FinSage: A Multi-aspect RAG System for Financial Filings Question Answering

Xinyu Wang[1,2], Jijun Chi[1,3], Zhenghan Tai[1,3], Tung Sum Thomas Kwok[1,4], Muzhi Li[5], Zhuhong Li[1,6], Hailin He[1], Yuchen Hua[1,2], Peng Lu[7], Suyuchen Wang[7,8], Yihong Wu[7], Jerry Huang[7,8], Jingrui Tian[9], Fengran Mo[7], Yufei Cui[2,10],Ling Zhou[11]
[1]SimpleWay.AI [2]McGill University [3]University of Toronto [4]Boston University
[5]The Chinese University of Hong Kong [6]Duke University [7]Université de Montréal [8]Mila - Quebec AI Institute
[9]University of California, Los Angeles [10]Noah's Ark Lab [11]CG Matrix Technology Limited
xinyu.wang5@mail.mcgill.ca,ferris.chi@mail.utoronto.ca

Abstract

Leveraging large language models in real-world settings often entails a need to utilize domain-specific data and tools in order to follow the complex regulations that need to be followed for acceptable use. Within financial sectors, modern enterprises increasingly rely on Retrieval-Augmented Generation (RAG) systems to address complex compliance requirements in financial document workflows. However, existing solutions struggle to account for the inherent heterogeneity of data (e.g., text, tables, diagrams) and evolving nature of regulatory standards used in financial filings, leading to compromised accuracy in critical information extraction. We propose the FinSage framework as a solution, utilizing a multi-aspect RAG framework tailored for regulatory compliance analysis in multi-modal financial documents. FinSage introduces three innovative components: (1) a multi-modal pre-processing pipeline that unifies diverse data formats and generates chunk-level metadata summaries, (2) a multi-path sparse-dense retrieval system augmented with query expansion (HyDE) and metadata-aware semantic search, and (3) a domain-specialized re-ranking module fine-tuned via Direct Preference Optimization (DPO) to prioritize compliance-critical content. Extensive experiments demonstrate that FinSage achieves an impressive recall of 92.51% on 75 expert-curated questions derived from surpasses the best baseline method on the FinanceBench question answering datasets by 24.06% in accuracy. Moreover, FinSage has been successfully deployed as financial question-answering agent in online meetings, where it has already served more than 1,200 people.

Keywords

LLM, Information Retrieval, Document Pre-Processing, Large Language Models, Retrieval Augmented Generation, Generative Models, Synthetic Datasets, Generative Retrieval

1 Introduction

In the real world, various regulatory constraints define the manners in which tools and systems can be permissibly integrated into modern workflows, complicating the manners in which technologies such as large language models (LLMs) can be directly play a role in fulfilling specific tasks. One such domain is the financial sector, where adherence to regulations is essential to mitigate legal risk and maintain stakeholder confidence. However, financial

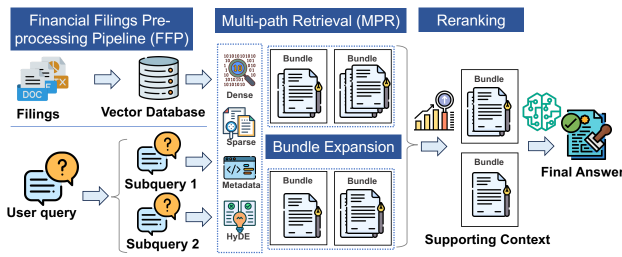

Figure 1: An overview of our proposed FinSage.

regulations can evolve with time, presenting complex and significant challenges even for well-structured institutions with expert teams dedicated for this specific purpose [34, 35]. More practically, these challenges have led to a strong demand for a Question Answering (QA) system capable of efficiently retrieving and analyzing compliance-related information from financial filings [37], with the hope of reducing manual labor and minimizing human error. Such systems often require the integration of retrieval-augmented generation (RAG), which enables machine learning-powered systems to directly search through databases to extract relevant information that can empower greater decision making.

However, while RAG systems have proven helpful for general LLM reliability and reasoning, integration into financial compliance tasks can be difficult due to inherent domain-related limitations. First, financial filings combine unstructured text (narrative disclosures), semi-structured data (tables, graphs), and contextual metadata - a multi-modal composition that conventional text-centric retrieval architectures fail to process cohesively, leading to fragmented or incomplete context representation [31, 41, 42, 46]. Next, while existing RAG systems predominantly employ dense retrieval [19, 43] or sparse lexical matching [32], these can fail due to the need for domain-specific fine-tuning or an inability to capture implicit regulatory relationships [17, 38, 45]. Finally, existing retrieval methods often rely on ranking relevant passages based on semantic similarity rather than domain-specific criteria, leading to the critical risk of faulty or incomplete reasoning [1, 5, 6]. Such limitations collectively undermine the precision and accountability of automated compliance workflows, exposing enterprises to operational and regulatory hazards.

As a solution to these challenges, we introduce FinSage, an end-to-end financial question-answering framework optimized for regulatory compliance that leverages the heterogeneous nature of financial data for better reasoning. As shown in Figure 1, FinSage consists of three core components: (1) Financial Filings Pre-processing (FFP) pipeline, (2) Multi-path Retrieval (MPR) pipeline, and (3) Domain-Specialized Document Re-ranker, which collectively overcome the aforementioned challenges of multi-modal documents domain-agnostic retrieval, and ranking ambiguity. First, the *FFP module* tackles the multi-modal heterogeneity in financial documents by unifying text, tables, and graphs into a structured vector database through modality-specific encoders. Next, the *MPR module* augments traditional dense/sparse retrieval with two tailored domain-aware strategies, ensuring robust coverage of both explicit and implicit domain-specific relationships. Finally, the *Document Re-ranker* addresses precision limitations through direct preference optimization (DPO) [49], prioritizing chunks containing legally significant phrases (e.g., “material weakness” or “non-compliance”) and suppressing irrelevant semantic matches. By jointly optimizing retrieval diversity and precision, FinSage establishes a new paradigm for compliance-critical financial QA systems. The experimental results demonstrate that FinSage significantly outperforms traditional RAG approaches in financial regulatory compliance tasks, offering a practical and scalable solution for financial institutions. In summary, our contributions are as follows:

- We present an end-to-end financial QA system that integrates pre-processing, retrieval, and re-ranking to provide a robust solution for regulatory compliance.
- We address key challenges in multi-modal financial document processing and domain-specific question answering, bridging the gap between traditional RAG models and regulatory compliance applications.
- We conduct extensive experiments and ablation studies to evaluate the effectiveness of our retrieval, re-ranking, and response generation mechanisms, analyzing the impact of different retrieval paths and re-ranking strategies.

2 Related Work

2.1 Retrieval Augmented Generation

Despite their strong empirical performance, large language models (LLMs) are inherently bottlenecked by their fixed parametric size, limiting their ability to store factual knowledge, which can affect their performance on specific tasks. Retrieval-Augmented Generation (RAG) [9, 27] emerged as a solution by integrating external knowledge that the LLM could access through retrieval within an external database. A retrieval module first selects relevant documents from sources like news, academic papers, or social media, which are then combined with the input query and fed into a language model to generate a response. This approach leverages both the model’s internal memory and the retrieved corpus [50], ensuring more accurate and contextually grounded outputs.

Recent studies have explored various improvements, such as feedback tokens for relevance [3], self-referential knowledge acquisition [40], iterative self-feedback [25], and domain-specific hierarchies [39]. In cases where initial queries lack sufficient detail, generative retrieval enhances information retrieval by generating queries or synthesizing passages beyond traditional keyword-based search [2, 22, 30]. Some studies further work to use multi-head RAG to extract information from diverse sources [6, 41], aiming to maximize answer accuracy through quantity-based retrieval. Others focus on enhancing answer precision via quality retrieval, leveraging re-rankers [13, 16, 21] to preserve high answer relevancy.

2.2 Multi-Modal Retrieval

Information is not limited to text, meaning that knowledge from various modalities can be used for improved reasoning. Some approaches, such as MuRAG [11] and generative retrieval frameworks [26], enhance retrieval by leveraging multi-modal memory, generative retrieval, and knowledge-guided decoding, improving performance in multi-modal QA tasks. Others, including VisRAG [44] and multi-modal RAG for industrial applications [31], explore vision-language models to retain document structure and improve retrieval accuracy, particularly in industrial and document-heavy settings.

2.3 Retrieval for Financial Data

Despite recent advances, RAG-LLM systems remain underdeveloped for financial filings. Islam et al. [15] introduced FinanceBench, a dataset specifically designed for financial document understanding. Building on this, Unstructured [17] explores different chunking strategies to enhance financial document processing. Setty et al. [36] proposes a RAG pipeline to facilitate retrieval in financial reports. However, existing approaches still face significant challenges: they struggle to handle complex multi-modal information—such as figures and tables—and their simple cross-encoder-based re-rankers often lack precision, negatively impacting retrieval performance.

3 Methodology

To reliably use RAG for financial document processing, the system must address three fundamental challenges: (1) handling multi-modal data such as tables, figures, and structured text; (2) extracting a comprehensive scope of relevant information from multiple sources; and (3) generating accurate and precise responses by effectively summarizing and reasoning over retrieved content. To tackle these challenges, we propose FinSage, an end-to-end financial QA system tailored for regulatory compliance.

This section introduces the three key components of FinSage, each designed to address specific challenges in financial RAG (Figure 2). Section 3.1 presents the **Financial Filings Pre-processing Pipeline (FFP)**, which transforms financial documents into a structured machine-readable format while enriching their semantic content. Section 3.2 details the **Multi-path Retrieval Pipeline (MPR)**, which enhances retrieval accuracy and contextual relevance by integrating multiple retrieval strategies. Section 3.3 discusses the **Document Re-ranker**, which improves retrieval precision by refining and prioritizing the most relevant document chunks.

3.1 Financial Filings Pre-processing pipeline

We create a Financial Filings Pre-processing (FFP) pipeline to convert multi-modal financial filings into machine-readable text chunks. Figure 3 illustrates how FFP operates in two steps: textual encoding and semantic enhancement.

