Reducing Hallucinations in Complex Question Answering using Simple Graph-based Retrieval-Augmented Generation

Christopher J. Wedge, Joshua Stutter, Danny Dixon, Jacek Cała
National Innovation Centre for Data, Newcastle University
Newcastle upon Tyne, UK
{Chris.Wedge,Joshua.Stutter,D.J.Dixon2,Jacek.Cala}@ncl.ac.uk

ABSTRACT

Large language models (LLMs) have fundamentally transformed the landscape of Natural Language Processing, although they remain susceptible to errors. Retrieval-augmented generation (RAG) systems have emerged as a common deployment scenario seeking to both avoid the well known risk of the LLM "hallucinating" information, and to enable reasoning and question answering over proprietary information that the LLM did not have access to during training without resorting to expensive model fine-tuning.

In this work, we explore the idea of using a lightweight graph structure with a relatively simple graph schema, to support the RAG subsystem via a dedicated toolset. We design an agentic system with a variety of vector search and graph query tools operating over a structured dataset based on a curated subset of English Wikipedia articles, and evaluate its performance on questions from MoNaCo, a challenging Wikipedia based benchmark of complex question answering (QA) tasks.

Our results show that the introduction of graph-based tools can significantly increase the precision and recall of factual correctness, can halve the number of hallucinated answers, and achieves the highest fine-grained truthfulness score among the three evaluated scenarios. All this with a modest increase in token usage.

VLDB Workshop Reference Format:
Christopher J. Wedge, Joshua Stutter, Danny Dixon, Jacek Cała. Reducing Hallucinations in Complex Question Answering using Simple Graph-based Retrieval-Augmented Generation. VLDB 2026 Workshop: Agent+Graph.

VLDB Workshop Artifact Availability:
The source code, data, and/or other artifacts have been made available at https://github.com/NICD-UK/graph-based-rag-qa/.

1 INTRODUCTION

Since the advent of the transformer architecture many competing Large language models (LLMs) have been developed with strong capabilities in understanding, reasoning, summarisation, and question answering. Their ability to generate coherent and contextually relevant responses has led to widespread adoption in many domains. Despite these advances, LLMs and LLM-based systems remain prone to a variety of failure modes.

Proceedings of the VLDB Endowment. ISSN 2150-8097.

Type I (false positive) and Type II (false negative) errors provide a useful framework for describing fundamental failure modes in machine-learning models, including LLMs. These failures can be severe: false positives manifest as hallucinated content, incorrect context, or inappropriate tool use, while false negatives lead to omissions, missed context, or failure to invoke the right tools. In practice, such errors often occur simultaneously in multiple stages of system operation [9]. Therefore, a valuable improvement reduces both error types simultaneously rather than merely shifting performance along the precision-recall tradeoff. This consideration becomes even more vital for complex question answering problems, such as multi-hop (MHQA), cross-document (CDQA) and multi-entity question answering (MEQA) [5, 29, 41], where knowledge from multiple sources must be retrieved, aggregated, and reasoned over in order to provide an answer.

As an example, consider the following question:

> *Can you name all the battles between the Dutch and English in the First, Second and Third Anglo-Dutch Wars, and list the victor of each battle?*

To answer this question, one needs to perform a sophisticated retrieval and reasoning process. In fact, it requires multi-entity and multi-hop reasoning, and cross-document access, all at once [34].

First, as with MEQA, the process must operate on a set of entities (in this case nations, wars, battles and victors) and needs to perform per-entity analysis and filtering. Second, as with MHQA, it requires hierarchical traversal, repeated relational expansion and nested decomposition. Finally, as with CDQA, it needs to fetch information from multiple sources: war documents, battle descriptions, and historical records. A single document is unlikely to answer this question; the information required is typically dispersed across dozens of documents.

These types of questions pose a significant challenge to current state-of-the-art LLM-based systems. There are multiple reasons for this, but in this work we focus on one aspect: Retrieval-Augmented Generation (RAG) [27]. RAG is a common element of many LLM-based systems, specifically those used for document processing, analytics, and question answering. In this context, we address complex QA problems, like the one presented above, using a unified vector and graph database with a series of pre-defined tools to improve retrieval from a knowledge base (KB). We compare our solution, vector+graph RAG, with simple vector RAG and zero-shot approaches. With agents prompted for safe refusal – instructed to answer 'unknown' when the necessary information cannot be found in the provided KB, our system more than halved the proportion of complex questions that the agent refused to answer. At the same

time, it improved the ratio of correct over incorrect answers compared to the zero-shot approach. Thus, it shows a promising direction of research in which graphs are coupled with more traditional vector-based retrieval methods.

The use of knowledge retrieval systems (RAG) to extend the capability of LLM agents is the preferred method by which general-purpose LLMs are applied to reason over specialised and / or proprietary knowledge [13]. By augmenting a user query with data queried by a knowledge retrieval system – data which may not be in the LLM's training data – the LLM can answer the question using in-context learning, extending its capabilities and reducing hallucinations. Whilst early RAG systems consisted of a single retrieval-then-answer pipeline [18], it is now typically expected that RAG systems should be agentic for all but the simplest queries, so as to allow the LLM to practise question decomposition, reflection, and follow-up or validation queries using Chain of Thought (CoT) reasoning. Limiting ourselves to systems querying a defined KB (rather than Internet search), the most commonly-used tool within agentic RAG is vector search, where a vector database of text chunks from source documents is queried by an embedding model to assess semantic similarity [4, 26]. Beyond this, a variety of other tools have been developed that purport to improve retrieval in terms of its accuracy and / or efficiency.

Among these new retrieval methodologies, those that employ graphs are gaining popularity. Rather than operating over unstructured, chunked text documents, a knowledge graph (KG) is either harnessed directly or created using an LLM from a set of related documents (commonly known as *GraphRAG* [10]). However, a major challenge with graph-based solutions is the creation of a useful KG. On the one hand, KBs with rich structures are potentially desirable, but they necessitate a complex schema and thus risk filling the fixed context window of an LLM [6]. On the other hand, small schemas are more easily digestible by an LLM, but may not faithfully and comprehensively capture the true status of the KB.

In this work, we explore the latter idea, i.e. using a lightweight graph structure, with a relatively simple graph schema, to support the RAG subsystem via a dedicated toolset. We adopt this very practical approach in which a KB is constructed from semi-structured documents, including only a high-level structure of document titles, section titles, and constituent paragraphs, with links to other paragraphs or documents. This scenario is well suited to many real-world KBs that comprise various document types but lack a sophisticated and / or rich KG representation, such as those proposed by other work [8].

The tools we developed use the *Neo4j* graph database engine and its accompanying *Cypher* query language[1] to traverse the graph of Wikipedia articles, article sections and section paragraphs. We compare our graph-based approach with a typical vector-based RAG system on a set of sophisticated queries that require gathering and reasoning over information from multiple sources (articles, sections, and paragraphs).

Although work has been done to evaluate vector RAG against *GraphRAG* methods [14, 24, 30, 39], in this work we are concerned instead with evaluating vector RAG against knowledge base question answering (KBQA) from the perspective of unstructured and semi-structured data. There is as yet no research that directly evaluates vector RAG against KBQA on structured data. To address this deficiency, we aim to answer the following research question:

> *Do queries over structured knowledge bases, such as knowledge graphs, improve the performance of complex QA over unstructured RAG?*

We design an agentic system with a variety of vector search and graph query tools operating over a structured dataset based on a curated subset of English Wikipedia articles, and evaluate its performance on a challenging Wikipedia QA benchmark (MoNaCo, *vide infra*). Our experiments measure both answer accuracy and token usage, with the objective of a strong system being to maximise answer accuracy while minimising token usage.

Additionally, we compare both solutions against a zero-shot approach that relies purely on the knowledge acquired during model training. Our results show that the introduction of graph-based RAG significantly reduces hallucinated content and improves truthfulness scores, all with only a modest increase in token usage compared to the more conventional vector RAG approach, and using a lightweight and easy to construct graph knowledge base.

In summary, our main contributions are as follows. We:

- Identify a complex-QA benchmark suitable for comparing structured and unstructured retrieval.
- Design a unified *Cypher*-based toolset that affords efficient navigation over a knowledge graph.
- Conduct extensive experiments on the efficacy of RAG and KBQA tools.
- Release an evaluation KG based on a significant proportion of English Wikipedia, suitable for both RAG and KBQA.[2]

2 BACKGROUND AND SCOPE

As has recently been observed, the expectations of LLM-based and agentic systems are increasing beyond simple question answering problems [13]. Indeed, real-world queries often require retrieving information from multiple sources, summarising and reasoning before an answer can be produced. In this context, our reliance on systems that may hallucinate their answers, even if only at some of the substages of the whole answering pipeline, is problematic and undermines trustworthiness of these systems. Therefore, the need to design and develop solutions that can reduce hallucination (confabulation) and improve trust are critical.

There have been various attempts to address this problem. They revolve around improving four main aspects: query understanding and decomposition, evidence gathering (i.e. information retrieval), reasoning, and response synthesis and grounding (c.f. Figure 1).

