Literature Review

This literature review will begin by providing an overview of the state of the art technologies, frameworks and methodologies used in RAG applications using a graph knowledge base. This will include preprocessing of documents, knowledge graph creation, embedding, retrieval of information and generation using LLMs. Later we will discuss about some relevant studies about RAG applications in non-finance fields, particularly medical applications, because they may be helpful when targetting domain specific usage of the RAG architecture. And after that we will summarize some studies about the use RAG, data quality, etc, for FinTech applications, specially financial planning, which will be the focus of the proposal of this document.

Preprocessing and Data Storage

Graph databases have emerged as specialized systems designed to store and manage data in graph formats, emphasizing the relationships between data points. Technologies such as Neo4j, Amazon Neptune, and ArangoDB offer scalable solutions for handling complex graph structures. These databases enable efficient querying and traversal of relationships, making them ideal for applications requiring high-performance retrieval and manipulation of interconnected data, which is fundamental for Graph RAG systems.

Effective storage and retrieval in Graph RAGs rely on optimized data structures and indexing methods that facilitate rapid access to relevant nodes and edges. Techniques such as adjacency lists, path indexing, and graph partitioning are employed to manage large graphs efficiently. Query languages like Cypher and Gremlin allow for expressive querying capabilities, enabling RAG systems to retrieve pertinent subgraphs or relationships necessary for informed content generation.

Building upon these foundational technologies, **Peng et al. (2024)** discussed the construction and indexing of graph databases in the context of GraphRAG, emphasizing both open-source and self-constructed graph databases. They explored various indexing methods, including graph, text, and vector indexing, to improve retrieval efficiency [1]. Their work highlights the importance of tailored indexing strategies in enhancing the performance of RAG systems.

A knowledge graph is a structured representation of real-world entities and their interrelations, organized in a graph format where nodes represent entities and edges represent relationships. Building a knowledge graph entails creating a structured representation of information where entities (such as people, places, or concepts) are nodes, and the relationships between them are edges connecting these nodes. The process begins with collecting data from various sources, including structured databases, unstructured text documents, and web content. This diverse data is then preprocessed to ensure consistency and quality, which involves cleaning to remove inaccuracies and normalizing to standardize formats. [2]

The core of knowledge graph construction involves extracting entities and the relationships between them using natural language processing techniques like named entity recognition and relation extraction. An ontology or schema is developed to define the categories of entities and the permissible relationships, providing a formal structure that underpins the knowledge graph. This ontology serves as a blueprint that guides the integration of data and ensures semantic consistency across the graph. [3]

Data integration involves aligning and merging data from different sources, which may refer to the same entities in various ways. Techniques like entity resolution help in identifying and unifying these references to maintain a cohesive graph. The construction phase involves creating the graph structure by instantiating nodes for entities and edges for relationships as defined by the ontology, and enriching them with attributes and properties. [3]

Once the knowledge graph is constructed, validation is crucial to ensure its accuracy and coherence. Logical reasoning and inference mechanisms can be applied to detect inconsistencies and derive new insights from the existing data. Maintenance of the knowledge graph involves regular updates with new information, scaling to accommodate growth, and optimizing for efficient querying and retrieval. [3]

The integration of knowledge graphs into RAG systems has shown significant benefits. **Hu et al. (2024)** demonstrate how knowledge graphs serve as rich information sources, providing relevant entities and their relationships, which lead to more coherent and accurate responses from language models [4]. By leveraging the structured information in knowledge graphs, RAG systems can generate content that is both contextually relevant and semantically accurate.

In specific domains like medicine, the structure of the graph data becomes even more critical. **Cui et al. (2024)** explore the application of Directed Acyclic Graph (DAG) task decomposition in the medical domain. They propose that hierarchical graph structures, such as DAGs, provide a systematic way to process complex medical tasks, enabling large language models to organize and represent inter-task dependencies more efficiently. This allows for more structured task execution and the use of metadata for task nodes, thereby enhancing the management and understanding of medical knowledge. Their research shows that DAG structures, especially in retrieval-augmented generation, can optimize medical task processing by enabling a clear breakdown of subtasks [5].

When considering the choice between different database systems, it's essential to analyze the trade-offs to ensure optimal performance. Compared to traditional relational databases, graph databases offer superior performance in handling complex, interconnected data due to their inherent design focused on relationships. While relational databases excel in structured, tabular data management, they are less efficient for operations requiring multiple joins or traversals. This emphasizes the advantages of graph databases in supporting the dynamic retrieval needs of RAG systems.

Adding to this perspective, **Jin et al. (2024)** discuss the performance benefits of graph-based vector databases, such as HNSW, used in Retrieval-Augmented Generation (RAG) systems. They highlight that vector databases structured as graphs offer significant advantages in terms of efficient search and retrieval compared to other types of databases. Specifically, graph databases are shown to be more efficient in managing large-scale vector searches through hierarchical structures, reducing computational complexity and enhancing query performance. This contrasts with traditional databases that may not be optimized for high-dimensional vector searches. Their work demonstrates that incorporating graph databases into RAG systems can improve knowledge retrieval efficiency, particularly for applications requiring high-accuracy vector similarity searches [6].