Formally, let C denote the original set of multi-modal chunks extracted from a financial document, divided based on its chapters and

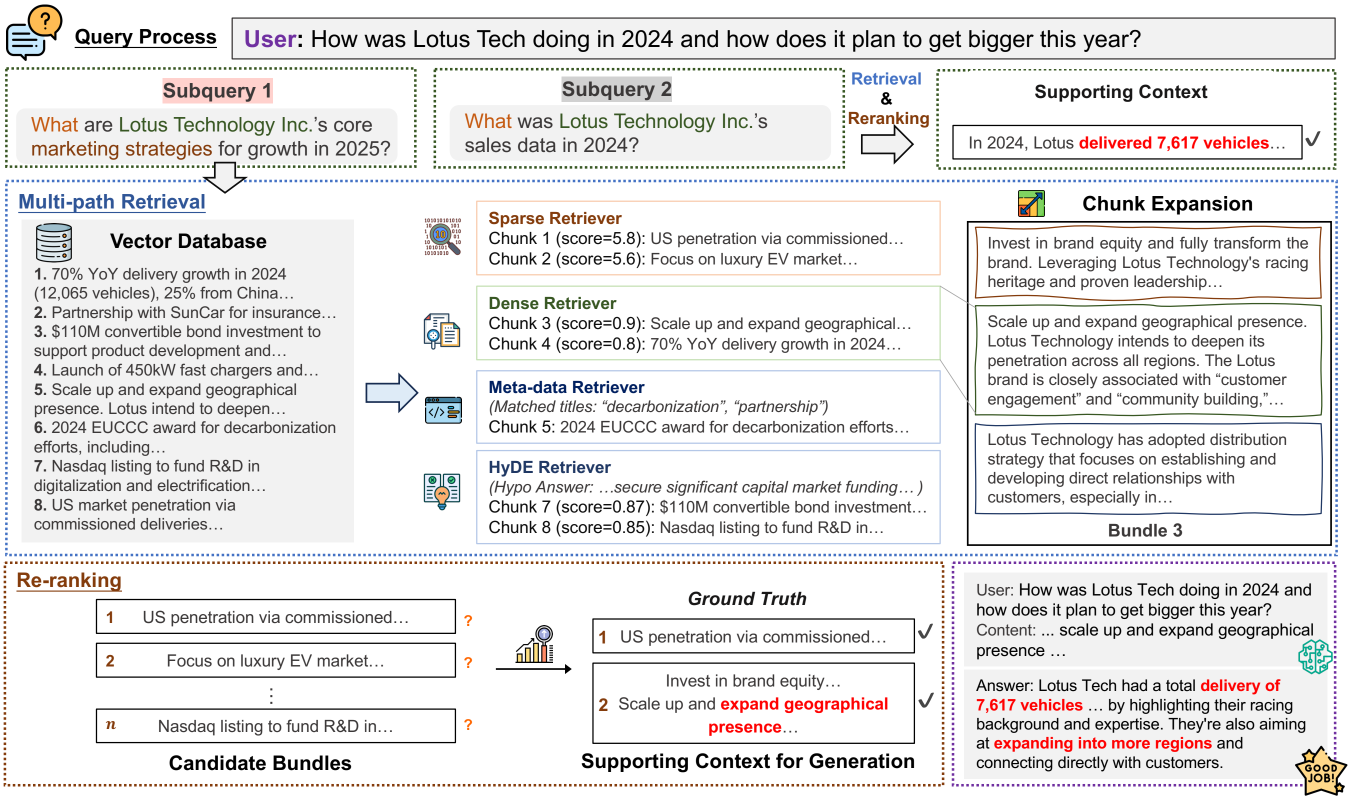

Figure 2: The complete FinSage RAG pipeline. The system processes the user's query and splits it into subqueries. For each subquery, multiple chunks are retrieved with multi-path retrieval, and these chunks are then expanded into candidate bundles by concatenating neighboring chunks. After re-ranking, the most relevant chunks and their bundles are included for answer generation.

a pre-defined token length. Each chunk $s_i = (t_i, m_i) \in C$ consists of raw content t_i and metadata m_i. The goal of the FFP is to transform chunks in C into $\tilde{C}$, a self-contained and condensed format that is better suited for downstream information retrieval tasks.

Textual Encoding of Multi-Modal Documents. The FFP pipeline first converts multi-modal financial documents into structured, machine-readable textual formats. Following the natural reading order, we use open-source PDF parsing tools, such as MinerU [12], to decompose each document into a sequence of chunks. Each chunk is labeled with its corresponding type (i.e., text, figure, or table). Inspired by Jimeno-Yepes et al. [17], we further transform figure and table chunks into descriptive textual narratives with large vision-language models. Specifically, figures are converted into structured captions summarizing their key insights, while tables are rewritten as textual statements capturing their primary data trends and relationships [47]. This transformation enables large language models (LLMs) to effectively interpret and utilize multi-modal content.

Semantic Enhancements. To improve the quality and coherence of the extracted texts, the FFP pipeline applies the following three key semantic enhancement steps.

(1) **Redundant chunk de-duplication.** We first de-duplicate documents to reduce redundancy, ensuring that duplicate content does not limit the diversity of retrieved information. For any two chunks $s_i, s_j \in C$ such that $i \neq j$, we randomly discard s_i or s_j if the cosine similarity of the dense embeddings of their raw texts $\cos\left(\mathbf{d}_i, \mathbf{d}_j\right)$ exceeds a predefined threshold τ_{dedup}.
(2) **Co-reference resolution.** Next, we disambiguate the textual context of each chunk by iteratively resolving co-references. Within each subtitle section, pronouns of entity mentions are replaced with their corresponding antecedents identified from preceding chunks. Specifically, for each chunk containing pronouns, we employ an LLM as a co-reference resolver to perform entity mention replacements. The LLM receives k previous chunks to identify proper antecedents, where k is fixed.
(3) **Metadata generation.** Due to the token-length limitation, document sections may be separated into multiple chunks where content may no longer be self-contained. Thus, we ensure that chunks within a section retain sufficient context from others in the same section for retrieval. We employ an LLM to summarize each section of the financial filing and append the generated summary to the metadata of every chunk obtained from that section. This ensures that each chunk retains enough contextual information while maintaining a minimal context length, enhancing the efficiency and accuracy of document retrieval.

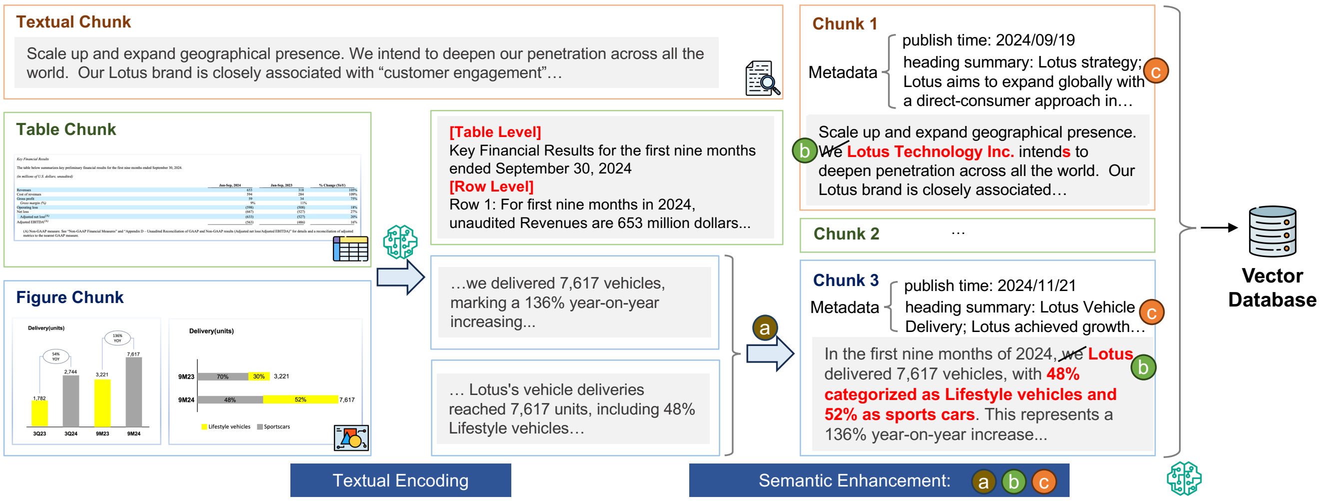

Figure 3: The FinSage FFP pipeline. Textual Encoding: PDF parsing tools extract multi-modal chunks (text, figures, tables), which are then converted into textual representations using large language models (LLMs). Semantic Enhancement: The textual representations are further refined by (a) eliminating redundant chunks through pairwise similarity comparison, (b) resolving co-reference within subtitle sections, and (c) generating subtitle-section-based summaries as metadata. These enriched chunks are then embedded in a vector database for further processing.

After the aforementioned process, for each chunk s_i, dense and sparse embeddings $\mathbf{d}_i$ and $\mathbf{p}_i$ are generated from $\tilde{t}_i$ using a text encoder and BM25, respectively. Furthermore, a dense embedding for the metadata $\mathbf{e}_i$ is also generated by a text encoder. The chunk is then represented as a 5-tuple $(\tilde{t}_i, \tilde{m}_i, \mathbf{d}_i, \mathbf{p}_i, \mathbf{e}_i)$. These enhancements ensure that extracted chunks are self-contained, contextually coherent, and semantically enriched, optimizing their utility for downstream processing. Further details regarding the selection of the multi-modal textualization model and the dense embedding model are found in Appendices A.3 and A.4, respectively.

3.2 Multi-path Retrieval pipeline

We introduce a Multi-path Retrieval (MPR) pipeline that enhances performance by leveraging both metadata and semantic information during the retrieval process. The pipeline consists of two primary modules: a Query Paraphrasing Module and a Retrieval Module for obtaining relevant answers from diverse sources.

Query Paraphrasing Module. An LLM agent is employed to reformulate each input query using FinSage's specialized vector store, which serves as a comprehensive knowledge base containing all company-related information. This paraphrasing process involves three key steps:

(1) **Query Decomposition:** Each query is segmented into multiple self-contained sub-queries. This ensures that each subquery can be independently addressed by mapping its keywords to the corresponding segments in the knowledge base.
(2) **Co-reference Resolution:** Co-references within each query are resolved by converting the text into standardized English. In this process, all pronouns are replaced with their corresponding antecedents.
(3) **Context Integration:** For queries that reference prior conversation history, relevant contextual information is incorporated to maintain completeness and clarity.

Retrieval Module. Traditional retrieval methods typically rely on either sparse or dense retrieval techniques, each presenting challenges when applied to domain-specific tasks. Dense retrievers generally require extensive fine-tuning with substantial resources, whereas sparse retrievers often depend on heuristic rules that can lead to inaccuracies. To address these limitations, we incorporate both metadata-based and hypothesis-based retrieval strategies, resulting in a multi-path retrieval system. Consequently, our MPR pipeline comprises four components:

(1) **BM25 Sparse Retriever:** This component employs the BM25 scoring function to compute relevance based on tf-idf between the query and documents [32].
(2) **BGE-M3 Dense Retriever:** Utilizing the BGE-M3 model [10], this retriever calculates the cosine similarity between query embeddings and chunk embeddings. The retrieval process is accelerated by FAISS [18], ensuring high-performance search.
(3) **Metadata Retriever:** Leveraging the same BGE-M3 model, this module computes the cosine similarity between the query and metadata embeddings, facilitating the integration of structured metadata into the retrieval process.
(4) **HyDE Retriever:** This component computes the cosine similarity between hypothesis and chunk embeddings.

We collect all results selected by those retrievers into a chunk set, which serves as the output of the retrieval module.
Our metadata retrieval approach is designed to address high-level search tasks, such as question answering that requires reasoning across multiple text segments. Traditional retrieval methods often struggle to capture all relevant chunks; therefore, we incorporate

metadata summaries (generated via FFP) under a consistent subtitle to enhance retrieval coherence. Specifically, our approach concatenates each chunk's heading and summary to form a unified metadata embedding. In our framework, every chunk within a semantic segment is assigned an identical heading-summary representation, ensuring that the retrieval of a single metadata instance inherently retrieves all associated chunks. This design not only reinforces the contextual linkage among chunks from the same document segment but also effectively mitigates retrieval ambiguity when dealing with multiple documents.

The HyDE retriever is designed to further address recall issues. In certain scenarios, target chunks and user queries exhibit only weak correlations, even when the queries are optimally formulated. In such cases, the HyDE retriever leverages Hypothetical Document Embeddings[48], a specialized query expansion technique which uses an LLM-generated response to the user query as a hypothesis for retrieving similar chunks. To further enhance HyDE's performance, we augment the LLM's domain-specific knowledge through instruction fine-tuning on our custom Company dataset, following the InPars method [7]. Additionally, a smaller model is subsequently trained on a synthesized dataset to emulate the generative capabilities of larger models. As a result, HyDE not only improves the matching accuracy between the generated hypothetical responses and the target chunks but also bolsters overall retrieval performance across diverse content types.

