A System for Vietnamese Financial Report Retrieval Using OCR and RAG

GROUP 4

Mai Phan Quoc Hung	11222607
Le Hoang Minh	11224192
Nguyen Phuong Hoai Ngoc	11224722
Pham Minh Anh	11220539

A thesis submitted in fulfillment of the requirements for the degree of

Machine Learning 1

in the

Faculty of Mathematics and Economics
National Economics University

Friday 28 March, 2025

Abstract

The use of Large Language Models (LLMs) for financial document analysis has seen significant advancements, yet substantial challenges remain due to the unstructured and varied formats of financial data, particularly in scanned documents. Our project of Group 4 presents an innovative chatbot framework that combines LLMs with Retrieval-Augmented Generation (RAG) and Optical Character Recognition (OCR) technologies to optimize financial document interaction. By leveraging OCR, the system transforms scanned and image-based documents into structured, machine-readable text, enabling more effective processing. The RAG framework enhances the LLM's capabilities by dynamically retrieving contextually relevant information from specialized financial databases, thereby enriching responses with precise, domain-specific insights. Extensive evaluations conducted on diverse financial datasets demonstrate that this integrated approach not only significantly improves response accuracy and contextual relevance but also effectively tackles the challenges of handling unstructured and imbalanced financial data. This solution offers a robust tool for automating financial document analysis, ultimately enhancing decision-making and customer support in financial services.

Contents

1 Introduction

Financial documents, ranging from quarterly reports and annual statements to regulatory filings, comprise of complicated pages filled with intricate numerical data, technical footnotes, and industry-specific terminology. Manual analysis of these documents is not only time-consuming but prone to human error, while the sheer volume of data makes it challenging to identify subtle patterns or potential red flags that could signal financial distress or opportunities. Furthermore, the dynamic nature of modern business environments means that financial data quickly becomes outdated, requiring constant updates and reanalysis to maintain relevant insights for decision-making.

In order to tackle these limitations, we developed an end-to-end system for financial report analysis and query handling that leverages the advanced capabilities of Large Language Models (LLMs) [36] combined with Optical Character Recognition (OCR) [21] and Retrieval-Augmented Generation (RAG) [26]. This innovative system is designed to automate the extraction, understanding, and processing of financial documents, addressing the inefficiencies of traditional methods.

At the core of the system is OCR technology, which enables the accurate digitization of complex financial documents [29], converting unstructured data from scanned reports, PDFs, and images into machine-readable text. This text is then processed by Large Language Models, such as GPT-4o and Gemini-1.5-pro, which bring unparalleled natural language understanding and generation capabilities to the analysis. LLM can interpret complex financial terminology, extract meaningful insights, and facilitate downstream tasks like summarization, question answering, and reasoning [14] with minimal human intervention.

To enhance the accuracy and relevance of the results, our system incorporates RAG techniques. By combining the retrieval of domain-specific information from a curated knowledge base with the generative capabilities of LLM, the system ensures that the outputs are both contextually accurate and highly informative. This approach mitigates common challenges associated with LLM, such as hallucination or misinterpretation of domain-specific terms, while providing users with reliable and actionable insights.

By integrating with LLM, our system represents a significant leap forward in financial document analysis. It helps finance analysts to navigate through vast amounts of data with unprecedented efficiency, extracting relevant information, answering complex queries, and supporting strategic decision-making in real-time. This report explores the development and application of this end-to-end pipeline, highlighting its potential to automate the finance report analysis process, make it more accurate and less time-consuming for professionals.

2 Background

2.1 OCR and Vision Language Models

Optical Character Recognition (OCR) is a transformative technology that converts text from images, scanned documents, or photographs into machine-readable and editable formats. This capability is essential for digitizing printed materials, enabling efficient data storage, retrieval, and analysis across various industries.

Recent years have seen significant advancements in OCR technology, a vital area in Computer Vision. Among the prominent tools, Tesseract [27] has been a cornerstone in the field. As an open-source engine, it is widely used for extracting text from structured and semi-structured documents and supports numerous languages. Building on these advancements, PP-OCRv3 [12], part of the PaddleOCR framework, introduces cutting-edge features such as LK-PAN for detecting irregular text, scene text detection, and text recognition.