In this work we focus on the second step – information retrieval – and aim to verify the extent to which the introduction of a knowledge graph store with a simple document graph structure within a complex QA system can improve overall system performance. To accomplish this, we embed our QA system within an evaluation framework that includes a benchmark dataset and an evaluation step. Our goal is to evaluate the end-to-end factual correctness and truthfulness of the QA system. Our comparison is deliberately scoped to generic vector RAG versus vector+graph RAG

[1] https://neo4j.com

[2] https://github.com/NICD-UK/graph-based-rag-qa/

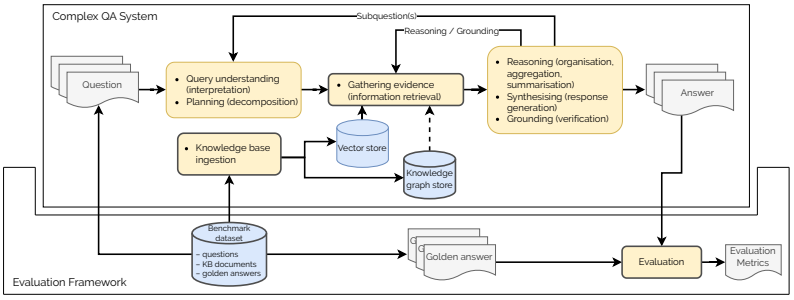


Figure 1: A Graph-based Question Answering pipeline embedded within evaluation framework. Highlighted in black are items of our focus.

over a simple document graph; more sophisticated *GraphRAG* [10] approaches and richer, LLM-constructed KG representations are beyond the scope of this study. Our simple graph is intended as an *enabler* of structure-aware retrieval rather than a competing heavyweight knowledge graph.

The following sections provide more details on these aspects.

2.1 Information Retrieval

Several paradigms have emerged within state-of-the-art RAG systems. Vector search is still the most common method, as it can be applied directly to unstructured text, such as a corpus of PDF documents on a topic. As a result, most RAG benchmarks and evaluations begin from the assumption that the initial data will be of an unstructured form [19]. For natively structured data, text-to-SQL [7, 28] or text-to-SPARQL [25, 33] are common techniques, where the LLM is tasked with directly querying a relational or graph database respectively. For the case of structured data querying over graphs, methods either:

- use graph metrics, such as community detection, alongside vector retrieval, to improve results by retrieving small subgraphs [10],
- verbalise entities and relationships into text for embedding [1], or
- generate structured queries, e.g. in SPARQL or *Cypher*, that are then executed directly against the graph [15].

Here, we propose an alternative approach. We expose the agent to specific tools, backed by handwritten queries, which enable flexible structured data retrieval while, at the same time, relieve the agent from the task of generating valid data queries – a potential source of failure and security vulnerability.

2.2 Benchmark dataset

An external knowledge base offers three advantages over relying on an LLM's parametric memory: it can be updated continuously rather than being frozen at training time [27, 32]; it supplies domain-specific or proprietary knowledge that the model lacks [23]; and it sidesteps the degraded performance of LLMs on long or complex inputs, since even large context windows are usably shorter than advertised [21]. Regardless of context size or parameter count, there will always be datasets too large or specialised for an LLM to reason over unaided.

Typical vector RAG and automatically-constructed *GraphRAG* both ingest *unstructured* documents. By contrast, a graph database operates on already-structured data, and its key advantage is a controlled schema: rather than *searching* for semantically similar text, it lets us *query* structure directly. Using this lens, we reviewed recent datasets and benchmarks to find one usable in a knowledge-graph RAG (KG RAG) system, employing the following criteria:

(1) Large dataset (>10,000 records); for a realistic scenario when KB does not fit into LLM context window.
(2) End-to-end QA; in contrast to retrieval only or generation only evaluation.
(3) Data appropriate for both structured and unstructured retrieval; for fair comparison between simple vector-based RAG and KG RAG.
(4) Multi-hop, multi-entity reasoning; to tackle the challenging task of complex QA which the current LLM-based systems struggle to solve.

A search within the literature for benchmark datasets for RAG, especially graph RAG, yielded many candidates. A summary of this survey is presented in [37, Appendix A].

In the end we selected the *MoNaCo* benchmark dataset [38]. Although it does not directly include a structured KB we built a simple knowledge graph to provide a structured index to the source documents (Wikipedia articles) necessary as context. This benchmark dataset, contains 1315 complex questions, one of which was shown earlier in the introduction. They are human written in natural language rather than being LLM-generated as in some alternatives. The answer to each question is a result to be determined by combining or reasoning over multiple pieces of information. The published, yet preliminary, studies of this dataset show poor performance by modern LLMs in a zero-shot or basic RAG context, which requires the use of LLM reasoning approaches [38]. Within an agentic vector RAG implementation, this might require multiple queries to retrieve the necessary information, but we postulate that using a graph RAG system it may be possible to retrieve information directly via a suitable graph query.

Although this dataset is less than ideal in that the QA pairs are based on publicly available Wikipedia data, the complexity of questions, and the reported failure of zero-shot LLMs to answer these accurately without a retrieval subsystem [38], ensure that the agent cannot rely solely on parametric knowledge.

2.3 Evaluation

The decision on the most appropriate RAG evaluation metrics must take into consideration the benchmark dataset in use, to ensure that the required information is available to compute the chosen metrics in both vector- and KG-RAG scenarios. For definitions of key metrics see the two recent reviews of RAG evaluation methods in [16, 42]. Here, we consider which of the many available metrics are appropriate.

RAG-supported systems comprise separate retrieval and generation components [27]. As a result, any evaluation experiment has the option to focus on only one of the two individual components, or the entire end-to-end pipeline [42]. For the results to be widely explainable, in this study we focus on metrics that are relevant to the end-to-end RAG system. We report a headline difference in overall performance when adding a KG to a RAG system, rather than focusing on a specific sub-component.

2.3.1 Traditional metrics. The simplest metrics for evaluating generated answers come from the domain of NLP and tend to work at a character level. These include Exact Match, BLEU and ROUGE. The challenge when applying such metrics to LLM output is that a correct answer, identical semantically but phrased differently to the reference answer, would be given a low score. They are, therefore, ineffective when evaluating free-form LLM responses [16, 40].

2.3.2 Embedding-based metrics. Metrics based on semantic similarity are more appropriate for our purpose, as they seek to evaluate whether the meaning of the generated and reference answers are the same. A typical example would be BERTScore, which generates embeddings from a BERT pre-trained transformer for both generated and reference answers, allowing token level similarity to be addressed [43]. Similar models include BART and RoBERTa [16]. These approaches have the advantage of capturing some semantic meaning, while still relying on a deterministic scoring method. However, they are less flexible and give lower evaluation quality than LLM prompting methods [17], and some exhibit specific blindspots [20].

2.3.3 LLM-based evaluators. The so-called *LLM-as-a-judge* methods rely on prompting an LLM to evaluate some material from the RAG chain against predefined criteria. This general approach can be used for a wide variety of different metrics assessing the RAG system at various stages; the required inputs to evaluate these are summarised in Table 1.

For example, the RAGAS framework [35] has *context precision* and *context recall* metrics, in which retrieved contexts are processed along with the user query and golden answer to determine whether they are useful in answering that query. The computation of these scores is achieved by decomposing the number of distinct claims within the contexts to assess precision and recall on a claim-by-claim basis, and the reference answer is used as a proxy for a golden context to allow evaluation even when a curated dataset containing the golden context is not available.

Two LLM-based metrics commonly seen to be important for RAG systems are *faithfulness* and *answer relevancy*. Both of these metrics can be computed using the user query and LLM-generated response without requiring a golden answer, and were introduced by RAGAS to address the specific challenge of RAG evaluation when a golden answer is not available [12]. *Answer relevancy* aims to measure alignment between the generated response and the user query, by generating three additional queries from the response and computing the average cosine similarity between the embedding of these queries and the actual query. *Faithfulness* additionally uses the retrieved contexts, and breaks down the individual claims in the LLM response to check these for support from the context. The ratio of supported claims to total claims is then returned as a score from 0 to 1 (e.g. a faithfulness score of 1 means that all claims in the answer are justified by the retrieved context).

Table 1: Summary of different LLM-based RAG evaluation metrics with required inputs and score range; underlined are metrics used in this work.

Metric	User Query	Retrieved Context	LLM Response	Golden Answer	Score Range
Answer Relevancy	•		•		0 to 1
Faithfulness	•	•	•		0 to 1
Context Precision	•	•		•	0 to 1
Context Recall	•	•		•	0 to 1
Factual Correctness			•	•	0 to 1
CRAG	•		•	•	1, 0 or -1

A potential challenge to the evaluation of faithfulness is that for complex questions with extensive reasoning or aggregation over multiple context chunks even correct answers will not have a direct corresponding claim in a single context chunk, which may lead to errors in the evaluation. As the MoNaCo benchmark does not provide specific golden context chunks directly, and prompting for metrics using context chunks is not developed for the scenario of reasoning and aggregating from multiple chunks, the context precision/recall and faithfulness metrics were deemed inappropriate and are not used in this study.

