HYBRID MODELS COMBINING LLMS AND SYMBOLIC REASONING IN FINANCE

A MINI-THESIS SUBMITTED TO

THE UNIVERSITY OF KENT

IN THE SUBJECT OF COMPUTER SCIENCE

FOR THE DEGREE

OF MSC.

Supervisor: Zhongtian Sun

Word Count: 6938

By

Simon Bandiera

September 2025

Abstract

In order to increase the accuracy of financial document analysis, this dissertation explores the integration of symbolic reasoning with large language models (LLMs). Although LLMs show great aptitude for natural language processing, their preposition for hallucinations lessens the confidence in high-stakes fields like finance, where accuracy and verifiability are crucial. In order to improve factual grounding and produce interpretable results, this work suggests a hybrid framework that combines knowledge graphs (KG) with small LLMs (no more than 7B parameters).

The process entails creating knowledge graphs from financial reports, inserting text into a Neo4j database, and using user queries to retrieve contextually relevant nodes. The inference and post-generation phases of the LLM are then enhanced by these nodes, which offer fact-based answers as well as clear explanations. When compared to a baseline LLM pipeline, the framework demonstrated some accuracy gains on the TruthfulQA benchmark with retrieval-augmented context (RAG).

In addition to finance, this work shows the potential of integrating LLMs with symbolic knowledge in other crucial domains where explainability and factual accuracy are crucial, like law, healthcare, and education.

Contents

Chapter 1

Introduction

1.1 LLM today

Large language models (LLMs) are becoming more and more important in terms of impact in domains such as education, law, decision making, and finance. Their capacity to understand natural language allows for a wide range of use in those domains, such as document analysis, question answering , scenario simulation ... However, a problem that inherently comes from how LLMs are designed is the uncertainty of their answer because they contain inaccuracies or nonsensical text Xu, Jain and Kankanhalli (2025), generally called hallucination. Those hallucinations make LLMs output untrustworthy in high-stake scenario such as the domains cited earlier Shah et al. (2025). Here we will focus on finance as a domain and, more particularly, the use of LLM in the analysis of large file such as 10-k reports and others. Hallucinations are a problem, especially in those types of scenario, since the user cannot simply validate what the LLM said because of the size of the files and high precision is needed in question-answering scenario. LLMs are also expensive to run, if sensitive data need to be given to an LLM for a task, especially like a large file, and the user want to be sure that their data will not be leaked or used, their only solution is to run them locally, which can be quite hard when recent models need a lot of GPU memory to work. However, some models are trained with size in mind, allowing users to run them locally. But another problem arise; in recent benchmark about LLM hallucination Bang et al. (2025), we can see a correlation between LLM size and how much they hallucinate. The smaller they are the more prone to hallucinations they become. To address this issue, a domain like symbolic reasoning is interesting to explore. It is widely used in finance today Mehra (2024) Xie et al. (2025) can be used for rule verification system. More particularly, symbolic knowledge, which is how information is stored in symbolic reasoning systems, offers a defined structure that we could use to enhance LLM capabilities and help explain LLM outputs. The goal of this dissertation is to combine the strength of LLMs in natural language processing and symbolic knowledge to enhance LLM capabilities over logical structure and explainability. This dissertation aims to explore the use of symbolic knowledge to limit small LLMs (up to 7 billion parameters), hallucinations by limiting their context to only relevant information of the document relative to the query. The workflow of the framework explored in this dissertation is as follows.

- creation of a knowledge graph based on a financial report by an LLM.
- user query to an agent with limited context on the document (only relevant KG nodes).
- validation of the LLM output by the user with the help of relevant nodes.

1.2 Background

1.2.1 Financial Document complexity

Finance requires a lot of knowledge to be understood, particularly legal documents. For example, the 10k filling is an annual financial reports that aim to give an overview on what the company did the last year and how well they are, it is very useful for the investors. Even if those types of document are well structured, they are lengthy, commonly going over 200 pages. It is challenging for a human to understand them and even harder to automate. LLMs could do a great job at understanding them since they are good natural language processors, but the risk of hallucination or context loss due to length is too important to delegate the tasks of analysis and reasoning to those types of artificial intelligence. Also, the enormous number of different notions (nested references, boards, dates, numbers, legal entities, etc.) make it hard for the LLM to understand all the relations between the concepts Zhang, Cao and Liao (2025).