Beyond general OCR, specialized models and frameworks have emerged for structured text recognition and layout analysis, including TableTransformer [22], Surya [30], and PDF Extract Kit [31]. For structured data extraction, these models focus on understanding and extracting information from tabular data, paragraph, figure structures within documents. By combining visual features with text encoding, it effectively models the relationships inherent in these structures, facilitating accurate data extraction.

Building on advances in Large Language Models (LLMs), Vision-Language Models (VLMs) integrate visual and textual data to enable tasks like image captioning, visual question answering, and multimodal retrieval. These models leverage deep learning to align visual and linguistic information, providing a holistic understanding of data. A key milestone is the Vision Transformer (ViT) [6], which processes images as sequences of patches, similar to words in text, and has gained significant attention in recent studies [3, 15, 8]. Additionally, Contrastive Language-Image Pretraining (CLIP) models [23] use large-scale image-text pairs to align representations in a shared embedding space, enabling zero-shot transfer to diverse vision tasks without fine-tuning.

Advancements like InternVL 1.5 [5] have scaled multimodal pretraining to handle dense visual and textual data, improving the model's performance on tasks requiring hierarchical understanding of layouts. PaliGemma [4] integrates OCR and VLM capabilities, offering modular functionalities for tasks such as document layout analysis, object detection, and image-text retrieval. Its design allows for adaptability across various applications, enhancing the versatility of VLMs. DeepSeek-VL [17] combines deep learning with multimodal understanding for document-oriented tasks. It aligns visual and textual features to enable advanced information retrieval, extending the capabilities of VLMs into specialized applications. Collectively, these developments highlight the rapid evolution of Vision-Language Models, showcasing their expanding capabilities in processing and understanding complex multimodal data across diverse applications.

VLMs have transcended research and found real-world applications through integration with LLMs. Notably, OpenAI's GPT-4 [1], in its multimodal variant GPT-4o, and Google's Gemini 1.5 [25] incorporate VLM capabilities, enabling advanced image understanding alongside text processing. These integrations unlock a range of practical applications, from enhanced document analysis and visual question answering to automated medical imaging diagnostics and creative tools for generating visual content. The seamless combination of vision and language further highlights the transformative potential of multimodal AI in industries such as healthcare, education, and e-commerce.

2.2 Financial Document Understanding

Financial document understanding [20] focuses on automating the extraction, analysis, and interpretation of data from financial documents such as invoices, contracts, statements, and reports. These documents often combine complex textual content with structured formats like tables, making them challenging to process using traditional methods.

The field has evolved from rule-based systems to advanced deep learning models, leveraging pre-trained language models [32] and multi-modal techniques. These approaches integrate textual, visual, and layout features, enabling more accurate understanding of document content and structure. Modern systems are particularly adept at handling diverse formats, noisy or scanned documents, and domain-specific language.

Many studies on document understanding focus on using and Training VLM for the task document understanding. DocFormer [2] present a multi-modal transformer architecture for Visual Document Understanding (VDU), leveraging text, vision, and spatial features through a novel multi-modal self-attention layer. Pre-trained with unsupervised tasks to encourage multi-modal interaction, DocFormer achieves state-of-the-art results on multiple datasets, outperforming larger models. BROS (BERT Relying On Spatiality) [9] introduced a pre-trained language model for key information extraction (KIE), effectively combining text and layout by encoding relative text positions in 2D space, achieving strong performance without visual features. UDOP [28] represent a unified Document AI model that integrates text, image, and layout modalities using a Vision-Text-Layout Transformer, achieving state-of-the-art performance across multiple Document AI tasks, including document understanding and generation, by leveraging self-supervised pretraining and innovative task-specific objectives.