A more general LLM-as-a-judge metric is the *factual correctness* score in RAGAS, which is computed by a direct comparison between the generated and reference answers. In this case, the LLM is asked to decompose each input into a number of individual claims. By assessing using an LLM judge whether each claim in the response list matches a claim in the reference list and vice versa, claims are labelled as true positives, false positives or false negatives, allowing the system to calculate precision, recall or F_1 score according to their usual definitions [35]. While score calculation is done analytically, the claim decomposition and verification steps rely on LLM judgement. This allows flexibility in answer interpretation to determine correctness without direct word-for-word text matching, but it does make the results non-deterministic.

2.3.4 Custom metrics. While all of the metrics discussed above are commonly used, there are several other useful metrics for RAG evaluation that are used for specific purposes. One is the *Comprehensive RAG Benchmark* (CRAG) score, introduced by [40]. A key observation made by the authors is that incorrect or hallucinated answers are far worse than missing or refused answers, as they can harm user confidence in the LLM system. The CRAG paper therefore assigned +1 to a correct answer, 0 to a missing answer, and −1 to a hallucinated answer. For human evaluation, the scoring was extended to include not only +1 for a perfect answer but

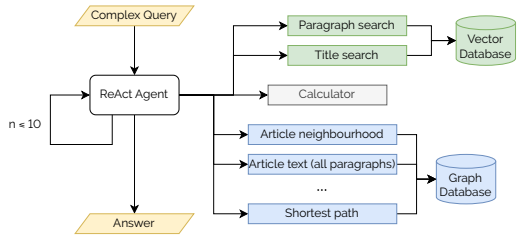


Figure 2: The architecture of our QA system with three types of tools: vector-based (green), graph-based (blue) and a simple calculator tool; middleware applies a soft limit of 10 iterations to the tool use loop.

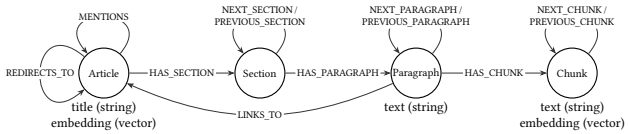


Figure 3: Graph schema for the English Wikipedia dataset

also +0.5 for an acceptable answer that contains useful information but with some minor errors or omissions. However, for auto evaluation with an LLM judge, both +1 and +0.5 were combined into the accurate class (+1). With this scoring system, a value can be reported for each answer in terms of whether it is accurate, missing, or hallucinated. From these can be calculated the proportions of each answer class, as well as an additional *truthfulness* metric, being the sum of all the scores given across a question batch (i.e. the number of accurate answers less the number of hallucinated answers).

3 A SOLUTION TO GRAPH-BASED COMPLEX QUESTION ANSWERING

To implement the QA pipeline presented earlier in Figure 1, we developed a system with a single reasoning agent and a set of tools used to retrieve information from vector and graph databases. Figure 2 shows the architecture of the proposed solution.

The reasoning agent was created using LangChain's *create_agent* function,[3] which provides tool calling capabilities. We equipped the agent with three types of tools: vector-based, graph-based and a simple calculation tool.

The agent was supplemented with standard middleware for tool call error handling, retries with exponential back-off for LLM failures, and a run limiter to prevent a never ending cycle of tool calls and limit token consumption. Finally, the agent was given a JSON structured output schema such that an answer, explanation for the answer, and references were produced separately.

The database, both vector store and graph DB, were populated with a semi-structured representation of a significant portion of English Wikipedia articles included in the MoNaCo benchmark.

3.1 MoNaCo reproducibility

The question-answer pairs of MoNaCo were created by 24 Amazon Mechanical Turk (AMT) crowdworkers over an unspecified period of time on live English Wikipedia [38]. Although this is methodologically straightforward, it creates issues for reproducibility. A key concern is that Wikipedia is constantly in flux: articles that AMT workers may have used to generate question-answer pairs may have since been altered, deleted, or more up-to-date articles created. Therefore, in this study we used the benchmark against a snapshot version of the English Wikipedia from August 2025. This was done after we conducted a data alignment experiment to ascertain which of the various snapshots of Wikipedia prior to the benchmark publication was capable of providing the most valid sources to perform the evaluation.

Each of the fetched snapshots, from January 2023 to August 2025, were scored according to the proportion of answerable questions, out of the MoNaCo total of 1315. The August 2025 snapshot contained the sources necessary to answer 1207 (91.7%) of the questions. The 108 questions deemed unanswerable by this snapshot were dropped from our evaluation.

3.2 Knowledge graph creation

Each article within the snapshot consists of *Wikitext*, a lightweight markup language used for authoring Wikipedia articles. Using Wikitext markers, we parsed sections and paragraphs from each article as well as article redirects and articles linked to from each paragraph. From these links, we also materialised links between articles themselves (mentions), which later aid path-finding within the dataset.

As we also wanted to evaluate vector RAG, paragraphs were further split into 500-character chunks, with a 40-character overlap, following the parameters recommended by Bratanič and Hane [3]. Preceeding and succeeding sections, paragraphs, and chunks were made into doubly-linked lists via the NEXT_* /PREVIOUS_* relationships. Article nodes were given a title property, and the text of each paragraph was stored in a text property, which was then replicated onto the relevant chunk. The resulting schema is shown in Figure 3.

Using the filtered list of 1207 MoNaCo questions, a canonical list of sources was created. The August 2025 snapshot was filtered to just these sources, to balance dataset size against information density. Nodes outside of this filtered subset that were linked to from inside the subset were reduced to stubs: for example, an article not listed as a source by MoNaCo but linked to by a paragraph within a MoNaCo source was reduced to an empty article node.

The resulting KG comprised 5,771,867 nodes and 22,088,251 relationships. Nearly half of all nodes were paragraph chunks, which were vectorised to enable vector search. As such, an embedding model was selected based on its performance on the retrieval task of the MTEB leaderboard [11].

At the time of writing, the top-performing models on the MTEB leaderboard for retrieval were almost entirely large embedding models with 8B parameters or more, such as *Qwen3-embedding*. Despite demonstrating high performance, such large models are unfeasible for this study, where the dataset within the chunk's text attribute runs to over 300 million tokens, and so the embedding process

[3] https://reference.langchain.com/python/langchain/agents/factory/create_agent.

would take an unreasonably long time. However, an outlier among the top retrieval embedding models on the leaderboard was Microsoft's *Harrier* 0.6B [22], which achieved top-5 performance on retrieval tasks (70.75), similar to *Qwen3-embedding* 8B (70.88), and top-10 performance overall (69.01), despite being a much smaller and faster model.

Apart from chunk embeddings, an embedding was also created for each article title, and the resulting dataset was imported to the Neo4j database. Vector indices were created for each chunk's text embedding and article title embedding, so that the graph could be queried directly via *Cypher*, or article titles / chunk texts surfaced via vector search. The resulting dataset could be queried both by traditional vector RAG methods and by KG tooling.

3.3 Retrieval tools

A key advantage of graph queries over vector RAG is that they are cheap operations capable of traversing many hops at once. Multihop QA datasets such as *MuSiQue* [34] rely on CoT reasoning, decomposing a complex question into single-node steps, e.g. "*Who succeeded the first President of Namibia*" becomes "*Who was the first President of Namibia?*" (Sam Nujoma), then "*Who succeeded Sam Nujoma?*". Each step is cheap, but requires a separate LLM call to reflect and generate the next step, on the assumption that retrieval returns one answer at a time from a single data point.

The same multihop reasoning can be seen in MoNaCo, in that it also provides question decomposition. However, a motivation of this study is to determine whether more intelligent retrieval tools can encourage reasoning over multiple hops at once, reducing both the number of LLM turns required as well as overall token usage.

One method of achieving this would be to provide a direct interface between the AI agent and the database, such as an MCP server that allows the agent to write and execute *Cypher* statements directly to the database.[4] This would, however, require the LLM to not only maintain knowledge of the question it is answering but also the schema of the database, possibly fracturing its thinking between its main task (answering the question) and the generation of syntactically correct *Cypher* queries.

Allowing an LLM to generate its own queries also raises prompt injection issues, where a malicious actor may be able to exfiltrate data that they were not supposed to access. Moreover, early attempts with Cypher query generation in this study revealed that AI agents were reluctant to generate complex Cypher queries that would go beyond a single hop, likely due to their preconditioning for typical RAG scenarios.

Instead, we decided that bespoke Cypher queries should be handwritten to query our KG. This allowed the LLM to focus more directly on answering the question, and encouraged it to use more complex queries than single-hop question decomposition. As such, several queries were written for vector search (for typical RAG and graph discovery), structural navigation, and relational queries. These are listed below:

- **Discovery**: finding entry points into the graph and vector RAG
  - *Title vector search*: return top-k articles whose title matches a query.
  - *Chunk vector search*: return top-k paragraphs with chunks matching a query.
  - *Article neighbourhood*: return all articles that are linked to an article within distance n.
- **Structural navigation**: reading the text of an article
  - *Article text*: walk all paragraphs in an article sequentially, returning article text.
  - *Section titles and infoboxes*: Wikipedia often includes so-called "infoboxes", short summaries and vital statistics about the object or person in question. Although extracting all the article text retrieves these along with the rest of the article, it may be possible to save tokens by extracting just the infobox. Similarly, section headings give a brief overview of the content of an article and allow for more incisive reading.
  - *Get sections*: given a list of section IDs, return the text of just those specific sections; this might be another way to reduce token usage.
  - *Window paragraph*: get the n paragraphs surrounding a particular paragraph
  - *Window section*: get the n sections surrounding a particular section.
- **Relational queries**: finding how other articles relate
  - *Get backlinks*: find all the paragraphs that link to an article.
  - *Shortest path*: get all of the interstitial paragraphs that link two articles together.