1.2.2 LLMs and NLP

LLMs are very good at tasks such as natural language processing, question answering, sentiment analysis, etc. Thanks to their architecture, they have access to all parts of the input sequence at the same time and can be trained on billions of parameters; this allows for state-of-the-art capabilities in natural language processing. However, depending on the field and complexity of what they have to process, some limitations appear, noticeably on identification and categorization, this is the case for advanced finance related analysis. Another significant limitation lies in bias and hallucinations that can affect the LLM. The context limit can also be discussed as a limitation even if today's high-end professional LLMs such as Open AI's o4 have multiple hundred thousand of tokens as context. The lack of proof and grounding in LLMs' answer is an issue in finance application. All of these limitations are further empathized with small LLMs.

1.2.3 Symbolic knowledge

Symbolic knowledge can take a lot of form; here we will use relations and entities to represent information from a context. By defining what entities can be found in a document and what relationship each entity can have with each other, we will be able to greatly limit the context size of our model. More importantly, by making the entities and relationships used by the LLM to formulate its answer available to the user, it greatly de-obfuscates what the LLM base itself on. It allows the user to have a deep understanding of the entities related to its question without having to understand the whole document he gives to the LLM. On top of that, depending on the field and the context of the question, symbolic reasoning can be performed, such as rule-based approach, or path-based methods to directly answer the user's question and totally bypass the LLM. In this dissertation, we will only focus on the interaction between symbolic knowledge and the LLM since otherwise the proposed solution would not be an hybrid method. The representation and storage of the symbolic knowledge will be done thanks to knowledge graphs. It allows us to store for each part of a text what entities is present in it, and what relation they have with each other or entities from other text parts.

1.2.4 End Goal

The goal of the proposed framework is to have a web UI that allows a user to upload PDF file containing 10K or 10Q fillings, and after embedding it in a knowledge graph, to allow the user to query the LLM with enhanced context thanks to the knowledge graph. Then, the user will be able to validate if the LLM answer is anchored in facts with the relevant KG triples returned to the user with the answer.

example: **User query**: *"How did Apple do in the last quarter based on this filling ?"* **Answer**: *"Based on the provided data, Apple did particularly well in the third quarter of 2024 thanks to the release of its new product"* **Node used:** *(Apple, released, Iphone 16), (Iphone 16, has made in revenue, 10 millions $)*

Thanks to the returned node, the user know for a fact that a new product was released since those nodes were created by the knowledge graph.

Chapter 2

Literature review

2.1 Hallucination and LLM in NLP

2.1.1 Knowledge Graphs, Large Language Models, and Hallucinations: An NLP Perspective Lavrinovics et al. (2025)

Analysis

In this study, the authors discuss a way to have large language model answers based on facts and consistent independently of the phrasing of the question asked by the user. To do that, they search for a way to limit LLM hallucination through the use of a knowledge graph. To do so, they consider three types of hallucinations that an LLM can have. First, the **input conflicting hallucination**, is when the answer given to the user is not related to its question. The second is the **context conflicting hallucination**, where the LLMs are creating a context from thin air, and lastly, the **fact conflicting hallucination**, where the answer has information that is simply false. They also talk about different benchmarks that one could use to test the hallucination of a model. Their conclusion is that only one benchmark satisfy all criterion for being good and relevant, the Definitive Answer Dataset Rahman et al. (2024). It is the only benchmark with multi-prompt evaluations, where each question is accompanied with 15 paraphrasing. The different steps where Knowledge graph can be integrated to the reasoning of an LLM is also discussed. The main part discussed is during training, inference and post-generation. They all have their downsides, but more specifically, it is hypothesized that doing that during inference is problematic at its core because the model becomes overly dependent on prompt engineering, constrained by limited context windows, and vulnerable to conflicts between its internal knowledge and the injected external evidence, ultimately leading to inefficiency and inconsistent outputs.

Limitations

Since this is a position paper, the authors propose no implementation of the different discussions mentioned earlier. Also, no discussion of how to create a knowledge graph is provided. It is an important issue since current public KGs are not always up to date and, in fast changing domains such as finance, if the creation of a knowledge graph is as expensive as the retraining of the model, it loses a bit of its purpose. An interesting point to talk about in the future direction would have been other directions to make LLMs more robust. For example, while staying in the field of symbolic reasoning, a rule-based engine could have been talked about.

