HybridRAG: Integrating Knowledge Graphs and Vector Retrieval Augmented Generation for Efficient Information Extraction

Bhaskarjit Sarmah
bhaskarjit.sarmah@blackrock.com
BlackRock, Inc.
Gurugram, India

Benika Hall
bhall@nvidia.com
NVIDIA
Santa Clara, CA, USA

Rohan Rao
rohrao@nvidia.com
NVIDIA
Santa Clara, CA, USA

Sunil Patel
supatel@nvidia.com
NVIDIA
Santa Clara, CA, USA

Stefano Pasquali
stefano.pasquali@blackrock.com
BlackRock, Inc.
New York, NY, USA

Dhagash Mehta
dhagash.mehta@blackrock.com
BlackRock, Inc.
New York, NY, USA

ABSTRACT

Extraction and interpretation of intricate information from unstructured text data arising in financial applications, such as earnings call transcripts, present substantial challenges to large language models (LLMs) even using the current best practices to use Retrieval Augmented Generation (RAG) (referred to as VectorRAG techniques which utilize vector databases for information retrieval) due to challenges such as domain specific terminology and complex formats of the documents. We introduce a novel approach based on a combination, called HybridRAG, of the Knowledge Graphs (KGs) based RAG techniques (called GraphRAG) and VectorRAG techniques to enhance question-answer (Q&A) systems for information extraction from financial documents that is shown to be capable of generating accurate and contextually relevant answers. Using experiments on a set of financial earning call transcripts documents which come in the form of Q&A format, and hence provide a natural set of pairs of ground-truth Q&As, we show that HybridRAG which retrieves context from both vector database and KG outperforms both traditional VectorRAG and GraphRAG individually when evaluated at both the retrieval and generation stages in terms of retrieval accuracy and answer generation. The proposed technique has applications beyond the financial domain.

1 INTRODUCTION

For the financial analyst, it is crucial to extract and analyze information from unstructured data sources like news articles, earnings reports, and other financial documents to have at least some chance to be on the better side of potential information asymmetry. These sources hold valuable insights that can impact investment decisions, market predictions, and overall sentiment. However, traditional data analysis methods struggle to effectively extract and utilize this information due to its unstructured nature. Large language models (LLMs) [1–4] have emerged as powerful tools for financial services and investment management. Their ability to process and understand vast amounts of textual data makes them invaluable for tasks such as sentiment analysis, market trend predictions, and automated report generation. Specifically, extracting information from annual reports and other financial documents can greatly enhance the efficiency and accuracy of financial analysts [5]. A robust information extraction system can help analysts quickly gather relevant data, identify market trends, and make informed decisions, leading to better investment strategies and risk management [6].

Although LLMs have substantial potential in financial applications, there are notable challenges in using pre-trained models to extract information from financial documents outside their training data while also reducing hallucination [7, 8]. Financial documents typically contain domain-specific language, multiple data formats, and unique contextual relationships that general purpose-trained LLMs do not handle well. In addition, extracting consistent and coherent information from multiple financial documents can be challenging due to variations in terminology, format, and context across different textual sources. The specialized terminology and complex data formats in financial documents make it difficult for models to extract meaningful insights, in turn, causing inaccurate predictions, overlooked insights, and unreliable analysis, which ultimately hinder the ability to make well-informed decisions.

Current approaches to mitigate these issues include various Retrieval-Augmented Generation (RAG) techniques [9], which aim to improve the performance of LLMs by incorporating relevant retrieval techniques. VectorRAG (the traditional RAG techniques that are based on vector databases) focuses on improving Natural Language Processing (NLP) tasks by retrieving relevant textual information to support the generation tasks. VectorRAG excels in situations where context from related textual documents is crucial for generating meaningful and coherent responses [9–11]. RAG-based methods ensure the LLMs generate relevant and coherent responses that are aligned with the original input query. However, for financial documents, these approaches have significant challenges as a standalone solution. For instance, traditional RAG systems often use paragraph-level chunking techniques, which assume the text in those documents are uniform in length. This approach neglects the hierarchical nature of financial statements and can result in the loss of critical contextual information for an accurate analysis[12, 13]. Due to the complexities in analyzing financial documents, the quality of the LLM retrieved-context from a vast and heterogeneous corpus can be inconsistent, leading to inaccuracies and incomplete analyses. These challenges demonstrate the need for more sophisticated methods that can effectively integrate and process the detailed and domain-specific information found in financial documents, ensuring more reliable and accurate results for informed decision-making.

Knowledge graphs (KGs) [14] may provide a different point of view to looking at the financial documents where the documents

are viewed as a collection of triplets of entities and their relationships as depicted in the text of the documents. KGs have become a pivotal technology in data management and analysis, providing a structured way to represent knowledge through entities and relationships and have been widely adopted in various domains, including search engines, recommendation systems, and biomedical research [15–17]. The primary advantage of KGs lies in their ability to offer a structured representation, which facilitates efficient querying and reasoning. However, building and maintaining KGs and integrating data from different sources, such as documents, news articles, and other external sources, into a coherent knowledge graph poses significant challenges.

The financial services industry has recognized the potential of KGs in enhancing data integration of heterogeneous data sources, risk management, and predictive analytics [18? –21]. Financial KGs integrate various financial data sources such as market data, financial reports, and news articles, creating a comprehensive view of financial entities and their relationships. This unified view improves the accuracy and comprehensiveness of financial analysis, facilitates risk management by identifying hidden relationships, and supports advanced predictive analytics for more accurate market predictions and investment decisions. However, handling large volumes of financial data and continuously updating the knowledge graph to reflect the dynamic nature of financial markets can be challenging and resource-intensive.