Each of these queries were integrated as tools into our LangChain agent, so that the Cypher queries themselves were hidden from the LLM and exposed via a simple API; see [37, Appendix B] for details.

3.4 The calculator tool

Given that some of the MoNaCo queries require calculations to be performed using the retrieved data, a calculator tool was created in addition to the retrieval tools discussed above. The tool allows basic arithmetic operations to be performed, and was designed to support aggregation tasks.

4 EVALUATION

To investigate whether queries over structured KBs can improve the performance of complex QA versus unstructured RAG, we set up our pipeline following Figures 1 and 2. Specifically, we:

- ingested an English Wikipedia dataset into both the vector store and knowledge base store. For this purpose we used the native *Neo4j* vector store and *Neo4j* graph database. The methods for chunk embedding and graph creation are documented in Section 3.2,
- looped over 510 complex questions, selected from our reproducible snapshot of the MoNaCo benchmark dataset,[5]
- performed query understanding, planning, reasoning and answer synthesis using a single agent equipped with a set of tools described in Section 3.3, along with a structured output parser,

[4] https://neo4j.com/docs/mcp/current/.

[5] To reduce computational expense 512 of 1207 questions were randomly selected, two were subsequently dropped due to repeated blocking by the LLM content guardrails.

- evaluated the answers produced by our QA system using the LLM-based evaluation metrics described in Section 2.3, i.e. *factual correctness*, *answer relevancy* and two CRAG-inspired custom metrics, along with additional scores calculated from these; we also recorded the token and tool usage for each of the three solutions.

4.1 Experiment configuration

To understand the impact of graph-based tools on the accuracy and performance of the agentic system, we ran three kinds of experiments, namely: vector RAG, vector+graph RAG, and zero-shot. They differed in the combination of tools the agent could use:

- vector RAG: the agent had access only to the retrieval tool, *chunk vector search*, and the calculator tool;
- vector+graph RAG: the agent had access to all the retrieval tools, plus the calculator tool; and
- zero-shot: no tools were available to the agent.

To account for stochasticity in LLM responses, for each scenario the QA task was run three times and minimum, median and maximum values are reported.

In our preliminary experiments, we gave identical prompts in all three scenarios, modified only by the inclusion of the names and descriptions of available tools. The analysis of tool usage showed, however, that in the vector+graph RAG scenario, the agent relied predominantly on the *Chunk vector search* and *Article text* tools, and rarely used other structural navigation tools. Using the two tools, the agent read in many whole articles consuming substantially more input tokens than in the *vector RAG* scenario. For this reason, we updated the prompts to give explicit instructions to use the tools to obtain information efficiently, with an example usage sequence: *Title vector search* → *Section titles* → *Get sections*. This prompting strategy improved the overall use of different tools, although some of the tools were still underutilised. More work is, therefore, needed to improve the prompts and tool use. The prompts themselves are available in [37, Appendix C].

Experiments were performed using a *LangChain* agent based on *GPT-5.4* which, at the time of the experiments, was a top-10 model in the MMLU-Pro leaderboard [36], with a score of 0.875. The model's reasoning effort was set to *medium*, as a trade-off between reasoning depth and response latency, to allow the experiments to be preformed repeatedly over a large question set.

As noted earlier in Section 3.2, this high performance reasoning model was used in conjunction with a top-performing embedding model, *Harrier-0.6B*. To avoid model "self-preference" bias [31], LLM-as-a-judge evaluations were performed using *Llama-4-Maverick* 17B parameter 128 Expert Instruct model. This model, although achieving somewhat lower performance than the GPT model (MMLU-Pro 0.805), was used for the simpler task of comparison between agent generated and expected answers. Where evaluation required embeddings, these were computed using the *text-embedding-3-large* model, again ensuring the use of an independent model family, and again sufficient for the simpler task of identifying semantic similarity between lists of generated and ground truth answers.

Answer evaluation metrics were calculated using the RAGAS library [35], with factual correctness calculated in both the precision and recall modes. The original CRAG metric could not be used directly, but inspired us to design two custom metrics, coarse and fine-grained CRAG, explained later.

4.2 Results

In our evaluation, we mainly compare the baseline vector RAG approach against the proposed vector+graph RAG solution. Additionally, we include the evaluation of the zero-shot scenario in which no external data was introduced to the model.

As shown later, we do note that in our results the highest number of correct answers and lowest token usage occur in the zero-shot mode. However, the dataset used for evaluating the MoNaCo benchmark is based on English Wikipedia. Wikipedia itself, as a largely dispassionate and factual source of information, is one of the key sources used in LLM training. For complex QA against a non-public knowledge base, the scores for the zero-shot approach would almost certainly be significantly lower than those using RAG, since the LLM would not have been exposed to the data in training, and without RAG would have no access to it whatsoever. Here we consider the zero-shot results as indicative of the limiting values possible for an LLM fine-tuned on the KB.

While model fine-tuning on relevant data is an option more efficient than training from scratch, it still requires considerable compute and is not generally an attractive proposition for KBs that are regularly updated. Indeed, RAG was specifically proposed as a method to reduce hallucination, improve factual correctness, and allow access to up-to-date information without costly fine-tuning [27]

Table 2 includes an overview of the results obtained. On top of the LLM-based metrics, we report derived scores to highlight key insights from the plots presenting the "CRAG scores", and the number of tokens each approach used.

4.2.1 Factual correctness and answer relevancy. As explained in Section 2.3.3, *factual correctness* is a RAGAS score computed by a direct comparison of the generated and reference answers. The factual correctness results indicate that the vector+graph RAG responds with much higher precision and recall than the basic vector RAG approach, and is not far from the "fine-tuned" zero-shot solution. This is a significant achievement for vector+graph RAG, meaning that even a simple, semi-structured KG can substantially improve the correctness score, both in terms of precision and recall. To reiterate, the zero-shot approach is only able to obtain comparable results due to the open public nature of the Wikipedia dataset, and this would not be achieved for QA tasks on proprietary data to which the model had not been exposed during training.

We note that in each implementation the precision and recall of factual correctness are relatively low. This is in part due to refusal ('unknown' scores 0), and also indicative of errors being a mixture of false positives and false negatives rather than one form of error predominating. It is not simply the case that failure to retrieve all claims related to a particular concept leads to false negatives lowering the recall score. The precision scores indicate that substantial incorrect answers (i.e. false positives) are also returned. However, in the case of aggregation queries the root cause could still be imperfect recall (e.g. the vector RAG approach fails on the question *"How many novels were written by Ernest Hemingway while he was living abroad?"* as while the correct time period is identified some

Table 2: Evaluation scores ($\text{median}^{\text{max}}_{\text{min}}$) for the three experiment scenarios each run three times; the best results are in bold, the second best are underlined. Scores are based on the same 510 question subset of the MoNaCo questions; some factual correctness evaluations failed due to long answer lists hitting embedding token limits so are based on successful evaluations only (median: precision 499, recall 467).

Score	Vector RAG	Vector+ graph RAG	Zero-shot
Factual correctness prec. (mean)	$0.15^{0.18}_{0.14}$	$\underline{0.36}^{0.38}_{0.34}$	$\mathbf{0.43}^{0.45}_{0.42}$
Factual correctness recall (mean)	$0.13^{0.15}_{0.11}$	$\underline{0.33}^{0.35}_{0.32}$	$\mathbf{0.39}^{0.41}_{0.38}$
Answer relevancy (mean)	$0.35^{0.36}_{0.31}$	$\underline{0.61}^{0.62}_{0.61}$	$\mathbf{0.80}^{0.81}_{0.80}$
Coarse truthfulness	$\mathbf{-31}^{-18}_{-31}$	$\underline{-49}^{-44}_{-61}$	-127^{-113}_{-137}
Fine-grained truthfulness	35^{43}_{25}	$\mathbf{63}^{73}_{56}$	$\underline{40.5}^{45}_{39}$
Coarse CRAG $\frac{\text{fully correct}}{\text{all}}$	$0.11^{0.13}_{0.10}$	$\underline{0.28}^{0.28}_{0.26}$	$\mathbf{0.34}^{0.35}_{0.33}$
Coarse CRAG $\frac{\text{any wrong}}{\text{all}}$	$\mathbf{0.16}^{0.17}_{0.16}$	$\underline{0.37}^{0.38}_{0.37}$	$0.59^{0.60}_{0.59}$
Fine CRAG $\frac{\text{fully + partially correct}}{\text{all}}$	$0.18^{0.20}_{0.17}$	$\underline{0.40}^{0.40}_{0.38}$	$\mathbf{0.51}^{0.52}_{0.50}$
Fine CRAG $\frac{\text{any wrong}}{\text{all}}$	$\mathbf{0.10}^{0.11}_{0.09}$	$\underline{0.26}^{0.27}_{0.25}$	$0.42^{0.43}_{0.42}$
Tokens used (median)			
Input tokens	$\underline{40094}^{41892}_{39063}$	43652^{45163}_{43144}	$\mathbf{490}^{490}_{490}$
Reasoning tokens	$\underline{852}^{858}_{834}$	1055^{1075}_{1050}	$\mathbf{516}^{516}_{516}$
Output tokens	$\underline{1486}^{1494}_{1474}$	1620^{1624}_{1618}	$\mathbf{618}^{620}_{614}$
Total	$\underline{41694}^{43636}_{40573}$	45695^{46871}_{44692}	$\mathbf{1108}^{1120}_{1106}$

novels written in this period are missed, and so the total returned is too low. But, since the golden answer provided is the numerical total only, this appears as a precision error; the additional graph tools allowed this question to be answered correctly).

