint arXiv:2502.15237v1. https://doi.org/10.48550/arXiv.2502.15237.

OpenAI (2025). GPT-4o mini model. [Blog post]. Retrieved January 21, 2026, from https://platform.openai.com/docs/models/gpt-4o-mini.

OpenAI (2025). Tiktoken: A tokenizer for openai models [software]. GitHub. Retrieved December 1, 2025, from https://github.com/openai/tiktoken.

Papadakis, G., Skoutas, D., Thanos, E., and Palpanas, T. (2020). Blocking and filtering techniques for entity resolution: A survey. *ACM Comput. Surv.*, 53(2). https://doi.org/10.1145/3377455.

Patterson, D., Gonzalez, J., Le, Q., Liang, C., Munguia, L.-M., Rothchild, D., So, D., Texier, M., and Dean, J. (2021). Carbon emissions and large neural network training. arXiv preprint arXiv:2104.10350v3. https://doi.org/10.48550/arXiv.2104.10350.

Peeters, R., Steiner, A., and Bizer, C. (2024). Entity matching using large language models. arXiv preprint arXiv:2310.11244v4. https://doi.org/10.48550/arXiv.2310.11244.

Peng, B., Zhu, Y., Liu, Y., Bo, X., Shi, H., Hong, C., Zhang, Y., and Tang, S. (2025). Graph retrieval-augmented generation: A survey. volume 44, pages 1–52, New York, NY, USA. Association for Computing Machinery. https://doi.org/10.1145/3777378.

Peng, C., Xia, F., Naseriparsa, M., and Osborne, F. (2023). Knowledge graphs: Opportunities and challenges. https://doi.org/10.1007/s10462-023-10465-9.

Potts, B. (2024). LazyGraphRAG sets a new standard for GraphRAG quality and cost. [Blog post]. Microsoft Research. Retrieved January 28, 2026 from https://www.microsoft.com/en-us/research/blog/lazygraphrag-setting-a-new-standard-for-quality-and-cost/.

Procko, T. T. and Ochoa, O. (2024). Graph retrieval-augmented generation for large language models: A survey. In *2024 Conference on AI, Science, Engineering, and Technology (AIxSET)*, pages 166–169. https://doi.org/10.1109/AIxSET62544.2024.00030.

Puri, R., Spring, R., Shoeybi, M., Patwary, M., and Catanzaro, B. (2020). Training question answering models from synthetic data. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5811–5826. https://doi.org/10.18653/v1/2020.emnlp-main.468.

Qwen Team (2025). Qwen3-coder-480b-a35b-instruct [Computer software]. https://artificialanalysis.ai/models/qwen3-coder-480b-a35b-instruct.

Rajpurkar, P., Zhang, J., Lopyrev, K., and Liang, P. (2016). SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392. https://doi.org/10.18653/v1/D16-1264.

Raza, M., Jahangir, Z., Riaz, M. B., Saeed, M. J., and Sattar, M. A. (2025). Industrial applications of large language models. *Scientific Reports*, 15(1):13755. https://doi-org.ludwig.lub.lu.se/10.1038/s41598-025-98483-1.

Renze, M. (2024). The effect of sampling temperature on problem solving in large language models. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, page 7346–7356, Miami, Florida, USA. Association for Computational Linguistics. http://dx.doi.org/10.18653/v1/2024.findings-emnlp.432.

Saad-Falcon, J., Khattab, O., Potts, C., and Zaharia, M. (2024). ARES: An automated evaluation framework for retrieval-augmented generation systems. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 338–354, Mexico City, Mexico. Association for Computational Linguistics. https://doi.org/10.18653/v1/2024.naacl-long.20.

Salinas, D., Swelam, O., and Hutter, F. (2025). Tuning LLM judge design decisions for 1/1000 of the cost. arXiv preprint arXiv:2501.17178v4. https://doi.org/10.48550/arXiv.2501.17178.