GraphRAG (Graph-based Retrieval-Augmented Generation) [22–27] is a novel approach that leverages knowledge graphs (KGs) to enhance the performance of NLP tasks such as Q&A systems. By integrating KGs with RAG techniques, GraphRAG enables more accurate and context-aware generation of responses based on the structured information extracted from financial documents. But GraphRAG generally underperforms in abstractive Q&A tasks or when there is not explicit entity mentioned in the question.

In the present work, we propose a combination of VectorRAG and GraphRAG, called HybridRAG, to retrieve the relevant information from external documents for a given query to the LLM that brings advantages of both the RAGs together to provide demonstrably more accurate answers to the queries.

1.1 Prior Work and Our Contribution

VectorRAG has been extensively investigated in the recent years and focuses on enhancing NLP tasks by retrieving relevant textual information to support generation processes [9–11, 26]. However, the effectiveness of the retrieval mechanism across multiple documents and longer contexts poses a significant challenge in extracting relevant responses. GraphRAG combines the capabilities of KGs with RAG to improve traditional NLP tasks [23–25]. Within our implementations of both VectorRAG GraphRAG techniques, we explicitly add information on the metadata of the documents that is also shown to improve the performance of VectorRAG [8].

To the best of our knowledge the present work is the first work that proposes a RAG approach that is hybrid of both VectorRAG and GraphRAG and demonstrates its potential of more effective analysis and utilization of financial documents by leveraging the combined strengths of VectorRAG and GraphRAG. We also utilize a novel ground-truth Q&A dataset extracted from publicly available

financial call transcripts of the companies included in the Nifty-50 index which is an Indian stock market index that represents the weighted average of 50 of the largest Indian companies listed on the National Stock Exchange[1].

2 METHODOLOGY

In this Section, we provide details of the proposed methodology by first discussing details of VectorRAG, then methodologies of constructing KGs from given documents and our proposed methodology of GraphRAG and finally the HybridGraph technique.

2.1 VectorRAG

The traditional RAG [9] process begins with a query that is related to the information possessed within external document(s) that are not a part of the training dataset for the LLM. This query is used to search an external repository, such as a vector database or indexed corpus, to fetch relevant documents or passages containing useful information. These retrieved documents are then fed back into the LLM as additional context. Hence, in turn, for the given query, the language model generates a response based not only on its internal training data but also by incorporating the retrieved external information. This integration ensures that the generated content is grounded in more recent and verifiable data, improving the accuracy and contextual relevance of the responses. By combining the retrieval of external information with the generative capabilities of large language models, RAG enhances the overall quality and reliability of the generated text.

In traditional VectorRAG, the given external documents are divided into multiple chunks because of the limitation of context size of the language model. Those chunks are converted into embeddings using an embeddings model and then stored into a vector database. After that, the retrieval component performs a similarity search within the vector database to identify and rank the chunks most relevant to the query. The top-ranked chunks are retrieved and aggregated to provide context for the generative model.

Then, the generative model takes this retrieved context along with the original query and synthesizes a response. Thus, it merges the real-time information from the retrieved chunks with its pre-existing knowledge, ensuring that the response is both contextually relevant and detailed.

The schematic diagram in Figure 1 provides details on the part of RAG that generates vector database from given external documents in the traditional VectorRAG methodology where we also include explicit reference of metadata [8]. Section 4.2 will provide implementation details for our experiments.

2.2 Knowledge Graph Construction

A KG is a structured representation of real-world entities, their attributes, and their relations, usually stored in a graph database or a triplet store, i.e., a KG consists of nodes that represent entities and edges that represent relations, as well as labels and attributes for both. A graph triplet is a basic unit of information in a KG, consisting of a subject, a predicate, and an object.

In most methodologies to build a KG from given documents, three main steps are involved: knowledge extraction, knowledge

---

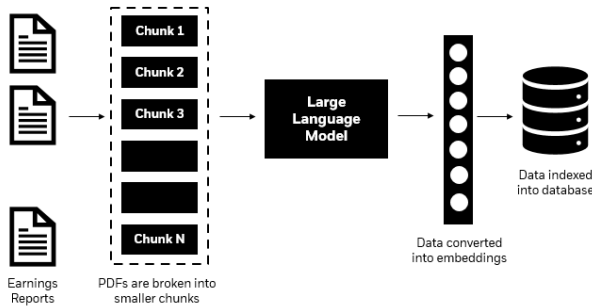[1]https://www.nseindia.com/products/content/equities/indices/nifty_50.htm

Figure 1: A schematic diagram describing the vector database creation of a RAG application.

improvement, and knowledge adaptation [28]. Here, we do not use knowledge adaptation and treat the KGs as static graphs.

Knowledge Extraction:- This step aims to extract structured information from unstructured or semi-structured data, such as text, databases, and existing ontologies. The main tasks in this step are entity recognition, relationship extraction, and co-reference resolution. Entity recognition and relationship extraction techniques use typical NLP tasks to identify entities and their relationships from textual sources [29]. Coreference resolution identifies and connects different references of the same entity, keeping coherence within the knowledge graph. For example, if the text refers to a company as both "the company" and "it", coreference resolution can link these mentions to the same entity node in the graph.