Considering answer relevancy, which measures alignment between the generated response and the user query (Section 2.3.3), adding a graph-based KB to the baseline vector RAG noticeably improves this score. The KG tools raise relevancy from about 0.35 to 0.61 which is over a 74% increase. As indicated later, this is due to the fact that with the KG tools the agent answers more of the user queries, whereas when using only vector tools the agent often states that the answer is unknown. While this "safe refusal" is preferable to a potentially hallucinated answer, it is likely to have received a low relevancy score from the LLM judge.

The highest relevancy score is achieved in the zero-shot case, which can be attributed to a further increase in the number of questions for which an answer was attempted. But this does not account for the correctness of the returned answer. The *faithfulness* score would add to answer relevancy a test for whether the response is grounded in the context. However, as noted earlier, this metric could not be implemented for the MoNaCo dataset given golden contexts are not provided.

4.2.2 Truthfulness scores. Following the idea of the CRAG benchmark, our *truthfulness* scores strongly penalise hallucinated information over a system that refuses to answer (see Section 2.3.4). However, due to the fact that many of the answers in the MoNaCo

Table 3: The number of answers ($\text{median}^{\text{max}}_{\text{min}}$), classified into three or five CRAG inspired score classes for the three experiment scenarios, each run three times. All scores are based on the same 510 question subset of the MoNaCo questions. The best results are in bold, the second best are underlined.

Correctness metric	Score	Vector RAG	Vector+ graph RAG	Zero-shot
Coarse CRAG	−1	$\mathbf{84}^{89}_{80}$	$\underline{191}^{195}_{189}$	301^{306}_{291}
	0	363^{381}_{360}	$\underline{177}^{181}_{176}$	$\mathbf{35}^{41}_{35}$
	1	58^{66}_{49}	$\underline{142}^{145}_{134}$	$\mathbf{174}^{178}_{169}$
Fine-grained CRAG	−1	$\mathbf{34}^{43}_{25}$	$\underline{83}^{89}_{81}$	136^{137}_{134}
	−0.5	$\mathbf{15}^{20}_{14}$	$\underline{48}^{53}_{45}$	80^{84}_{76}
	0	362^{380}_{360}	$\underline{175}^{178}_{172}$	$\mathbf{34}^{36}_{33}$
	0.5	32^{33}_{30}	$\underline{59}^{61}_{56}$	$\mathbf{86}^{89}_{82}$
	1	59^{68}_{55}	$\underline{145}^{150}_{135}$	$\mathbf{174}^{177}_{172}$

dataset are supposed to produce lists of claims rather than individual full-text responses, the answers can be fully or partially correct. Therefore, we report two versions of the truthfulness score. The first, and most strict *coarse truthfulness* assigns +1 only if the precision and recall of a response is 1. That is, the response includes all the claims from the golden answer and nothing more, which otherwise would be hallucinated claims.

In this view, the basic vector RAG solution achieves the highest score. It is, however, very conservative in answering complex questions and responds only in about 58 + 84 = 142 cases (28.1%) of which only 58 (11.4%) are fully correct. The number of answers in different categories is presented in Table 3, whilst the distribution of responses is shown in Figure 4 (left).

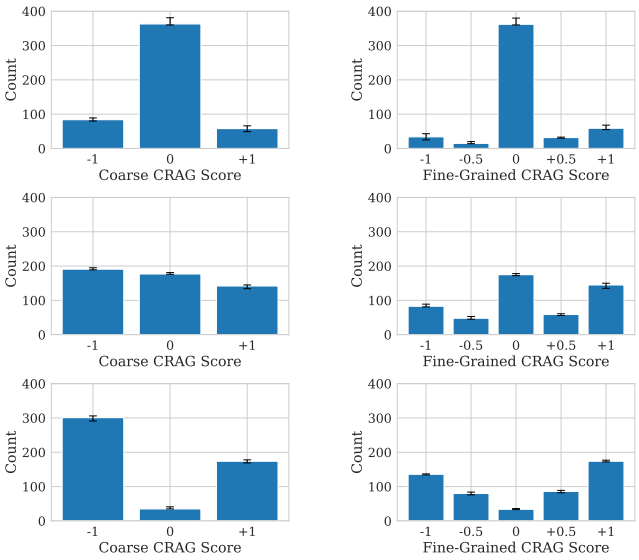


Figure 4: Coarse (left) and Fine-grained (right) "CRAG" scores in three scenarios (top-to-bottom): vector RAG, vector+graph RAG and zero-shot.

The vector+graph RAG approach obtains a slightly lower *coarse truthfulness* score, but significantly higher than the zero-shot solution, which provides fully correct answer for 174 questions (34.1%) but hallucinates in 301 cases (59%). The graph-supported solution is somewhat conservative in responding to the complex questions and provides answers for about 333 questions (65.3%), of which 142 (28.5%) are fully correct (cf. the left middle plot in Figure 4). Considering the *coarse truthfulness* and *coarse CRAG* scores, the vector+graph RAG system is intermediate between the other two scenarios in answering incorrectly, but the number of fully correct responses is much closer to the best, zero-shot approach, than it is to simple vector RAG.

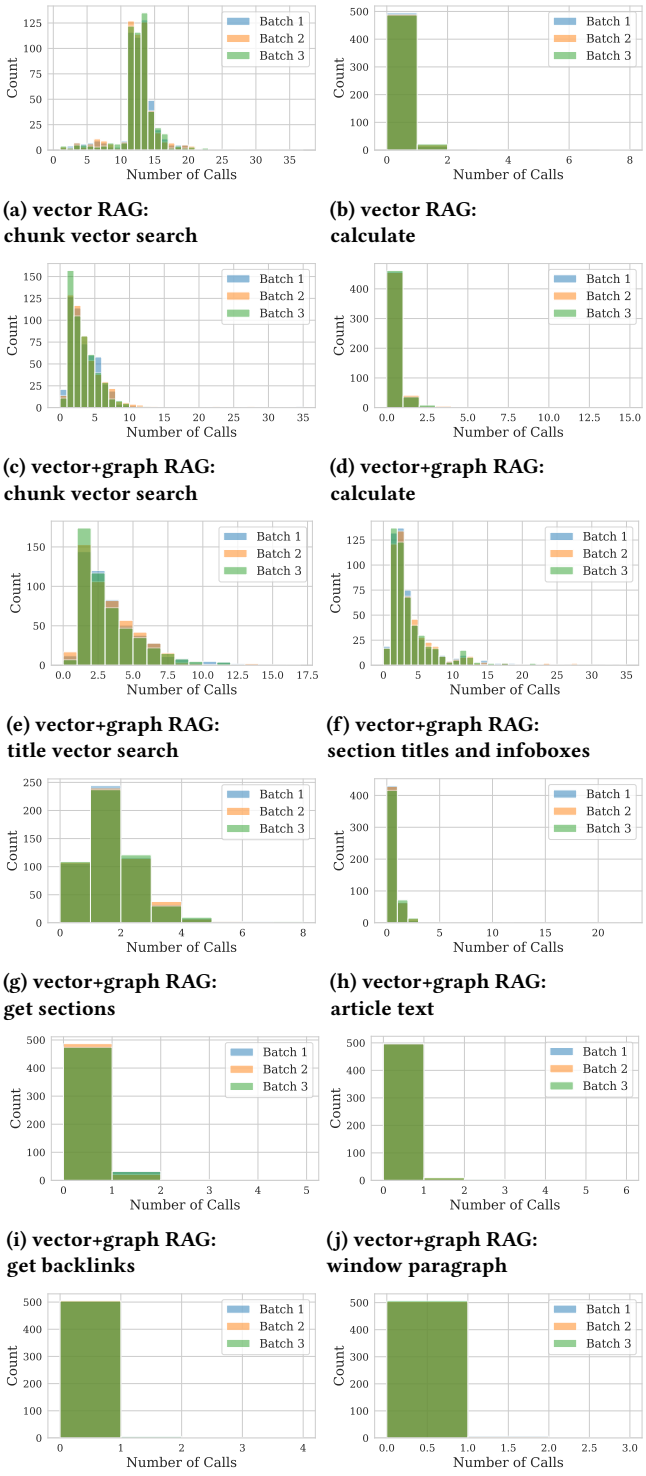
Looking at the answers via a more fine-grained lens (Figure 4, right), the CRAG scores are distributed differently. The *fine-grained truthfulness* still penalises hallucinations, but allows some missingness in the answers. A score of +1 is assigned only to fully correct answers (that is, precision = 1, recall = 1), 0.5 to answers with missing claims but no hallucinations (precision = 1, 0 < recall < 1), 0 to unanswered questions, −0.5 to answers which mix correct and hallucinated claims and −1 where all claims were wrong (i.e. recall = 0). In this case, the vector+graph RAG solution achieves the best score of 63 (median), which is much higher than the other scenarios, 35 and 40.5 for vector RAG and zero-shot, respectively.