Chunk Bundling Module. After retrieval, the MPR pipeline employs a chunk expansion mechanism to enhance contextual coherence among sequential chunks. This approach is particularly critical in financial documents, where key topics and discussions often span multiple chunks or paragraphs. Initially, each chunk is treated as an independent unit. These units are then iteratively expanded by evaluating adjacent chunks (preceding, following, or both) based on a predefined semantic similarity threshold. Formally, for each initial chunk s_i, we combine it with adjacent chunks s_{i-1} or s_{i+1} if the cosine similarity of their dense embeddings, namely $\cos(\mathbf{d}_i, \mathbf{d}_{i-1})$ or $\cos(\mathbf{d}_i, \mathbf{d}_{i-1})$) exceeds an empirically determined threshold τ_{exp}.

3.3 Document Re-ranking

The Document Re-ranker (DRR) in FinSage acts as an intelligent filter, addressing shortcomings of initial retrieval to enhance accuracy, coherence, and reliability in final generated responses. We employ the `BAAI/bge-reranker-v2-Gemma` model based on its superior performance in both English and multilingual applications. Within FinSage, the re-ranker refines retrieved documents before passing them to the LLM, providing more relevant contextual input for answer generation. Specifically, the Document Re-ranker scores candidate chunks $s \in \mathcal{R}_q$ as $\mathcal{K}(s, q)$ using a cross-encoder:

$$\mathcal{K}(s_i, q) = \sigma\left(\mathbf{w}^\top \text{CrossEncoder}\left(\text{concat}\left(q, \tilde{t}_i\right)\right) + \beta f\left(\tilde{m}_i\right) + b\right), \tag{1}$$

where $\mathbf{w}^\top \text{CrossEncoder}(\cdot) + b$ is a Transformer-based re-ranker scoring contextual alignment [4, 14], and $f(\tilde{m}_i)$ processes metadata to output a scalar bonus, scaled by β. The final score is transformed using a sigmoid function σ.

Time bonus. When ranking chunks for relevance, a common challenge arises when two similar chunks lead to favoring older content instead of the new, potentially more desirable entries. Recognizing that metadata, particularly publication dates, is crucial for evaluating chunk relevance, we incorporate a time bonus for chunk s_i, $f(\tilde{m}_i)$, into the overall ranking score in DRR. Specifically, if the publication date of the chunk is within a year of the query time, there will be an additional bonus. The closer the publication date is to the query time, the larger this bonus will be. This mitigates the issue where the DRR favors older contents over new one when the two chunks have identical embeddings.

Direct Preference Optimization. To further enhance ranking performance, we fine-tune the DRR using Direct Preference Optimization (DPO). This method frames preference learning as a binary classification task, where training data consists of preference pairs: a preferred (positive) response s^+ and a rejected (negative) response s^-. The model learns to prioritize high-quality responses by minimizing the cross-entropy loss:

$$\mathcal{L}(\text{DPO}) = -\mathbb{E}(q, s^+, s^-)\left[\log \frac{\mathcal{K}(s^+, q)}{\mathcal{K}(s^+, q) + \mathcal{K}(s^-, q)}\right], \tag{2}$$

ensuring that our re-ranker effectively distinguishes and prioritizes the most relevant document chunks.

The DRR pipeline is iteratively implemented as follows:

(1) **Retrieval and Annotation**: Extensive results are retrieved using FAISS in the Multi-Path Retrieval pipeline (Section 3.2) for a rich set of candidate documents in each query. Retrieved documents are annotated to create preference samples for DPO, distinguishing between relevant and irrelevant matches. The annotated pairs are then organized into a structured dataset with the balance between positive and negative samples maintained to prevent training bias.
(2) **Evaluation**: Multiple evaluation metrics, including Normalized Discounted Cumulative Gain (NDCG) and Mean Reciprocal Rank (MRR) [33], are employed to verify the model's ability to determine document relevancy and hence improve DRR's performance.
(3) **Model Adjustment**: When the model fails to perform on a new test set, the retrieved documents are re-annotated to better represent the underlying patterns and relationships via a careful review and adjustment of training data. The model is retrained on top of the previous version using the updated annotations, refining the ranking ability. This is repeated until a satisfactory performance on the new test set is attained.

After document re-ranking, the RAG pipeline in FinSage follows the traditional method to handle user queries, retrieve related data from the vector database, and generate answers accordingly (Figure 2).

4 Experiments

We conduct experiments across three settings to evaluate FinSage: (1) assessing the Multi-path Retrieval pipeline, (2) evaluating the Document Re-ranker, and (3) measuring overall answer quality.

4.1 Datasets and Baseline

To evaluate performance, a manually annotated dataset is typically regarded as the "golden dataset," with high-quality entries serving as the ground truth [20]. We utilize two such datasets:

FinanceBench. FinanceBench [15] is a financial question-answering dataset created to evaluate the performance of LLMs on open-book tasks. It includes 150 open-source questions relating to traded companies and provides raw financial documents in PDF format containing straightforward answers. We use 149 questions from FinanceBench and construct a single vector store by processing 83 documents that contain correct answers for these questions with FFP. A fixed length of 256 tokens is adopted when separating semantic segments into chunks. We compare our proposed RAG system with three baseline methods, namely Islam et al. [15], Jimeno-Yepes et al. [17] and Setty et al. [36] with their best-matching variants *Shared Vector Store*, *Base256*, and *Reranker*, respectively. These variants generally follow a setup similar to ours, utilizing a single vector store for all filings in FinanceBench, a chunk size of 256 tokens, as well as the retriever before generating answers. The baseline *reranker* is combined with a cross-encoder reranker which best aligns with our configuration.

In-depth Single Company Dataset. Apart from FinanceBench, we construct a custom QA dataset using 12 filings from a single company. These filings are segmented into a vector database, with each chunk fixed at 200 tokens. We manually compile 75 question-answer pairs from SEC 6-K and F-1 filings, associating each question with an average of 8.7 relevant text chunks containing the most pertinent passages. The questions cover various aspects of Lotus Technology Inc., including products, sales data, history, and corporate strategy. To ensure accuracy, we engage domain experts to review the manually annotated chunks, establishing them as the ground truth. The complexity of these questions often necessitates aggregating information from multiple documents or sections, making it challenging to locate answers within a single chunk. We refer to this dataset henceforth as the Company dataset.

4.2 Experiment Settings

4.2.1 MPR settings. We present the experiment settings for the Multi-Path Retrieval (MPR) pipeline, following the order of its retrieval workflow. After the query process, FinSage selects the top-K results from the vector database to match the query.

In our HyDE retriever, we train a `Qwen-2.5-7B-Instruct` to follow the HyDE instructions and acquire finance knowledge from the 72B model. To synthesize the training dataset, we first generate question-document pairs using `Qwen-2.5-72B-Instruct` and select the top 2,000 pairs based on similarity scores. Two hypothetical chunks are generated for each pair, encapsulating essential domain-specific nuances. After training, the model produces three hypothetical answers for each question during testing, which are then used to retrieve the corresponding set of top-K chunks.

For the metadata retriever, the process selects the top-K metadata entries and includes all chunks from the corresponding semantic segments as candidates. Accordingly, to maintain a consistent number of retrieved chunks across different methods, we adopt varying top-K values.

Finally, unique chunks are collected across different retrieval methods, after which we select $\tau_{\text{exp}} = 0.85$ for the chunk expansion threshold with a limit of $\mathcal{K} = 5$, to include the neighbor chunk with sufficient similarity.

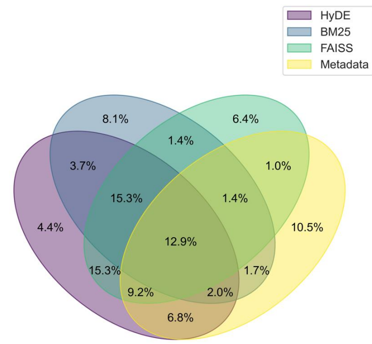

Figure 4: The percentage of relevant chunks selected by retrieval methods relative to the total number of retrieved relevant chunks.

4.2.2 DRR settings. We assess the impact of Document Re-ranker on retrieval effectiveness using different configurations (Top K + Different Re-ranker), as detailed in Figure 5. Here, Top K denotes the number of top-reranked chunks by the re-ranker. For re-ranking, we compare our **fine-tuned DRR**, trained on single-company financial filings, against `BAAI/bge-reranker-v2-gemma`, a general re-ranker. Our model is fine-tuned using Direct Preference Optimization (DPO) with a contrastive learning objective. The training dataset comprises annotated query-document pairs, each containing one relevant chunk and multiple irrelevant ones. This enables the model to adapt to SEC filings' terminology and structure, ranking relevant chunks higher for improved retrieval precision. The experiment aims to evaluate whether a domain-specific re-ranker enhances retrieval effectiveness.

4.2.3 Response settings. We evaluate the end-to-end question answering (QA) performance of FinSage by comparing it against question answering approach proposed in [15, 17, 36] on the FinanceBench dataset. Additionally, we conduct QA accuracy evaluation on our in-depth company dataset. For FinanceBench, we construct a single vector store (one store for all filings, chunk size = 256) by processing financial PDF documents using the Financial Filings Preprocessing Pipeline (FFP) and apply our multi-path retriever combined with a fine-tuned document re-ranker to retrieve relevant chunks before generating answers. This setup on FinanceBench resembles the setup of *Reranker* [36], *Base256* [17], and *Shared Vector Store* [15]. In addition, we conduct a separate end-to-end evaluation on the Company dataset, where we measure QA accuracy by comparing generated answers against ground truth answers. For response generation, we set $K = 5$ for each sub-query to efficiently manage LLM token consumption. This threshold is justified by empirical evidence indicating that retrieving five chunks is generally sufficient to generate effective answers [24].

4.3 Evaluation

MPR Evaluation. Our MPR pipeline is evaluated using various combinations of retriever paths by analyzing document chunk recommendation metrics (Recall, Precision, F1) on the in-depth dataset. By introducing additional retriever paths, we expect improvements in recall rates due to a more diverse information coverage [52].

Table 1: Retrieval Performance on the company dataset. "Avg Num Retrieved" represents the average number of chunks in the retrieved set. Recall, Precision, and F1 scores are calculated based on the relevant chunks retrieved. The bold denotes the best Recall, Precision and F1 produced from FinSage.

Category	Method	Avg Recall	Avg Precision	Avg F1	Avg Num Retrieved
FAISS Retrieval	FAISS (Baseline)	0.8194	0.0941	0.1646	75.6
	+ Bundle Expansion (Exp)	0.8103	0.0999	0.1733	70.25
BM25 Retrieval	+ BM25	0.8452	0.1147	0.1949	66.24
Metadata Retrieval	+ Metadata	0.8573	0.1198	0.2037	64.87
HyDE Retrieval	HyDE-1: HyDE(Qwen7B)	0.8228	0.1076	0.1830	69.09
	HyDE-2: HyDE(Qwen7B-SFT)	0.8567	0.1072	0.1831	72.92
	HyDE-3: HyDE(Qwen72B)	0.8323	0.1078	0.1844	69.77
FinSage	+ BM25+Metadata+HyDE-2 + Exp	**0.9251**	**0.1272**	**0.2156**	68.75

DRR Evaluation. We assess the effectiveness of the Document Re-ranker by comparing the retrieved chunks with the manually annotated ground truth dataset. The re-ranker prioritizes and selects up to K chunks per sub-query, where K is an adjustable parameter. We then measure the re-ranking performance with Precision, Normalized Recall, Mean Reciprocal Rank (MRR), and Binary Normalized Discounted Cumulative Gain (nDCG). To ensure a fair assessment, we introduce an adjusted **Normalized Recall** metric:

$$\text{Normalized Recall} = \frac{\text{correctly retrieved chunks}}{\min(\text{ground truth chunks}, K)}.$$

We cap denominator to $\min(\text{ground truth chunks}, K)$, making it an upper bound for recall calculation. Traditional recall would underestimate and unfairly penalize the model if we were to divide by the actual number of ground truth chunks, despite the reranker functioning optimally within its retrieval limit.