Knowledge Improvement:- This step aims to enhance the quality and completeness of the KG by removing redundancies and addressing gaps in the extracted information. The primary tasks in this step are KG completion and fusion. KG completion technique infers missing entities and relationships within the graph using methods such as link prediction and entity resolution. Link prediction predicts the existence and type of a relation between two entities based on the graph structure and features, while entity resolution matches and merges different representations of the same entity from different sources.

Knowledge fusion combines information from multiple sources to create a coherent and unified KG. This involves resolving conflicts and redundancies among the sources, such as contradictory or duplicate facts, and aggregating or reconciling the information based on rules, probabilities, or semantic similarity.

We utilized a robust methodology for creating KG triplets from unstructured text data, specifically focusing on corporate documents such as earnings call transcripts, adapted from Ref. [18?]. This process involves several interconnected stages, each designed to extract, refine, and structure information effectively.

We implement a two-tiered LLM chain for content refinement and information extraction. The first tier employs an LLM to generate an abstract representation of each document chunk. This refinement process is crucial as it distills the essential information while preserving the original meaning and key relationships between concepts that serves as a more focused input for subsequent processing, enhancing the overall efficiency and accuracy of our

triplet extraction pipeline. The second tier of our LLM chain is dedicated to entity extraction and relationship identification.

Both the steps are executed using carefully performed prompt engineering on a pre-trained LLM. A detailed discussion on implementation of the methodology will be provided in Section 4.1

2.3 GraphRAG

KG based RAG [22], or GraphRAG, also begins with a query based on the user's input same as VectorRAG. The main difference between VectorRAG and GraphRAG lies in the retrieval part. The query here is now used to search the KG to retrieve relevant nodes (entities) and edges (relationships) related to the query. A subgraph, which consists of these relevant nodes and edges, is extracted from the full KG to provide context. This subgraph is then integrated with the language model's internal knowledge, by encoding the graph structure into embeddings that the model can interpret. The language model uses this combined context to generate responses that are informed by both the structured information from the KG and its pre-trained knowledge. Crucially, when responding to user queries about a particular company, we leveraged the metadata information to selectively filter and retrieve only those document segments pertinent to the queried company [8]. This integration helps ensure that the generated outputs are accurate, contextually relevant, and grounded in verifiable information.

A schematic diagram of the retrieval methodology of GraphRAG is given in Figure 2. Here we first write a prompt to clean the data and then write another prompt in the second stage to create knowledge triplets along with metadata. It will be covered in more detail in section 4.1
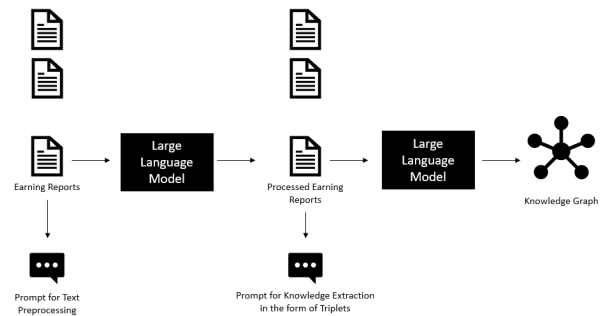

Figure 2: A schematic diagram describing knowledge graph creation process of GraphRAG.

2.3.1 HybridRAG. For the HybridRAG technique, we propose to integrate the aforementioned two distinct RAG techniques: VectorRAG and GraphRAG. This integration involves a systematic combination of contextual information retrieved from both the traditional vector-based retrieval mechanism and the KG-based retrieval system, the latter of which was constructed specifically for this study.

The amalgamation of these two contexts allows us to leverage the strengths of both approaches. The VectorRAG component provides a broad, similarity-based retrieval of relevant information, while the GraphRAG element contributes structured, relationship-rich

contextual data. This combined context is then utilized as input for a LLM to generate the final responses. Details on the implementation of the HybridRAG will be provided in Section 4.4.

2.4 Evaluation Metrics

To assess the efficacy of this integrated approach, we conducted a comparative analysis among the three approaches in a controlled experimental set up: VectorRAG, GraphRAG and HybridRAG. The responses generated using the combined VectorRAG and GraphRAG contexts were juxtaposed against those produced individually by VectorRAG and GraphRAG. This comparative evaluation aimed to discern potential improvements in response quality, accuracy, and comprehensiveness that might arise from the synergistic integration of these two RAG methodologies.

To objectively evaluate different RAG approaches (VectorRAG and GraphRAG in their case), Ref. [22] utilized metrics such as comprehensiveness (i.e., the amount of details the answer provides to cover all aspects and details of the question?); diversity (i.e., the richness of the answer in providing different perspectives and insights on the question); empowerment (i.e., the helpfulness of the answer to the reader understand and make informed judgements about the topic); and, directness (i.e., clearness of the answer in addressing the question). here, the LLM was provided tuples of question, target metric, and a pair of answers, and was asked to assess which answer was better according to the metric and why.

These metrics though compare the final generated answers, do not necessarily directly evaluate the retrieval and generation parts separately. Instead, here we implement a comprehensive set of evaluation metrics which are designed to capture different aspects of a given RAG system's output quality, focusing on faithfulness, answer relevance, and context relevance [30]. Each metric provides unique insights into the system's capabilities and limitations.

