GraphRAG based QA system for financial professionals

Olga Kopaneva

May 2025

Abstract

This work is focused on testing the hypothesis that QA systems based on GraphRAG technique are able to provide more accurate / reliable answers (in particular less 'hallucination') than 'open domain LLM only' or 'open domain LLM + RAG' approaches. To test this, we will create a GraphRAG-based QA system that will answer questions from the CFA (Chartered Financial Analyst) certification programme, which is the global standard of professionalism in the financial sector.
Link to project: `https://github.com/OlgaKopaneva/graphrag_for_finance`.

1 Introduction

Against the background of active development and implementation of LLMs, improving the quality of the answers they generate is especially relevant, in particular, to reduce the frequency of 'hallucination' models (by 'hallucination' we will mean that the generated content is meaningless or does not correspond to the original content provided).

In context of the social sciences there is an additional problem: at the pre-training stage a large number of very diverse ideas about these sciences enter the LLM - facts are mixed with assumptions and opinions and non-specialised models may provide unprofessional answers to professional questions, which limits the potential for their use in work tasks and professional learning. That is why the financial domain (a subfield of economics) was chosen for the project as it is susceptible to this additional distortion.

It is important to note that this work does not consider fine-tuned models with domain knowledge or specialised models. This is because it is a different task requiring other resources. The aim is to improve the quality of the basic open domain model without altering the model itself (achieving satisfactory results at the expense of minimal resources).

GraphRAG technique is proposed as a way to improve the model's factual accuracy and reliability. This approach combines all the advantages of Retrieval-Augmented Generation (RAG) and structured knowledge (knowledge graph).

RAG has made significant strides in overcoming key limitations of LLMs without retraining. By referencing an external knowledge source, RAG enhances the quality of LLM outputs, effectively reducing problems like "hallucinations," gaps in domain-specific expertise, and reliance on outdated information. However, the complex structure of relationships between entities in databases is a challenge for traditional RAG. To address this, GraphRAG utilizes the structural connections among entities to enable more accurate and comprehensive information retrieval, capturing relational insights and supporting more precise, contextually informed responses [14] .

The GraphRAG technique often uses a knowledge graph as the graph basis, we will follow the same approach. A knowledge graph is a structured database that connects entities through well-defined relationships.

1.1 Team

Olga Kopaneva - sole author of this project.

2 Related Work

Depending on the degree of generalisation of the problem, different groups of related work can be identified (sources directly describing GraphRAG will be cited in the next section on model description).

Firstly, there are the works that set the main directions for solving the problem of improving the factorial accuracy of models (we still not considering expensive variants, arising at the model training stage):

1. **RAG** [23], which involves augmenting the user's query with additional information from external sources (via a document database or a real-time query to a search engine) and **its variations** (LongRAG [21] for long-context question answering, ChunkRAG [8] filtering method for RAG Systems, HtmlRAG [15] with idea that HTML is better than plain text for RAG Systems, FastRAG [2] for semi-structured data, Astute RAG [4] as the way to overcome knowledge conflicts);

2. reduction of the **model temperature** (the parameter responsible for diverse or creative responses);

3. various **step-by-step reasoning** (through prompting) approaches: Chain of Thoughts [7], Self Consistency with Chain of Thoughts [25], Tree of Thoughts [26] (these approaches are currently less relevant than before due to the emergence of an internal reasoning in LLM);

4. **self-control** of the model (it critically re-examines its own results) [11].

Secondly, there are works that are not embodiments of GraphRAG, nevertheless they offer a similar idea - the different use of knowledge graphs (and similar structures) in the text generation process:

1. Notable is the work [27], which covers various ways of **combining LLMs and knowledge graphs**: 1) KG-enhanced LLMs, which incorporate KGs during the pre-training and inference phases of LLMs, or for the purpose of enhancing understanding of the knowledge learned by LLMs; 2) LLM-augmented KGs, that leverage LLMs for different KG tasks such as embedding, completion, construction, graph-to-text generation, and question answering; and 3) Synergized LLMs + KGs, in which LLMs and KGs play equal roles and work in a mutually beneficial way to enhance both LLMs and KGs for bidirectional reasoning driven by both data and knowledge;

2. the paper [18] considers the implementation of a KG-to-text model using a multiple attention mechanism that encodes input semantic triples via a bidirectional GRU (guided recurrent unit);

3. the papers [17] and [19] focus on the use of **dynamic multi-way reasoning** in pre-trained language models based on multiple links derived from an external knowledge graph (an approach that combines knowledge graph learning and traditional learning on multiple corpora of texts into a single model);

4. in articles [10] and [9], the authors propose the concept of a **cognitive graph** and its application to text generation. The cognitive graph is related to the knowledge graph, but it solves the problems of the complexity of construction and scaling of knowledge graphs by using the theory of the dual process of human thinking. According to this theory, humans have two stages of making judgments: the first stage is fast and intuitive (heuristic), and involves selecting information relevant to the current situation on the basis of experience; the second stage is slow and analytical, and is under the control of consciousness, forming rational opinions and attitudes. Implementing this approach improves the quality of NLG models. Pre-training the GPT-2 model, as demonstrated by the authors of article [10], produced results that surpassed those of models employing text generation based on knowledge graphs, as measured by BLEU, METEOR and ROUGE;

5. the paper [22] provides an overview of the application of knowledge graphs in machine learning models (in particular, pointing out existing challenges and promising directions).