Sarmah, B., Mehta, D., Hall, B., Rao, R., Patel, S., and Pasquali, S. (2024). HybridRAG: Integrating knowledge graphs and vector retrieval augmented generation for efficient information extraction. In *Proceedings of the 5th ACM International Conference on AI in Finance*, ICAIF '24, page 608–616, New York, NY, USA. Association for Computing Machinery. https://doi.org/10.1145/3677052.3698671.

Sarthi, P., Abdullah, S., Tuli, A., Khanna, S., Goldie, A., and Manning, C. D. (2024). RAPTOR: Recursive abstractive processing for tree-organized retrieval. arXiv preprint arXiv:2401.18059v1. https://doi.org/10.48550/arXiv.2401.18059.

Schmidt, M., Bartezzaghi, A., and Vu, N. T. (2024). Prompting-based synthetic data generation for few-shot question answering. arXiv preprint arXiv:2405.09335v1. https://doi.org/10.48550/arXiv.2405.09335.

Sclar, M., Choi, Y., Tsvetkov, Y., and Suhr, A. (2024). Quantifying language models' sensitivity to spurious features in prompt design or: How i learned to start worrying about prompt formatting. arXiv preprint arXiv:2310.11324v2. https://doi.org/10.48550/arXiv.2310.11324.

Shao, Z., Gong, Y., Shen, Y., Huang, M., Duan, N., and Chen, W. (2023). Enhancing retrieval-augmented large language models with iterative retrieval-generation synergy. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 9248–9274, Singapore. Association for Computational Linguistics. https://doi.org/10.18653/v1/2023.findings-emnlp.620.

Siddiq, M. L., Islam-Gomes, A., Sekerak, N., and Santos, J. C. S. (2025). Large language models for software engineering: A reproducibility crisis. arXiv preprint arXiv:2512.00651v1. https://doi.org/10.48550/arXiv.2512.00651.

Statistikmyndigheten SCB (2025). Artificial intelligence in enterprises 2025. https://www.scb.se/en/finding-statistics/statistics-by-subject-area/research-and-the-digital-society/ovrigt/artificial-intelligence-in-sweden/pong/statistical-news/artificiell-intelligens-i-sverige-2025/.

Takahashi, K., Omi, T., Arima, K., and Ishigaki, T. (2023). Training generative question-answering on synthetic data obtained from an instruct-tuned model. arXiv preprint arXiv:2310.08072v2. https://doi.org/10.48550/arXiv.2310.08072.

Tang, Y. and Yang, Y. (2024). MultiHop-RAG: Benchmarking retrieval-augmented generation for multi-hop queries. arXiv preprint arXiv:2401.15391v1. https://doi.org/10.48550/arXiv.2401.15391.

Team, Q. (2025). Qwen3-coder-480b-a35b-instruct. [Technical Report]. https://huggingface.co/Qwen/Qwen3-Coder-480B-A35B-Instruct.

The Associated Press (2025). Conservative activist robby starbuck suing meta over ai responses about him. Retrieved January 28, 2026, from https://apnews.com/article/robby-starbuck-meta-ai-delaware-eb587d274fdc18681c51108ade54b095.

Traag, V. A., Waltman, L., and van Eck, N. J. (2019). From louvain to leiden: guaranteeing well-connected communities. *Scientific Reports*, 9:5233. http://dx.doi.org/10.1038/s41598-019-41695-z.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, 30.

Verga, P., Hofstatter, S., Althammer, S., Su, Y., Piktus, A., Arkhangorodsky, A., Xu, M., White, N., and Lewis, P. (2024). Replacing judges with juries: Evaluating LLM generations with a panel of diverse models. arXiv preprint arXiv:2404.18796v2. https://doi.org/10.48550/arXiv.2404.18796.

Vrandečić, D. and Krötzsch, M. (2014). Wikidata: a free collaborative knowledgebase. *Commun. ACM*, 57(10):78–85. https://doi.org/10.1145/2629489.

Wang, S., Fang, Y., Zhou, Y., Liu, X., and Ma, Y. (2025). ArchRAG: Attributed community-based hierarchical retrieval-augmented generation. arXiv preprint arXiv:2502.09891v3. https://doi.org/10.48550/arXiv.2502.09891.