2.1.2 GraphEval: A Knowledge-Graph Based LLM Hallucination Evaluation Framework Sansford et al. (2024)

Analysis

This paper presents a new framework to validate LLM's output with a knowledge graph. Here is the proposed pipeline: the LLM output is fed to a KG constructor, that will then output a set of triples, containing two entities and their relations. Finally, with an out-of-the-box hallucinator detection, every triple that is inconsistent with the context is marked and returned to the user. The authors also talk more in depth on how the KG creator works. The process is divided into three steps, the **entity detection** (that can represent company, people, object, and many more). The **co-reference detection**, used to find all mentions of the same entity, even if it is called by pronouns. The **relation extraction**, finding the relationships between entities.

Two main prompting methods are also mentioned, chain of thoughts and in context learning. It is also mentioned that their framework outperforms simple model prompting when it comes to limit hallucinations.

	SummEval	QAGS-C	QAGS-X
HHEM	66.0	63.5	75.5
HHEM + GraphEval	71.5	72.2	75.2
TRUE	61.3	61.8	72.6
TRUE + GraphEval	72.4	71.7	73.9
TrueTeacher	74.9	75.6	79.0
TrueTeacher + GraphEval	**79.2**	**78.1**	**79.6**

Table 1: Performance comparison with and without GraphEval.

Limitations

One question that we can have regarding this pipeline is, if the LLM output contains hallucinations, more particularly input and context conflicting hallucinations, how can the knowledge graph created based on this output can counteract the hallucinations ? By the methods they described, triples are matched against the original context and flagged if in opposition, but we don't know what happens when it is not mentioned in it. Based on this, we can say that the proposed framework is focused only on fact conflicting hallucination. Another limitation is the way the knowledge graph is constructed. With the process relying basically on prompt engineering, such as chain of thoughts and context learning, if the model responsible for the task hallucinates himself, or output triples to a wrong format, it could be an issue. It also makes the framework non-deterministic, which would be the ideal since hallucination of a said output regarding a context is.

2.2 LLM in Finance documents analysis

2.2.1 An Approach to the Analysis of Financial Documents Using Generative AI Amri, Bani and Bani (2024)

Analysis

In this paper, the exploration of the usage of generative ai, more specifically LLMs, to analyze financial documents is discussed. Two main approaches are explored. The **RAG** method, which combine a retriever (Llama index here) and an LLM for dynamic enhance summarization. The **BART** model with a vector database, which allows a precise enhancement of domain-specific knowledge. A deep dive on their front-end and back-end is also made in the paper. The front-end is used mainly for the user to interact with the back-end, upload context such as PDF files. The back-end has many more feature, since it is where the main pipeline is executed. When the user uploads a PDF, it is embedded in a vector database. Pre-processing tools are also used on the input of the user. After that, the LLM and the vector database are used together to summarize document: the vector database retrieves the most relevant snippets based on semantic similarity, and the LLM synthesizes this information to generate a concise, coherent, and contextually accurate summary. This seamless integration ensures that the system delivers fast, interactive, and insightful results tailored to the user's needs.

Limitations

In the study, a lot of information is not provided. The authors do not specify how the vector database is populated, or which one they use. Same for the LLM, nothing is specified. More insight on the authors' choices would have been a great addition to the paper. Furthermore, the paper is mainly focused on explaining how the front-end and back-work work together, and less on the actual pipeline workflow. Also, it is not specified which financial document is analyzed. Another limitation of the paper is the absence of a benchmark to test their tool. Examples of the pipeline in action are not given in the paper, and no GitHub repository or alternative resource is provided for further exploration or replication. Finally, limiting the analysis of the document to only summarization restricts the potential applications of the tools. Question answering, trend analysis, sentiment detection, and such would significantly enhance its utility and relevance in real-world financial analysis scenarios.

2.2.2 A Scalable Data-Driven Framework for Systematic Analysis of SEC 10-K Filings Using Large Language Models Daimi and Iqbal (2024)

Analysis

In this paper, the authors propose a novel data driver framework to analyze 10K fillings. To do so, they are leveraging LLM (Command-R+) text evaluators' capacities. In addition to analyzing the fillings, the proposed framework also automates the extraction and pre-processing of these large files. The framework pipeline includes the extraction of specific segments using regular expressions. Those segments are pre-defined and common to all fillings. After that, the LLM will rate the quality of the text on different performance metrics such as confidence rating, innovation ration, etc. To do so, the model is prompt-engineered to output a specific format depending on the selected criterion. A Web application is also mentioned where a user can input a company's stock ticker and get a comprehensive analysis in return.