Moreover, the organization of data within graphs can further enhance retrieval processes. **Chang et al. (2024)** introduce CommunityKG-RAG, a novel framework that leverages community structures in knowledge graphs (KGs) to enhance retrieval-augmented generation systems. They emphasize that traditional RAG systems suffer from poor integration of structured knowledge, limiting their effectiveness in fact-checking tasks. By focusing on community detection within KGs, CommunityKG-RAG provides a more contextually rich retrieval process compared to standard databases or flat knowledge structures. This approach significantly enhances the retrieval of relevant information by utilizing the multi-hop relationships in KGs, showing the advantages of graph-based data structures over other types of databases in complex, fact-checking applications [7].

An essential aspect of improving retrieval in RAG systems is the use of semantic embeddings. Semantic embeddings represent words, phrases, or entities in continuous vector spaces, capturing their meanings and relationships. In RAG systems, these embeddings are crucial for mapping between the language model and the knowledge base. They enable the system to retrieve

semantically relevant information based on the context of the input, facilitating more coherent and accurate generation.

To generate these embeddings, various techniques are employed. Traditional methods like Word2Vec and GloVe capture semantic relationships based on word co-occurrence, while contextual models like BERT and GPT account for word meaning based on surrounding context. For graph data, methods like Node2Vec and GraphSAGE generate embeddings that capture structural information within the graph. Combining these approaches allows RAG systems to create rich representations that reflect both linguistic and relational properties. [1]

In the context of textual graph knowledge bases—where both nodes and edges have textual notations—pre-trained language models prove particularly beneficial. **He et al. (2024)** stated that using models such as **SentenceBERT** can enhance the system's ability to understand and retrieve relevant information [8]. This approach leverages the strengths of language models in capturing semantic nuances, thereby improving the retrieval process in RAG systems.

Semantic embeddings not only improve retrieval relevance but also enable better generalization across different contexts. Applications extend to semantic search, recommendation systems, and personalization. The benefits include more accurate responses, improved user satisfaction, and the ability to handle ambiguous or novel queries effectively. By integrating semantic embeddings, RAG systems become more adept at handling the complexities of natural language and the intricacies of user intent.

Concluding this discussion, **Peng et al. (2024)** further elaborated on how GraphRAG enhances retrieval performance by incorporating semantic embeddings derived from graph structures. This integration allows for a more nuanced understanding of relationships within the data, improving the quality of information retrieval for downstream tasks [1]. Their insights reinforce the critical role of semantic embeddings in advancing the capabilities of RAG systems.

Retrieval

Retrieval is a critical component in Graph Retrieval-Augmented Generation (Graph RAG) systems, determining the relevance and accuracy of the information drawn from the knowledge base for the generation process. Unlike traditional RAG systems that retrieve unstructured text, Graph RAGs leverage the structural properties of graphs, retrieving not just textual information but also relational data embedded within nodes, edges, and subgraphs. This allows for more nuanced responses, which are essential for applications requiring deep contextual understanding and interconnected knowledge. Efficient retrieval methods are essential for ensuring that the system can effectively process large, complex graph structures while maintaining response accuracy and speed.

The query processing stage in Graph RAG systems is designed to map a user's natural language input to relevant graph structures. This involves transforming the input into a query that can be understood by the underlying graph database. Techniques such as graph traversal algorithms, entity linking, and semantic matching are often employed to extract nodes, edges, or subgraphs relevant to the query [1]. The unique structure of graphs allows for retrieval methods that can account for both direct and indirect relationships between entities, thus providing richer context for the generation model.

For instance, **He et al. (2024)** propose a k-nearest neighbors (k-NN) retrieval approach that identifies relevant nodes and edges by measuring the cosine similarity between the query and the embeddings of graph elements. This method retrieves the top-k most relevant nodes and edges, ensuring that the query is mapped to meaningful graph structures and relationships, providing the system with precise data for subsequent generation tasks [8]. Similarly, **Hu et al. (2024)**

demonstrate how WeKnow leverages structured knowledge from graphs to improve the accuracy and relevance of information retrieved by language models, especially for complex queries requiring relational understanding. They propose a domain-specific RAG model based on a scraping mechanism, making the graph knowledge base the domain from which to scrape relevant information. This approach facilitates the assembly of a larger knowledge base with fewer resources such as memory or compute power [9].