Wang, S., Liu, J., Song, S., Cheng, J., Fu, Y., Guo, P., Fang, K., Zhu, Y., and Dou, Z. (2024a). DomainRAG: A chinese benchmark for evaluating domain-specific retrieval-augmented generation. arXiv preprint arXiv:2406.05654v2. https://doi.org/10.48550/arXiv.2406.05654.

Wang, Y., Lipka, N., Rossi, R. A., Siu, A., Zhang, R., and Derr, T. (2024b). Knowledge graph prompting for multi-document question answering. In *Proceedings of the Thirty-Eighth AAAI Conference on Artificial Intelligence*, volume 38, pages 2141–2149. https://doi.org/10.1609/aaai.v38i17.29889.

Wei, H., He, S., Xia, T., Liu, F., Wong, A., Lin, J., and Han, M. (2025). Systematic evaluation of LLM-as-a-judge in LLM alignment tasks: Explainable metrics and diverse prompt templates. arXiv preprint arXiv:2408.13006v2. https://doi.org/10.48550/arXiv.2408.13006.

Wissler, L., Al-Mashraee, M., Monett, D., and Paschke, A. (2014). The gold standard in corpus annotation. In *Proceedings of the 5th IEEE Germany Student Conference*. https://doi.org/10.13140/2.1.4316.3523.

Wu, J., Zhu, J., Qi, Y., Chen, J., Xu, M., Menolascina, F., and Grau, V. (2024). Medical graph RAG: Towards safe medical large language model via graph retrieval-augmented generation. arXiv preprint arXiv:2408.04187v2. https://doi.org/10.48550/arXiv.2408.04187.

Xiang, Z., Wu, C., Zhang, Q., Chen, S., Hong, Z., Huang, X., and Su, J. (2025). When to use graphs in RAG: A comprehensive analysis for graph retrieval-augmented generation. arXiv preprint arXiv:2506.05690v2. https://doi.org/10.48550/arXiv.2506.05690 Accepted for publication at ICLR 2026 (OpenReview, Jan 2026).

Xiao, Q., Tsang, H. T., and Bai, J. (2025). TERAG: Token-efficient graph-based retrieval-augmented generation. arXiv preprint arXiv:2509.18667v3. https://doi.org/10.48550/arXiv.2509.18667.

Yang, Z., Qi, P., Zhang, S., Bengio, Y., Cohen, W., Salakhutdinov, R., and Manning, C. D. (2018). HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2369–2380, Brussels, Belgium. https://doi.org/10.18653/v1/D18-1259.

Yasunaga, M., Ren, H., Bosselut, A., Liang, P., and Leskovec, J. (2021). QA-GNN: Reasoning with language models and knowledge graphs for question answering. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 535–546. https://doi.org/10.18653/v1/2021.naacl-main.45.

Yu, H., Gan, A., Zhang, K., Tong, S., Liu, Q., and Liu, Z. (2025a). *Evaluation of Retrieval-Augmented Generation: A Survey*, volume 2301. Springer, Singapore. http://dx.doi.org/10.1007/978-981-96-1024-2_8.

Yu, T., Zhou, W., Yang, L., Shukla, A., Madugula, M., Gundecha, P., Burnett, N., Xu, A., Seth, V., Bar, T., Akkiraju, R., and Zhang, V. (2025b). EKRAG: Benchmark RAG for enterprise knowledge question answering. In *Proceedings of the 4th International Workshop on Knowledge-Augmented Methods for Natural Language Processing*, pages 152–159, Albuquerque, New Mexico, USA. https://doi.org/10.18653/v1/2025.knowledgenlp-1.13.

Yuan, J., Li, H., Ding, X., Xie, W., Li, Y.-J., Zhao, W., Wan, K., Shi, J., Hu, X., and Liu, Z. (2025). Understanding and mitigating numerical sources of nondeterminism in llm inference. arXiv preprint arXiv:2506.09501v2. https://doi.org/10.48550/arXiv.2506.09501.