Limitations

The need to categorize the relevant sections in 10K fillings requires an expert in the field and is quite a complex task. It also limits the freedom of the model to pick which information it thinks is useful. The intrinsic biases of LLMs are also mentioned but not explored in detail. Another limitation in the proposed framework is that its process rely solely on LLMs, and is in consequence prone to hallucinations. This is a critical issue in a field as important as finance document analysis. Even if LLMs have been explored as text evaluators, they are not perfect in this field because of their intrinsic statistical-based design.

2.3 Common issues

With those different papers, we can observe a bridge to gap in finance analysis where no symbolic-reasoning-based framework has been proposed yet. Also, the different reviewed frameworks integrating a symbolic reasoning part, more precisely knowledge graph, in other fields either lack proof that are theorized in the paper, or are incomplete regarding the use of KG to help LLM give accurate, deterministic answers, and limit their hallucinations.

2.3.1 Problem definition

In the following section, we will try to answer these questions.

”Can Knowledge Graph help large language models give a fact-based answer and limit their hallucinations ?”

”Can Knowledge Graph help large de-obfuscate language models answers ?”

In a more focused way to the finance field, we will explore how knowledge graph can capture entities and relationships in a finance document such as 10Q and 10K fillings. The different ways to integrate KG information into the LLM pipeline. And finally, integration of KG into the LLM pipeline helps in question answering and reduce hallucinations. We will also work in a resource-limited environment. The LLM used will have no more than 7 billion parameters and the solution that will be developed will be able to run locally on a medium to high-end personal computer.

Chapter 3

Methodology

3.0.1 Proposed solutions

The multiple possibilities of proposed solutions come mainly from the different places where the integration of the knowledge graph can take place.

Inference boosted with knowledge graph

One way to try to bridge the gap between knowledge graphs and LLMs is during inference. This approach falls into the category of prompt engineering. With the other techniques common to the two approach, we will add in the context window of the LLM the triples that are relevant depending on the user question. This has some upsides and downsides. As stated in Lavrinovics et al. (2025), it is a pretty naive method since it depends on the user's query. Relevant triples could be put in context or not with how the user created its query and what keyword he put in it. This issue is fixed with multi-prompt, but it would not be a user-friendly experience, nor consistent to input a question multiple times rephrased. In a non-resource-limited environment, we can simply put the whole KG in the context window.

Post-generation boosted with knowledge graph

Another way would be to let the LLM answer the user's query without the help of the KG, and validate its answer using the KG. This does not help the LLM achieve a more accurate answer or limit hallucination, but this empowers the user by allowing him to verify what the LLM said by checking the triples related to the answer of the model. If a high number of triples is found, the user knows the answer is fact-based and true, else the user knows the LLM is potentially hallucinating and cannot be trusted for this answer. The main downside of this approach is that the user analysis skill is needed in the pipeline to be complete. If the user does not check the nodes that are returned with the question, it is as there is no KG at all. This technique is also more complex to benchmark since the benefits of the technique are not in the model's answer in itself.