Finally, we identify papers on the use of knowledge graphs to organise knowledge about finance for different systems:

1. Source [3] discusses the **synergy between knowledge graphs and LLM** in finance. It highlights that the finance industry requires advanced reasoning capabilities to extract data from texts, particularly for interpreting complex terms and concepts. For instance, accurately interpreting the phrase 'an indicator is calculated as the ratio of revenue for the last 12 months before the reporting date to revenue for the 12 months before the

same reporting date of the previous year' necessitates an understanding of several concepts and the ability to connect them. Integrating a knowledge graph into an LLM can solve this problem.
Another issue is that financial science contains visually similar terms whose differences consist of essential nuances that cannot be inferred from the context and can only be identified by experts. LLMs can substitute such terms and mislead users, whereas knowledge graphs can support a complex system of concepts, including synonymous terms, different terms, and terms in hierarchical and 'equal' and 'not equal' relations.
Furthermore, combining LLMs and knowledge graphs increases explainability, which is critical in finance.

2. paper about formation of the ontology of the financial sphere for its implementation in the system of intelligent business intelligence [12].

3 Model Description

First, we will describe the GraphRAG approach in general, after that we will come to the hybrid GraphRAG model.

GraphRAG

First of all: basic repository with implementation of the approach of GraphRAG: **GraphRAG** [1] and paper to it [13]. Also noteworthy: **Azure GraphRAG** (extends the concept by offering a solution accelerator built on the graphrag Python package) [2], **Fast GraphRAG** (takes a performance-oriented approach to the GraphRAG concept by leveraging asynchronous operations and parallelized graph querying) [3].

Fig. 1 provides a visual comparison of direct LLM, RAG and GraphRAG. Direct LLMs often provide shallow or unspecific answers. RAG improves the responses somewhat by fetching relevant / domain-specific textual information. However, as text can be lengthy and entity relationships are expressed in flexible natural language, RAG struggles to emphasize 'influence' relationships, which are often central to a user's query. GraphRAG method retrieves relevant relational knowledge from graph databases instead of text corpus, these approach offers a more precise solution. Using explicit entity and relationship representations found in graph data, they can retrieve structured information, providing more accurate and specific answers, particularly for questions involving influence.

Another field very close to GraphRAG is Knowledge Base Question Answering (KBQA). The goal of KBQA in natural language processing (NLP)

[1] microsoft.github.io/graphrag/
[2] Azure-Samples/graphrag-accelerator.git
[3] circlemind-ai/fast-graphrag.git

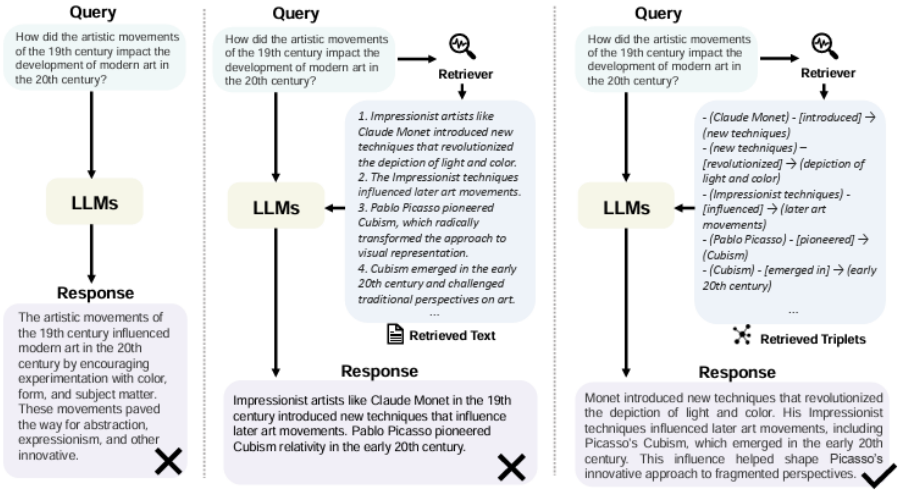

Figure 1: Comparision between Direct LLM, RAG, and GraphRAG [14].

is to answer user questions using external knowledge bases. This capability is vital for tasks like verifying facts, improving search results, and deepening text comprehension. Existing KBQA methods are generally classified as either Information Retrieval (IR)-based or Semantic Parsing (SP)-based. IR-based approaches function by retrieving pertinent data from a knowledge graph to guide the answer generation. In contrast, SP-based methods translate queries into a logical form, which is then executed against the knowledge base to derive the answer. Significantly, GraphRAG and KBQA are intertwined, with IR-based KBQA methods being a specialized form of GraphRAG tailored for specific applications.