In the realm of fact-checking, retrieval processes demand even greater precision and contextual awareness. **Chang et al. (2024)** contribute to improving retrieval processes by integrating knowledge graphs (KGs) with RAG systems using communities. They introduce a community-aware retrieval mechanism that leverages multi-hop pathways within KGs to improve the precision of retrieved information. This enhances the relevance and contextual accuracy of the data retrieved for fact-checking tasks, surpassing traditional retrieval methods. Their system also performs well in zero-shot learning settings, adapting to new queries without additional fine-tuning, making it highly scalable [7].

Due to the potentially large size of graph databases, especially in real-world applications, optimization strategies are necessary to enhance the efficiency of retrieval. Techniques such as graph indexing, approximate nearest neighbor search, and multi-stage retrieval methods are utilized to narrow down the search space and reduce computational overhead. In Graph RAGs, the trade-off between retrieval accuracy and speed is crucial, particularly in applications requiring real-time responses. Hybrid retrieval methods, which combine both parametric (e.g., neural models) and non-parametric techniques, are increasingly popular for balancing performance with scalability.

To address these challenges, **He et al. (2024)** introduce a method that constructs subgraphs based on the Prize-Collecting Steiner Tree (PCST) algorithm. This method assigns relevance scores (or "prizes") to nodes and edges, constructing a subgraph that maximizes relevance while minimizing retrieval cost. The adaptation of PCST for both nodes and edges allows for efficient subgraph construction that maintains manageable size without sacrificing important relational data [8]. This strategy is particularly effective in ensuring that only pertinent information is retrieved, reducing noise and computational load.

One of the limitations of Graph RAG systems is multi-hop reasoning, which involves arriving at answers that require several steps of inference within the knowledge graph. **Fang et al. (2024)** propose a solution to this problem by creating ground-based reasoning chains—chains of nodes and edges used to construct a response to a query. Essentially, they build a chain of thought where, given any point in the chain, one can move forward, with the last node being the answer to the query [10]. This method enhances the system's ability to handle complex queries that require understanding multiple relationships.

In specialized domains like medicine, the need for precise and efficient retrieval is even more pronounced. **Cui et al. (2024)** contribute to improving retrieval in RAG systems by proposing the Bailicai framework. They highlight the importance of refining retrieval processes to reduce noise from irrelevant documents and optimizing the timing of retrieval in large language models (LLMs). By utilizing adaptive strategies, such as self-knowledge boundary identification, Bailicai ensures that retrieval is only invoked when necessary, improving efficiency. This approach addresses the challenge of document noise and improves accuracy in medical retrieval tasks, significantly enhancing the integration of external knowledge in LLMs [5].

Similarly, **Jin et al. (2024)** make significant contributions to retrieval in RAG systems by introducing the RAGCache framework. They focus on optimizing retrieval processes by caching intermediate knowledge states, effectively reducing redundant computations. By leveraging

multilevel dynamic caching strategies, RAGCache enhances retrieval efficiency, especially for high-frequency document requests, which are cached to minimize retrieval latency. The study also discusses the performance bottlenecks of retrieval steps in RAG and how caching intermediate results can drastically cut down on retrieval and computational costs [6].

In the medical field, **Wu et al. (2024)** introduce a U-retrieve method, which enhances the retrieval efficiency of RAG systems in medical applications. U-retrieve balances global awareness and indexing efficiency by generating summaries at different graph levels and performing top-down matching. This ensures that the system retrieves highly relevant entities and their relationships across multiple layers of medical knowledge. Their method enables precise, evidence-based responses to medical queries, dramatically improving the reliability of retrieval-augmented language models in critical medical scenarios [11].

An essential component that enhances retrieval in Graph RAG systems is the utilization of Graph Neural Networks (GNNs). GNNs are a class of neural networks tailored to process graph-structured data by capturing the dependencies between nodes via message-passing mechanisms. They extend traditional neural networks by incorporating the topology of graphs into the learning process, enabling models to learn representations that reflect both the features of individual nodes and their structural context within the graph.

Several GNN architectures, such as Graph Convolutional Networks (GCNs), Graph Attention Networks (GATs), and Graph Recurrent Neural Networks (GRNNs), are particularly relevant to RAG systems. These architectures differ in how they aggregate and propagate information across the graph. For instance, GATs leverage attention mechanisms to weigh the importance of neighboring nodes, which can enhance the retrieval process in RAGs by focusing on more relevant relationships.

Peng et al. (2024) highlight the role of GNNs in GraphRAG systems, explaining that GNNs are used to represent graph structures and enhance the retrieval and generation processes. GNN-based retrievers are particularly suited for tasks involving complex relationships in graph-structured data [1]. In practice, GNNs are utilized to encode graph knowledge bases into embeddings that can be integrated with language models. Applications include improving entity recognition, relation extraction, and providing context-aware responses. GNNs enable the system to understand complex interactions within the graph, leading to more accurate retrieval and generation of information in response to user queries.