Response Evaluation. We compare between the quality of system-generated responses and expert-provided answers through automatic evaluation using `GPT-4o`. Specifically, we use the *CorrectnessEvaluator* module from LlamaIndex, which utilizes `GPT-4o` to systematically score generated responses based on alignment of FinSage's outputs with ground truth reference answers. This follows prior work [28, 29, 51] which utilize language models to judge the quality of system-generated responses. To complement automatic evaluation, our team manually labeled the responses after careful discussion. Human evaluation focuses on factual correctness, coherence, and completeness, however, minor factual inconsistencies may be tolerated if they do not alter the overall meaning or intent of the response. Details are provided in the appendix. These two metrics together provide a measurement to assess the FinSage's ability to generate responses that align with the annotated ground truth answers both textually and semantically.

5 Results and Analysis

5.1 MPR Results

Table 1 demonstrates FinSage's superior performance through its novel multi-path retrieval architecture. While traditional single-path approaches like FAISS and BM25 show decent performance in isolation, they each capture only limited aspects of the complex financial domain based on superficial semantic matching. FinSage's innovative combination of multiple specialized retrievers, particularly its integration of metadata-aware retrieval with fine-tuned HyDE hypothetical document generation, allows it to comprehensively capture both the structural and semantic aspects of financial documents. This synergistic approach enables FinSage to maintain high precision while significantly improving recall.

Our experimental results reveal the fine-tuned `Qwen2.5-7B-SFT` model surpasses a significantly larger `Qwen2.5-72B` counterpart across key metrics, particularly in recall performance. This highlights the critical role of fine-tuning with high-quality domain-specific data, reinforcing the necessity of FinSage for precise and accurate question-answering in the complex structure of financial filings. The strong performance of our smaller fine-tuned model aligns with emerging research demonstrating that well-adapted smaller language models can match or exceed the performance of larger models in specialized domains [8], while offering substantial computational efficiency advantages.

To better understand the uniqueness of each retrieval method, we analyze their overlap on the Company dataset. As shown in Figure 4, while each method uniquely contributes to retrieving positive chunks, there is significant overlap, suggesting that hybrid approaches could leverage their combined strengths. Notably, FAISS and HyDE retrieve highly similar positive chunks, with an overlap rate of 52.7% $(15.3\% + 12.9\% + 15.3\% + 9.2\%)$, compared to their individual recovery rates of 62.9% and 69.6%. This overlap likely arises because HyDE generates hypothetical documents by referencing the original document, preserving much of its context.

5.2 DRR Results

We compare re-rankers' performance in Figure 5, where we compare our trained re-ranker with `bge-reranker-v2-Gemma`. These results demonstrate that our Document Re-ranker significantly and consistently outperforms the general re-ranker across all settings and metrics in top-5 and top-10 configurations. Notably, our trained re-ranker retrieves more relevant chunks, evidenced by an approximate 15% improvement in recall across all setups. Moreover, performance gains are not limited to retrieving relevant chunks, it also effectively filters out irrelevant information, leading to a substantial precision boost from 5.6% to 18.5%.

It is worth noting that precision, MRR, and binary nDCG all decline as more chunks are extracted by the reranker: precision drops from

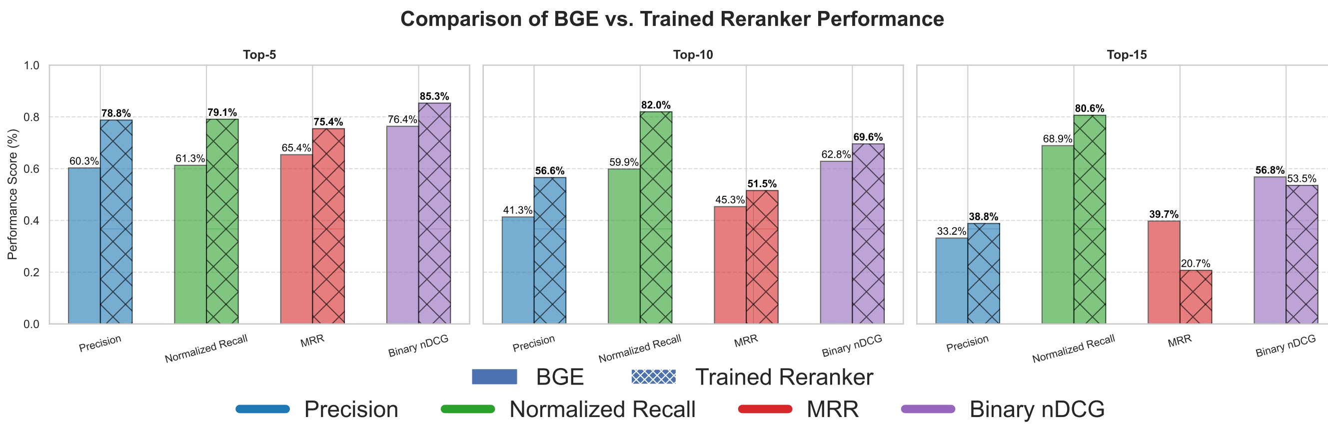

Figure 5: Performance of different re-rankers under different retrieval settings.

78.8% to 38.8%, MRR from 75.4% to 20.7%, and binary nDCG from 85.3% to 53.5%. This consistent trend across benchmark scenarios indicates that extracting additional chunks increases the burden on the re-ranker, resulting in less optimal ranking in chunk selection during re-ranking. Moreover, the rapid degradation in these metrics between Top-10 and Top-15 aligns with previous studies suggesting that retrieving only 5 to 10 chunks is generally sufficient for producing effective answers [24].

5.3 Response Results

Table 2: Performance of End-To-End QA. The bold is from our implementation, which has the top LLM and Manual scores. Results with $*$ are from our experiments, while others are from the original papers.

Dataset	Method	LLM	Manual
FinanceBench	Islam et al. [15]	-	0.1900
FinanceBench	Jimeno-Yepes et al. [17]	0.3262	0.3688
FinanceBench	Setty et al. [36]	0.2560	-
FinanceBench	FinSage*	**0.4966**	**0.5705**
Company	FinSage*	0.8533	0.8800

Table 2 shows performance on the FinanceBench and Company datasets. FinSage achieves a significant improvement over prior methods on FinanceBench, with an LLM accuracy of 49.66%, outperforming Jimeno-Yepes et al. [17] (32.62%) and Setty et al. [36] (25.60%, aligns most closely with our configuration). Similarly, manual evaluation also reflects an accuracy of 57.05%, substantially higher than the closest competitor Jimeno-Yepes et al. [17] (36.88%). These results indicate that FinSage's retrieval and ranking mechanisms contribute significantly to improved answer quality. In the Company dataset, FinSage also demonstrates strong performance, which achieves an LLM accuracy of 85.33% and a manual accuracy of 88.00%. The consistently higher scores across both datasets indicate that FinSage is particularly adept at processing structured financial filings with clear contextual information. By retrieving semantically relevant passages across multiple documents, FinSage significantly improves response accuracy, making it highly effective for complex financial question-answering tasks.

Table 3: Response Time Comparison (in seconds)

Method	Mean	Median	Min	Max
GraphRAG	16.90	15.27	9.86	40.67
LightRAG	12.16	8.84	3.41	859.51
FinSage	19.34	18.57	8.57	40.02

Table 4: Faithful Evaluation Score Comparison

Method	Questions	Mean	Median	Min	Max	Pass %
GraphRAG	71(w/ 4 failures)	3.46	3.50	2.0	5.0	42.50%
LightRAG	75	2.45	2.00	1.0	5.0	13.67%
FinSage	75	4.31	5.00	1.00	5.00	82.67%

5.4 Comparison with Other RAG Solutions

We also compare our method with graph-based RAG solutions **GraphRAG** and **LightRAG** due to their mainstream usage. Table 3 and 4 shows the advantages of Finsage in terms of response time and performance. This suggests that the additional complexity introduced by graph-based representation does not lead to improved response quality in our experimental context, which emphasizes the importance of balancing response speed with better performance when selecting RAG implementations for specific application domains. Detailed experimental settings and analysis can be found in Appendix F.

6 Conclusion

In this paper, we propose FinSage, a RAG solution for the question-answering task on financial filings. FinSage comprises a multi-modal data pre-processing pipeline, a domain-aware multi-path retrieval system specifically designed for financial data, and a domain-specialized re-ranking module fine-tuned using DPO. Our system demonstrated state-of-the-art performance on expert-curated questions derived from financial filings. Moreover, FinSage has been deployed as a financial question-answering agent in online meetings organized by affiliated entrepreneurs.

References

[1] Garima Agrawal, Tharindu Kumarage, Zeyad Alghamdi, and Huan Liu. 2024. Mindful-RAG: A Study of Points of Failure in Retrieval Augmented Generation. arXiv:2407.12216 [cs.IR] https://arxiv.org/abs/2407.12216

[2] Anonymous. 2024. TabMeta: Table Metadata Generation with LLM-Curated Dataset and LLM-Judges. In *Submitted to ACL Rolling Review - June 2024*. https://openreview.net/forum?id=NXYVm3AjG2 under review.

[3] Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. 2024. Self-RAG: Learning to retrieve, generate, and critique through self-reflection. In *Proceedings of the Twelfth International Conference on Learning Representations*.

[4] Arian Askari, Amin Abolghasemi, Gabriella Pasi, Wessel Kraaij, and Suzan Verberne. 2023. Injecting the BM25 Score as Text Improves BERT-Based Re-rankers. arXiv:2301.09728 [cs.IR] https://arxiv.org/abs/2301.09728

[5] Scott Barnett, Stefanus Kurniawan, Srikanth Thudumu, Zach Brannelly, and Mohamed Abdelrazek. 2024. Seven Failure Points When Engineering a Retrieval Augmented Generation System. arXiv:2401.05856 [cs.SE] https://arxiv.org/abs/2401.05856

[6] Maciej Besta, Ales Kubicek, Roman Niggli, Robert Gerstenberger, Lucas Weitzendorf, Mingyuan Chi, Patrick Iff, Joanna Gajda, Piotr Nyczyk, Jürgen Müller, Hubert Niewiadomski, Marcin Chrapek, Michał Podstawski, and Torsten Hoefler. 2024. Multi-Head RAG: Solving Multi-Aspect Problems with LLMs. arXiv:2406.05085 [cs.CL] https://arxiv.org/abs/2406.05085

[7] Luiz Bonifacio, Hugo Abonizio, Marzieh Fadaee, and Rodrigo Nogueira. 2022. InPars: Unsupervised Dataset Generation for Information Retrieval. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval* (Madrid, Spain) *(SIGIR '22)*. Association for Computing Machinery, New York, NY, USA, 2387–2392. doi:10.1145/3477495.3531863

[8] Martin Juan José Bucher and Marco Martini. 2024. Fine-Tuned 'Small' LLMs (Still) Significantly Outperform Zero-Shot Generative AI Models in Text Classification. arXiv:2406.08660 [cs.CL] https://arxiv.org/abs/2406.08660

[9] Deng Cai, Yan Wang, Lemao Liu, and Shuming Shi. 2022. Recent Advances in Retrieval-Augmented Text Generation. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval* (Madrid, Spain) *(SIGIR '22)*. Association for Computing Machinery, New York, NY, USA, 3417–3419. doi:10.1145/3477495.3532682

[10] Jianlv Chen, Shitao Xiao, Peitian Zhang, Kun Luo, Defu Lian, and Zheng Liu. 2024. Bge m3-embedding: Multi-lingual, multi-functionality, multi-granularity text embeddings through self-knowledge distillation. *arXiv preprint arXiv:2402.03216* (2024).