Different sources provide different ways of dividing the GraphRAG framework into stages with different levels of detail. However, there are three main stages in total, which are shown in Fig. 2: **Indexing, Retrieval, Generation**.

Indexing: this stage is dedicated to construction and indexing of graph databases and knowledge graphs (Note: it should be clarified here that other graph structures can be used instead of the knowledge graph - social graphs, molecular graphs, infrastructure graph and so on). As this project uses a ready-made knowledge graph, we will not be stopping at this stage.

Retrieval: this is the centrepiece of the method. Let's take a closer look at it (Fig. 3). *The first part* of this process is the processing of the received query (not all steps are necessary, apply depending on the specific task at hand): Named Entity Recognition, relational extraction, query structuration (if required, transforms queries into formats tailored to specific data sources and tasks - Cypher,

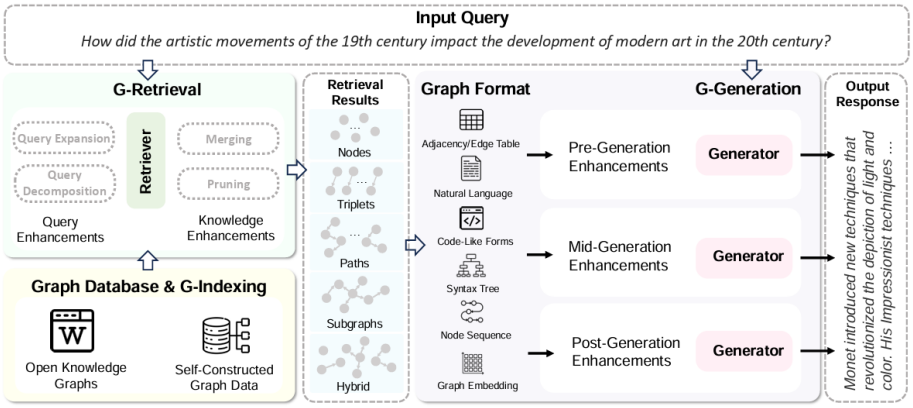

Figure 2: The overview of the GraphRAG framework for question answering task [14].

GraphQL, SPARQL, other), query decomposition (split the input query into multiple distinct subqueries), query expansion (adding meaningful terms with similar significance). *The second part* is retriever itself. As input/output format of GraphRAG differs from that of traditional RAG we need special GraphRAG retrievers: heuristic-based retriever (use predefined rules, domain-specific insights, and hard-coded algorithms to extract relevant information from graph data sources), learning-based retriever (capture deeper, more abstract, and task-relevant relations between the query and objects in data sources, which avoid relying solely on hard-coded rules). *The third part* is organizer (post-process and refine the retrieved content to better adapt it for generator consumption). [24]

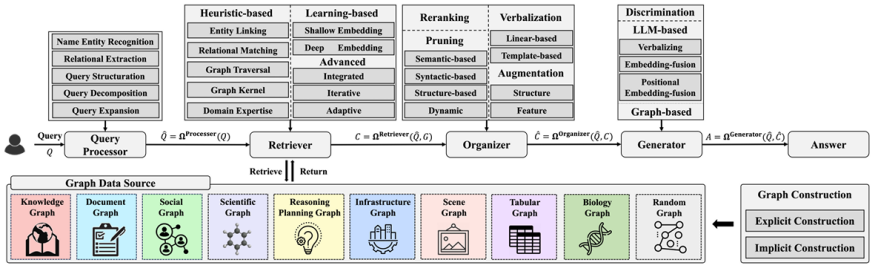

Figure 3: GraphRAG framework and representative techniques for its key components [24].

Generation: the selection of generators often depends on the type of task

at hand: Graph neural networks (GNN) are applicable for discriminative tasks (e.g., multi-choice question answering) or generative tasks that can be formulated as discriminative tasks (e.g., KBQA); LLM are applicable for text-based tasks (directly generate text answers).

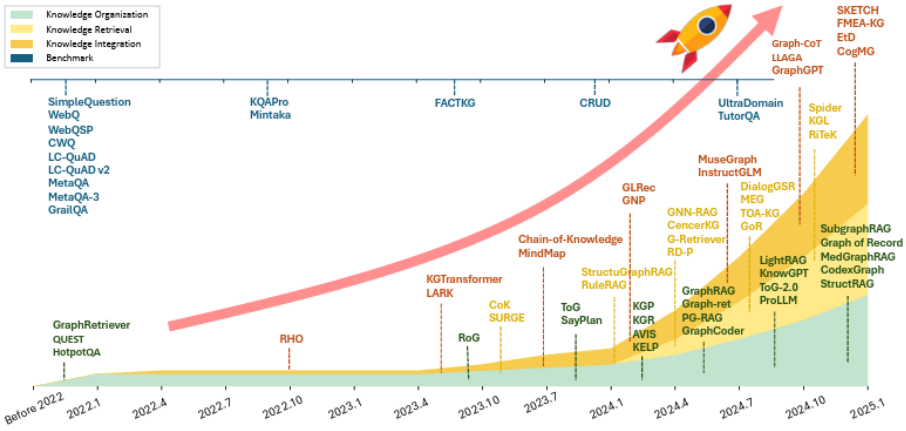

Figure 4: The development trends in the field of GraphRAG with representative works [1].