2.4.1 Faithfulness. Faithfulness is a crucial metric that measures the extent to which the generated answer can be inferred from the provided context. Our implementation of the faithfulness metric involves a two-step process:

Statement Extraction:- We use an LLM to decompose the generated answer into a set of concise statements. This step is crucial for breaking down complex answers into more manageable and verifiable units. The prompt used for this step is:

"Given a question and answer, create one or more statements from each sentence in the given answer. question: [question] answer: [answer]".

Statement Verification:- For each extracted statement, we employ the LLM to determine if it can be inferred from the given context. This verification process uses the following prompt:

"Consider the given context and following statements, then determine whether they are supported by the information present in the context. Provide a brief explanation for each statement before arriving at the verdict (Yes/No). Provide a final verdict for each statement in order at the end in the given format. Do not deviate from the specified format. statement: [statement 1] ... statement: [statement n]".

The faithfulness score (F) is $F = |V|/|S|$, where $|V|$ is the number of supported statements and $|S|$ is the total number of statements.

2.4.2 Answer Relevance:-. The answer relevance metric assesses how well the generated answer addresses the original question, irrespective of factual accuracy. This metric helps identify cases of incomplete answers or responses containing irrelevant information. Our implementation involves the following steps:

Question Generation: We prompt the LLM to generate n potential questions based on the given answer:

"Generate a question for the given answer. answer: [answer]".

Then, we obtain embeddings for all generated questions and the original question using OpenAI's text-embedding-ada-002 model[2]. We then calculate the cosine similarity between each generated question's embedding and the original question's embedding.

Finally, the answer relevance score (AR) is computed as the average similarity across all generated questions: $AR = \frac{1}{n}\sum(sim(q, q_i))$, where $sim(q, q_i)$ is the cosine similarity between the embedding of the original question q and the embeddings of each of the n generated questions q_i.

2.4.3 Context Precision. It is a metric used to evaluate the relevance of retrieved context chunks in relation to a specified ground truth for a given question[3]. It calculates the proportion of relevant items that appear in the top ranks of the context. The formula for context precision at K is the sum of the products of precision at each rank k and a binary relevance indicator v_k, divided by the total number of relevant items in the top K results. Precision at each rank k is determined by the ratio of true positives at k to the sum of true positives and false positives at k. This metric helps in assessing how well the context supports the ground truth, aiming for higher scores which indicate better precision.

2.4.4 Context Recall. It is a metric used to evaluate how well the retrieved context aligns with the ground truth answer, which is considered the definitive correct response[4]. It is quantified by comparing each sentence in the ground truth answer to see if it can be traced back to the retrieved context. The formula for context recall is the ratio of the number of ground truth sentences that can be attributed to the context to the total number of sentences in the ground truth. Higher values, ranging from 0 to 1, indicate better alignment and thus better context recall. This metric is crucial for assessing the effectiveness of information retrieval systems in providing relevant context.

3 DATA DESCRIPTION

Although there do exist some public financial datasets, none of them were suitable for the present experiments: e.g., FinQA [31], TAT-QA [32], FIQA [33], FinanceBench [34], etc. datasets are limited to specific usecases such as benchmarking LLMs' abilities to perform complex numerical reasoning or sentiment analysis. On the other hand, FinTextQA [35] dataset was not publicly available at the time of writing the present work. In addition, in most of these datasets, access to the actual documents from which the ground-truth Q&As were created is not available, making it impossible to use them for our RAG techniques evaluation purposes. Hence, we resorted to a dataset of our own though through publicly available

---

[2]https://platform.openai.com/docs/guides/embeddings/embedding-models
[3]https://docs.ragas.io/en/stable/concepts/metrics/context_precision.html
[4]https://docs.ragas.io/en/stable/concepts/metrics/context_recall.html

documents but such that we finally have access to both the actual financial documents and the ground-truth Q&As. Datasets like FinanceBench[5] provides question-context-answer triplets but they are not useful as here we are comparing VectorRAG, GraphRAG and HybridRAG and they do not provide the context generated from a KG. A recent paper [22] has not made the KG and triplets constructed by their algorithm public to the best of our knowledge either.

In short, there is no publicly available benchmark dataset to compare VectorRAG and GraphRAG either for financial or general domains to the best of our knowledge. Hence, we had to rely on our own dataset constructed as explained below.

We used transcripts from earnings calls of Nifty 50 constituents for our analysis. The NIFTY 50 is popular index in the Indian stock market that represents the weighted average of 50 of the largest Indian companies listed on the National Stock Exchange (NSE). The dataset of the earning call documents of Nifty 50 companies is widely recognized in the investment realm and is esteemed as an authoritative and extensive collection of earnings call transcripts. In our investigation, we focus on data spanning the quarter ending in June, 2023 i.e. the earnings reports for Q1 of the financial year 2024 (A financial year in India starts on the 1st April and ends in 31st March, so the quarter from 1st April to 30th June is the first quarter of 2024 for the Indian market).

Our dataset encompasses 50 transcripts for this quarter, spanning over 50 companies within Nifty 50 universe from diverse range of sectors including Infrastructure, Healthcare, Consumer Durables, Banking, Automobile, Financial Services, Energy - Oil & Gas, Telecommunication, Consumer Goods, Pharmaceuticals, Energy - Coal, Materials, Information Technology, Construction, Diversified, Metals, Energy - Power and Chemicals providing a substantial and diverse foundation for our study.

We start the data collection process focused on acquiring earnings reports from company websites within the Nifty 50 universe by developing and deploying a custom web scraping tool to navigate through the websites of each company within the Nifty 50 index, systematically retrieving the pertinent earnings reports for Q1 of the financial year 2024. By utilizing this web scraping approach, we aimed to compile a comprehensive dataset encompassing the earnings reports of the constituent companies.