[11] Wenhu Chen, Hexiang Hu, Xi Chen, Pat Verga, and William Cohen. 2022. MuRAG: Multimodal Retrieval-Augmented Generator for Open Question Answering over Images and Text. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang (Eds.). Association for Computational Linguistics, Abu Dhabi, United Arab Emirates, 5558–5570. doi:10.18653/v1/2022.emnlp-main.375

[12] MinerU Contributors. 2024. MinerU: A One-stop, Open-source, High-quality Data Extraction Tool. https://github.com/opendatalab/MinerU.

[13] Gabriel de Souza P. Moreira, Ronay Ak, Benedikt Schifferer, Mengyao Xu, Radek Osmulski, and Even Oldridge. 2024. Enhancing Q&A Text Retrieval with Ranking Models: Benchmarking, fine-tuning and deploying Rerankers for RAG. arXiv:2409.07691 [cs.IR] https://arxiv.org/abs/2409.07691

[14] Hervé Déjean, Stéphane Clinchant, and Thibault Formal. 2024. A Thorough Comparison of Cross-Encoders and LLMs for Reranking SPLADE. arXiv:2403.10407 [cs.IR] https://arxiv.org/abs/2403.10407

[15] Pranab Islam, Anand Kannappan, Douwe Kiela, Rebecca Qian, Nino Scherrer, and Bertie Vidgen. 2023. FinanceBench: A New Benchmark for Financial Question Answering. arXiv:2311.11944 [cs.CL]

[16] Mathew Jacob, Erik Lindgren, Matei Zaharia, Michael Carbin, Omar Khattab, and Andrew Drozdov. 2024. Drowning in Documents: Consequences of Scaling Reranker Inference. arXiv:2411.11767 [cs.IR] https://arxiv.org/abs/2411.11767

[17] Antonio Jimeno-Yepes, Yao You, Jan Milczek, Sebastian Laverde, and Renyu Li. 2024. Financial Report Chunking for Effective Retrieval Augmented Generation. *ArXiv* abs/2402.05131 (2024). https://api.semanticscholar.org/CorpusID:267547721

[18] Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2019. Billion-scale similarity search with GPUs. *IEEE Transactions on Big Data* 7, 3 (2019), 535–547.

[19] Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense Passage Retrieval for Open-Domain Question Answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 6769–6781.

[20] klu.ai. [n. d.]. klu.ai. https://docs.klu.ai/

[21] Muzhi Li, Minda Hu, Irwin King, and Ho-fung Leung. 2024. The Integration of Semantic and Structural Knowledge in Knowledge Graph Entity Typing. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, Kevin Duh, Helena Gomez, and Steven Bethard (Eds.). Association for Computational Linguistics, Mexico City, Mexico, 6625–6638. doi:10.18653/v1/2024.naacl-long.369

[22] Xiaoxi Li, Jiajie Jin, Yujia Zhou, Yuyao Zhang, Peitian Zhang, Yutao Zhu, and Zhicheng Dou. 2024. From Matching to Generation: A Survey on Generative Information Retrieval. arXiv:2404.14851 [cs.IR] https://arxiv.org/abs/2404.14851

[23] Jerry Liu. 2022. *LlamaIndex*. doi:10.5281/zenodo.1234

[24] Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2023. Lost in the Middle: How Language Models Use Long Contexts. arXiv:2307.03172 [cs.CL] https://arxiv.org/abs/2307.03172

[25] Yanming Liu, Xinyue Peng, Xuhong Zhang, Weihao Liu, Jianwei Yin, Jiannan Cao, and Tianyu Du. 2024. RA-ISF: Learning to answer and understand from retrieval augmentation via iterative self-feedback. *arXiv preprint arXiv:2403.06840* (2024).

[26] Xinwei Long, Jiali Zeng, Fandong Meng, Zhiyuan Ma, Kaiyan Zhang, Bowen Zhou, and Jie Zhou. 2024. Generative multi-modal knowledge retrieval with large language models. In *Proceedings of the Thirty-Eighth AAAI Conference on Artificial Intelligence and Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence and Fourteenth Symposium on Educational Advances in Artificial Intelligence (AAAI'24/IAAI'24/EAAI'24)*. AAAI Press, Article 2089, 9 pages. doi:10.1609/aaai.v38i17.29837

[27] Shengjie Ma, Chengjin Xu, Xuhui Jiang, Muzhi Li, Huaren Qu, Cehao Yang, Jiaxin Mao, and Jian Guo. 2025. Think-on-Graph 2.0: Deep and Faithful Large Language Model Reasoning with Knowledge-guided Retrieval Augmented Generation. In *The Thirteenth International Conference on Learning Representations*. https://openreview.net/forum?id=oFBu7qaZpS

[28] Steven Moore, Huy Anh Nguyen, Tianying Chen, and John C. Stamper. 2023. Assessing the Quality of Multiple-Choice Questions Using GPT-4 and Rule-Based Methods. *CoRR* abs/2307.08161 (2023). doi:10.48550/ARXIV.2307.08161 arXiv:2307.08161

[29] Ben Naismith, Phoebe Mulcaire, and Jill Burstein. 2023. Automated evaluation of written discourse coherence using GPT-4. In *Proceedings of the 18th Workshop on Innovative Use of NLP for Building Educational Applications (BEA 2023)*, Ekaterina Kochmar, Jill Burstein, Andrea Horbach, Ronja Laarmann-Quante, Nitin Madnani, Anaïs Tack, Victoria Yaneva, Zheng Yuan, and Torsten Zesch (Eds.). Association for Computational Linguistics, Toronto, Canada, 394–403. doi:10.18653/v1/2023.bea-1.32

[30] Shashank Rajput, Nikhil Mehta, Anima Singh, Raghunandan H. Keshavan, Trung Vu, Lukasz Heldt, Lichan Hong, Yi Tay, Vinh Q. Tran, Jonah Samost, Maciej Kula, Ed H. Chi, and Maheswaran Sathiamoorthy. 2023. Recommender Systems with Generative Retrieval. arXiv:2305.05065 [cs.IR] https://arxiv.org/abs/2305.05065

[31] Monica Riedler and Stefan Langer. 2024. Beyond Text: Optimizing RAG with Multimodal Inputs for Industrial Applications. arXiv:2410.21943 [cs.CL] https://arxiv.org/abs/2410.21943

[32] Stephen Robertson, Hugo Zaragoza, et al. 2009. The probabilistic relevance framework: BM25 and beyond. *Foundations and Trends® in Information Retrieval* 3, 4 (2009), 333–389.

[33] Idan Schwartz. 2021. Ensemble of MRR and NDCG models for Visual Dialog. arXiv:2104.07511 [cs.AI] https://arxiv.org/abs/2104.07511

[34] SEC. 2024. JP Morgan Affiliates to Pay 151 Million to Resolve SEC Enforcement Actions. https://www.sec.gov/newsroom/press-releases/2024-178

[35] SEC. 2024. SEC Levies More Than 3.8 Million in Penalties in Sweep of Late Beneficial Ownership and Insider Transaction Reports. https://www.sec.gov/newsroom/press-releases/2024-148

[36] Spurthi Setty, Harsh Thakkar, Alyssa Lee, Eden Chung, and Natan Vidra. 2024. Improving Retrieval for RAG based Question Answering Models on Financial Documents. arXiv:2404.07221 [cs.IR] https://arxiv.org/abs/2404.07221

[37] Shalin Shah, Srikanth Ryali, and Ramasubbu Venkatesh. 2024. Multi-Document Financial Question Answering using LLMs. arXiv:2411.07264 [cs.IR] https://arxiv.org/abs/2411.07264

[38] Dewang Sultania, Zhaoyu Lu, Twisha Naik, Franck Dernoncourt, David Seunghyun Yoon, Sanat Sharma, Trung Bui, Ashok Gupta, Tushar Vatsa, Suhas Suresha, Ishita Verma, Vibha Belavadi, Cheng Chen, and Michael Friedrich. 2024. Domain-specific Question Answering with Hybrid Search. arXiv:2412.03736 [cs.CL] https://arxiv.org/abs/2412.03736

[39] C. Wang, Q. Long, M. Xiao, et al. 2024. Biorag: A RAG-LLM framework for biological question reasoning. *arXiv preprint arXiv:2408.01107* (2024).

[40] Yile Wang, Peng Li, Maosong Sun, and Yang Liu. 2023. Self-knowledge guided retrieval augmentation for large language models. In *Findings of the Association for Computational Linguistics: EMNLP 2023*. 10303–10315.

[41] Peng Xia, Kangyu Zhu, Haoran Li, Tianze Wang, Weijia Shi, Sheng Wang, Linjun Zhang, James Zou, and Huaxiu Yao. 2024. MMed-RAG: Versatile Multimodal RAG System for Medical Vision Language Models. arXiv:2410.13085 [cs.LG] https://arxiv.org/abs/2410.13085

[42] Qianqian Xie, Dong Li, Mengxi Xiao, Zihao Jiang, Ruoyu Xiang, Xiao Zhang, Zhengyu Chen, Yueru He, Weiguang Han, Yuzhe Yang, Shunian Chen, Yifei Zhang, Lihang Shen, Daniel Kim, Zhiwei Liu, Zheheng Luo, Yangyang Yu, Yupeng Cao, Zhiyang Deng, Zhiyuan Yao, Haohang Li, Duanyu Feng, Yongfu Dai, Vijaya-Sai Somasundaram, Peng Lu, Yilun Zhao, Yitao Long, Guojun Xiong, Kaleb Smith, Honghai Yu, Yanzhao Lai, Min Peng, Jianyun Nie, Jordan W. Suchow, Xiao-Yang Liu, Benyou Wang, Alejandro Lopez-Lira, Jimin Huang, and Sophia Ananiadou.

2024. Open-FinLLMs: Open Multimodal Large Language Models for Financial Applications. arXiv:2408.11878 [cs.CL] https://arxiv.org/abs/2408.11878

[43] Lee Xiong, Chenyan Xiong, Ye Li, Kwok-Fung Tang, Jialin Liu, Paul Bennett, Junaid Ahmed, and Arnold Overwijk. 2020. Approximate nearest neighbor negative contrastive learning for dense text retrieval. *arXiv preprint arXiv:2007.00808* (2020).