HybridGraphRAG

This project uses an extended version of the GraphRAG framework, in which a knowledge graph is used for navigation to more accurate chunks of unstructured domain-specific textual information. The idea is based on the work HybridRAG [16], in which the authors have combined GraphRAG (based on knowledge graph) with VectorRAG (the traditional RAG techniques that are based on vector databases). A similar approach is also mentioned in the article [1]. This project does not use a vector database, so instead of VectorRAG, we will use a combination of basic RAG.

As can be seen from Fig. 5, retrieving chunks for augmentation involves two parts. On the one hand, a standard RAG is used (searching chunks via hybrid FIASS + BM5). On the other hand: 1) beforehand, the chunks are annotated according to the available knowledge graph (i.e. entities from the knowledge graph); 2) after receiving a query from the user, the key entities are selected from the query and the corresponding entities in the knowledge graph (i.e. the ontology) are searched; 3) annotated chunks corresponding to these selected entities are then searched in file from stage 1. Then, the chunks obtained by both methods are mixed in the following proportions: 70% from GraphRAG (annotated) and 30% from RAG. This mixture of chunks is then passed on for the generation stage.

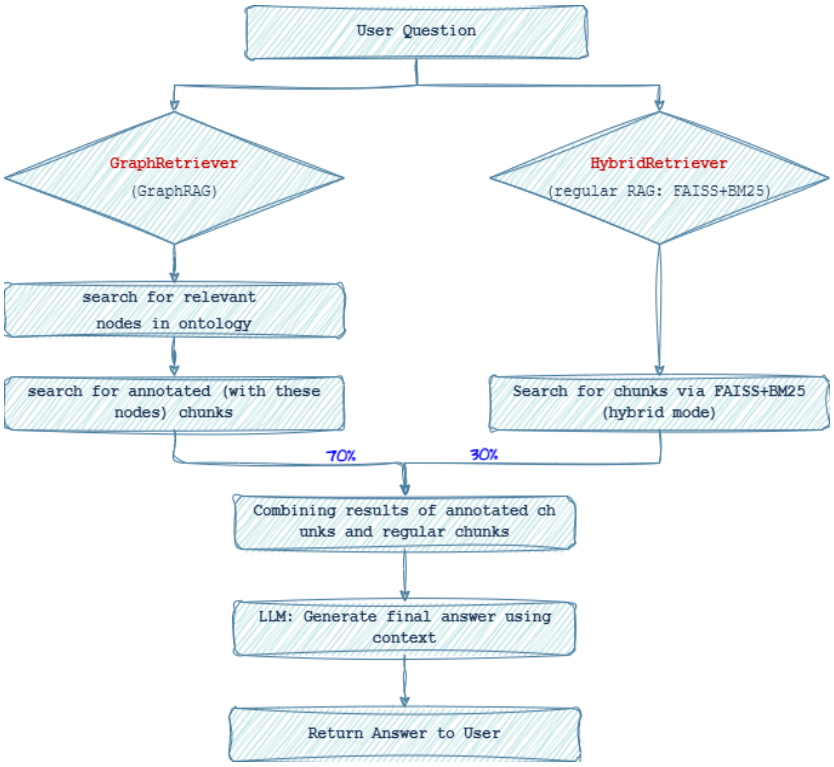

Figure 5: The process used in the project

4 Dataset

These project uses the following data:

Part 1: Data for the HybridGraphRAG model, including:

1. The Financial Industry Business Ontology (FIBO): an international formal ontology originally created by the EDM Council (a not-for-profit, cross-industry data and analytics management association) to provide a common framework for financial contracts worldwide. It emerged as a result of the need to standardise terminology for regulatory reporting and other analytics globally, which was identified during the 2008 financial crisis. It was first published in 2014 and has grown and improved continuously since then. Changes are made on an ongoing basis by several domain-specific teams (e.g. securities, derivatives and indices) and are available on FIBO GitHub [4] and special web-page [5], with formal updates

[4] edmcouncil/fibo.git
[5] spec.edmcouncil.org/fibo/

published quarterly.

FIBO is developed as an ontology in the Web Ontology Language (OWL). OWL is currently the World Wide Web Consortium's (W3C) recommended language for ontologies. The basic elements of the language are properties, classes, and constraints. Using logic ensures that each FIBO concept is formulated in a single-valued way, making it readable by both humans and machines. Since January 2020, the community has been developing FIBO in an open process to provide a machine-readable, unambiguous standard for financial terminology.