For example, **He et al. (2024)** use a Graph Attention Network (GAT) to encode the subgraph response from the retrieval phase using a pooling operation, followed by a multilayer perceptron, and finally using a text embedder to turn the output into textual form. This results in a smaller amount of tokens required to perform a question-answer operation, enhancing efficiency [8]. Similarly, **Fang et al. (2024)** propose the use of a GNN in the retrieval phase of the RAG system to retrieve the top-k relevant triplets (head, edge, tail) in the knowledge graph, improving the system's ability to handle complex queries [10].

Efficient query processing is critical for the performance of Graph RAG systems. Techniques such as graph traversal algorithms, indexing strategies, and approximate nearest neighbor search are employed to retrieve relevant information swiftly. Optimizing these processes ensures that the retrieval component does not become a bottleneck in the generation pipeline. **Hu et al. (2024)** highlight the importance of efficient graph querying and representation learning, embedding entities and relations into a shared space with the language model to facilitate seamless interaction between structured and unstructured data [4].

Optimization strategies in Graph RAGs focus on reducing latency and computational overhead. Methods include precomputing embeddings, using caching mechanisms, and employing parallel

processing. Additionally, pruning irrelevant parts of the graph and leveraging hierarchical structures can improve retrieval efficiency. **Peng et al. (2024)** explore how GraphRAG enhances query processing through techniques like query expansion and decomposition. These methods ensure more precise retrievals by refining the original query with additional context or breaking it down into smaller, more manageable parts [1].

Building upon these strategies, **Cui et al. (2024)** contribute to query optimization within the context of RAG systems by introducing mechanisms that determine whether a medical query requires external retrieval or can be resolved using internal model knowledge. Their self-knowledge boundary identification module optimizes the retrieval process by avoiding unnecessary external data retrieval, reducing computational time and resource consumption. The combination of this method with task decomposition ensures that each query is handled efficiently, either through internal knowledge or by accessing the relevant external documents. This dynamic and targeted retrieval process offers significant improvements in query processing for medical applications [5].

Jin et al. (2024) introduce several key optimizations for query processing in RAG systems. One of the primary contributions is the development of dynamic speculative pipelining, which overlaps the retrieval and inference stages to minimize query processing time. This technique ensures that as retrieval results are generated, they are immediately used in inference, reducing latency. Additionally, their cache-aware scheduling and a prefix-aware replacement policy ensure that frequently retrieved documents are quickly accessible, improving the overall query performance. This approach is shown to significantly reduce end-to-end query time, offering a scalable solution for systems handling large volumes of complex queries [6].

Moreover, **Huang et al. (2023)** introduce optimization strategies such as Retriever as Answer Classifier (RAC) and Dense Knowledge Similarity (DKS), which focus on retrieving passages that are knowledge-relevant. This optimization ensures that the system retrieves the most meaningful passages for a given query, improving both retrieval and generation performance. Their work significantly enhances query efficiency by focusing on dense knowledge matching rather than relying solely on token-level matching [12].

In addition, **Chang et al. (2024)** optimize the query process by introducing a hierarchical, community-driven retrieval system within their knowledge graph framework. This approach dynamically selects the most relevant communities and sentences based on their relationship to the query, improving both the speed and accuracy of fact-checking. The zero-shot capability of their system means that query optimization occurs without the need for retraining, making the solution efficient and scalable for various fact-checking and retrieval tasks [7].

Wu et al. (2024) also introduce several optimizations to the query process through their U-retrieve mechanism, which involves top-down matching of queries with graph-based structures. This approach minimizes the search space by dynamically indexing and matching queries with relevant graph nodes, enabling faster and more accurate retrieval. Additionally, the use of hierarchical graph structures allows the system to retrieve information at varying levels of detail, ensuring that even complex medical queries are handled efficiently and effectively [11].

Evaluating the performance of retrieval techniques in Graph RAGs involves assessing both the accuracy of the retrieved subgraphs and the efficiency of the retrieval process. Metrics such as precision, recall, and F1-score are commonly used to measure retrieval relevance. Additionally, specific metrics related to graph-based retrieval, such as subgraph quality and path relevance, are considered. Speed metrics, including query latency and retrieval time, are also critical for applications where real-time generation is required. Benchmarking on standard datasets helps ensure that retrieval methods can generalize to various domains while maintaining performance.

In conclusion, retrieval in Graph RAG systems is a multifaceted process that benefits from the integration of advanced techniques such as GNNs, optimization strategies, and efficient query processing methods. The collective contributions from various researchers highlight the ongoing advancements in this field, aiming to enhance the accuracy, efficiency, and scalability of retrieval processes. These improvements are essential for the development of sophisticated Graph RAG systems capable of handling complex, real-world applications, such as financial planning and advising services.

Generation

The integration of Retrieval-Augmented Generation (RAG) into Natural Language Processing (NLP) tasks significantly enhances language models by providing access to external knowledge during text generation. This approach improves the factual accuracy and relevance of outputs in tasks such as question answering, dialogue systems, and summarization. By retrieving pertinent information from graph knowledge bases, RAGs can fill knowledge gaps inherent in standalone language models, leading to more informed and contextually appropriate responses.