3.0.2 Chosen solution

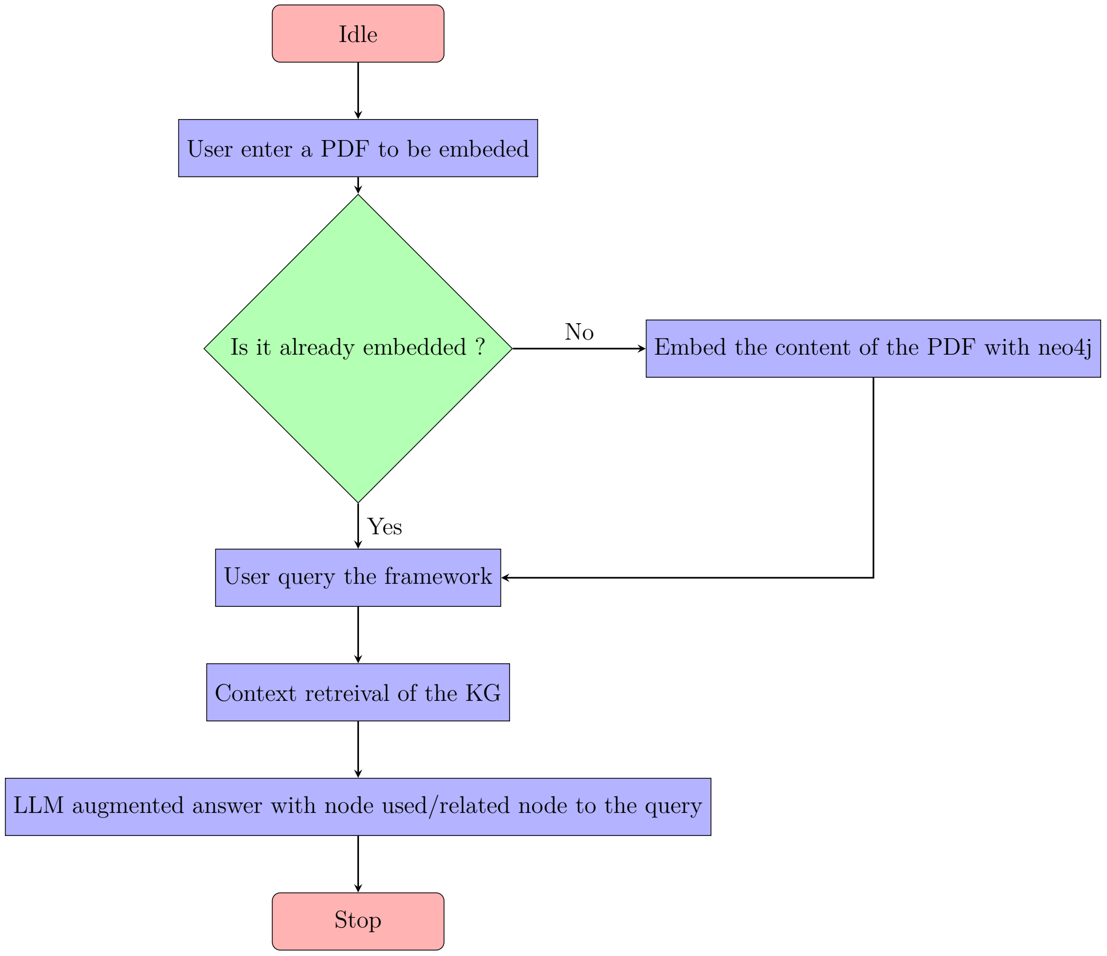
This section will outline the workflow of the proposed framework. As both solutions have their upsides and downsides, the final framework will integrate both of them, and let the user choose at what step of the LLM pipeline he wants to integrate knowledge graph. This choice has been mainly done to still have a way to benchmark the framework. In real world usage, the best usage will probably be the integration of knowledge graph at post-generation, for reasons explained in the section above. The following sections will focus on the interaction between the KG and LLM at the inference step.

3.0.3 Workflow

The proposed workflow contains three steps.

- **Knowledge graph construction**
- **Nodes retrieval based on user query**

- **Knowledge graph and LLM interactions**


Idle
User enter a PDF to be embeded
Is it already embedded ?
No
Embed the content of the PDF with neo4j
Yes
User query the framework
Context retreival of the KG
LLM augmented answer with node used/related node to the query
Stop

Knowledge graph construction

For the construction of the KG, Neo4j is used. It is a graph database that simplifies the creation of knowledge graphs. After the user gives a PDF to analyze, a multiple of things happens.

The most important step is the process of chunking. It transforms unstructured files into coherent segments called chunks. They can represent multiples things such as sentences, paragraphs and other entities of text. Each chunk is represented as a node in the Neo4J database, and then connected to their parent document, and they are also linked at their neighbors. Those chunks allow for an easier understanding of the document's architecture. After the creation of chunks, the embedding comes into play. Its role is to allow for semantic comparison and retrieval of the data. Here, the Nomic-embed-text is used. It will map each chunk to vectors that will allow comparison between the chunks through the representation of vector dimension and geometrical calculus to find the similarity of two nodes. For example, a query talking about the new release of a product will follow the same process of embedding, like the chunks, and compared to find the most relevant chunks.

Nomic is also open-source, and its embedding specializes in semantic clustering, which is perfectly fitting for this task. An LLM agent is also used to extract the different relationships and entities of the document. The model used here is **mistral:7b-instruct**. Between all the models tested, it was the one that was more consistent with respecting the answer format imposed based on a defined schema. To help the LLM find the entities and relations that we want, a schema representing them and their interactions is feed to the LLM. You can check the schema in the Appendix A: Knowledge gragh shema construction

Note that the schema can be changed depending on the documents that are expected. Also, the three last options in the schema $additional_node_types$, $additional_patterns$ and $additional_relationship_types$ allow the LLM to create and add new relationship and entities if it finds the need to do so. This can be useful in fields like finance, where deep knowledge is needed to describe potential entities and relationships. Those options allow us to automatically find the relevant one.