However, few studies have focused on retrieval systems for querying financial documents. [13] focus on extracting information from unstructured plain text or text PDFs, transforming the data using LLMs before storing it in a database. This research, however, overlooks the case of extracting information from images and structured resources, which are also prevalent in financial documents. In an effort to improve retrieval approaches, [35] enhances LLMs' ability to extract critical information from hybrid financial reports, employing a retrieval system with split text. However, a limitation of this approach is that it summarizes all the information in the financial report into text embeddings, including important tables, which may result in data loss, particularly in capturing numerical or tabular data essential for financial analysis. For analysis-specific tasks, [11] demonstrates that GPT-4 can outperform human financial analysts in predicting earnings changes from standardized financial statements and generate valuable insights, with trading strategies based on its predictions yielding superior performance. While this study shows great promise in predictive analysis, it does not fully address how the model handles multimodal data, such as financial statements containing both textual and tabular information, which remains an important area for future exploration.

Other retrieval approaches have been proposed for extracting relational information from financial news via knowledge graphs [24], which can serve as tools for social listening in the context of financial events and market sentiment. However, these models primarily focus on structured relationships and may not fully capture the nuanced, multimodal aspects of financial documents. Additionally, some previous studies [16, 10], which are mostly encoder-only transformer models, have been designed for sentiment classification tasks on summarized financial text, often as part of financial text mining. While these models have shown success in sentiment analysis, they tend to limit the scope of their application to simpler text-based tasks and may not leverage the full potential of more complex multimodal financial data.

Despite significant advancements, challenges remain, including the variability in document layouts, data privacy concerns, and ensuring model robustness across different industries and use cases. As reviewed, recent studies have primarily focused on extracting information from financial documents using Vision-Language Models or OCR models, but less attention has been given to deep interpretation, retrieval, and analysis of the extracted data. This gap highlights the need for more advanced methods that not only extract but also contextualize and analyze financial information effectively.

2.3 Prompt Engineering

Prompt engineering [33] is a key enabler in developing systems for financial document understanding, ensuring that Large Language Models (LLMs) can effectively interpret and process diverse financial data. Financial documents, often composed of unstructured text, numerical tables, and complex layouts, require precise instructions to guide LLMs in performing tasks such as data extraction, trend analysis, and summarization.

The complexity of financial documents necessitates detailed and context-driven prompts. For example, prompts can specify the type of document (e.g., balance sheets, income statements, or cash flow statements) and the specific analytical task to be performed, such as calculating liquidity ratios or identifying year-over-year revenue trends. This specificity allows the LLM to focus on relevant portions of the document while ignoring extraneous data, thereby improving accuracy. Moreover, prompts can account for variations in document layouts, ensuring that the system adapts to different formats effectively.

To further enhance the system's accuracy and reasoning capabilities, Chain-of-Thought (CoT) [34] prompting is integrated into the prompt engineering process. CoT is a technique that breaks down complex queries into smaller, logical steps, allowing the LLM to process and respond to each step sequentially. For instance, if a user requests an analysis of profitability over a three-year period, the CoT approach guides the LLM to extract revenue and net profit data for each year from the financial statements, calculate the profitability ratios for each year, compare the results to identify trends, and summarize the insights in a structured and user-friendly format. By decomposing tasks into manageable components, CoT ensures that the LLM does not overlook critical details and delivers more accurate and interpretable outputs. derived metrics like total equity or working capital.

Incorporating Retrieval-Augmented Generation (RAG) further enhances the system's ability to process financial documents. RAG enables the dynamic retrieval [7] of relevant information from large document repositories or databases, supplementing the LLM's contextual under -standing. CoT prompts play a critical role here by guiding the retrieval process in stages. For instance, when tasked with identifying trends in asset growth, the system can first retrieve balance sheet data for each fiscal year, then calculate year-over-year changes, and finally analyze the trends in context with other financial indicators. This stepwise reasoning ensures that the retrieved information is not only relevant but also processed in a logical sequence to produce actionable insights.

Prompt engineering also standardizes the output format of the system, ensuring clarity and consistency in responses. Financial professionals often require structured outputs, such as tabular presentations of financial ratios or concise textual explanations of trends and anomalies. By leveraging CoT, prompts can guide the system to organize information logically and present it in a format that aligns with user expectations. For instance, a CoT-enabled prompt might instruct the system to first calculate a company's current ratio, then provide a bullet-pointed explanation of its implications for liquidity, followed by recommendations for improvement.