Zhang, B. and Soh, H. (2024). Extract, define, canonicalize: An LLM-based framework for knowledge graph construction. arXiv preprint arXiv:2404.03868v2. https://doi.org/10.48550/arXiv.2404.03868.

Zhang, Q., Chen, S., Bei, Y., Yuan, Z., Zhou, H., Hong, Z., Chen, H., Xiao, Y., Zhou, C., Dong, J., Chang, Y., and Huang, X. (2025). A survey of graph retrieval-augmented generation for customized large language models. arXiv preprint arXiv:2501.13958v3. https://doi.org/10.48550/arXiv.2501.13958.

Zhang, T., Kishore*, V., Wu, F., Weinberger, K. Q., and Artzi, Y. (2020). BERTScore: Evaluating text generation with BERT. In *International Conference on Learning Representations*. arXiv preprint arXiv:1904.09675v3. https://doi.org/10.48550/arXiv.1904.09675.

Zhao, W. X., Zhou, K., Li, J., Tang, T., Wang, X., Hou, Y., Min, Y., Zhang, B., Zhang, J., Dong, Z., Du, Y., Yang, C., Chen, Y., Chen, Z., Jiang, J., Ren, R., Li, Y., Tang, X., Liu, Z., Liu, P., Nie, J.-Y., and Wen, J.-R. (2025). A survey of large language models. arXiv preprint arXiv:2303.18223v16. https://doi.org/10.48550/arXiv.2303.18223.

Zheng, L., Chiang, W.-L., Sheng, Y., Zhuang, S., Wu, Z., Zhuang, Y., Lin, Z., Li, Z., Li, D., Xing, E., et al. (2023). Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in neural information processing systems*, 36:46595–46623. https://proceedings.neurips.cc/paper_files/paper/2023/file/91f18a1287b398d378ef22505bf41832-Paper-Datasets_and_Benchmarks.pdf.

Zheng, Y., Yang, D., Li, J., Shang, L., Chen, L., Xu, J., and Luan, S. (2025). Less is more: Denoising knowledge graphs for retrieval augmented generation. arXiv preprint arXiv:2510.14271v1. https://arxiv.org/abs/2510.14271.

Zhou, A., Xu, X., Raghunathan, R., Lal, A., Guan, X., Yu, B., and Li, B. (2024). KnowGraph: Knowledge-enabled anomaly detection via logical reasoning on graph data. In *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security*, CCS '24, page 168–182, New York, NY, USA. Association for Computing Machinery. https://doi.org/10.1145/3658644.3690354.

Zhou, Y., Su, Y., Sun, Y., Wang, S., Wang, T., He, R., Zhang, Y., Liang, S., Liu, X., Ma, Y., and Fang, Y. (2025). In-depth analysis of graph-based RAG in a unified framework. arXiv preprint arXiv:2503.04338v1. https://doi.org/10.48550/arXiv.2503.04338.

Zhu, Y., Wang, X., Chen, J., Qiao, S., Ou, Y., Yao, Y., Deng, S., Chen, H., and Zhang, N. (2024). LLMs for knowledge graph construction and reasoning: Recent capabilities and future opportunities. *World Wide Web*, 27(5):58. https://doi.org/10.1007/s11280-024-01297-w.

Appendices

Appendix A

Nomenclature

LLM	Large Language model
RAG	Retrival Augumented Generation
KG	Knowledge Graph
NLP	Natural Language Processing
GPT	Generative Pre-trained Transformer
AI	Artificial Intelligence
ER	Entity Resolution
QA	Question-Answering
KGC	Knowledge Graph Construction