The ontology includes 10 thematic ontologies on the main sections of finance (business entities, business process, corporate actions and events, derivative financial instruments, indices and indicators, loans, market data (time-related entities), securities and 2 blocks of general concepts). The ontology contains 1159 classes (entities) - Fig. 6 shows the top-level classes (the most general concepts, for example: document, product, program, law, etc.) most of them have subordinate classes connected to the superior ones by the SubClassOf relation (the figure shows the hierarchy of the 'Strategy' class for example).

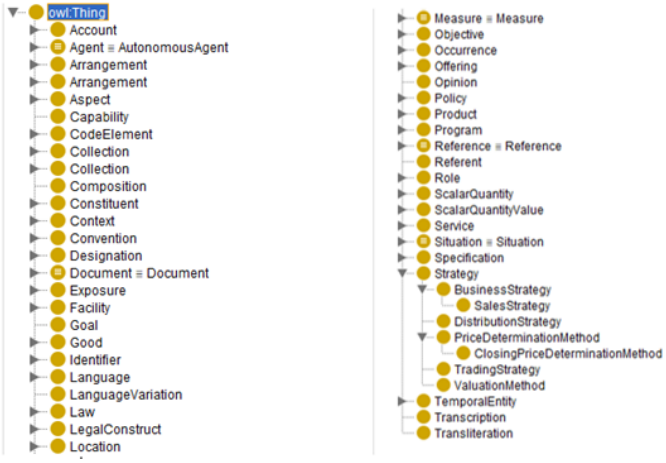

Figure 6: Ontology classes (top level)

There are 764 relations, which are subdivided into two types: ObjectProperty (linking a subject and an object) and DataProperty (linking a subject to constant). The main ObjectProperty relations (more than 100 occurrences): appliesTo, comprises (includes), hasArgument, isAPartyTo, isDefinedIn, isPlayedBy, isProvidedBy. The main DataProperty relationships (more than 50 occurrences): hasDurationValue and hasTextValue. The data structure also contains about 15,000 axioms and 52 annotations.

Pre-processing: full ontology, including all relationships, entities and details, was saved in the project folder. Then was a script (parse_fibo_to_kg.py) was implemented, that, using RDF processing methods, extracts all details and put the RDF files into JSON-ready format. The extracted information was then converted into a knowledge graph using the NetworkX library and finally saved in Gpickle format. So we ended up with a ready-made knowledge graph that will be used directly for the annotation of chunks and the extraction of the most relevant data for the query.

2. Domain-specific texts: for this part was used materials for preparation for the CFA (Chartered Financial Analyst) exams (CFA is an internationally recognised professional certificate for financial analysts and a key certificate in the field of finance and investments) - 14 books of about 400-500 pages each - this source was chosen because it is the most structured, standardised and covers in a single structure the key areas of financial science, also it's recognised by the professional community. Expansion of this part is possible in the future - as there are 33 such books in total + it is possible to use other textbooks and professional dictionaries / thesaurii.

   Pre-processing: all data are originally in pdf format, so the first step is to recognise them and convert into text format (using the PyPDF2 library); the second step is to discard irrelevant/technical information (selecting the actual chapters and discarding the rest via using regular expressions) and clean up unnecessary characters; the third step is the standard for RAG approach: division texts into 200-word chunks (number of words can be changed).

Part 2: Dataset for evaluating:

Initially it was supposed to find a purely financial dataset used in GraphRAG benchmarks. But, as it turned out, there are not many datasets specially designed for GraphRAG and they mostly cover multiple task domains. For example, STARK benchmarks LLM Retrieval on semi-structured knowledge bases covering three domains, including product search, academic paper search, and queries in precision medicine to access the capacity of current GraphRAG systems. Graph Reasoning Benchmark (GRBENCH) is constructed to facilitate the research of augmenting LLMs with graphs, which contains 1,740 questions that can be answered with the knowledge from 10 domain graphs. CRAG provides a structured query dataset, with additional mock APIs to access information from underlying mock KGs to achieve fair comparison [14]. At the time of writing, we could not find such financial dataset.

The second idea was - all the same, but for RAG, here another problem was revealed: these datasets assume that it is necessary to use those documents for

RAG, which are supplied with the dataset (for example, it is proposed to count financial indicators on the statements of specific companies), which is also not suitable given the specifics of the model implementation.

The final idea was to use any QA dataset specialising in financials. Since the most popular datasets at the moment are mostly multimodal, which is also not suitable for us, the best option we could find at the moment was **yymYYM/stock_trading_QA dataset** [6] hosted on Hugging Face.

The dataset **yymYYM/stock_trading_QA** is quite large for specialised QA - 7165 rows, 2 columns: 'question' and 'answer' in text format. Unfortunately, this is not a benchmark, so the quality on it of third-party models is not known.
Dataset is downloaded automatically when you run the evaluate_graphrag script (see repository). But it can be downloaded separately by running *dataset = load_dataset('yymYYYM/stock_trading_QA')*

5 Experiments

The most difficult part of these project is the part with automatic estimation of the obtained model:
- because of the difficulties with the evaluating dataset, as mentioned above;
- because of in general difficult measurability of the quality of responses;
- because of the memory limitations of the local machine, which prevents the use of strong models for generation.