In Graph RAG systems, the generation component is responsible for producing natural language outputs that are informed by the retrieved graph data. This involves integrating the structured information from the knowledge graph into the language model's generation process. Techniques such as conditional generation, where the model conditions its output on the retrieved data, are commonly employed. The fusion of retrieval and generation enables models to produce outputs that are contextually enriched and factually consistent with the external knowledge base.

Semantic parsing plays a crucial role in this integration by converting natural language into structured, machine-understandable representations. Graph RAGs contribute to this process by retrieving and incorporating relevant graph structures that reflect the semantic meaning of the input. This integration allows for more precise interpretation of queries and the generation of responses that are aligned with the underlying knowledge base. By mapping user inputs to specific entities and relations within the graph, the system can generate outputs that are both semantically accurate and contextually relevant.

RAG-based representations enhance NLP models by embedding retrieved knowledge directly into the generation process. Techniques such as joint training and attention mechanisms over retrieved content are explored to maximize the synergy between retrieval and generation components. For instance, attention mechanisms enable the model to focus on the most relevant parts of the retrieved graph data during generation, improving the coherence and relevance of the output. This fusion ensures that the generated responses are not only fluent but also grounded in the external knowledge base.

Moreover, advanced models like Transformer-based architectures are employed in the generation step to handle complex language patterns and dependencies. Pre-trained language models such as GPT-3 and BERT can be fine-tuned to incorporate graph-structured knowledge, enhancing their ability to generate contextually rich and factually accurate text. Tools and libraries like Hugging Face Transformers provide frameworks for integrating retrieval and generation models, facilitating the development of sophisticated Graph RAG systems.

Peng et al. (2024) addressed limitations in traditional NLP models by leveraging graph structures to retrieve relational knowledge. They demonstrated that GraphRAG outperforms conventional RAG systems by incorporating both textual and structural information, enabling more context-aware responses [1]. Their work highlights the importance of integrating graph-based retrieval with advanced generation techniques to enhance the overall performance of language models.

In addition to these approaches, graph-to-text generation techniques are employed to translate structured graph data into natural language. This involves using models that can interpret the nodes and edges of a graph and generate coherent textual descriptions or explanations. Graph Neural Networks (GNNs) play a significant role in this process by encoding graph structures into embeddings that capture both semantic and relational information. These embeddings are then used by sequence-to-sequence models to generate the final textual output [1], to enhance modeling capabilities for graph data, thereby improving performance on downstream tasks such as node classification, edge prediction, graph classification, and others.

Evaluation of the generation component in Graph RAG systems is crucial to ensure the quality and reliability of the responses. Metrics such as BLEU, ROUGE, and METEOR scores are commonly used to assess the fluency and relevance of generated text. Additionally, specialized evaluation methods are employed to measure factual accuracy and consistency with the knowledge graph, ensuring that the generated content accurately reflects the underlying data [1].

The generation step also involves handling challenges such as controlling the level of detail and managing ambiguities in the retrieved data. Techniques like controlled generation and response refinement are explored to allow the system to adjust the specificity of the output based on the user's needs. This is particularly important in applications like financial planning or advising services, where precise and accurate information is paramount.

In summary, the generation component of Graph RAG systems is a critical aspect that determines the effectiveness of the overall system. By integrating advanced generation techniques with retrieved graph knowledge, Graph RAGs can produce responses that are not only contextually relevant but also factually accurate. Ongoing research continues to explore new methods and tools to further enhance the generation capabilities of these systems, expanding their applicability across various NLP tasks.

Non-Financial RAG Applications and Studies

Graph Retrieval-Augmented Generation (Graph RAG) systems have found significant applications beyond the financial domain, particularly in fields that require the integration of complex, structured knowledge. One prominent area is healthcare, where Graph RAGs assist in clinical decision support by retrieving relevant medical knowledge in response to patient data. They enable personalized treatment plans by considering complex relationships between symptoms, diseases, and treatments. This application requires handling sensitive data and ensuring compliance with privacy regulations.

Cui et al. (2024) demonstrate that the Bailicai framework enhances the performance of Large Language Models (LLMs) in medical tasks. By integrating medical knowledge injection and self-knowledge boundary identification, the framework effectively mitigates hallucinations, a common issue in medical text generation. Bailicai surpasses proprietary models like GPT-3.5 and achieves state-of-the-art performance across several medical benchmarks. The researchers focus on curating datasets from the UltraMedical dataset, ensuring that the model learns from high-quality medical data. The framework's ability to handle medical question-answering tasks with precision makes it a significant advancement in medical AI applications [5].