Prompt engineering, augmented by Chain-of-Thought reasoning, is an indispensable component of systems designed for financial document understanding. CoT prompts enable the system to handle complex queries with stepwise precision [19, 18], ensuring that LLMs, OCR, and RAG work cohesively to manage the intricacies of financial data. By leveraging CoT alongside well-designed prompts, the system can extract, retrieve, and analyze financial information with exceptional accuracy and contextual relevance, meeting the diverse demands of financial analysis, reporting, and decision-making.

3 Approach

3.1 System Design

To enhance LLM-based document understanding, we have designed a comprehensive end-to-end system tailored for efficient financial report analysis and query handling. The primary objective is to extract, structure, and store all the essential information from financial reports in a database. This structured data serves as a foundation for efficiently answering user queries. When a user submits a question, the system retrieves relevant information from the database based on the query context and forwards it to the LLM. The LLM then processes this data to generate precise, context-aware responses tailored to the user's question. Notably, the LLM is not only responsible for answering the question but also plays a crucial role in intelligently querying the database by interpreting the user's intent and mapping it to the underlying structured data.

This approach ensures seamless integration between data storage, retrieval, and reasoning, enabling the system to handle complex queries. By offloading the retrieval task to the LLM, the system can dynamically adapt to diverse user questions without rigid pre-defined query structures. This design maximizes the utility of both the database and the LLM's advanced natural language understanding capabilities, creating a robust solution for financial data analysis and interactive query resolution.

Figure 1 illustrates the workflow for processing documents and supporting the chatbot. The architecture is divided into three main components: Front-End Chat, Document Processing, and Back-End Logic. Users interact with the system through a Streamlit-based Chatbot UI in the front-end chat, submitting queries that are routed to an API gateway. The API gateway acts as a mediator, forwarding user queries to the back-end for processing by the LLM. The LLM retrieves relevant information by consulting stored data in the database, using logical reasoning and FAQs, and generates responses tailored to user queries. Logical routing ensures that the appropriate path is followed for each query type.

The system workflow consists of two main phases. In the first phase, document owners upload files via a cloud-based front-end interface, which triggers the OCR process. A segmentation module extracts text and entities from the documents, and metadata is further enriched using large language models (LLMs). For text recognition in metadata and explanation processing, we employed Tesseract. However, since Tesseract performs poorly on tabular text data, we utilized Vision-Language Models (enhanced by GPT-4o and Gemini-1.5-pro) to extract information from tables. The key tables include the Balance Sheet, Income Statement, and Cash Flow Statement. The explanation content is then split into chunks, embedded, and uploaded to a vector database. Extracted content is stored in a relational database for structured data and in the VectorDB for semantic search and indexing.

In the second phase, when a user submits a query, the LLM dynamically retrieves relevant information from the database or VectorDB based on the context and intent of the question. Specifically, if the query requests information from a financial report, the LLM either generates SQL queries to fetch structured data or uses semantic search to retrieve relevant