The schema still, however, constrains the LLM on it's output which is limiting the hallucinations it can have. The interaction between the embedding layer and the interpretation layer is mandatory for the success of this step. It makes this step a real hybrid between the two worlds.

Nodes retrieval based on user query

As for the retrieval of the context based on the user query, the process can be defined in one single step. As stated above, the first step is the embedding of the query, following the same steps that we did to embed the chunks. This will allow us to compare the query vector with our chunk and find the most relevant one based on a similarity score. This technique allows to find relation between the query and the chunks even if they don't share the exact same wording. The comparison between the vectors is made thanks to geometry. Once the text has been transformed into a vector, it can be represented in a high-dimensional space. After that, the nearest neighbors can be retrieved to the user. This process is implemented using the Neo4j RAG package, which leverages the vector similarity search to efficiently retrieve and rank the most relevant context for the user's query. This approach ensures that the generated response is accurate and contextually grounded.

Knowledge graph and LLM interactions

Lastly, the nodes returned by the retriever is given as context to the LLM that serves as interface with the user. To ensure a consistent way for the LLM to use the data that have been retrieved, we use a system prompt to answer the user's query. In it, an output format is specified where the LLM needs to follow a strict pattern with two tags, ¡response¿¡/response¿ and ¡justification¿¡/justification¿. In the justification tag, the nodes used to answer the user's query are returned, they come from the one we extracted from the KG. Not all nodes are used in most cases. In the response tag, the answer to the user's query is given with fact-based information. The format in which we give the LLM the nodes that we retrieved can be found in Appendix B: Return format of nodes.

This gives an overview of the relation between entities. This is not the regular triple format, but this allows one node to have multiple entities link to himself at once. It has also been shown that the format changes the understanding of LLMsWedholm (2024), and triples is a much less known format than JSON. Depending on the resources and context size that we want to allocate to our model, we can retrieve only the top K matches to our query, the depth at which we search for the relationship of each node. Since one of the goals of the framework is to work in a resource-limited environment, the number K of nodes recovered is 5, and the depth of the relationship is 1. Even with those minimal settings, a pretty large context size is needed, particularly in financial document analysis, since an entity can have dozens of relationships with others.

Chapter 4

Analysis and evaluation

4.1 Approach

To evaluate the framework, the truthful Q&A context benchmark Portkey (2024). It is an extension of the truthful Q&A benchmark Lin, Hilton and Evans (2021) specially designed for models that use RAG. The benchmark is composed of a question to be answered based on a context, a category, and a list of correct and incorrect answers. The lists of answers allow for a more precise notation on the question by comparing all the answers to the LLM's answer. The context of this benchmark is nothing near as long as a 10K/Q fillings or as complex in terms of relations but it can still demonstrate performance of the framework in question-answering context. Because of this, the prompt system of the framework and the schema used to build the KG was modified to be more fitted to the genereal question answering task. To evaluate the result of the framework, the F1 score has been used. Two metrics in particular are evaluated, the mean and the best F1 score between the answer of the LLM and all correct and incorrect answers. The F1 score was chosen as a performance metric as it is a widely adopted metric for evaluating LLM Hu and Zhou (2024). To obtain a general accuracy of the benchmarked model, we then verify which F1 score is higher between correct and incorrect answers. Since we are evaluating two metrics, average F1 and max F1, it gives us two global averages to rank models. Since resources is also something in mind, the pipeline boosted with the knowledge graph data will have a lesser context window, for the following results, the context window was shifted from 32768 to 16384. Because of how computer-expensive it is to run the framework (with the need to embed and chunk the context) the benchmark wasn't run in it's entirety. In real world use, you generally would ask multiple questions on a same context but this is not the case of the benchmark rendering the framework less relevant for benchmarking purposes.

4.2 Results

Model	Accuracy Max	Accuracy Average
mistral-7B:instruct	40.6%	62.5%
mistral-7B:instruct + KG at inference	48.4%	54.8%

Table 2: Performance comparison with and without KG data.