Building upon these advancements, **Wu et al. (2024)** propose a graph-based Retrieval-Augmented Generation framework specifically designed for medical applications, called MedGraphRAG. This framework constructs a hierarchical graph from medical documents, textbooks, and authoritative medical dictionaries, significantly enhancing the capability of LLMs to process and retrieve medical information. By employing a graph-based structure, the system preserves complex relationships between medical entities, providing superior retrieval accuracy compared to flat databases or simple

key-value retrieval systems. The hierarchical nature of the graph allows for more precise linking and retrieval, ensuring that the generated responses are evidence-based and contextually accurate [11].

The primary focus of Wu et al.'s work is on enhancing LLM performance in the medical domain. Their MedGraphRAG system addresses the specific needs of medical professionals by ensuring that generated responses are backed by evidence from credible medical sources, such as textbooks and peer-reviewed papers. The hierarchical graph structure links user-provided documents with established medical knowledge, ensuring transparency and interpretability in medical diagnoses and recommendations. MedGraphRAG's performance on multiple medical Q&A benchmarks, including PubMedQA, MedMCQA, and USMLE, demonstrates its effectiveness, surpassing even fine-tuned models in generating accurate, evidence-based medical information [11].

Beyond healthcare, Graph RAGs enhance **social network analysis** by generating insights based on the structure and dynamics of social interactions. They can identify influential nodes, detect communities, and predict relationship evolution. This information is valuable for marketing strategies, trend analysis, and understanding social phenomena. By analyzing complex social graphs, organizations can leverage Graph RAGs to make data-driven decisions in rapidly changing social environments.

In the field of **Geographic Information Systems (GIS)**, Graph RAGs facilitate the retrieval of spatial and relational data for applications like route planning, resource management, and environmental monitoring. By integrating various geospatial datasets, RAG systems can generate comprehensive analyses and support decision-making processes that consider multiple geographic factors. The graph-based approach allows for modeling intricate spatial relationships and dependencies, improving the accuracy and usefulness of generated insights in GIS applications.

The versatility of Graph RAGs is further demonstrated in domain-specific applications that require structured and reliable information retrieval. **He et al. (2024)** represent a significant advance in this area by creating a framework of questions and answers for a Graph RAG system. By paving the path for more structured and less hallucination-prone domain-specific chatbots, their work addresses common challenges in chatbot applications, such as maintaining context and ensuring factual accuracy of responses [8].

In conclusion, Graph RAG systems are making impactful contributions across various non-financial domains. Their ability to integrate complex, structured knowledge into language models enables more accurate, contextually rich, and reliable outputs. Whether in healthcare, social network analysis, GIS, or specialized chatbot development, Graph RAGs provide powerful tools for advancing AI applications and supporting sophisticated decision-making processes.

Finance RAG Applications and Studies

The financial sector has increasingly adopted Retrieval-Augmented Generation (RAG) systems to enhance various services, including data quality management, customer advisory, and financial planning. These systems leverage Large Language Models (LLMs) and external knowledge bases to improve accuracy, relevance, and personalization. While significant progress has been made, there remains a gap in applying **Graph RAG** systems specifically for financial planning or advisory services. This section reviews existing studies in the domain and highlights the need for further research integrating Graph RAG into financial advisory.

Enhancing Data Quality Management with RAG

Raja (2024) developed a Retrieval-Augmented Generation framework to improve data quality management in the financial sector, focusing on contextual data validation [13]. Financial data poses unique challenges due to stringent regulatory, security, and consistency requirements. Traditional

tools often lack the necessary context awareness to address these issues effectively. Raja's framework leverages LLMs in conjunction with knowledge graphs to enhance the accuracy of data quality processes.

Key components of the framework include:

1. **Web Scraping for Data Collection**
   - Utilized tools like **Scrapy** and **BeautifulSoup** to extract relevant financial content from publicly available blogs and articles.
   - Cleaned and structured the collected data into PDF documents for analysis.
2. **Embeddings and Model Architecture**
   - Generated sentence embeddings using Hugging Face's **all-mpnet-base-v2**, mapping text into a 768-dimensional space for better semantic understanding.
   - Employed a vector index store for faster query retrieval and improved contextualization.
3. **System Architecture and Configuration**
   - Integrated **Llama 2** with pre-trained models consisting of 7 billion parameters to enhance response quality.
   - Developed a query engine leveraging the indexed vector store to provide context-aware answers to financial data-related queries.

The framework was evaluated using three use cases relevant to financial data quality:

1. **Timeliness Check**
   - **Query**: "How to check if stock data is stale?"
   - **Result**: Provided detailed recommendations for evaluating stale stock data, going beyond traditional timestamp checks.
2. **Completeness Check**
   - **Query**: "What strategies should we use if customer income data is missing?"
   - **Result**: Suggested context-aware strategies such as regression imputation and explained the consequences of various imputation techniques.
3. **Consistency Check**
   - **Query**: "Are these symbols ($, €, &&, ☐) valid currencies?"
   - **Result**: Recommended ISO-based currency validation, outperforming regex-based methods used by SQL Server DQS.