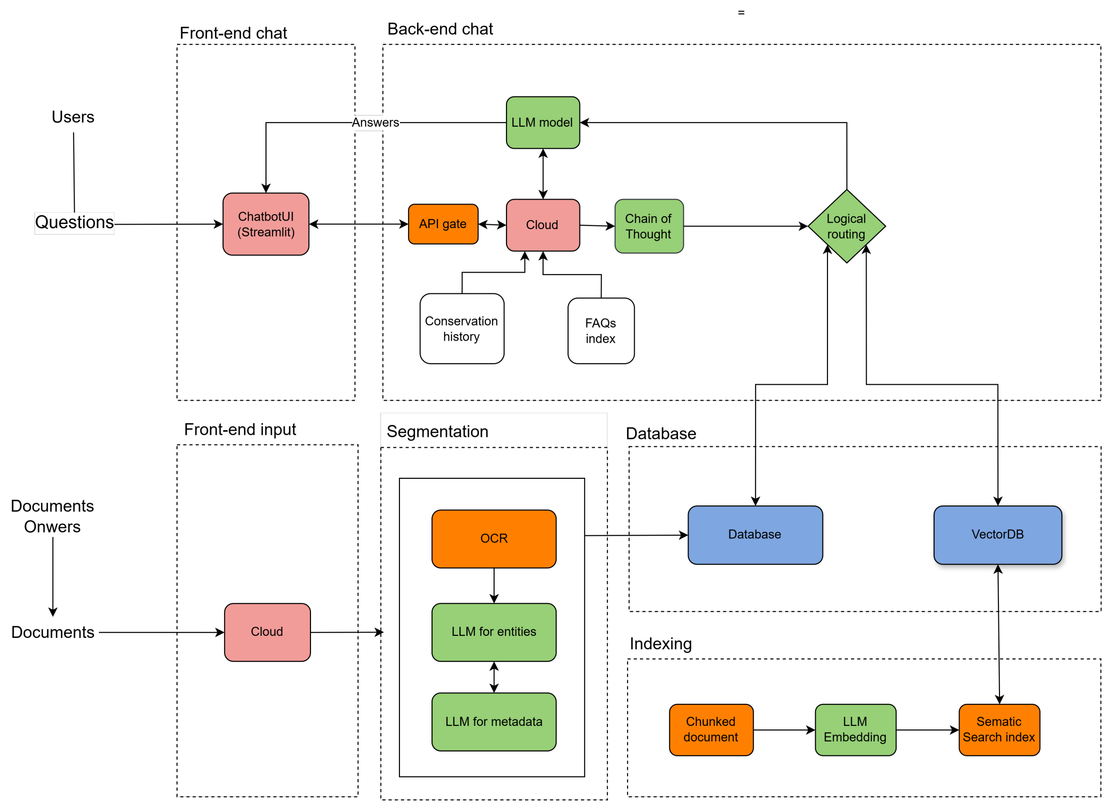


Figure 1: System design

embeddings. The retrieved information is then processed by the LLM to generate precise, contextually appropriate responses.

By separating document analysis and query handling into distinct phases, the system ensures efficient processing of financial documents and accurate real-time answers to user questions. This modular approach optimally integrates OCR, database storage, and LLM capabilities to meet diverse user needs.

The ERD design (**Figure 2**) for our database is structured to handle financial reports processed by the OCR phase. Once a financial report is uploaded and processed, three main tables—Balance Sheet, Income Statement, and Cash Flow—along with chunked explanations, are stored in the database. The financial tables store data by account name and account number, following the regulations set by the State Bank of Vietnam for Vietnamese banking standards. Additionally, a Metadata table serves as a central hub for organizing data, while a Bank table stores information about the bank. Account names are also embedded to enable efficient matching with those extracted from the financial reports during the OCR phase.