Table 1 summarizes basic statistics of the documents we will be experimenting with in the remainder of this work.

Number of companies/documents	50
Average number of pages	27
Average number of questions	16
Average number of tokens	60,000

Table 1: Summary Statistics for the call transcript documents used in the present work.

These call transcripts documents consist of questions and answers between financial analysts and the company representatives for the respective companies, hence, there already exist certain Q&A pairs within these documents along with additional text. We examined the earnings reports within the Nifty50 universe, systematically curated a comprehensive array of randomly selected 400

questions posed during the earnings calls from all the documents, and gathered the exact responses corresponding to these questions. These questions constitute the specific queries articulated by financial analysts to the management during these calls.

4 IMPLEMENTATION DETAILS

In this Section, we provide details of implementation of the proposed methodology.

4.1 Knowledge Graph Construction

The initial phase of our approach centers on document preprocessing. We utilize the PyPDFLoader[6] to import PDF documents, which are subsequently segmented into manageable chunks using the RecursiveCharacterTextSplitter. This chunking strategy employs a size of 2024 characters with an overlap of 204 characters, ensuring comprehensive coverage while maintaining context across segment boundaries.

Following the preprocessing stage, we implement the two-tiered language model chain for content refinement and information extraction. It is not possible to include the exact prompt here due to the limited space, but a baseline prompt can be found in Ref. [18].

Entity Type	Examples
Companies and Corporations	Official names, abbreviations, informal references
Financial Metrics and Indicators	Revenue, profit margins, EBITDA
Corporate Executives and Key Personnel	CEOs, CFOs, board members
Products and Services	Tangible products and intangible services
Geographical Locations	Headquarters, operational regions, markets
Corporate Events	Mergers, acquisitions, product launches, earnings calls
Legal and Regulatory Information	Legal cases, regulatory compliance

Table 2: Entities extracted from earnings call transcripts

Table 2 summarizes details on entities extracted from the earning calls transcripts using our prompt based method. Concurrently, LLM identifies relationships between these entities using a curated set of verbs, capturing the nuanced interactions within the corporate narrative. A key improvement in our methodology is the enhanced prompt engineering to generate the structured output format for knowledge triplets. Each triplet is represented as a nested list ['h', 'type', 'r', 'o', 'type', 'metadata'], where 'h' and 'o' denote the head and object entities respectively, 'type' specifies the entity category, 'r' represents the relationship, and 'metadata' encapsulates additional contextual information. This format allows for a rich, multidimensional representation of information, facilitating more nuanced downstream analysis.

Our process incorporates several advanced features to enhance the quality and utility of the extracted triplets. Entity disambiguation techniques are employed to consolidate different references to the same entity, improving consistency across the KG. We also prioritize conciseness in entity representation, aiming for descriptions of less than four words where possible, which aids in maintaining a clean and navigable graph structure.

The extraction pipeline is applied iteratively to each document chunk, with results aggregated to form a comprehensive knowledge graph representation of the entire document, allowing for scalable processing of large documents while maintaining local context within each chunk. We have added explicit instruction on obtaining metadata following Ref. [8] for both VectorRAG and GraphRAG.

Finally, we implement a data persistence strategy, converting the extracted triplets from their initial string format to Python data structures and storing them in a pickle file. This facilitates easy retrieval and further manipulation of the knowledge graph data in subsequent analysis stages.

Our methodology represents a significant advancement in automated knowledge extraction from corporate documents. By combining advanced NLP techniques with a structured approach to information representation, we create a rich, queryable knowledge base that captures the complex relationships and key information present in corporate narratives. This approach opens up new possibilities for financial analysis and automated reasoning in the business domain that will be explored further in the future.

4.2 VectorRAG

Our methodology builds upon the concept of RAG [9] which allows for the creation of a system that can provide context-aware, accurate responses to queries about company financial information, leveraging both the power of large language models and the efficiency of semantic search.

At the core of our system is a Pinecone vector database[7], which serves as the foundation for our information retrieval process. We employ OpenAI's text-embedding-ada-002 model to transform textual data from earnings call transcripts into high-dimensional vector representations. This vectorization process enables semantic similarity searches, significantly enhancing the relevance and accuracy of retrieved information. Table 3 provides summary of the configuration of the set up in use for our experiments.

LLM	GPT-3.5-Turbo
LLM Temperature	0
Embedding Model	text-embedding-ada-002
Framework	LangChain
Vector Database	Pinecone
Chunk Size	1024
Chunk Overlap	0
Maximum Output Tokens	1024
Chunks for Similarity Algorithm	20
Number of Context Retrieved	4

Table 3: VectorRAG Configuration

The Q&A pipeline is constructed using the LangChain framework[8]. The begins with a context retrieval step, where we query the Pinecone vector store to obtain the most relevant document chunks for a given question. This retrieval process is fine-tuned with specific filters for quarters, years, and company names, ensuring that the retrieved information is both relevant and current.

Following retrieval, we implement a context formatting step that consolidates the retrieved document chunks into a coherent context string. This formatted context serves as the informational basis for the language model's response generation. We have developed a sophisticated prompt template, that instructs the language model to function as an expert Q&A system, emphasizing the importance of utilizing only the provided context information and avoiding direct references to the context in the generated responses.