The RAG-based approach demonstrated actionable insights not achievable with standard data quality tools, showing high answer relevancy and recall. However, challenges related to scalability due to memory constraints and limitations in domain-specific knowledge were identified. Future enhancements include training the model with real-world datasets from Jira logs and Notion entries to improve performance.

While Raja's study showcases the potential of RAG frameworks in improving data validation processes in finance, it primarily focuses on data quality management rather than financial advisory services.

Multi-Agent Systems and RAG in Financial Services

Sepanosian et al. (2024) present an industry-oriented framework to guide scalable AI adoption in financial institutions [14]. Their work includes a proof-of-concept implementation for retirement planning using multi-agent systems and RAG technologies. The framework categorizes AI solutions into problem areas such as information retrieval, data analysis, content generation, decision support,

and task automation. Central to the framework is RAG technology, ensuring the use of up-to-date, accurate information without needing constant model retraining.

The proof-of-concept utilizes **crewAI**, a multi-agent orchestration framework, deploying four specialized agents:

1. **Retirement Policy Expert**
   - Provides information on contribution limits and regulations.
2. **Industry Expert**
   - Delivers insights on investment trends and average savings.
3. **Advisory Expert**
   - Aggregates information and delivers personalized advice.
4. **Quality Assurance (QA) Agent**
   - Ensures accuracy by cross-referencing information and flagging inconsistencies.

These agents interact sequentially, using RAG tools and LangChain functionalities to dynamically retrieve relevant data. Testing with a fictional client demonstrated the agents' ability to generate personalized reports comparing retirement savings with industry averages. The agents identified that the client's investments were performing competitively, advised on potential additional contributions, and provided regulatory insights based on recent updates.

Despite valuable results, challenges emerged related to explainability and non-deterministic outputs inherent to LLMs. The QA agent often required manual review due to inconsistent or ambiguous outputs. Security risks were also noted, particularly when using external tools like OpenAI's API, introducing concerns about data security. Scalability challenges included preprocessing overhead for complex data types and redundant API calls reducing performance.

Although this study demonstrates the potential of LLMs and RAG tools in personalized financial advisory applications, it does not specifically employ **Graph RAG** systems. The focus remains on multi-agent systems and the use of RAG for information retrieval, without leveraging the structural advantages of graph-based knowledge representation.

Integrating LLMs with Financial Planning Models

De Zarzà et al. (2024) propose an innovative financial planning framework that combines traditional budgeting models with AI-powered LLMs to enhance financial management at both individual and cooperative (household) levels [15]. The core objective is to democratize financial planning by providing personalized recommendations that optimize savings and budgeting through accessible AI tools.

The study focuses on two interconnected financial planning models:

1. **Individual Budget Optimization Model**
   - Aims to maximize savings by distributing monthly income efficiently across expense categories (e.g., rent, groceries, utilities).
   - A utility function prioritizes saving while meeting necessary expenses, defined as:

$$U(I, E) = \alpha \log(S) - \beta \sum_{o=1}^{n} w_o \log(E_o)$$

     where I income, S savings, E are categorized expenses, and w are personalized preference weights.

- LLM-generated recommendations serve as a guiding baseline to suggest feasible allocations.

2. **Cooperative Budgeting Model for Households**
   - Extends the individual model to address shared financial responsibilities among household members.
   - The cooperative model ensures financial equity by considering individual priorities while optimizing total savings:

$$\mathrm{TS} = \sum_{o} = 1^{n} \left(I_o - \sum_{z} E_{\mathrm{zo}} \right)$$

     where TS is the total household savings, I_o is each member's income, and $E_{\{\mathrm{zo}\}}$ are their respective expenses per category z
   - LLM-informed recommendations offer advice on efficient shared budget planning and propose strategies to minimize redundant expenses.

The integration of LLMs offers several advantages:

- **Personalization**: Tailors recommendations based on individual and household-specific data.
- **Dynamic Adaptability**: Adjusts recommendations to reflect economic trends and evolving financial goals.
- **Enhanced Savings Strategies**: Includes advice on building emergency funds, retirement savings, and managing short-term vs. long-term priorities.
- **Consumption Smoothing**: Uses cooperative game theory principles to maintain consistent living standards over time.

The study evaluates scenarios using both individual and cooperative models, guided by LLM recommendations. Key insights include:

1. **Short-term Budgeting**
   - LLM suggests emergency fund allocations and prioritizes debt repayments.
2. **Retirement Planning**
   - In a household scenario, LLMs guide members to optimize savings, balancing retirement contributions with daily expenses.
3. **Long-term Financial Goals**
   - Recommends setting up a child fund to accommodate future expenses like education and healthcare.