Appendix B

Prompts

B.1 Prompt P1: Triple Extraction

The following system prompt was used to extract triples, e.g. subject-relation-object pairs, from text chunks of the corpus.

```
You are an information extraction assistant specialized in product
    and technical documentation.
Your task is to extract factual subjectrelationobject triples from
    the provided text.

CORE INSTRUCTIONS:
- Extract all factual relationships as subject-relation-object
    triples from text provided by the user.
- Focus on clear, atomic facts: e.g. organizations, people,
    locations, products, and relations between them.
- Include optional short descriptions for subject, relation and
    object if available.
- Do not include fields with null or empty values.

RESPONSE STYLE:
Return ONLY compact JSON with no whitespace, newlines, or formatting.
Output should only contain valid JSON objects and nothing else (no
    markdown, no explanation, no extra text).
JSON must strictly follow this schema and be syntactically valid
    (double quotes, no trailing commas). Use exactly the same key
```

```
    names as shown.

## Example format:
[{{"s":"subject","r":"relation","o":"object","s_description":"optional",
"r_description":"optional","o_description":"optional"}}]

## Example:
Input Text: Tesla manufactures electric cars in the United States.
    Elon Musk is the CEO of Tesla.
Output: [{{"s":"Tesla","r":"manufactures","o":"electric
    cars","s_description":"car company","o_description":"vehicles
    powered by electricity"}},{{"s":"Elon Musk","r":"is CEO
    of","o":"Tesla","s_description":"entrepreneur"}}]
```

B.2 Prompt P2: Entity Resolution prompt

The following prompt was used for the LLM-based entity merging.

```
You are a data processing assistant. Your task is to identify
    duplicate entities by analyzing their names, descriptions, and
    usage contexts.

Rules for identifying duplicates:
1. Entities with minor typographical differences should be considered
    duplicates.
2. Entities with different formats but the same content should be
    considered duplicates.
3. Entities that refer to the same real-world object or concept, even
    if described differently, should be considered duplicates.
4. Do NOT merge if they refer to different concepts, numbers, dates,
    or products.
5. Pay attention to the context windows - if the entities appear in
    very different contexts, they are likely different entities.

Respond with three parts separated by a comma:
1. "true" or "false" for whether to merge
2. If merging, respond with "1" or "2" to indicate which entity name
    to keep, choose the most informative and explaining one
3. If merging, provide a description for the merged entitys OR none
   - Use existing descriptions if they are informative, adds new
    information about the entity, and is not redundant
   - If both descriptions are missing, or redundant and repeating the
    entity name, respond with "none"
   - If not already mentioned in the entity name, include
    abbreviations and alternative names of the entity in the
```

```
    description
   - If the merged entities provides two alternative names for the
    same concept/object/subject, include the secondary name in the
    description
   - The description should be very consise and brief
   - Focus on not losing any information after merging the nodes

Example responses:
- "true, 1, Founder and CEO of Tesla"
- "true, 2, Also referred to as MJ"
- "true, 1, none"
- "false"

No explanation or extra text.
```

B.3 Prompt P3: Community Generation

The following system prompt was used to generate the community summaries.

```
You are a helpful assistant responsible for writing a comprehensive
    summary of the data provided below.
```

We used a different user prompt depending on what level of the community the summary where produced for.

Summaries for communities of level 0 where generated using the following user prompt.

```
Generate a comprehensive but readable summary based on a list of
    facts (and possible additional context if helpfull).

Rules:
- Include every distinct fact.
- Do not invent facts.
- 5-10 sentences; technical, neutral tone.

INPUT:
{community_info}

OUTPUT:
A single natural-language summary (no preamble).
```

Summaries for communities of level 1 and higher where generated using the following user prompt.

```
Generate a clear, concise summary that integrates the provided
    textual summaries into a cohesive overview.

Rules:
- Use only the given input summaries as source material; do not add
    new facts.
- Combine and reconcile the input summaries for coherence and
    completeness.
- Highlight key themes, relationships, or implications emerging from
    the combined content.
- Include at most one sentence about the overall significance or
    impact based on the input.
- Limit output to 510 sentences in a technical, neutral tone.

INPUT:
{community_info}

OUTPUT: A single natural-language paragraph synthesizing the input
    summaries without introducing unsupported information.
```