[44] Shi Yu, Chaoyue Tang, Bokai Xu, Junbo Cui, Junhao Ran, Yukun Yan, Zhenghao Liu, Shuo Wang, Xu Han, Zhiyuan Liu, and Maosong Sun. 2025. VisRAG: Vision-based Retrieval-augmented Generation on Multi-modality Documents. In *The Thirteenth International Conference on Learning Representations*. https://openreview.net/forum?id=zG459X3Xge

[45] Ye Yuan, Chengwu Liu, Jingyang Yuan, Gongbo Sun, Siqi Li, and Ming Zhang. 2024. A Hybrid RAG System with Comprehensive Enhancement on Complex Reasoning. arXiv:2408.05141 [cs.CL] https://arxiv.org/abs/2408.05141

[46] Wenjia Zhai. 2024. Self-adaptive Multimodal Retrieval-Augmented Generation. arXiv:2410.11321 [cs.CL] https://arxiv.org/abs/2410.11321

[47] Le Zhang, Yihong Wu, Fengran Mo, Jian-Yun Nie, and Aishwarya Agrawal. 2023. MoqaGPT: Zero-Shot Multi-modal Open-domain Question Answering with Large Language Model. *arXiv preprint arXiv:2310.13265* (2023).

[48] Le Zhang, Yihong Wu, Qian Yang, and Jian-Yun Nie. 2024. Exploring the Best Practices of Query Expansion with Large Language Models. In *Findings of the Association for Computational Linguistics: EMNLP 2024*. 1872–1883.

[49] Shengyu Zhang, Linfeng Dong, Xiaoya Li, Sen Zhang, Xiaofei Sun, Shuhe Wang, Jiwei Li, Runyi Hu, Tianwei Zhang, Fei Wu, and Guoyin Wang. 2024. Instruction Tuning for Large Language Models: A Survey. arXiv:2308.10792 [cs.CL] https://arxiv.org/abs/2308.10792

[50] Qingfei Zhao, Ruobing Wang, Yukuo Cen, Daren Zha, Shicheng Tan, Yuxiao Dong, and Jie Tang. 2024. LongRAG: A Dual-Perspective Retrieval-Augmented Generation Paradigm for Long-Context Question Answering. arXiv:2410.18050 [cs.CL] https://arxiv.org/abs/2410.18050

[51] Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. 2023. Judging LLM-as-a-Judge with MT-Bench and Chatbot Arena. arXiv:2306.05685 [cs.CL] https://arxiv.org/abs/2306.05685

[52] Kunlun Zhu, Yifan Luo, Dingling Xu, Ruobing Wang, Shi Yu, Shuo Wang, Yukun Yan, Zhenghao Liu, Xu Han, Zhiyuan Liu, et al. 2024. Rageval: Scenario specific rag evaluation dataset generation framework. *arXiv preprint arXiv:2408.01262* (2024).

A Data Preprocessing Details

To effectively convert PDF files into structured data, we utilize the Mineru tool [1], which can extract content from PDFs and export it in various formats. The preprocessing workflow is described below.

A.1 PDF to JSON Conversion Process

We can utilize the Mineru to to process PDF files with the following command:

```
magic-pdf -p {some_pdf} -o {some_output_dir} -m auto
```

Users may need to specify the input PDF file and the output directory. The tool generates the following output files:

- `some_pdf.md`: Markdown representation of the PDF content.
- `images/`: A directory containing extracted images, including tables.
- `some_pdf_layout.pdf`: A visualization of the document's layout.
- `some_pdf_middle.json`: Intermediate processing results in JSON format.
- `some_pdf_model.json`: Model inference results in JSON format.
- `some_pdf_origin.pdf`: The original PDF file.
- `some_pdf_spans.pdf`: A PDF file showing the bounding box positions at the smallest granularity.
- `some_pdf_content_list.json`: A structured JSON file representing the document content in reading order.

Among these files, `some_pdf_content_list.json` stores the document content into blocks, each of which contains the following fields: `type`, `text`, `text_level`, and `page_idx`. The `type` field differentiates different content types, such as:
- `text`: Regular text blocks. - `table`: Table elements. - `img_path`: Image paths referencing extracted tables.

A.2 Examples of JSON Structure

Text Block Example:

```
{
    "type": "text",
    "text": "U.S. SEC Washington, D.C. 20549",
    "text_level": 1,
    "page_idx": 1
}
```

In this example, `type` = `text` signifies a text block, while `text_level` = 1 designates it as a title.

Table Block Example:

[1] https://github.com/opendatalab/MinerU

Table 5: Retrieval Performance on the Company Dataset with a Fixed top-K. Avg Num Retrieved represents the average number of chunks in the retrieved set. Top-K is set to 10, so each retriever selects its top 10 results for the retrieval set.

Category	Method	Avg Recall	Avg Precision	Avg F1	Avg Num Retrieved
FAISS Retrieval	FAISS (Baseline)	0.5984	0.4427	0.4861	10.8
	+ Bundle Expansion(Exp)	0.6206	**0.4440**	**0.4960**	11.28
HyDE Retrieval	**HyDE-1**: HyDE(Qwen7B, w/ Exp)	0.7019	0.2119	0.3084	28.73
	HyDE-2: HyDE(Qwen7B-SFT, w/ Exp)	0.7186	0.2438	0.3450	25.48
	HyDE-3: HyDE(Qwen72B, w/ Exp)	0.7310	0.2160	0.3178	29.49
FinSage	+ BM25+Metadata+**HyDE-2**	**0.9251**	0.1272	0.2156	68.75

```
{
    "type": "table",
    "img_path": "images/aba30...36.jpg",
    "table_caption": ["Shareholder structure ..."],
    "table_footnote": [],
    "page_idx": 9
}
```

Here, `type` = `table` denotes a table, with `img_path` pointing to the extracted table image.

Image Block Example:

```
{
    "type": "image",
    "img_path": "images/c22a5...41.jpg",
    "img_caption": [
        "Our Holding Company Structure "
    ],
    "img_footnote": [],
    "page_idx": 17
}
```

Here, `type` = `image` denotes a image, with `img_path` pointing to the extracted image.

A.3 Image Processing and Text Parsing

To extract table information, the Mineru2Base tool leverages the GPT-4o API to convert table images into text. The system iterates through each document block in `some_pdf_content_list.json`. For blocks containing an `img_path`, the tool retrieves the preceding and succeeding blocks as context. The table image is then converted to Base64 format and passed to the GPT-4o API, which generates a structured textual representation.

A.4 Embedding Model

Two embedding models are used in this project, each serving a distinct purpose: TF-IDF for text deduplication and BGE-M3 for storing text in the database.

For the deduplication task, we employ the TF-IDF tool to convert the text into vector representations. This approach helps to identify and remove redundant text segments by comparing their vector similarities. The model uses term frequency and inverse document frequency to generate vectors, which are then compared using cosine similarity to detect high similarity between text segments. Text with similarities above a predefined threshold is removed, reducing redundancy.

For storing text in the Chroma database, we use the BGE-M3 embedding model. This model, trained on large-scale datasets, transforms text into high-dimensional vectors that capture the semantic meaning of the content. These vectors are then stored in the database, enabling efficient indexing and retrieval for future similarity searches and semantic analysis.

A.5 Duplicate Removal

To remove redundant content, a de-duplication step is performed using text similarity measurements. The method includes:

1.Chinese word segmentation using the Jieba library. 2.ransforming text into a term-frequency matrix via PyTorch, where the text is vectorized using a TF-IDF approach implemented with torch for efficient matrix computation. 3.Computing cosine similarity between document blocks using PyTorch's cosine similarity function.

If the similarity score exceeds 0.7 (default threshold), the duplicate block is removed.

A.6 Coreference Resolution

Coreference resolution is an important task in natural language processing, aiming to identify the entities that pronouns and other referring expressions point to within a text. In simple terms, it analyzes pronouns (such as "he," "it," "we," etc.) and determines the specific object or entity they refer to, making the text more coherent and clear. The relationship between pronouns and their referred entities can significantly affect comprehension, and thus coreference resolution plays a crucial role in improving text readability and reducing ambiguity.

In practical implementation, the algorithm traverses a JSON file containing multiple chunks, each with a title and content field. For each chunk, the coreference resolution follows a specific process: it first searches up to four preceding chunks as references. The search is limited to preceding chunks that share the same title as the current one. If the previous chunk belongs to a different title, it is considered to have low relevance to the current chunk and is thus not considered as a reference.

In this way, the coreference resolution algorithm effectively scans the related content within the text, using the GPT-4o-mini API to resolve pronoun references and replace them with specific entities. This approach not only improves the clarity and coherence of

the text but also reduces the ambiguity that pronouns may cause, ensuring the accurate transmission of information.
Detail prompts provided to GPT-4o-mini during the anaphora resolution step are shown in Figure 6:

A.7 Title Summarization

Each block receives a summary stored in the `title_summary` field. The process involves:
1. Generating a concise summary for each block. 2. Aggregating summaries to form a higher-level title summary. 3. Storing the final summary in `title_summary`.
For example, the original JSON format:

```
{
    "id": 1,
    "content": "Some text content here...",
    "page_number": 1,
    "title": "Example Title",
    "type": "text"
}
```

After summarization:

```
{
    "id": 1,
    "content": "Some text content here...",
    "page_number": 1,
    "title": "TITLE AAA",
    "type": "text",
    "title_summary": "Title+Summary"
}
```

A.8 Text Segmentation

To maintain manageable block sizes, text segmentation is applied. The segmentation rules are:
- If a block has fewer than 200 characters, subsequent content is appended until it exceeds 200. - If a block exceeds 200 characters, the last sentence is moved to the next block. - Segmentation is restricted within the same title to prevent cross-title segmentation.
Through this structured preprocessing pipeline, PDF content is efficiently converted into JSON format, ensuring accurate extraction of text, tables, and other elements. This process facilitates subsequent analysis, model applications, and further information retrieval.

A.9 Loading Chunk into a Vector Database

In this study, text data is loaded into a Chroma-based vector database, using the BGE-M3 embedding model to vectorize the text. The process begins by reading a series of JSON-format files from a specified directory, each containing metadata related to page content and the text itself. Metadata such as page range and publication date are extracted and processed along with the corresponding content. To avoid duplicate data, each piece of text is uniquely identified by calculating its SHA-256 hash, and in case of duplicate content, the system updates the record with the latest version based on the publication date.
Next, the text and associated metadata are stored into the Chroma database in batches. During the storage process, the system also maintains the sequential relationship between documents by adding "previous" and "next" chunk identifiers for texts within the same file. Finally, to further optimize information retrieval, a BM25 index is built based on the text data stored in the Chroma database, improving retrieval efficiency in subsequent searches.
Through this process, we can effectively convert large amounts of text data into vectors and provide efficient database support for future retrieval and analysis.

B Reranker Finetune Details

B.1 Data Preparation

The training data is stored in a JSONL format, where each entry contains the following fields: `query`, `pos`, `neg`, and optionally, `pos_scores`, `neg_scores`, and `prompt`. The key fields are as follows:

- `query`: The query string.
- `pos`: A list containing one positive candidate related to the query.
- `neg`: A list containing multiple negative candidates unrelated to the query.
- `pos_scores` (optional): A list of scores corresponding to the positive candidates.
- `neg_scores` (optional): A list of scores corresponding to the negative candidates.
- `prompt` (optional): A prompt string used for additional customization of the training data.