To model the interaction of financial agents as an adaptive system, the extended coevolutionary (EC) theory is employed. Agents iteratively adjust their spending based on LLM recommendations and their financial environment. For an agent o at time t:

$$E_o(t+1) = (1-\beta)\Big(E_o(t) + \alpha \nabla U_o\Big(E_{o(t)}, E_{-o}(t)\Big)\Big) + \beta R_{o,t}$$

where $R_{o,t}$ is the LLM recommendation, and α and β control learning and LLM influence.

While the integration of LLMs yields optimized budget plans that align with best practices in personal finance, human oversight is essential to validate LLM suggestions and avoid potential errors. Similar to the previous studies, this research does not incorporate **Graph RAG** systems. The recommendations are generated by LLMs without the explicit use of graph-structured knowledge bases, potentially limiting the system's ability to capture complex financial relationships.

The Need for Graph RAG in Financial Planning and Advisory

The reviewed studies illustrate the growing interest in applying RAG systems and LLMs within the financial sector, including data quality management, retirement planning, and budgeting. However, none of the studies specifically explore the use of **Graph RAG** systems for financial planning or advisory services.

Graph RAG systems leverage the structural properties of graphs to represent complex relationships and dependencies inherent in financial data. By retrieving and incorporating relational data embedded within nodes, edges, and subgraphs, Graph RAGs can provide more nuanced and contextually rich recommendations. This is particularly valuable in financial planning and advisory services, where understanding intricate relationships between financial products, market trends, regulatory requirements, and individual client profiles is crucial.

Integrating Graph RAG systems into financial advisory could enhance the personalization and accuracy of recommendations, enable multi-hop reasoning over financial knowledge graphs, and improve the system's ability to handle complex queries. Additionally, using graph-based knowledge representations can facilitate better compliance with regulatory standards by providing transparent and interpretable insights.

Conclusion

While significant advancements have been made in applying RAG systems and LLMs within the financial sector, there remains a gap in utilizing **Graph RAG** systems specifically for financial planning and advisory services. The potential benefits of Graph RAG, including improved handling of complex financial relationships and enhanced personalization, highlight the need for further research in this area. This thesis aims to address this gap by investigating the integration of Graph RAG systems into financial planning and advisory, exploring their potential to revolutionize the delivery of financial services.

Bibliography

[1] BOCI PENG *et al.*, "Graph Retrieval-Augmented Generation: A Survey," vol. 37, no. 4.

[2] Lisa Ehrlinger and Wolfram Wöß, "Towards a Definition of Knowledge Graphs."

[3] Paulheim H., "Knowledge Graph Refinement: A Survey of Approaches and Evaluation Methods."

[4] Yuntong Hu, Zhihan Lei, Zheng Zhang, Bo Pan, Chen Ling, and Liang Zhao, "GRAG: Graph Retrieval-Augmented Generation."

[5] Cui Long, Yongbin Liu, Chunping Ouyang, and Ying Yu, "Bailicai: A Domain-Optimized Retrieval-Augmented Generation Framework for Medical Applications."

[6] Chao Jin *et al.*, "RAGCache: Efficient Knowledge Caching for Retrieval-Augmented Generation."

[7] Rong-Ching Chang and Jiawei Zhang, "CommunityKG-RAG: Leveraging Community Structures in Knowledge Graphs for Advanced Retrieval-Augmented Generation in Fact-Checking."

[8] Xiaoxin He *et al.*, "G-Retriever: Retrieval-Augmented Generation for Textual Graph Understanding and Question Answering."

[9] Zetian Hu, Weijian Xie, Xuefeng Liang, Yuhui Liu, Kaihua Ni, and Hong Cheng, “WeKnow-RAG: An Adaptive Approach for Retrieval-Augmented Generation Integrating Web Search and Knowledge Graphs.”

[10] Jinyuan Fang, Zaiqiao Meng, and Craig Macdonald, “TRACE the Evidence: Constructing Knowledge-Grounded Reasoning Chains for Retrieval-Augmented Generation.”

[11] Junde Wu, Jiayuan Zhu, and Yunli Qi, “Medical Graph RAG: Towards Safe Medical Large Language Model via Graph Retrieval-Augmented Generation.”

[12] Wenyu Huang, Mirella Lapata, Pavlos Vougiouklis, Nikos Papasarantopoulos, and Jeff Z. Pan, “Retrieval Augmented Generation with Rich Answer Encoding.”

[13] Komal Azram Raja, “Domain Specific Data Quality Framework.”

[14] Thomas Sepanosian, Zoran Milosevic, and Andrew Blair, “Scaling AI adoption in finance: modellingframework and implementation study.”

[15] I. de Zarzà, J. de Curtò, Gemma Roig, and Carlos T. Calafate, “Optimized Financial Planning: Integrating Individual and Cooperative Budgeting Models with LLM Recommendations.”

[16] Jinyuan Fang, Zaiqiao Meng, and Craig Macdonald, “REANO: Optimising Retrieval-Augmented Reader Models through Knowledge Graph Generation,” vol. 1.