B.4 Prompt P4: QA prompt for RAG and GraphRAG

The same prompt was used for both the RAG and GraphRAG for question-answering.

```
--Role--
You are a helpful assistant responding to user queries.

--Goal--
Generate direct and concise answers based strictly on the provided
    knowledge base.
Respond in plain text without explanations or formatting.
Maintain conversation continuity and use the same language as the
    query.
If the answer is unknown, respond with "I don’t know".
```

This second prompt was used in the answer generation on the HR Axis data.

```
    You are a factual question-answering assistant that uses provided
    context to answer questions accurately and concisely.

## Core Instructions:
- Focus on answering the question directly and do not include any
```

```
    unnecessary additional context.
- Answer strictly using the information in the context chunks
- When possible, respond with a single word, name, or short sentence
- If the information is insufficient, respond exactly with "Not
    enough information"
- Do not explain your answer or reasoning

## Response Guidelines:
- Be precise and factual
- Use the same language as the query.
- Do not speculate or add information not present in the context

## User Input Format:
The user will provide:

QUESTION: <contents of question>

--Context--
-<first context chunk>
-<second context chunk>
-<third context chunk>
...
```

B.5 Prompt P5: LLM-as-a-judge prompt

The following prompt was used for our own LLM-as-a-judge implementation applied to the Axis data. The LLM was prompted to return a score from 1-5 according to some stated evaluation criteria.

```
You are an expert evaluator that assesses how well produced answers
    respond to questions based on correct reference answers.

## Your Task:
Evaluate the quality of a produced answer by comparing it to the
    correct answer for a given question.

## User Input Format:
The user will provide:
- QUESTION: The original question being asked
- PRODUCED ANSWER: The answer generated by a system that needs
    evaluation
- CORRECT ANSWER: The reference/ground truth answer

## Evaluation Criteria:
Rate the produced answer on a scale of 1-5 based on:
```

```
- **Factual accuracy**: How well it matches the correct answer
- **Completeness**: Whether it addresses all important aspects
- **Relevance**: How directly it answers the question
- **Overall quality**: Clarity and usefulness of the response

## Rating Scale:
- **1 = Poor**: Completely incorrect, irrelevant, or contradicts the
    correct answer. Answers claiming "Not enough information" when the
    correct answer demonstrates the question can be
  answered should receive the lowest score.
- **2 = Below Average**: Partially correct but has significant errors
    or omissions
- **3 = Average**: Generally correct but missing important details or
    has minor inaccuracies
- **4 = Good**: Accurate and relevant, with only minor gaps or
    differences in expression
- **5 = Excellent**: Comprehensive, accurate, and fully addresses the
    question

## Response Format:
Provide your evaluation in exactly this format:
‘‘‘
RATING: [number from 1-5]
```

B.6 Prompt P6: Synthetic question generation prompts

The three following prompt were used for synthetic QA pair data generation.

Prompt following an answer-guided strategy.

```
You are a QA generator. For each answer below, generate one clear
    question that is answerable from the text. Generate at most 3 QA
    pairs.

Answers: {answers}

Text:
{text}

Return the response as a JSON array with this format:
[{"question": "...", "answer": "..."}, ...]
```

Prompt with zero-shot strategy for QA pair generation.

```
Generate exactly 3 question-answer pairs based on the text. Questions
    must be specific and answerable from the text.

Text:
{text}

Return the response as a JSON array with this format:
[{"question": "...", "answer": "..."}, ...]
```

Prompt for multi-hop QA pair generation.

```
Generate 3 multi-hop reasoning questions. Each question must require
    combining at least two different facts from the text to answer.
Text:
{text}
Return the response as a JSON array with this format:[{"question":
    "...", "answer": "..."}, ...]
```

Appendix C

Example JSON objects

An example of a triple extracted from Axis's HR data set.

```
{
  "s": "Axis",
  "r": "provides access to",
  "o": "Convini vending machines",
  "s_description": "organization",
  "o_description": "vending machines for food and beverages",
  "chunk_id": 25,
  "document_id": 15,
  "document_filename": "food-and-beverage.md",
  "document_chunk_index": 2,
  "article_url": "xxx"
}
```