In Table 2, we can see the performance of the LLM with context augmented by the knowledge graph versus without. Looking at our two metrics, we can see a 7.8% increase in accuracy based on the maximal F1 values between all correct answers and a 7.7% decrease when basing accuracy on the average F1 of all correct answers. Those shifts in accuracy could mean that the model that has the KG data as context is better at pin pointing the main component in the answer but less efficient at finding all the surrounding components to have a complete answer. This suggests that the integration of the knowledge graph does have a utility in the LLM pipeline, but further testing on more relevant benchmarks is needed to have a clear proof of the out-performing or under-performing of the framework compared to baseline.

Chapter 5

Conclusions

To allow for a more precise and fact-based answer, the proposed framework researched the use of knowledge graph as a symbolic reasoner and LLM as a natural language analyzer. The final goal was to use the KG data to enhance the capacitates of LLMs in a resource-limited environment, and limited context size window, while still keeping accurate answers with fewer hallucinations. The framework also aimed at bridge a gap in the finance field where LLMs are not yet a perfect fit for a field that requires a lot of precision and where decisions must be taken with the full picture in mind. With this pipeline, the context size window of the LLM could be reduced but still keep relevant context through the use of the KG node retrieval system. The LLM was also able to state the limit of the information he could extract with the document rather than hallucinating false information to maintain alignment with the bias in the user question. Although the complete pipeline could not be tested in its optimal settings and field, the truthful Q&A dataset showed encouraging results compared to a non-modified LLM pipeline. However, the goal to limit the LLM number's of parameters to 7B was a real challenge to overcome and probably pushed research in a niche of optimization that is not the most align with real-world usage. It is really great for deploying local agent, but further research without this limitation would probably yield interesting results as well.

5.0.1 Future work

The findings of this dissertation can be extended to other fields with ease. Noticeably, the use of an LLM agent in coding interfaces is a growing practice, and these agents tend to become less efficient when the code base becomes too large. The knowledge graph would be a perfect way to represent entities found in code such as classes, variables, instances, functions, libraries, etc. and relations between entities would also be much more simple to find since the syntax of code contains much more rules than a financial paper analysis. More exploration on the construction of the knowledge graph could also be beneficial. The current framework implementation uses the base prompt and the settings of the Neo4j library. A custom implementation of the embedding would allow for more precise entity and relation extraction, even more if the implementation is field-related. Another interesting exploration would have been to try a different symbolic reasoner to replace the knowledge graph such as a rule-base engine. Those other methods would probably have allowed the validation of the output of the LLM and the knowledge embedded from the fillings to be automatic and to remove human interaction as it is needed in this framework. During the later experimentation, I found an interesting benchmark Minzheng Wang (2024) to test the framework. It is focused on long-context multi-documents Q&A, and have three real world category tests, with one of them being finance. It would have been much more accurate to test the framework with it, but required quite a bit of adaptation from the given source code, which I ran out of. Finally, the UI of the framework is also an important part of the framework since its how the user interacts with the whole system. More functionalities such as chat history and tools integration like web search, etc. to have a more robust context. But because of time constraints, it could not be done in time. I would have loved to continue working on this subject in the context of this dissertation, but unfortunately I had other obligations to attend. In the next months I will probably continue to explore those options.

5.0.2 Final word

Using knowledge graphs, we have shown that it is possible to provide users with not only more accurate outputs, but also an interpretable chain of validation. This contributes to bridging the gap between the flexibility of LLMs and the rigor of symbolic approaches, offering a pathway toward more reliable AI systems in high-stakes environments such as finances.

The results suggest that computing power alone is not a sufficient solution to the shortcomings of LLMs. Instead, the integration of symbolic reasoning opens new avenues for making smaller, locally deployable models more trustworthy, cost-effective, and usable in professional workflows. Beyond finance, such a framework has the potential to extend to other critical domains, such as law, education, and healthcare, where the combination of probabilistic reasoning and rule-based validation can deliver both accuracy and explainability.

Bibliography

Amri, S., Bani, R. and Bani, S. (2024). An approach to the analysis of financial documents using generative ai. In *Proceedings of the 7th International Conference on Networking, Intelligent Systems and Security*, New York, NY, USA: Association for Computing Machinery, NISS '24.

Bang, Y. et al. (2025). Hallulens: Llm hallucination benchmark. `2504.17550`.

Daimi, S. A. and Iqbal, A. (2024). A scalable data-driven framework for systematic analysis of sec 10-k filings using large language models. `2409.17581`.