For the core language processing task, we integrate OpenAI's GPT-3.5-turbo model which processes the formatted context and query to generate natural language responses that are informative, coherent, and contextually appropriate.

To evaluate the performance of our system, we developed a comprehensive framework that includes the preparation of a custom dataset of question-answer pairs specific to each company's earnings call. Our system processes each question in this dataset, generating answers based on the retrieved context. The evaluation results, including the original question, generated answer, retrieved contexts, and ground truth, are compiled into structured formats (CSV and JSON) to facilitate further analysis. The outputs generated by our system are stored in both CSV and JSON formats, enabling easy integration with various analysis tools and dashboards. This approach facilitates both quantitative performance metrics and qualitative assessment of the system's responses, providing a comprehensive view of its effectiveness.

By parameterizing company names, quarters, and years, we can easily adapt the system to different datasets and time periods. This design choice allows for seamless integration of new data and expansion to cover multiple companies and earnings calls.

4.3 GraphRAG

For GraphRAG, we developed an Q&A system specifically designed for corporate earnings call transcripts. Our implementation of GraphRAG leverages several key components and techniques:

LLM	GPT-3.5-Turbo
LLM Temperature	0
Framework	LangChain
KG Manipulation	Networkx
Chunk Size	1024
Chunk Overlap	0
Number of Triplets	13950
Number of nodes	11405
Number of edges	13883
DFS Depth	1

Table 4: GraphRAG Configuration

Knowledge Graph Construction:- We begin by constructing a KG from a set of knowledge triplets extracted from corporate documents using the prompt engineering based methodology as described in Section 4.1. These triplets, stored in a pickle file, represent structured information in the form of subject-predicate-object relationships. We use the NetworkxEntityGraph class from the LangChain library to create and manage this graph structure. Each triple is added to the graph, which encapsulates the head entity, relation, tail entity, and additional metadata.

We implement the Q&A functionality using the GraphQAChain class from LangChain. This chain combines the KG with an LLM (in our case, OpenAI's GPT-3.5-turbo) to generate responses. The GraphQAChain traverses the KG to find relevant information and uses the language model to formulate coherent answers based on the retrieved context. A summary of configuration of our LLM models and other libraries used for GraphRAG is shown in Table 4.

In KG as the information is stored in the form of entities and relationships and there can be multiple relations emanating from one single entity, in this experiment, to extract relevant information from the KG, we employ a depth-first search strategy constrained by a depth of one from the specified entity.

To prepare for assessing the performance of our GraphRAG system, we follow the below steps: **Dataset Preparation:-** We use a pre-generated CSV file containing question-answer pairs specific to the earnings call transcript.

Iterative Processing:- For each question in the dataset, we run the GraphQAChain to generate an answer.

Result Compilation:- We compile the results, including the original questions, generated answers, retrieved contexts, and ground truth answers, into a structured format.

Finally, the evaluation results are saved in both CSV and JSON formats for further analysis and comparison. We then fed these outputs into our RAG evaluation pipeline. For each Q&A pair in our dataset, we compute all three metrics: faithfulness, answer relevance, context precision and context recall.

4.4 HybridRAG

For the proposed HybridRAG technique, upon obtaining all the contextual information from VectorRAG and GraphRAG, we concatenate these contexts to form a unified context utilizing both techniques. This combined context is then fed into the answer generator model to produce a response. The context used for response generation is relatively larger, which affects the precision of the generated response. The context from VectorRAG is appended first, followed by the context from GraphRAG. Consequently, the precision of the generated answer depends on the source context. If the answer is generated from the GraphRAG context, it will have lower precision, as the GraphRAG context is added last in the sequence of contexts provided to the answer generator model, and vice versa.

5 RESULTS

We evaluate both the retrieval and generation parts of RAG for the three different RAG pipelines. Evaluating the RAG outputs is also an active area of research there is no standard tool which is universally accepted as of yet, though we use a currently popular framework RAGAS[30] to evaluate the three RAG pipelines in the

present work where we have modified them a bit to make more precise comparisons.

The results of our comparative analysis reveal notable differences in performance among VectorRAG, GraphRAG, and HybridRAG approaches as summarized in Table 5. In terms of Faithfulness, GraphRAG and HybridRAG demonstrated superior performance, both achieving a score of 0.96, while VectorRAG trailed slightly with a score of 0.94. Answer relevancy scores varied across the methods, with HybridRAG outperforming the others at 0.96, followed by VectorRAG at 0.91, and GraphRAG at 0.89. Context precision was highest for GraphRAG at 0.96, significantly surpassing VectorRAG (0.84) and HybridRAG (0.79). However, in context recall, both VectorRAG and HybridRAG achieved perfect scores of 1, while GraphRAG lagged behind at 0.85.

Overall, these results suggest that GraphRAG offers improvements over VectorRAG, particularly in faithfulness and context precision. Furthermore, HybridRAG emerges as the most balanced and effective approach, outperforming both VectorRAG and GraphRAG in key metrics such as faithfulness and answer relevancy, while maintaining high context recall.

The relatively lower context precision observed for HybridRAG (0.79) can be attributed to its unique approach of combining contexts from both VectorRAG and GraphRAG methods. While this integration allows for more comprehensive information retrieval, it also introduces additional content that may not align precisely with the ground truth, thus affecting the context precision metric. Despite this trade-off, HybridRAG's superior performance in faithfulness, answer relevancy, and context recall underscores its effectiveness. When considering the overall evaluation metrics, HybridRAG emerges as the most promising approach, balancing high-quality answers with comprehensive context retrieval.