5.1 Metrics

BLEU (Bilingual Evaluation Understudy) is a metric used in natural language processing (NLP) and machine translation to assess the quality of the generated text compared to one or more high-quality reference translations. It measures how similar the machine-generated text is to one or more human-generated reference texts.

BLEU compares n-grams (sequences of n consecutive words) between the generated text and the reference texts. It calculates precision by considering how many n-grams in the generated text match that in the reference text(s). The precision score is then modified by a brevity penalty to avoid favouring shorter translations.

$$p_n = \frac{\sum_{C \in Candidates} \sum_{gram_n \in C} Count_{clip}(gram_n)}{\sum_{C' \in Candidates} \sum_{gram'_n \in C'} Count(gram'_n)}$$

[6] https://huggingface.co/datasets/yymYYM/stock_trading_QA

$$BP = \begin{cases} 1 & \text{if } c > r \\ e^{1-r/c} & \text{if } c \leqslant r. \end{cases}$$

$$Blue = BP \cdot exp\left(\sum_{n=1}^{N} w_n log p_n\right)$$

We first compute the geometric average of the modified $gram_n$ precisions, p_n, using $gram_n$ up to length N and positive weights w_n summing to one. Next, let c be the length of the candidate translation and r be the effective reference corpus length. We compute the brevity penalty BP. And finally compute Blue [6].

ROUGE (Recall-Oriented Understudy for Gisting Evaluation) is a generative AI quality evaluation metric that measures how well generative AI assets perform tasks. **Rouge-1** measures the precision of unigram (single-word) overlap between the generated text and a reference (human-generated) text. This evaluation method matches the reference words but doesn't consider the position of words.

$$ROUGE_N = \frac{\sum_{S \in ReferemceSummaries} \sum_{gram_n \in S} Count_{match}(gram_n)}{\sum_{S \in ReferemceSummaries} \sum_{gram_n \in S} Count(gram'_n)}$$

[20]

It is quite similar to BLEU, but:
- BLEU focuses on precision: how much the words (and/or n-grams) in the candidate model outputs appear in the human reference.
- ROUGE focuses on recall: how much the words (and/or n-grams) in the human references appear in the candidate model outputs.

METEOR (Metric for Evaluation of Translation with Explicit ORdering) is a machine translation evaluation metric, which is calculated based on the harmonic mean of precision and recall, with recall weighted more than precision.

METEOR is based on a generalized concept of unigram matching between the machine-produced translation and human-produced reference translations. Unigrams can be matched based on their surface forms, stemmed forms, and meanings. Once all generalized unigram matches between the two strings have been found, METEOR computes a score for this matching using a combination of unigram-precision, unigram-recall, and a measure of fragmentation that is designed to directly capture how well-ordered the matched words in the machine translation are in relation to the reference.

METEOR evaluates a translation by computing a score based on explicit word-to-word matches between the translation and a reference translation. If

more than one reference translation is available, the given translation is scored against each reference independently, and the best score is reported [5].

P - unigram precision (computed as the ratio of the number of unigrams in the system translation that are mapped (to unigrams in the reference translation) to the total number of unigrams in the system translation)
R - unigram recall (ratio of the number of unigrams in the system translation that are mapped (to unigrams in the reference translation) to the total number of unigrams in the reference translation)

$$Fmean = \frac{10PR}{R + 9P}$$

chunks - all the unigrams in the system translation that are mapped to unigrams in the reference translation are grouped into the fewest possible number of chunks such that the unigrams in each chunk are in adjacent positions in the system translation, and are also mapped to unigrams that are in adjacent positions in the reference translation

$$Penalty = 0.5 \cdot \left(\frac{\#chunks}{\#unigramsmatched} \right)^3$$

$$METEOR_{Score} = Fmean * (1 - Penalty)$$

Values of all metrics are in the range from 0 to 1, the closer to 1, the better.

Note: Although metrics in this section was originally used for evaluating machine translation, they are also well suited for evaluating generation in the presence of correct answers (as is the case here).

5.2 Experiment Setup

To evaluate the proposed approach, a series of experiments were conducted on the open dataset `yymYYYM/stock_trading_QA` (it was described in detail in section 4) containing 7165 question-answer pairs in financial topics. A random subsample of 500 examples was used to speed up the evaluation.

Was compared the following approaches:

1. **Regular RAG**: a classical Retrieval-Augmented Generation implementation with hybrid search (FAISS + BM25)
2. **Modified GraphRAG**: a modification of GraphRAG that uses mix of two approaches: GraphRAG based on FIBO and Regular RAG to refine the user query.

The values of parameter `top-k` (number of documents returned) which were tested for each model: 5, 10 и 15.

Response generation was performed using the compact model `TinyLlama-1.1B-Chat-v1.0` (larger models do not fit in the memory of the local machine).

Response quality was evaluated automatically using the BLEU, ROUGE-1, and METEOR metrics compared to reference responses from the dataset.