The minimal required data format is as follows:

```
{
  "query": "str",
  "pos": ["str"],
  "neg": ["str"]
}
```

In each labeled sample, the pos field is structured as a list containing only a single positive instance. This positive instance typically represents the most relevant or ideal answer in relation to the query, thereby providing the model with a clear and representative positive reference. In contrast, the neg field consists of a list of multiple negative candidate strings, each reflecting content that is either irrelevant or only weakly related to the query. This data construction facilitates a significant enhancement of the model's sensitivity and discriminative capability in assessing relevance, achieved through contrastive learning between positive and negative samples during the fine-tuning of the reranker model.
Example of the labeled training data:

```
{
  "query": "question1",
  "pos": ["aaa"],
  "neg": ["ccc", "dddd", "eee"]
}
```

```
You are a language model assistant. Your task is to enhance the given text by replacing ambiguous or context-dependent words with more
specific and clear alternatives, based on the context of company's financial reports.

## Here are some guidelines for replacements:
- Replace pronouns like "we" with the appropriate entity, but only use entities that appear in the current context:
       CORRECT: If the text mentions "Lotus Technology's R\&D department" earlier, then "we" → AAA Technology's R\&D department"
       INCORRECT: Don't introduce new entities or products that aren't mentioned in the context

- When replacing "it" or other pronouns:
       CORRECT: Only use specific names/entities that were previously mentioned in the same or immediately preceding paragraph
       CORRECT: If the specific reference is unclear, use a descriptive but general term (e.g., "the company", "this initiative")
       INCORRECT: Don't introduce specific product names or entities that aren't in the source text

- For product references:
       CORRECT: Only use exact product names that appear in the text
       CORRECT: If the specific product name isn't clear, use general terms like "the vehicle", "this model"
       INCORRECT: Don't make assumptions about which specific product is being discussed

    Examples:

    . . .

    INSTRUCTIONS:

    1. Use the reference context ONLY to understand what pronouns and references refer to
    . . .


    Please note:
    1. Use only the reference context to understand the direction of pronouns and referents
    . . .
```

Figure 6: The instruction prompts for anaphora resolution.

```
{
  "query": "question2",
  "pos": ["bbb"],
  "neg": ["ccc", "dddd", "eee"]
}
```

The above format is used for fine-tuning a reranker model, where the goal is to reorder the candidate passages based on their relevance to the query.

B.2 Selection of Positive and Negative Samples

In this study, the selection of positive and negative samples is governed by a systematic and rigorous procedure, meticulously designed to ensure that the chosen samples effectively facilitate both the training and subsequent optimization of the model. The process comprises several carefully orchestrated stages, which are detailed as follows:

Initially, a comprehensive set of questions was curated to represent a wide array of potential query scenarios that the model might encounter in real-world applications. These questions were selected to capture the inherent diversity and complexity of user inquiries, thereby ensuring that the dataset is both representative and robust. For each question, a dual retrieval strategy was implemented using the Faiss and BM25 algorithms within the Chroma vector database. Faiss, renowned for its efficiency in high-dimensional vector similarity search, and BM25, a well-established probabilistic retrieval model, complement each other by capturing both semantic nuances and lexical patterns. This combined approach ensures that each query retrieves a diverse set of candidate answers—typically exceeding twenty per question—thus providing a rich pool of potential responses with varying degrees of relevance and quality.

Subsequent to the retrieval phase, the candidate results underwent an exhaustive manual labeling process. Expert annotators meticulously evaluated each candidate to assess its capacity to accurately and comprehensively address the corresponding question. Candidates that met stringent criteria for precision, relevance, and completeness were designated as positive samples. These positive samples are intended to exemplify the optimal answers, providing a clear and reliable reference for the model during training. Conversely, any candidate that failed to deliver an effective or accurate answer was classified as a negative sample. This deliberate bifurcation into positive and negative samples ensures that the model is exposed to a distinct contrast between high-quality and substandard responses, thereby enhancing its ability to discriminate between relevant and irrelevant content.

To ensure fairness, consistency, and reproducibility throughout the sample selection process, each question was processed using an identical evaluation protocol. Strict adherence to uniform criteria and standardized methods during the manual labeling phase served to minimize potential biases that might otherwise arise from automated selection procedures. This rigorous approach not only guarantees the high quality of both positive and negative samples but also underpins the overall robustness of the training data. Ultimately, this meticulous curation process is instrumental in bolstering the model's performance, particularly in its capacity to accurately discern and rank relevant content in response to diverse queries.

B.3 Training Parameters

During the fine-tuning of the reranker model, Low-Rank Adaptation (LoRA) was used to reduce the number of parameters that need to be updated during training, thus enhancing training efficiency. Specifically, LoRA was enabled, with the rank set to 32 and the scaling factor set to 64. The learning rate was set to 1e-4 to ensure stable training and prevent gradient explosion. To save memory and improve stability, the batch size per device was set to 1. Additionally, gradient accumulation was enabled, with gradients updated after processing two batches, allowing for more efficient training while maintaining a small batch size. The model was trained using 10 epochs, and mixed precision training was employed with the bf16 setting to further optimize performance and memory usage.

C RAG Evaluation

To evaluate the quality of the Retrieval-Augmented Generation (RAG) model, this study employs two assessment methods: large model evaluation and human evaluation. First, a representative set of questions is collected, covering a range of query scenarios that the model may encounter. For each question, the entire RAG process is executed, including retrieval, generation, and ranking steps, ultimately producing the model's answers. To facilitate subsequent analysis and evaluation, these generated results are stored in a standardized JSON file format, ensuring traceability and structured data. Through this approach, we can comprehensively assess the performance of the RAG model, evaluating its performance under automated evaluation metrics while also incorporating human review to further analyze the accuracy and relevance of its generated content.

C.1 Environment Setting

Our experiments were conducted on a NVIDIA A100-SXM4-80GB server, where we deploy the Qwen2.5-72B-Instruct-AWQ 4bit model. The model was configured using the following command:

Table 6: Model deployment parameters using VLLM.

Parameter	value
max-model-len	6144
gpu-memoty-utilization	0.7
engorce-eager	True
swap-space	36

C.2 LLM Evaluation

In this approach, we utilize the open-source tool *llamaindex* [23] to systematically evaluate RAG model outputs by loading two JSON files: one containing generated QA pairs from the RAG model, and the other providing reference answers with contextual data in `content_list`. Each question is mapped to its reference information, and a `Response` object is constructed by combining the RAG-generated answer with its concatenated context. We then employ the `CorrectnessEvaluator` module (backed by a GPT-4o api) to assess each `Response`, returning a binary `passing` flag along with detailed `feedback`.

To facilitate the evaluation, we integrate the correctness assessment module from LlamaIndex's `CorrectnessEvaluator`. This module assigns a numerical score (ranging from 1 to 5) to each response based on alignment with reference answers, using a predefined evaluation template. A threshold-based scoring mechanism determines whether a response meets the required correctness level. The evaluator asynchronously processes input queries, system responses, and reference answers using a structured prompt, ensuring consistency and scalability in judgment. The predefined evaluation prompt template encapsulated in the module is as follows:
During the evaluation, logs and statistics are recorded, including total queries processed, successful evaluations, and skipped items, while exceptions are noted and the evaluation continues for remaining data. Finally, the results—comprising the question, generated answer, reference answer, evaluation status, and feedback—are stored in a structured JSON file for further analysis, providing a robust, LLM-driven metric for measuring and tracing the fidelity of RAG-generated responses.
An example of the result is as below:

C.3 Re-ranker Evaluation

Binary Normalized Discounted Cumulative Gain (BnDCG) measures ranking quality by applying position-aware discounting with logarithmic scaling. Since our manually annotated dataset lacks explicit ranking scores for ground truth chunks, we adopt a binary relevance model (1 for relevant, 0 for irrelevant). DCG and its ideal counterpart (IDCG) are computed as:

$$\mathrm{DCG} = \sum_{i=1}^{n} \frac{rel_i}{\log_2(i+1)} \tag{3}$$

where rel_i is the binary relevance score of the retrieved chunk at position i, taking values of either 1 (relevant) or 0 (irrelevant).
The Ideal DCG (IDCG) is computed by sorting all relevant chunks to the top positions:

$$\mathrm{IDCG} = \sum_{i=1}^{n^*} \frac{1}{\log_2(i+1)} \tag{4}$$

where n^* is the total number of relevant chunks in the dataset, and $rel_i = 1$ for all positions i in the ideal ranking, ensuring that all relevant chunks appear first. Finally, BnDCG is given by:

$$\mathrm{BnDCG} = \frac{\mathrm{DCG}}{\mathrm{IDCG}} \tag{5}$$

ensuring a normalized ranking score between 0 and 1, where 1 represents a perfect ranking.

D Efficiency and cost analysis

To evaluate computational efficiency, we conducted detailed benchmarks using an NVIDIA H20 GPU. This hardware surpasses the minimum system requirement of a GPU with 24GB of memory, making it available to increase the batch size for the re-ranker model, resulting in higher inference throughput. Table 8 summarizes key performance metrics, specifically the latency and estimated cost per user request. Latency is defined as the time elapsed from receiving a user request to generating the first token of the final response. Our analysis indicates that the primary computational costs originate

```
DEFAULT_SYSTEM_TEMPLATE = """
You are an expert evaluation system for a question answering chatbot.

You are given the following information:
- a user query, and
- a generated answer

You may also be given a reference answer to use for reference in your evaluation.

Your job is to judge the relevance and correctness of the generated answer.
Output a single score that represents a holistic evaluation.
You must return your response in a line with only the score.
Do not return answers in any other format.
On a separate line provide your reasoning for the score as well.

Follow these guidelines for scoring:
- Your score has to be between 1 and 5, where 1 is the worst and 5 is the best.
- If the generated answer is not relevant to the user query, you should give a score of 1.
- If the generated answer is relevant but contains mistakes, you should give a score between 2 and 3.
- If the generated answer is relevant and fully correct, you should give a score between 4 and 5.

Example Response:
4.0
The generated answer has the exact same metrics as the reference answer, but it is not as concise.

"""

DEFAULT_USER_TEMPLATE = """
## User Query
{query}

## Reference Answer
{reference_answer}
```

Figure 7: Predefined evaluation prompt template used in LlamaIndex's `CorrectnessEvaluator`.

```
{
  "summary": {
    "total_questions": 75,
    "passing_questions": 64,
    "passing_rate": 85.33333333333334,
    "skipped_questions": {
      "missing_fields": [],
      "not_in_correct_dict": [],
      "evaluation_error": []
    }
  },
  "detailed_results": [
    {
      "question": "question1",
      "generated_answer": "FinSage's output answer",
      "reference_answer": "FinSage's reference answer (domain expert reviewed)",
      "passing": true,
      "feedback": "Feedback provided by GPT-4"
    },
    // more questions
}
```

Figure 8: Example question answering results

Table 7: Performance comparison of different re-rankers under different retrieval settings.