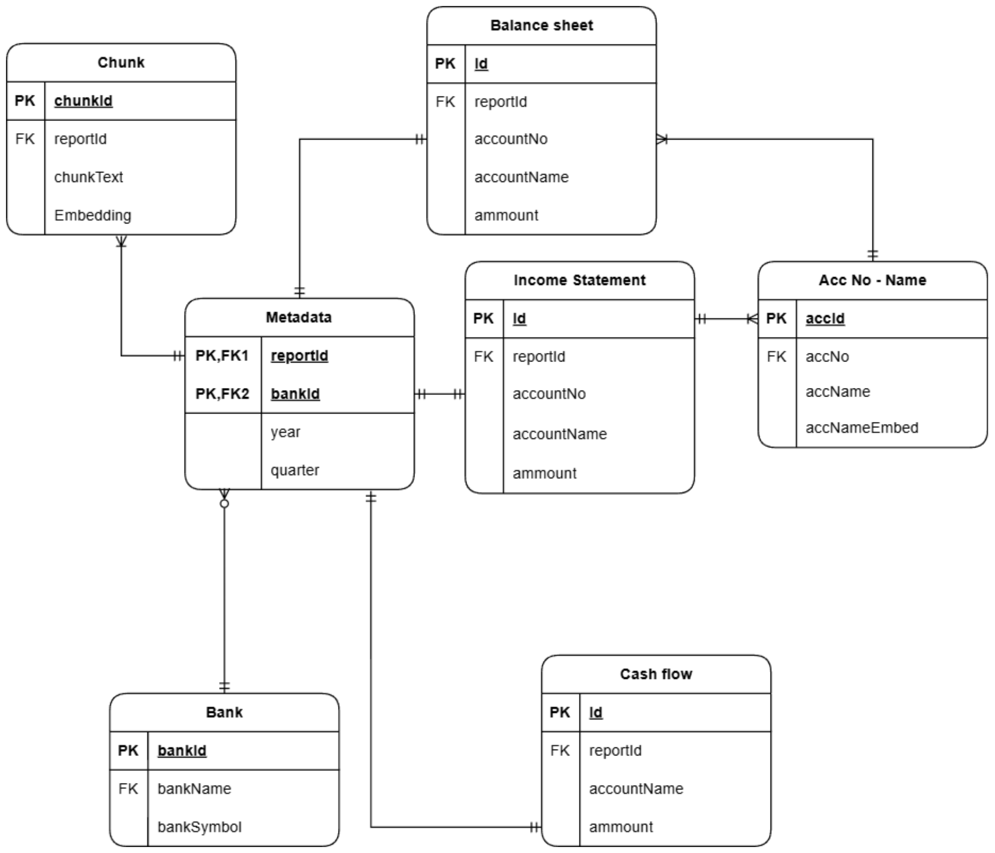


Figure 2: Database architecture ERD.

3.2 OCR

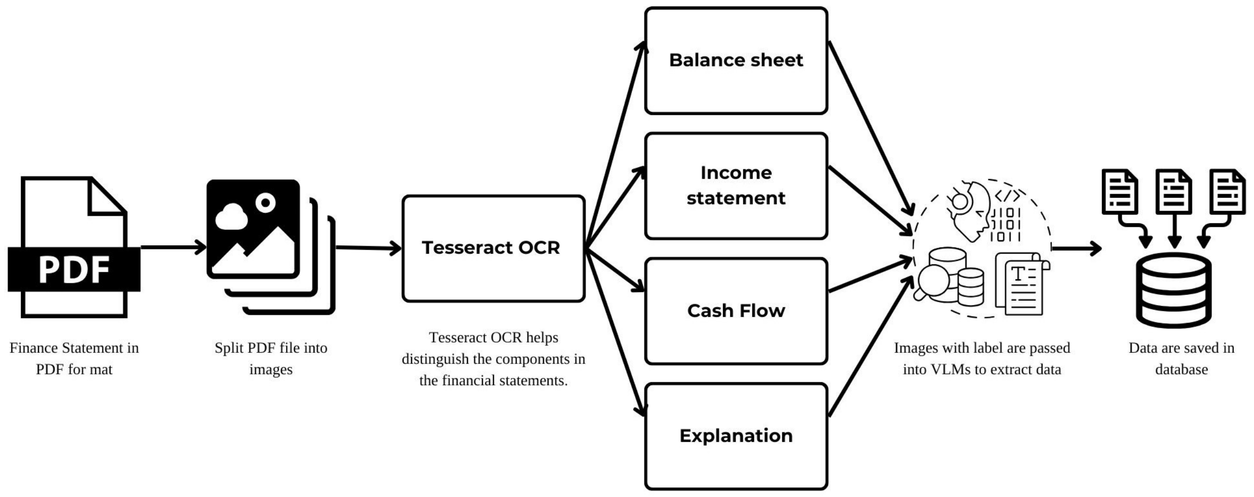

PDF
Finance Statement in PDF for mat
Split PDF file into images
Tesseract OCR
Tesseract OCR helps distinguish the components in the financial statements.
Balance sheet
Income statement
Cash Flow
Explanation
Images with label are passed into VLMs to extract data
Data are saved in database

Figure 3: OCR structure

The first step in our workflow involves splitting the PDF into individual images, one per page. We then use **Tesseract OCR** to label the data into three types: Balance Sheet, Income Statement, and Cash Flow. The main challenge is accurately interpreting tabular data within these images, as OCR tools struggle with table formats. Additionally, each bank presents financial statements differently, with significant variations in format and layout. Addressing these variations within a single semester was not feasible. Therefore, we integrated VLMs into our workflow despite the additional cost. VLMs offer a faster and more adaptable solution for interpreting diverse financial statement formats compared to traditional OCR methods. While LLMs may not always match OCR's precision in extracting numerical details, they significantly reduce manual workload and processing time, making it easier to handle variability across different financial statements. After extraction, the data is automatically imported into our database for storage and further analysis. Although this structure includes both OCR and VLM, we refer to it as the OCR structure for short, as shown in Figure **Figure 3**.