5.3 Baselines

Existing approaches have been described in detail earlier in Sections 2 and 3. The current situation with benchmarks was mentioned in section 4 (in the part about Dataset for evaluating).

6 Results

The experimental results are summarised in the table Tab. 1

Method	**Top-K**	**ROUGE-1**	**BLEU**	**METEOR**
Modified GraphRAG	5	0.036	0.005	0.11
Modified GraphRAG	10	0.021	0.002	0.07
Modified GraphRAG	15	0.015	0.001	0.05
Regular RAG	5	0.033	0.004	0.10
Regular RAG	10	0.018	0.002	0.06
Regular RAG	15	0.013	0.001	0.04

Table 1: Statistics of the experiments.

In absolute terms, the results can hardly be considered indicative. The most likely reason for this is the significant difference between the validation dataset and the text dataset from which RAG extracts the additional context. Other reasons: the weakness of the generative model, and the forced reduction of the number of possible annotations for chunks when forming GraphRAG due to memory limitations. The limited amount of domain data for RAG is also influential.
However, it should be noted that the **modified GraphRAG approach proposed in the project consistently demonstrated better performance than the classical RAG approach**.

The sample could be found in Tab. 2. Question: What is EBITDA?

EBITDA is a source of funds to pay interest, dividends, and taxes. EBITDA can be viewed as a source of funds to estimate enterprise value. EV multiples are widely used in Europe, with EV/EBITDA arguably the most common. EBITDA may include other non- cash expenses.

Table 2: Output sample.

7 Conclusion

In this project has been **developed** and successfully applied (compared to the conventional RAG) the **HybridGraphRAG approach or modified GraphRAG** - an extended version of the GraphRAG framework - knowledge graph is used for navigation to more accurate chunks of unstructured domain-specific textual information. Retrieving chunks for augmentation involves two parts. On the one hand, a standard RAG is used (searching chunks via hybrid FIASS + BM5). On the other hand: 1) beforehand, the chunks are annotated according to the available knowledge graph (i.e. entities from the knowledge graph); 2) after receiving a query from the user, the key entities are selected from the query and the corresponding entities in the knowledge graph (i.e. the ontology) are searched; 3) the annotated chunks corresponding to these selected entities are then searched in file from stage 1. Then, the chunks obtained by both methods are mixed in the following proportions: 70% from GraphRAG (annotated) and 30% from RAG. This mixture of chunks is then passed on for the generation stage.

Knowledge graph (from ontology) and a corpus of domain-specific texts were also collected and processed.

Then the approach was tested on an entirely different dataset, results confirmed to a first approximation the theory that using an ontological structure improves the relevance of the returned contexts, and thus the quality of the model response. However, more experiments are required to substantiate the quality of the approach, since the absolute values of the metrics are low.

Possible future improvements include expanding the corpus of domain texts, improving the annotation of chunks by introducing hierarchies and sorting, using a different generation model and tuning the links of chunks to the knowledge graph more carefully.

References

1. A Survey of Graph Retrieval-Augmented Generation for Customized Large Language Models / Q. Zhang [et al.]. — 2025. — arXiv: 2501.13958 [cs.CL]. — URL: https://arxiv.org/abs/2501.13958.

2. *Abane A.*, *Bekri A.*, *Battou A.* FastRAG: Retrieval Augmented Generation for Semi-structured Data. — 2024. — arXiv: 2411.13773 [cs.NI]. — URL: https://arxiv.org/abs/2411.13773.

3. *Andonians V.* The Synergy Between Knowledge Graphs and Large Language Models. — 2024. — URL: https://www.datanami.com/2024/05/01/the-synergy-between-knowledge-graphs-and-large-language-models/ (visited on 06/10/2025).

4. Astute RAG: Overcoming Imperfect Retrieval Augmentation and Knowledge Conflicts for Large Language Models / F. Wang [et al.]. — 2025. — arXiv: 2410.07176 [cs.CL]. — URL: https://arxiv.org/abs/2410.07176.

5. *Banerjee S.*, *Lavie A.* METEOR: An Automatic Metric for MT Evaluation with Improved Correlation with Human Judgments // Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization / ed. by J. Goldstein [et al.]. — Ann Arbor, Michigan : Association for Computational Linguistics, 06/2005. — P. 65–72. — URL: https://aclanthology.org/W05-0909/.

6. Bleu: a Method for Automatic Evaluation of Machine Translation / K. Papineni [et al.] // Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics / ed. by P. Isabelle, E. Charniak, D. Lin. — Philadelphia, Pennsylvania, USA : Association for Computational Linguistics, 07/2002. — P. 311–318. — DOI: 10.3115/1073083.1073135. — URL: https://aclanthology.org/P02-1040/.

7. Chain-of-Thought Prompting Elicits Reasoning in Large Language Models / J. Wei [et al.]. — 2023. — arXiv: 2201.11903 [cs.CL]. — URL: https://arxiv.org/abs/2201.11903.