This is a significant result which reinforces *coarse truthfulness* in that the vector+graph RAG solution produces much less hallucinated content than zero-shot. But it also indicates that despite the responses still not always being fully correct, adding a simple knowledge-graph can substantially improve correctness and trust into a QA system, especially for complex questions.

4.2.3 Token usage. Token usage and tool use were evaluated by capturing all messages from the agent during QA tasks, and subsequent evaluation and post-processing. We extracted details from the *usage_metadata* dictionary of messages of type *AIMessage*. As shown in Table 2, for the two RAG implementations, around 96% of total token usage is for input tokens, with over 40,000 input tokens used in each case. This is due to the reading of material retrieved from the RAG system. In the zero-shot case input token usage is minimal because the knowledge is embedded in the model parameters, and so tokens are spent on the system prompt, user query and reasoning only.

While the vector RAG implementation has slightly lower total token usage, hence lower cost than the vector+graph RAG solution, this should be considered in relation to the truthfulness scores discussed above. In the vector+graph RAG approach tokens are used to answer many more questions than plain vector RAG (i.e. the fewer responses of 'unknown') while the generated answers are more correct. Overall, the graph-supported solution provides much better value to the end user.

4.2.4 Tool use. Figure 5 shows the distribution of tool calls by both the vector RAG (two tools) and vector+graph RAG scenarios (11 tools). Unsurprisingly, the vector RAG scenario called the vector search tool almost exclusively, often with similar search terms and increasing number of retrieved chunks k as it widened its search to include more results in its context window. The vector+graph RAG also used the chunk vector search tool regularly, but in far fewer turns. It used this tool in conjunction with the other



(a) vector RAG: chunk vector search

(b) vector RAG: calculate

(c) vector+graph RAG: chunk vector search

(d) vector+graph RAG: calculate

(e) vector+graph RAG: title vector search

(f) vector+graph RAG: section titles and infoboxes

(g) vector+graph RAG: get sections

(h) vector+graph RAG: article text

(i) vector+graph RAG: get backlinks

(j) vector+graph RAG: window paragraph

(k) vector+graph RAG: window section

(l) vector+graph RAG: article neighbourhood

Figure 5: Distribution of tools called per question for the vector RAG and vector+graph RAG scenarios. The shortest path tool available in the vector+graph RAG scenario was never used so is omitted from the figure.

vector tool (title search) to retrieve more relevant results, and then used combinations of the other tools to determine its answer. The article neighbourhood tool was used extremely rarely, indicating that links between articles were of little importance once the agent had reached an article.

The article text tool was also rarely used, indicating that the agent limited token usage by focussing on specific sections rather than reading whole articles. Indeed, the tool to fetch section titles and infoboxes was regularly used to fetch basic information about the article: many queries could be answered in part by infoboxes, and section titles give a clear overview of the article's content before the agent begins reading. The agent could then select the sections that it wished to read from the article, and so used the get sections tool in a majority of cases, reducing token usage.

The "windowing" functions for paragraphs and sections were not used often. It suggests the agent correctly retrieved the intended sections and paragraphs in the first try, rather than having to read forwards and backwards from its current location. Similarly, unused were the backlinks, shortest path, and calculate tools. Seemingly, the relationships between articles were not as important as argued by the authors of the MoNaCo benchmark, nor the calculation and aggregation of intermediate results.

The observed reluctancy to call graph tools over vector searches may be due to the task-specific training of LLMs. Vector RAG is seen as a key business case for LLMs, and AI providers such as OpenAI are clearly including RAG in their post-training.[6] Therefore, the use of graph tools to perform complex QA, as explored in this paper, is likely under-represented in LLM training. LLMs are conservative in their tool selection, and are biased towards tools that align with tools seen during training [2]. It is, therefore, not surprising that even when encouraged to use graph tools by the system prompt, the tested model fell back on simpler vector searches for complex queries.

4.3 Non-deterministic evaluation

It is important to state that the results presented above show scores calculated automatically using the LLM-as-a-judge pattern and without human validation. Inherently, this process is non-deterministic and resulted in scoring discrepancies, which stem for randomness embedded in the inference of LLM models.

To illustrate the extent of non-determinism, in Figure 6 we show correlation heatmaps between the coarse and fine-grained trustfulness scores. Following the definition, an ideal heatmap would classify all results of x-axis (fine-grained CRAG) less then 1 and non-zero under the score -1 on the y-axis (coarse CRAG). All zeros on the x-axis should correspond to zeros on y-axis, and all ones on the x-axis should correspond to ones on y-axis.

The heatmaps show that there was a small number of inconsistently classified responses. For example, in the bottom right corner 4, 14 and 26 answers in the vector RAG, vector+graph RAG and zero-shot scenarios, respectively, were classified by the 'fine-grained CRAG' judge as +1. But the same answers were given -1 by the 'coarse CRAG' judge. Similarly, 2, 3 + 6 and 3 + 5 + 10 answers classified by the 'coarse CRAG' judge as +1 were classified

[6] https://developers.openai.com/api/docs/guides/retrieval.

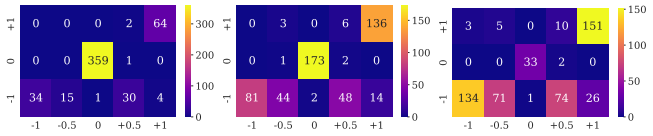


Figure 6: Correlation between coarse CRAG (y-axis) and fine-grained CRAG (x-axis) scores from a single experiment run in three scenarios (left-to-right): vector RAG, vector+graph RAG and zero-shot.

with scores below +1 by the fine-grained judge. Further investigation would be required to confirm whether these effects are due to the stochastic nature of the LLM generative processes, or whether they are systematic errors related to specific questions.

5 LIMITATIONS

One benchmark, one model. Although the results presented are very promising, one of the key limitations of this work is that we evaluated our scenarios using only a single benchmark dataset, MoNaCo, and only a 510 question-answer pair subset from the 1,207 QA pairs available for which corresponding source articles could be found. Furthermore, we used only a single reasoning model for answer generation. In favour of our analysis, MoNaCo uses Wikipedia as its source of ground truth, and despite this, we were still able to show that vector+graph RAG improves precision and recall while reducing hallucinations. Nevertheless, it would be useful to validate these results using a larger number of questions, additional benchmark datasets, and alternative reasoning models.

Human validation. The majority of the results presented in this paper were generated using the LLM-as-a-judge paradigm; only limited human validation was conducted to assess potential scoring errors and inconsistencies in the evaluation values (cf. Section 4.3). As the observed scoring discrepancies were not significant, we used a relatively large number of complex questions from MoNaCo, and three independent experiment runs, the results may be considered valid. Nevertheless, repeated evaluations and thorough error analysis would further increase confidence.

Isolating the contribution of individual tools. Our experiments compare vector RAG against vector+graph RAG as whole configurations, and so do not isolate the marginal contribution of each graph tool. A full ablation would re-run the entire pipeline (510 questions, three repeats, an LLM agent per run) for every tools configuration: while a leave-one-out study scales linearly in the number of tools, isolating the *interactions* between tools grows combinatorially (up to 2^n subsets for n tools), making a thorough ablation substantially more expensive than the experiments reported here. As partial evidence in lieu of such a study, the per-question distribution of tool calls (Figure 5) already indicates which tools carry the load – chunk and title vector search, section-title retrieval, and *get_sections* – and which are near-unused – article neighbourhood, backlinks, shortest path, and the windowing tools – providing a coarse signal of each tool's importance.

Tool use depends on prompting. The graph tools require accompanying prompt instructions to be used effectively: as described

in the experiment configuration, we supply an example usage sequence (*Title vector search → Section titles → Get sections*) to steer the agent towards efficient, structure-aware retrieval. We regard this prompting as an intended and integral part of the proposed method rather than a confound: the contribution is the *combination* of a lightweight document graph, a dedicated toolset, and the prompting that drives them. The agent's residual reluctance to call the more complex graph tools might be further reduced with few-shot examples – e.g. by working through the MoNaCo-provided decomposition steps – or by fine-tuning on graph-specific traces, which we leave to future work.

Latency. Due to asynchronous batching of LLM calls by RAGAS and unhandled API errors on some LLM calls we do not assess the relative answer latency of vector vs vector+graph approaches. We note that an assessment of such features would be strongly dependent on the precise deployment details of the vector and graph DB systems, and is beyond the scope of this work.

Scope. This study compares vector RAG against vector+graph RAG built on a *simple* document graph. Richer knowledge graphs and more sophisticated GraphRAG variants are out of scope. We do not claim that a graph database is the only way to realise the approach, nor that our simple graph is optimal; rather, it is the lightweight enabler that makes structure-aware retrieval practical on semi-structured collections.