Overall GraphRAG performs better in extractive questions compared to VectorRAG. And VectorRAG does better in abstractive questions where information is not explicitly mentioned in the raw data. And also GraphRAG sometimes fails to answer questions correctly whenever there is no entity explicitly mentioned in the question. So HybridRAG does a good job overall, as whenever VectorRAG fails to fetch correct context in extractive questions it falls back to GraphRAG to generate the answer. And whenever GraphRAG fails to fetch correct context in abstractive questions it falls back to VectorRAG to generate the answer.

	VectorRAG	GraphRAG	HybridRAG
F	0.94	0.96	0.96
AR	0.91	0.89	0.96
CP	0.84	0.96	0.79
CR	1	0.85	1

Table 5: Performance Metrics for Different RAG Pipelines. Here, F, AR, CP and CR refer to Faithfulness, Answer Relevence, Context Precision and Context Recall.

6 CONCLUSION AND FUTURE DIRECTIONS

Among the current approaches to mitigate issues regarding information extraction from external documents that were not part of

training data for the LLM, Retrieval Augmented Generation (RAG) techniques have emerged as the most popular ones that aim to improve the performance of LLMs by incorporating relevant retrieval mechanisms. RAG methods enhance the LLMs' capabilities by retrieving pertinent documents or text to provide additional context during the generation process. However, these approaches encounter significant limitations when applied to the specialized and intricate domain of financial documents. Furthermore, the quality of the retrieved context from a vast and heterogeneous corpus can be inconsistent, leading to inaccuracies and incomplete analyses. These challenges highlight the need for more sophisticated methods that can effectively integrate and process the detailed and domain-specific information found in financial documents, ensuring more reliable and accurate outputs for informed decision-making.

In the present work, we have introduced a novel approach that significantly advances the field of information extraction from financial documents through the development of a hybrid RAG system. This system, called HybridRAG, which integrates the strengths of both Knowledge Graphs (KGs) and advanced language models, represents a leap forward in our ability to extract and interpret complex information from unstructured financial texts. The hybrid RAG system, by combining traditional vector-based RAG and KG-based RAG, has shown superior performance in terms of retrieval accuracy and answer generation, marking a pivotal step towards more effective financial analysis tools.

Through a comparative analysis using objecive evaluation metrics, we have highlighted the distinct performance advantages of the HybridRAG approach over its vector-based and KG-based counterparts. The HybridRAG system excels in faithfulness, answer relevancy, and context recall, demonstrating the benefits of integrating contexts from both VectorRAG and GraphRAG methods, despite potential trade-offs in context precision.

The implications of this research extend beyond the immediate realm of financial analysis. By developing a system capable of understanding and responding to nuanced queries about complex financial documents, we pave the way for more sophisticated AI-assisted financial decision-making tools that could potentially democratize access to financial insights, allowing a broader range of stakeholders to engage with and understand financial information.

Future directions for this research include expanding the system to handle multi-modal inputs, incorporating numerical data analysis capabilities, and developing more sophisticated evaluation metrics that capture the nuances of financial language and the accuracy of numerical information in the responses. Additionally, exploring the integration of this system with real-time financial data streams could further enhance its utility in dynamic financial environments.

7 ACKNOWLEDGEMENT

REFERENCES

[1] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. 2013.

[2] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. Advances in neural information processing systems, 30, 2017.

[3] Yi Yang, Mark Christopher Siy Uy, and Allen Huang. Finbert: A pretrained language model for financial communications. arXiv preprint arXiv:2006.08097, 2020.

[4] Shijie Wu, Ozan Irsoy, Steven Lu, Vadim Dabravolski, Mark Dredze, Sebastian Gehrmann, Prabhanjan Kambadur, David Rosenberg, and Gideon Mann. Bloomberggpt: A large language model for finance. arXiv preprint arXiv:2303.17564, 2023.

[5] Yuqi Nie, Yaxuan Kong, Xiaowen Dong, John M Mulvey, H Vincent Poor, Qingsong Wen, and Stefan Zohren. A survey of large language models for financial applications: Progress, prospects and challenges. arXiv preprint arXiv:2406.11903, 2024.

[6] Huaqin Zhao, Zhengliang Liu, Zihao Wu, Yiwei Li, Tianze Yang, Peng Shu, Shaochen Xu, Haixing Dai, Lin Zhao, Gengchen Mai, Ninghao Liu, and Tianming Liu. Revolutionizing finance with llms: An overview of applications and insights, 2024.

[7] Chen Ling, Xujiang Zhao, Jiaying Lu, Chengyuan Deng, Can Zheng, Junxiang Wang, Tanmoy Chowdhury, Yun Li, Hejie Cui, Xuchao Zhang, et al. Domain specialization as the key to make large language models disruptive: A comprehensive survey. arXiv preprint arXiv:2305.18703, 2023.

[8] Bhaskarjit Sarmah, Dhagash Mehta, Stefano Pasquali, and Tianjie Zhu. Towards reducing hallucination in extracting information from financial reports using large language models. In Proceedings of the Third International Conference on AI-ML Systems, pages 1–5, 2023.

[9] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. Advances in Neural Information Processing Systems, 33:9459–9474, 2020.

[10] Gautier Izacard and Edouard Grave. Leveraging passage retrieval with generative models for open domain question answering. arXiv preprint arXiv:2007.01282, 2021.