8. ChunkRAG: Novel LLM-Chunk Filtering Method for RAG Systems / I. S. Singh [et al.]. — 2025. — arXiv: 2410.19572 [cs.CL]. — URL: https://arxiv.org/abs/2410.19572.

9. Cognitive Graph for Multi-Hop Reading Comprehension at Scale / M. Ding [et al.] // Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics. — 2019. — P. 2694–2703.

10. CogNLG: Cognitive Graph for KG-to-text Generation / P. Lai [et al.] // Expert Systems. — 2024. — Vol. 41, no. 1. — e13461. — DOI: 10.1111/exsy.13461. — e13461.

11. CRITIC: Large Language Models Can Self-Correct with Tool-Interactive Critiquing / Z. Gou [et al.]. — 2024. — arXiv: 2305.11738 [cs.CL]. — URL: https://arxiv.org/abs/2305.11738.

12. *Dudycz H.*, *Korczak J.* Conceptual design of financial ontology // Proceedings of the Federated Conference on Computer Science and Information Systems. Vol. 5. — 2015. — P. 1505–1511. — (ACSIS). — DOI: 10.15439/2015F162.

13. From Local to Global: A Graph RAG Approach to Query-Focused Summarization / D. Edge [et al.]. — 2025. — arXiv: 2404.16130 [cs.CL]. — URL: https://arxiv.org/abs/2404.16130.

14. Graph Retrieval-Augmented Generation: A Survey / B. Peng [et al.]. — 2024. — arXiv: 2408.08921 [cs.AI]. — URL: https://arxiv.org/abs/2408.08921.

15. HtmlRAG: HTML is Better Than Plain Text for Modeling Retrieved Knowledge in RAG Systems / J. Tan [et al.] // Proceedings of the ACM on Web Conference 2025. — ACM, 04/2025. — P. 1733–1746. — (WWW '25). — DOI: 10.1145/3696410.3714546. — URL: http://dx.doi.org/10.1145/3696410.3714546.

16. HybridRAG: Integrating Knowledge Graphs and Vector Retrieval Augmented Generation for Efficient Information Extraction / B. Sarmah [et al.]. — 2024. — arXiv: 2408.04948 [cs.CL]. — URL: https://arxiv.org/abs/2408.04948.

17. KGPT: Knowledge-Grounded Pre-Training for Data-to-Text Generation / W. Chen [et al.]. — 2020. — arXiv: 2010.02307 [cs.CL]. — URL: https://arxiv.org/abs/2010.02307.

18. Knowledge graph based natural language generation with adapted pointer-generator networks / W. Li [et al.] // Neurocomputing. — 2020. — Vol. 382. — P. 174–187.

19. Language Generation with Multi-Hop Reasoning on Commonsense Knowledge Graph / H. Ji [et al.] // Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP). — Association for Computational Linguistics, 2020. — P. 725–736.

20. *Lin C.-Y.* ROUGE: A Package for Automatic Evaluation of Summaries // Text Summarization Branches Out. — Barcelona, Spain : Association for Computational Linguistics, 07/2004. — P. 74–81. — URL: https://aclanthology.org/W04-1013/.

21. LongRAG: A Dual-Perspective Retrieval-Augmented Generation Paradigm for Long-Context Question Answering / Q. Zhao [et al.]. — 2024. — arXiv: 2410.18050 [cs.CL]. — URL: https://arxiv.org/abs/2410.18050.

22. Machine Learning and Knowledge Graphs: Existing Gaps and Future Research Challenges / C. D'Amato [et al.] // Transactions on Graph Data and Knowledge (TGDK). — 2023. — Vol. 1, no. 1. — P. 1–35. — DOI: 10.4230/TGDK.1.1.8.

23. Retrieval-Augmented Generation for Large Language Models: A Survey / Y. Gao [et al.]. — 2024. — arXiv: 2312.10997 [cs.CL]. — URL: https://arxiv.org/abs/2312.10997.

24. Retrieval-Augmented Generation with Graphs (GraphRAG) / H. Han [et al.]. — 2025. — arXiv: 2501.00309 [cs.IR]. — URL: https://arxiv.org/abs/2501.00309.

25. Self-Consistency Improves Chain of Thought Reasoning in Language Models / X. Wang [et al.]. — 2023. — arXiv: 2203.11171 [cs.CL]. — URL: https://arxiv.org/abs/2203.11171.

26. Tree of Thoughts: Deliberate Problem Solving with Large Language Models / S. Yao [et al.]. — 2023. — arXiv: 2305.10601 [cs.CL]. — URL: https://arxiv.org/abs/2305.10601.

27. Unifying Large Language Models and Knowledge Graphs: A Roadmap / S. Pan [et al.] // IEEE Transactions on Knowledge and Data Engineering. — 2024. — July. — Vol. 36, no. 7. — P. 3580–3599. — ISSN 2326-3865. — DOI: 10.1109/tkde.2024.3352100. — URL: http://dx.doi.org/10.1109/TKDE.2024.3352100.