6 CONCLUSIONS

In this paper we describe a QA system capable of tackling complex questions that require sophisticated multi-hop and multi-entity reasoning, access to multiple documents, and strong summarisation and aggregation capabilities. To evaluate the proposed approach, we used the MoNaCo complex QA dataset. This we filtered to over 1200 questions with golden answers referring to over 28 thousand semi-structured documents drawn from a static snapshot of English Wikipedia. From the dataset, we extracted a simple graph structure that does not require sophisticated graph construction methods. Instead, it relies on basic structural information, such as document and section titles, together with links to the relevant document fragments.

Typical LLM-based solutions, such as vector-based RAG and zero-shot prompting, are often inadequate for addressing complex QA tasks. The former often refuse to answer the question, being unable to retrieve, collate, and reason over all relevant context effectively. The latter are often overconfident and respond with a lot of hallucinated content.

The results shown indicate that augmenting a basic vector RAG subsystem with a simple graph-based KB and corresponding tools can significantly reduce the amount of hallucinated content (the coarse truthfulness score improved from about -127 to -49 for vector+graph RAG vs zero-shot). At the same time, the vector+graph RAG approach attempted to answer more than twice as many complex questions as the baseline vector RAG solution. We also show that, when partially correct answers are taken into account, vector+graph RAG achieves the highest score across all three evaluated scenarios; the fine-grained truthfulness score was 80% higher than for vector RAG. Additionally, the factual correctness results indicate that vector+graph RAG achieves more than twice the precision and recall of the system based solely on vector RAG.

These results show a substantial improvement over the baseline vector RAG, allowing us to answer our research question positively and with confidence: structured knowledge bases, such as knowledge graphs, can improve the performance of complex QA systems. By increasing both precision and recall while reducing hallucinations, the proposed solution is a promising direction towards increasing trust in LLM-based QA systems.

CONFLICT OF INTEREST STATEMENT

This research was funded by Neo4j through a project at the National Innovation Centre for Data. The authors declare no competing financial interests.

REFERENCES

[1] Jinheon Baek, Alham Fikri Aji, Jens Lehmann, and Sung Ju Hwang. 2023. Direct Fact Retrieval from Knowledge Graphs without Entity Linking. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics*. Toronto, Canada. https://doi.org/10.18653/v1/2023.acl-long.558

[2] Thierry Blankenstein, Jialin Yu, Zixuan Li, Vassilis Plachouras, Sunando Sengupta, Philip Torr, Yarin Gal, Alasdair Paren, and Adel Bibi. 2026. BiasBusters: Uncovering and Mitigating Tool Selection Bias in Large Language Models. In *Proceedings of the Fourteenth International Conference on Learning Representations*. Rio de Janeiro, Brazil. https://doi.org/10.48550/arXiv.2510.00307

[3] Tomaž Bratanič and Oskar Hane. 2025. *Essential GraphRAG: Knowledge Graph-Enhanced RAG.* Manning Publications, Shelter Island, NY.

[4] Andrew Brown, Muhammad Roman, and Barry Devereux. 2025. A Systematic Literature Review of Retrieval-Augmented Generation: Techniques, Metrics, and Challenges. *Big Data and Cognitive Computing* 9 (2025), 320. Issue 12. https://doi.org/10.3390/bdcc9120320

[5] Avi Caciularu, Arman Cohan, Iz Beltagy, Matthew Peters, Arie Cattan, and Ido Dagan. 2021. CDLM: Cross-Document Language Modeling. In *Findings of the Association for Computational Linguistics*. Punta Cana, Dominican Republic. https://doi.org/10.18653/v1/2021.findings-emnlp.225

[6] Abir Chakraborty. 2024. Multi-hop Question Answering over Knowledge Graphs using Large Language Models. arXiv:2404.19234

[7] Minghang Deng, Ashwin Ramachandran, Canwen Xu, Lanxiang Hu, Zhewei Yao, Anupam Datta, and Hao Zhang. 2025. ReFoRCE: A Text-to-SQL Agent with Self-Refinement, Format Restriction, and Column Exploration. https://openreview.net/forum?id=OuFIfDBwQd. In *Proceedings of the ICLR 2025 Workshop VerifAI: AI Verification in the Wild*. Singapore.

[8] Xin Dong, Evgeniy Gabrilovich, Geremy Heitz, Wilko Horn, Ni Lao, Kevin Murphy, Thomas Strohmann, Shaohua Sun, and Wei Zhang. 2014. Knowledge vault: A Web-scale Approach to Probabilistic Knowledge Fusion. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. New York, NY, 601–610. https://doi.org/10.1145/2623330.2623623

[9] Nouha Dziri, Ximing Lu, Melanie Sclar, Xiang Lorraine Li, Liwei Jiang, Yuchen Lin, Sean Welleck, Peter West, Chandra Bhagavatula, Ronan Le Bras, Jena D. Hwang, Soumya Sanyal, Xiang Ren, Allyson Ettinger, Zaid Harchaoui, and Yejin Choi. 2023. Faith and Fate: Limits of Transformers on Composability. In *Proceedings of the 37th Conference on Neural Information Processing Systems*. New Orleans, LA. https://openreview.net/pdf?id=Fkckkr3ya8

[10] Darren Edge, Ha Trinh, Newman Cheng, Joshua Bradley, Alex Chao, Apurva Mody, Steven Truitt, Dasha Metropolitansky, Robert Osazuwa Ness, and Jonathan Larson. 2025. From Local to Global: A Graph RAG Approach to Query-Focused Summarization. arXiv:2404.16130 [cs.CL]

[11] Kenneth Enevoldsen, Isaac Chung, Imene Kerboua, Márton Kardos, Ashwin Mathur, David Stap, Jay Gala, Wissam Siblini, Dominik Krzemiński, Genta Indra Winata, Saba Sturua, Saiteja Utpala, Mathieu Ciancone, Marion Schaeffer, Gabriel Sequeira, Diganta Misra, Shreeya Dhakal, Jonathan Rystrøm, Roman Solomatin, Ömer Çağatan, Akash Kundu, Martin Bernstorff, Shitao Xiao, Akshita Sukhlecha, Bhavish Pahwa, Rafał Poświata, Kranthi Kiran GV, Shawon Ashraf, Daniel Auras, Björn Plüster, Jan Philipp Harries, Loïc Magne, Isabelle Mohr, Mariya Hendriksen, Dawei Zhu, Hippolyte Gisserot-Boukhlef, Tom Aarsen, Jan Kostkan, Konrad Wojtasik, Taemin Lee, Marek Šuppa, Crystina Zhang, Roberta Rocca, Mohammed Hamdy, Andrianos Michail, John Yang, Manuel Faysse, Aleksei Vatolin, Nandan Thakur, Manan Dey, Dipam Vasani,

Pranjal Chitale, Simone Tedeschi, Nguyen Tai, Artem Snegirev, Michael Günther, Mengzhou Xia, Weijia Shi, Xing Han Lù, Jordan Clive, Gayatri Krishnakumar, Anna Maksimova, Silvan Wehrli, Maria Tikhonova, Henil Panchal, Aleksandr Abramov, Malte Ostendorff, Zheng Liu, Simon Clematide, Lester James Miranda, Alena Fenogenova, Guangyu Song, Ruqiya Bin Safi, Wen-Ding Li, Alessia Borghini, Federico Cassano, Hongjin Su, Jimmy Lin, Howard Yen, Lasse Hansen, Sara Hooker, Chenghao Xiao, Vaibhav Adlakha, Orion Weller, Siva Reddy, and Niklas Muennighoff. 2025. MMTEB: Massive Multilingual Text Embedding Benchmark. In *The Thirteenth International Conference on Learning Representations*. Singapore. https://doi.org/10.48550/arXiv.2502.13595

[12] Shahul Es, Jithin James, Luis Espinosa Anke, and Steven Schockaert. 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*. Association for Computational Linguistics, St. Julians, Malta, 150–158. https://aclanthology.org/2024.eacl-demo.16

[13] Wenqi Fan, Yujuan Ding, Liangbo Ning, Shijie Wang, Hengyun Li, Dawei Yin, Tat-Seng Chua, and Qing Li. 2024. A Survey on RAG Meeting LLMs: Towards Retrieval-Augmented Large Language Models. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. Barcelona, Spain. https://doi.org/10.1145/3637528.3671470

[14] Tengfei Feng and Liang He. 2025. RGR-KBQA: Generating Logical Forms for Question Answering Using Knowledge-Graph-Enhanced Large Language Model. In *Proceedings of the 31st International Conference on Computational Linguistics*. Abu Dhabi, UAE, 3057–3070. https://aclanthology.org/2025.coling-main.205/

[15] Yanlin Feng, Simone Papicchio, and Sajjadur Rahman. 2025. CypherBench: Towards Precise Retrieval over Full-scale Modern Knowledge Graphs in the LLM Era. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics*. Vienna, Austria. https://doi.org/10.18653/v1/2025.acl-long.438

[16] Aoran Gan, Hao Yu, Kai Zhang, Qi Liu, Wenyu Yan, Zhenya Huang, Shiwei Tong, and Guoping Hu. 2025. Retrieval Augmented Generation Evaluation in the Era of Large Language Models: A Comprehensive Survey. arXiv:2504.14891 [cs.CL]