Hu, T. and Zhou, X.-H. (2024). Unveiling llm evaluation focused on metrics: Challenges and solutions. `2404.09135`.

Lavrinovics, E., Biswas, R., Bjerva, J. and Hose, K. (2025). Knowledge graphs, large language models, and hallucinations: An nlp perspective. *Journal of Web Semantics*, 85, p. 100844.

Lin, S., Hilton, J. and Evans, O. (2021). Truthfulqa: Measuring how models mimic human falsehoods. `2109.07958`.

Mehra, A. (2024). Hybrid ai models: Integrating symbolic reasoning with deep learning for complex decision- making.

Minzheng Wang, C. F. S. L. X. Z. B. W. H. Y. N. X. L. Z. R. L. Y. L. M. Y.

F. H. Y. L., Longze Chen (2024). Leave no document behind: Benchmarking long-context llms with extended multi-doc qa.

Portkey, I. (2024). Enhancing truthfulqa with context.

Rahman, A. B. M. A., Anwar, S., Usman, M. and Mian, A. (2024). Defan: Definitive answer dataset for llms hallucination evaluation. `2406.09155`.

Sansford, H., Richardson, N., Maretic, H. P. and Saada, J. N. (2024). Grapheval: A knowledge-graph based llm hallucination evaluation framework. `2407.10793`.

Shah, A., Ye, L., Jaskowski, S., Xu, W. and Chava, S. (2025). Beyond the reported cutoff: Where large language models fall short on financial knowledge. `2504.00042`.

Wedholm, W. (2024). *EXPLORING THE INFLUENCE OF DATA FORMATS ON THE CONSISTENCY OF LARGE LANGUAGE MODELS OUTPUTS*. Ph.D. thesis.

Xie, Z. et al. (2025). Finchain: A symbolic benchmark for verifiable chain-of-thought financial reasoning. `2506.02515`.

Xu, Z., Jain, S. and Kankanhalli, M. (2025). Hallucination is inevitable: An innate limitation of large language models. `2401.11817`.

Zhang, Z., Cao, Y. and Liao, L. (2025). Xfinbench: Benchmarking llms in complex financial problem solving and reasoning. `2508.15861`.

Appendix A

Knowledge graph schema creation

Listing A.1: Schema definition

```
NODE_TYPES = [
    "Company",
    "Filing",
    "BusinessSegment",
    "Product",
    "RiskFactor",
    "FinancialMetric",
    "Executive",
    "Subsidiary",
    "Competitor",
    "Regulation",
    "Geography",
    "Event",
    "SustainabilityGoal",
    "IntellectualProperty",
]

RELATIONSHIP_TYPES = [
    "FILED",
    "HAS_SEGMENT",
    "OFFERS",
    "MENTIONS_RISK",
    "REPORTS_METRIC",
    "RUN_BY",
    "OWNS_SUBSIDIARY",
    "COMPETES_WITH",
    "SUBJECT_TO",
    "OPERATES_IN",
    "PARTICIPATED_IN",
    "SETS_GOAL",
    "HOLDS_IP",
]

PATTERNS = [
    ("Company", "COMPETES_WITH", "Competitor"),
    ("Company", "FILED", "Filing"),
    ("Filing", "MENTIONS_RISK", "RiskFactor"),
    ("Company", "HAS_SEGMENT", "BusinessSegment"),
    ("BusinessSegment", "OFFERS", "Product"),
    ("Filing", "REPORTS_METRIC", "FinancialMetric"),
    ("Company", "OPERATES_IN", "Geography"),
    ("Company", "SUBJECT_TO", "Regulation"),
    ("Executive", "RUN_BY", "Company"),
    ("Filing", "REPORTS_METRIC", "FinancialMetric")
]

SCHEMA = {
    "node_types": NODE_TYPES,
    "relationship_types": RELATIONSHIP_TYPES,
    "patterns": PATTERNS,
    "additional_node_types": True,
    "additional_patterns": True,
    "additional_relationship_types": True,
}
```

Appendix B

Return format of nodes

Listing B.1: Example of an extracted node

```
{
    "node_info": {
        "name": "Our Company",
    },


    "relationships": [
        {
            "type": "SUBJECT_TO",
            "target": {
                "name": "Various U.S. federal, state and
   foreign privacy laws",
            }
        },
        {
            "type": "FILED",
            "target": {
                    "name": "The United Kingdom\u2019s
    Age-Appropriate Design Code",
               }
          },
     ]
}
```