[11] Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Ming-Wei Chang. Realm: Retrieval-augmented language model pre-training. arXiv preprint arXiv:2002.08909, 2020.

[12] Antonio Jose Jimeno Yepes et al. Financial report chunking for effective retrieval augmented generation. arXiv preprint arXiv:2402.05131, 2024.

[13] SuperAcc. Retrieval-augmented generation on financial statements. SuperAcc Insights, 2024.

[14] Shaoxiong Ji, Shirui Pan, Erik Cambria, Pekka Marttinen, and S Yu Philip. A survey on knowledge graphs: Representation, acquisition, and applications. IEEE transactions on neural networks and learning systems, 33(2):494–514, 2021.

[15] Heiko Paulheim. Knowledge graphs: State of the art and future directions. Semantic Web Journal, 10(4):1–20, 2017.

[16] Aidan Hogan, Eva Blomqvist, Michael Cochez, Claudia D'Amato, Gerard de Melo, Claudio Gutiérrez, and Maria ... Maleshkova. Knowledge graphs. arXiv preprint arXiv:2003.02320, 2021.

[17] Lisa Ehrlinger and Wolfram Wöß. Towards a definition of knowledge graphs. In SEMANTiCS (Posters, Demos, SuCCESS), pages 1–4, 2016.

[18] Xiaohui Victor Li and Francesco Sanna Passino. Findkg: Dynamic knowledge graphs with large language models for detecting global trends in financial markets. arXiv preprint arXiv:2407.10909, 2024.

[19] Shourya De, Anima Aggarwal, and Anna Cinzia Squicciarini. Financial knowledge graphs: A novel approach to empower data-driven financial applications. In IEEE International Conference on Big Data (Big Data), pages 1311–1320, 2018.

[20] Marina Petrova and Birgit Reinwald. Knowledge graphs in finance: Applications and opportunities. Journal of Financial Data Science, 2(2):10–19, 2020.

[21] Wei Liu, Jun Zhang, and Li Pan. Utilizing knowledge graphs for financial data integration and analysis. Data Science Journal, 18(1):1–15, 2019.

[22] Darren Edge, Ha Trinh, Newman Cheng, Joshua Bradley, Alex Chao, Apurva Mody, Steven Truitt, and Jonathan Larson. From local to global: A graph rag approach to query-focused summarization. arXiv preprint arXiv:2404.16130, 2024.

[23] Xuchen Yao, Yanan Sun, Zhen Huang, and Dong Li. Retrieval-augmented generation for knowledge-intensive nlp tasks. arXiv preprint arXiv:2101.07554, 2021.

[24] Weizhi Zhao, Haoyu Chen, Kai Liu, and Jun Zhao. Graph-based retrieval-augmented generation for open-domain question answering. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 36, pages 3098–3106, 2022.

[25] Bill Yuchen Lin, Yuanhe Liu, Ming Shen, and Xiang Ren. Kgpt: Knowledge-grounded pre-training for data-to-text generation. In Findings of the Association for Computational Linguistics: EMNLP 2020, pages 710–724, 2020.

[26] Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, and Haofen Wang. Retrieval-augmented generation for large language models: A survey. arXiv preprint arXiv:2312.10997, 2023.

[27] Tyler Procko. Graph retrieval-augmented generation for large language models: A survey. Available at SSRN, 2024.

[28] Lingfeng Zhong, Jia Wu, Qian Li, Hao Peng, and Xindong Wu. A comprehensive survey on automatic knowledge graph construction. ACM Computing Surveys, 56(4):1–62, 2023.

[29] Ishani Mondal, Yufang Hou, and Charles Jochim. End-to-end nlp knowledge graph construction. arXiv preprint arXiv:2106.01167, 2021.

[30] Shahul Es, Jithin James, Luis Espinosa-Anke, and Steven Schockaert. Ragas: Automated evaluation of retrieval augmented generation. arXiv preprint arXiv:2309.15217, 2023.

[31] Zhiyu Chen, Wenhu Chen, Charese Smiley, Sameena Shah, Iana Borova, Dylan Langdon, Reema Moussa, Matt Beane, Ting-Hao Huang, Bryan Routledge, et al. Finqa: A dataset of numerical reasoning over financial data. arXiv preprint arXiv:2109.00122, 2021.

[32] Macedo Maia, Siegfried Handschuh, André Freitas, Brian Davis, Ross McDermott, Manel Zarrouk, and Alexandra Balahur. Www'18 open challenge: financial opinion mining and question answering. In Companion proceedings of the the web conference 2018, pages 1941–1942, 2018.

[33] Fengbin Zhu, Wenqiang Lei, Youcheng Huang, Chao Wang, Shuo Zhang, Jiancheng Lv, Fuli Feng, and Tat-Seng Chua. Tat-qa: A question answering benchmark on a hybrid of tabular and textual content in finance. arXiv preprint arXiv:2105.07624, 2021.

[34] Pranab Islam, Anand Kannappan, Douwe Kiela, Rebecca Qian, Nino Scherrer, and Bertie Vidgen. Financebench: A new benchmark for financial question answering. arXiv preprint arXiv:2311.11944, 2023.

[35] Jian Chen, Peilin Zhou, Yining Hua, Yingxin Loh, Kehui Chen, Ziyuan Li, Bing Zhu, and Junwei Liang. Fintextqa: A dataset for long-form financial question answering. arXiv preprint arXiv:2405.09980, 2024.