[17] Mingqi Gao, Xinyu Hu, Jie Ruan, Xiao Pu, and Xiaojun Wan. 2025. LLM-based NLG Evaluation: Current Status and Challenges. arXiv:2402.01383 [cs.CL]

[18] Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, Meng Wang, and Haofen Wang. 2024. Retrieval-Augmented Generation for Large Language Models: A Survey. arXiv:2312.10997 [cs.CL]

[19] Haoyu Han, Yu Wang, Harry Shomer, Kai Guo, Jiayuan Ding, Yongjia Lei, Mahantesh Halappanavar, Ryan A. Rossi, Subhabrata Mukherjee, Xianfeng Tang, Qi He, Zhigang Hua, Bo Long, Tong Zhao, Neil Shah, Amin Javari, Yinglong Xia, and Jiliang Tang. 2025. Retrieval-Augmented Generation with Graphs (GraphRAG). arXiv:2501.00309 [cs.IR]

[20] Tianxing He, Jingyu Zhang, Tianle Wang, Sachin Kumar, Kyunghyun Cho, James Glass, and Yulia Tsvetkov. 2023. On the Blind Spots of Model-Based Evaluation Metrics for Text Generation. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki (Eds.). Association for Computational Linguistics, Toronto, Canada, 12067–12097. https://aclanthology.org/2023.acl-long.674/

[21] Cheng-Ping Hsieh, Simeng Sun, Samuel Kriman, Shantanu Acharya, Dima Rekesh, Fei Jia, and Boris Ginsburg. 2024. RULER: What's the Real Context Size of Your Long-Context Language Models?. In *Proccedings of the First Conference on Language Modeling 2024*. https://openreview.net/forum?id=kIoBbc76Sy

[22] Xiaolong Huang, Liang Wang, Furu Wei, Jingwen Lu, Knut Risvik, and Jason Li. 2026. Microsoft Open-Sources Industry-Leading Embedding Model. https://blogs.bing.com/search/April-2026/Microsoft-Open-Sources-Industry-Leading-Embedding-Model.

[23] Nikhil Kandpal, Haikang Deng, Adam Roberts, Eric Wallace, and Colin Raffel. 2023. Large Language Models Struggle to Learn Long-Tail Knowledge. In *Proceedings of the 40th International Conference on Machine Learning*. Honolulu, Hawaii, 15696–15707. https://dl.acm.org/doi/10.5555/3618408.3619049

[24] Catherine Kosten, Philippe Cudré-Mauroux, and Kurt Stockinger. 2023. Spider4SPARQL: A Complex Benchmark for Evaluating Knowledge Graph Question Answering Systems. In *2023 IEEE International Conference on Big Data (BigData)*. IEEE, 5272–5281. https://doi.org/10.1109/bigdata59044.2023.10386182

[25] Liubov Kovriguina, Roman Teucher, Daniil Radyush, and Dmitry Mouromtsev. 2023. SPARQLGEN: One-Shot Prompt-based Approach for SPARQL Query Generation. In *Proceedings of SEMANTiCS 2023*. Leipzig, Germany. https://ceur-ws.org/Vol-3526/paper-08.pdf

[26] Kenton Lee, Ming-Wei Chang, and Kristina Toutanova. 2019. Latent Retrieval for Weakly Supervised Open Domain Question Answering. In *Procedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Florence, Italy. https://doi.org/10.18653/v1/P19-1612

[27] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (Eds.), Vol. 33. Curran Associates, Inc., 9459–9474. https://proceedings.neurips.cc/paper_files/paper/2020/file/6b493230205f780e1bc26945df7481e5-Paper.pdf

[28] Jinyang Li, Binyuan Hui, Ge Qu, Jiaxi Yang, Binhua Li, Bowen Li, Bailin Wang, Bowen Qin, Ruiying Geng, Nan Huo, Xuanhe Zhou, Ma Chenhao, Guoliang Li, Kevin Chang, Fei Huang, Reynold Cheng, and Yongbin Li. 2023. Can LLM Already Serve as a Databaes Interface? A Big Bench for Large-Scale Database Grounded Text-to-SQLs. In *Proceedings of the 37th Conference on Neural Information Processing Systems*, Vol. Datasets and Benchmarks Track. New Orleans, LA. https://openreview.net/pdf?id=dI4wzAE6uV

[29] Teng Lin, Yuyu Luo, Honglin Zhang, Jicheng Zhang, Chunlin Liu, Kaishun Wu, and Nan Tang. 2025. MEBench: Benchmarking Large Language Models for Cross-Document Multi-Entity Question Answering. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*. Suzhou, China. https://doi.org/10.18653/v1/2025.emnlp-main.77

[30] Shicheng Liu, Sina Semnani, Harold Triedman, Jialiang Xu, Isaac Dan Zhao, and Monica Lam. 2024. SPINACH: SPARQL-Based Information Navigation for Challenging Real-World Questions. In *Findings of the Association for Computational Linguistics: EMNLP 2024*. Association for Computational Linguistics, Miami, FL, 15977–16001. https://doi.org/10.18653/v1/2024.findings-emnlp.938

[31] Arjun Panickssery, Samuel R. Bowman, and Shi Feng. 2024. LLM Evaluators Recognize and Favor Their Own Generations. arXiv:2404.13076 [cs.CL]

[32] Fabio Petroni, Tim Rocktäschel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, Alexander H. Miller, and Sebastian Riedel. 2019. Language Models as Knowledge Bases?. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Hong Kong, China, 2463–2473. https://doi.org/10.18653/v1/D19-1250

[33] Tommaso Soru, Edgard Marx, Diego Moussallem, Gustavo Publio, André Valdestilhas, Diego Esteves, and Ciro Baron Neto. 2017. SPARQL as a Foreign Language. https://ceur-ws.org/Vol-2044/paper14/paper14.pdf. In *Proceedings of SEMANTiCS 2017*. Amsterdam, Netherlands.

[34] Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2022. MuSiQue: Multihop Questions via Single-hop Question Composition. arXiv:2108.00573 [cs.CL]

[35] VibrantLabs. 2024. Ragas: Supercharge Your LLM Application Evaluations. https://github.com/vibrantlabsai/ragas.

[36] Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, Tianle Li, Max Ku, Kai Wang, Alex Zhuang, Rongqi Fan, Xiang Yue, and Wenhu Chen. 2024. MMLU-Pro: A More Robust and Challenging Multi-Task Language Understanding Benchmark. arXiv:2406.01574 [cs.CL]

[37] Christopher J. Wedge, Joshua Stutter, Danny Dixon, and Jacek Cała. 2026. Reducing Hallucinations in Complex Question Answering using Simple Graph-based Retrieval-Augmented Generation (long version). arXiv:2606.05901 [cs.CL]

[38] Tomer Wolfson, Harsh Trivedi, Mor Geva, Yoav Goldberg, Dan Roth, Tushar Khot, Ashish Sabharwal, and Reut Tsarfaty. 2026. MoNaCo: More Natural and Complex Questions for Reasoning Across Dozens of Documents. *Transactions of the Association for Computational Linguistics* 14 (Jan. 2026), 23–46. https://doi.org/10.1162/TACL.a.64

[39] Yilin Xiao, Junnan Dong, Chuang Zhou, Su Dong, Qian wen Zhang, Di Yin, Xing Sun, and Xiao Huang. 2025. GraphRAG-Bench: Challenging Domain-Specific Reasoning for Evaluating Graph Retrieval-Augmented Generation. arXiv:2506.02404 [cs.CL]

[40] Xiao Yang, Kai Sun, Hao Xin, Yushi Sun, Nikita Bhalla, Xiangsen Chen, Sajal Choudhary, Rongze Daniel Gui, Ziran Will Jiang, Ziyu Jiang, Lingkun Kong, Brian Moran, Jiaqi Wang, Yifan Ethan Xu, An Yan, Chenyu Yang, Eting Yuan, Hanwen Zha, Nan Tang, Lei Chen, Nicolas Scheffer, Yue Liu, Nirav Shah, Rakesh Wanga, Anuj Kumar, Wen-tau Yih, and Xin Luna Dong. 2024. CRAG – Comprehensive RAG Benchmark. In *Advances in Neural Information Processing Systems*, A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (Eds.), Vol. 37. Curran Associates, Inc., 10470–10490. https://doi.org/10.52202/079017-0335

[41] Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018. HotpotQA: A Dataset for Diverse, Explainable Multi-hop Question Answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Brussels, Belgium. https://doi.org/10.18653/v1/D18-1259

[42] Hao Yu, Aoran Gan, Kai Zhang, Shiwei Tong, Qi Liu, and Zhaofeng Liu. 2025. *Evaluation of Retrieval-Augmented Generation: A Survey*. Springer Nature Singapore, 102–120. http://dx.doi.org/10.1007/978-981-96-1024-2_8

[43] Tianyi Zhang, Varsha Kishore, Felix Wu, Killian Q. Weinberger, and Yoav Artzi. 2020. BERTScore: Evaluating Text Generation with BERT. In *Proceedings of the 2020 International Conference on Learning Representations*. Addis Ababa, Ethiopia. https://openreview.net/pdf?id=SkeHuCVFDr