Method	Precision	Normalized Recall	MRR	Binary nDCG
5 Candidate Results per Retriever (R=5):				
Top-5 BGE	0.5913	0.6043	0.6570	0.7540
Top-5 trained Reranker	**0.7324**	0.7456	**0.7078**	**0.8124**
Top-10 BGE	0.3771	0.6881	0.3854	0.6011
Top-10 trained Reranker	0.4424	**0.7783**	0.3338	0.6097
10 Candidate Results per Retriever (R=10):				
Top-5 BGE	0.6028	0.6130	0.6540	0.7638
Top-5 trained Reranker	**0.7878**	0.7910	**0.7545**	**0.8533**
Top-10 BGE	0.4133	0.5985	0.4533	0.6285
Top-10 trained Reranker	0.5657	**0.8196**	0.5155	0.6958
Top-15 BGE	0.3318	0.6889	0.3974	0.5680
Top-15 trained Reranker	0.3883	0.8064	0.2071	0.5354

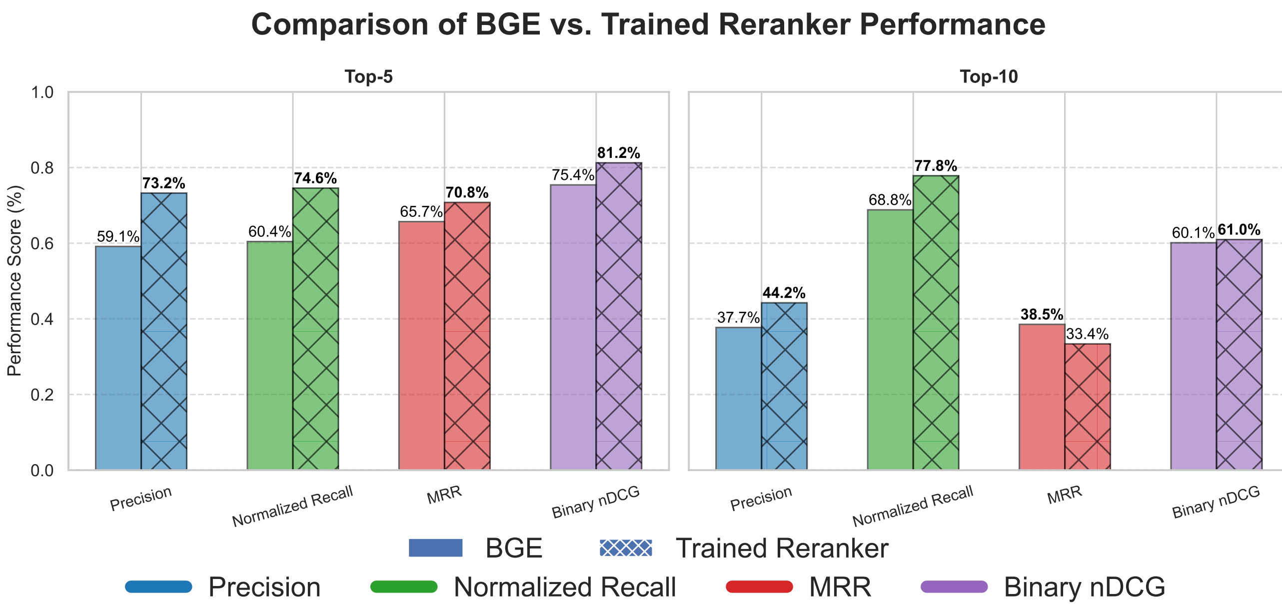

Figure 9: Performance comparison of different re-rankers under different retrieval settings when limiting the retrieval bundle to 5.

Table 8: System Efficiency and Deployment Cost

Step	Process	Time (s)	Model Used	Token Usage	Cost Estimation
1	Query rewriting	∼2.5	GPT-4o	∼1200	∼\$0.005
2	HyDE per sub-query (async)	∼4.2	GPT-4o	∼500	∼\$0.002 × n
3	Retrieval & reranking	∼4.7	Local	N/A	24GB GPU
4	Sub-query answering (async)	∼4.7	GPT-4o	∼2500	∼\$0.012 × n
5	Final answer merging	∼1.7	GPT-4o	∼200 + 200 × n	∼\$0.002 + \$0.002 × n
Total	–	**∼13–17**	–	∼3.7k + 3k × n	**∼\$0.017 + \$0.016 ×** n

n **denotes the number of sub-queries.**

Table 9: Retrieval Latency.

Method	Avg Time (s)
FAISS	0.057
Metadata(FAISS)	0.050
BM25	0.014
Total	**0.121**

from two main sources: the LLM inference steps, where latency scales with the length of the generated output, and the re-ranker model inference. And the estimated average cost to process a request involving a single sub-query is approximately 13 seconds with $0.03. Furthermore, Table 9 details the retrieval latency for the In-depth company dataset, demonstrating the efficiency of our multi-path retriever implementation.

E End-user feedback analysis

Our system is actively used by a major industrial client, CG Matrix Technology Limited, where FinSage has processed 2,702 user queries with a strong average satisfaction score of 4.19/5.0. The satisfaction score is the rating users give to each response generated by FinSage, with scores ranging from 1-5 (including half-point ratings like 4.5). Overall score distribution in table 10 reveals that 81.87% of all queries received a score of between 4 and 5, whcih indicates high end-user satisfaction with FinSage's responses.

As shown in table 11, we also categorized the questions into RAG questions (91.9%) and Non-RAG questions (8.1%). The distribution indicates that the predominant use of the system focuses on retrieving domain-specific financial information, which aligns with our intended design goals. The difference of average satisfaction score between RAG (4.13/5.0) and Non-RAG questions (4.80/5.0) derives from the inherent complexity of financial queries requiring precise retrieved context versus more general knowledge questions that LLM that already acquired.

For RAG questions, we categorize them into 4 categories, each addressing distinct informational needs in the financial domain as shown in table 12:

- **Business, Products, Competition (35.2%)**: This category encompasses queries about product lines, pricing strategies, sales channels, market positioning, and competitive advantages. Users frequently inquire about specific product offerings, international market presence, competitive landscape analysis, research and development investments, and strategic business advantages. These questions require the system to synthesize information across product specifications, market analysis documents, and competitive intelligence reports.
- **Basic Information, Equity Structure (28.3%)**: Questions in this category focus on fundamental company information, stock data, shareholder composition, management profiles, and corporate governance details. Users seek information about headquarters location, stock exchange listings, major shareholders, board composition, and corporate relationships. Answering these queries requires precise extraction of factual data points from corporate filings, annual reports, and governance documents.
- **Financial Status, Performance (25.8%)**: This category contains queries about financial metrics, operational performance, profitability measures, and forward-looking projections. Users typically request specific financial indicators such as gross margins, operational profits/losses, delivery volumes, sales forecasts, and financial ratios. These questions often involve temporal reasoning and numerical analysis across multiple financial statements and reports.
- **Regulatory Policy (2.5%)**: The smallest but highly specialized category addresses regulatory frameworks, tariff structures, geopolitical influences, and policy impacts. Users inquire about tariff applications, subsidy programs, tax incentives, and regulatory compliance requirements. These complex queries require the system to interpret regulatory documents and assess their implications on business operations.

While examining specific RAG question categories (table 12), we notice **Business, Products, Competition** queries were most frequent (35.2%) with medium average satisfication score, while for **Regulatory, Policy** questions, even it represents only 2.5% of total queries, it receives relatively high satisfaction scores (4.40/5.0), which demonstrates FinSage's effectiveness in handling complex compliance-related inquiries. However, **Financial status, Performance** queries receive the lowest average satisfaction score (3.88/5.0) among RAG categories, this suggests the area for potential improvement, which requires more precise numerical data retrieval and temporal reasoning across multiple chunks that FinSage might have to incoporate.

Table 10: Detailed Score Distribution

Score	Count	Percentage
1.0	100	3.70%
2.0	76	2.81%
2.5	33	1.22%
3.0	206	7.62%
3.5	75	2.78%
4.0	227	8.40%
4.5	1466	54.26%
5.0	519	19.21%

Table 11: Distribution and Performance of RAG vs Non-RAG Questions

Question Type	Count	Percentage	Avg. Score
RAG questions	2,482	91.9%	4.13/5.0
Non-RAG questions	220	8.1%	4.80/5.0

Table 12: Distribution and Performance of user feedback by RAG Question Category. Percentages shown are relative to all RAG questions, which constitute a subset of total questions in the user queries.

RAG Question Category	Count	Percentage	Avg. Score
business_products_competition	952	38.4%	4.23/5.0
basic_info_equity_structure	765	30.8%	4.22/5.0
financial_status_Performance	697	28.1%	3.88/5.0
regulatory_policy	68	2.7%	4.40/5.0

F Alternative Methodology comparison

We also explore the usage of graph-based representations, we conducte additional experiments with **GraphRAG** and **LightRAG** due to their mainstream usage for graph-based retrieval, showing unsatisfactory performance from both graph-based retrieval models (Table 3 and 4).

F.1 GraphRAG Configuration

GraphRAG is a graph-based retrieval system that uses knowledge graph representations to enhance contextual understanding. The configuration is detailed in Table 13.

Table 13: GraphRAG Configuration Parameters and Methods

Parameter / Method	Value / Description
Chat Model	
Model	gpt-4o
Concurrent Requests	25
Async Mode	threaded
Embedding Model	
Model	text-embedding-3-small
Concurrent Requests	25
Data Processing	
Chunk Size	1200
Chunk Overlap	100
Group By	id
Vector Store	lancedb
Graph Processing	
Entity Types	organization, person, geo, event
Extract Claims	disabled (optional)
Max Cluster Size	10
Summary Max Length	500
Query Methods	
local_search	Context-dependent information
global_search	Map-reduce approach
drift_search	Handles concept drift
basic_search	Simple retrieval mechanism

F.2 LightRAG Configuration

LightRAG represents a more streamlined approach which prioritizes efficiency while maintaining retrieval quality. The detailed configuration is shown in Table 14.

F.3 Comparative Analysis

Our experiment results presented in Tables 3 and 4 reveal the important insights regarding the performance trade-offs between GraphRAG and LightRAG implementations, compared to FinSage. While both retrieval models do achieve enhanced contextual understanding through structural relationships, we notice their unsatisfactory performance across several key metrics. **GraphRAG** achieved moderate faithful evaluation scores (mean: 3.46) with a 42.50% pass rate but at the cost of longer response times (mean: 16.09s). **LightRAG** optimized for speed (mean: 12.16s) but delivered the poorest faithful evaluation performance with only a 13.67% pass rate, which is not acceptable in the financial question answering domain. In general, FinSage demonstrates superior faithful evaluation scores (mean: 4.31) with an impressive 82.67% pass rate, although with slightly longer response times (mean: 19.34s). This suggests that the additional complexity introduced by graph-based representation does not translate to improved response quality in our experimental context. These findings emphasize the importance of balancing response speed with response quality when selecting RAG implementations for specific application domains.

G Additional Results

G.1 Re-ranker Results

We also assess the impact of Document Re-ranker on retrieval effectiveness using different R, as detailed in Table 7. R denotes the number of candidate bundles for each retriever. The retrieval system under this assessment integrates four retrievers—FAISS (baseline), BM25, Metadata, and HyDE—retrieving at least $4 \times R$ chunks. While increasing R enhances the chances of retrieving relevant chunks, it also intensifies the re-ranker's challenge of filtering out irrelevant ones.

Table 14: LightRAG Configuration Parameters

Parameter	**Value / Description**
Language Model	
Model	GPT-4o
Configuration	custom_llm_model_func
Embedding Model	
Model	text-embedding-3-small
Dimension	1536 (auto-detected)
Max Token Size	8192
Parallel Processing	
embedding_batch_num	128
embedding_func_max_async	64
max_parallel_insert	8
llm_model_max_async	16
insert_batch_size	32
Query Parameters - Retrieval Modes	
mode	mix (default), local, global, hybrid, naive
top_k	60 (default, configurable)
Query Parameters - Token Limits	
max_token_for_text_unit	4000
max_token_for_global_context	4000
max_token_for_local_context	4000
Response Controls	
response_type	"Multiple Paragraphs"
stream	false
only_need_context	false
only_need_prompt	false
Keyword Parameters	
hl_keywords	High level keywords: retrieval prioritization
ll_keywords	Low level keywords: refinement