Example of a synthetically generated QA pair, the produced response from GraphRAG, and judges' scores (1–5) on Axis's HR dataset.

```
{
  "question": "What items are included in the Axis baby kit?",
  "produced_answer": "The Axis baby kit includes two drool bibs,
  a baby body in size 68, and a toiletry bag.",
  "correct_answer": "The kit includes a toiletry bag,
  a baby body (size 68), and two drool bibs.",
  "judge_scores": {
    "judge_1": {
```

```
      "scores": [
        5.0
      ]
    },
    "judge_2": {
      "scores": [
        5.0
      ]
    },
    "judge_3": {
      "scores": [
        5.0
      ]
    }
  }
}
```

INSTITUTIONEN FÖR DATAVETENSKAP | LUNDS TEKNISKA HÖGSKOLA | PRESENTERAD 2026-02-17

EXAMENSARBETE Knowledge Graph-Enhanced RAG for Enterprise Question-Answering Systems
STUDENTER Ebba Rakuljic Feldt, Elin Hellström
HANDLEDARE Marcus Klang (LTH)
EXAMINATOR Jacek Malec (LTH)

Grafer som hjälper AI förstå intern data

POPULÄRVETENSKAPLIG SAMMANFATTNING **Ebba Rakuljic Feldt, Elin Hellström**

Många företag sitter idag på stora mängder intern information och data som kan vara svår att överblicka. AI-assistenter och chatbotar används i allt större utsträckning för att besvara frågor kring intern företagsdata. Detta ställer högre krav på systemens förmåga att förstå relationer i datan, utan att hitta på information som riskerar att förvilla användaren. I detta examensarbete undersöker vi om AI kan ge bättre och mer korrekta svar genom att använda kunskap som organiserats i kunskapsgrafer.

Stora språkmodeller, såsom ChatGPT, är generalister som tränas på stora mängder offentlig textdata från nätet. Modellen har bara kunskap som finns i dess träningsdata, och riskerar därmed att snabbt bli inaktuell. Vill man att modellerna även ska kunna använda intern data, såsom privat företagsdata eller domänspecifik information, måste man utöka modellens kunskapsbas. Traditionellt görs detta genom att träna om modellen på ny data, men detta är en resurskrävande process. De mest framstående modellerna idag är dessutom stängda modeller som inte går att bygga vidare på. När en språkmodell saknar kunskap riskerar den att hitta på svar, ett fenomen som kallas hallucinationer. En lösning på dessa problem är RAG (Retrieval-Augmented Generation). Modellen får då relevant kontext innan den svarar, vilket utökar dess kunskapsbas och minskar hallucinationer. RAG kan dock ha svårt för mer komplexa eller sammanfattande frågor som kräver resonemang och en djupare förståelse för datan.

I detta arbete undersöker vi därför en variant av RAG, där data organiseras i en graf med noder (t.ex. personer eller koncept) och relationer mellan dessa. Vidare jämför vi grafbaserad RAG mot traditionell RAG. Vi undersöker också tillförlitligheten i att syntetiskt generera testdataset och i att använda språkmodeller för utvärdering, tekniker vi tror kan påskynda införandet av RAG-system inom industrin, där referensdata ofta saknas.

Våra resultat visar att traditionell RAG fortfarande presterar bättre än grafbaserad RAG på fråge-svarsdata, samtidigt som den är enklare och mer kostnadseffektiv att sätta i drift. Resultaten utesluter inte att grafbaserad RAG kan vara användbar för vissa typer av frågor eller att grafer har andra fördelar, till exempel för att visualisera data. Vi bedömer att kunskapsgrafer är lovande för RAG, men att det krävs mer forskning och innovation för att se om dessa system kan överträffa traditionell RAG. Vi konstaterar också att både syntetisk datagenerering och språkmodeller är lovande vid utvärdering av AI-system där annoterad valideringsdata saknas.