To be more specific, we prompted the VLM (see **Listing 2**) to process each page of the Balance Sheet, Income Statement, and Cash Flow Statement, extracting information and returning it in a structured JSON format. The VLM leverages its ability to treat an image as a sequence of word tokens, effectively performing OCR tasks by interpreting visual data as text. To guide the model, we provided a one-shot example of the desired JSON output format, enabling the model to generate results aligned with the structure and semantics of the financial tables it processed. This approach demonstrates the VLM's capability for multimodal reasoning, combining vision and language understanding.

Listing 1: Example prompt for VLM

```
# Example prompt
'''
Given this image of an Balance Sheet: {Image}
Extract all the information in this image, return the result of the year 2024.
Return only Json format, no further explanation, Json must be in flat format not
   nested. Expecting property name enclosed in double quotes.
Return the account names in English. Specify positive and negative numbers.
Do not use decimal number and do not add any comma or dot into the amount.
For example:
{
    "Cash and gold": amount,
    "Deposits at the State Bank": amount,
    "Deposits at and loans to other credit institutions": amount,
    "Deposits at other credit institutions": amount,
    "Loans to other credit institutions": amount,
    "Provision for deposits at and loans to other credit institutions": amount,
    "Trading securities": amount,
    ...
}
'''
```

After the model generates results from the financial tables, a key issue arises: account names may vary because the VLM, as a generative model, cannot be explicitly constrained to produce standardized account names. To address this, we established a predefined set of standard account names and stored them in a database, along with their corresponding numerical index and embeddings. Once the VLM produces the JSON output, we apply semantic search to match the generated account names with the saved standard account names. However, during experimentation, we observed that financial account names (e.g., "Deposits at other credit institutions," "Loans to other credit institutions") are often short, highly specialized terms. General-purpose embedding models, such as OpenAI's text-embedding-3 or Google's text-embedding-004, fail to effectively capture the nuanced meanings of such domain-specific terms. To resolve this, we utilized voyage-finance-2, a domain-specific embedding model from VoyageAI, which is fine-tuned for financial terminology. This customization significantly improved the accuracy of semantic matching, addressing the challenges posed by the variability in account names.

3.3 Retrieval-Augmented Generation (RAG)

Following the extraction and storage of financial report information into the database, as outlined in the previous section, the subsequent phase involves the application of prompt engineering and Retrieval-Augmented Generation (RAG). This method is employed to develop a robust system that enables users to interact with the database effectively, facilitating precise and efficient retrieval of the required financial report information. The workflow of this process is illustrated in **Figure 4**

To optimize the generation of SQL queries for complex, multi-query requests, we instruct the agent to decompose the problem into a list of simpler tasks. By iterating through this list, we can obtain the desired simple SQL queries instead of forcing the agent to create a highly complex SQL query for a multi-query request. This involves extracting essential information such as company names, financial periods, and indicators from the user's query and formulating SQL queries based on the database structure and the identified details. After then, to ensure the accuracy of the SQL queries, LLM will explain and align with the query requirements. Additionally, integrating the database schema and a predefined list of account names and bank-mapping is vital, as it enables the system to interpret user queries accurately within the context of the specific database structure. This workflow ensures precision, efficiency, and comprehensive satisfaction of financial information retrieval needs when compared with the basic Chain-of-Thought technique.

When the agent returns a list of SQL queries finally, they execute them to get their corresponding results. Finally, based on these results and the initial query, the agent provides a comprehensive answer to the user by combining the results from the individual SQL queries.

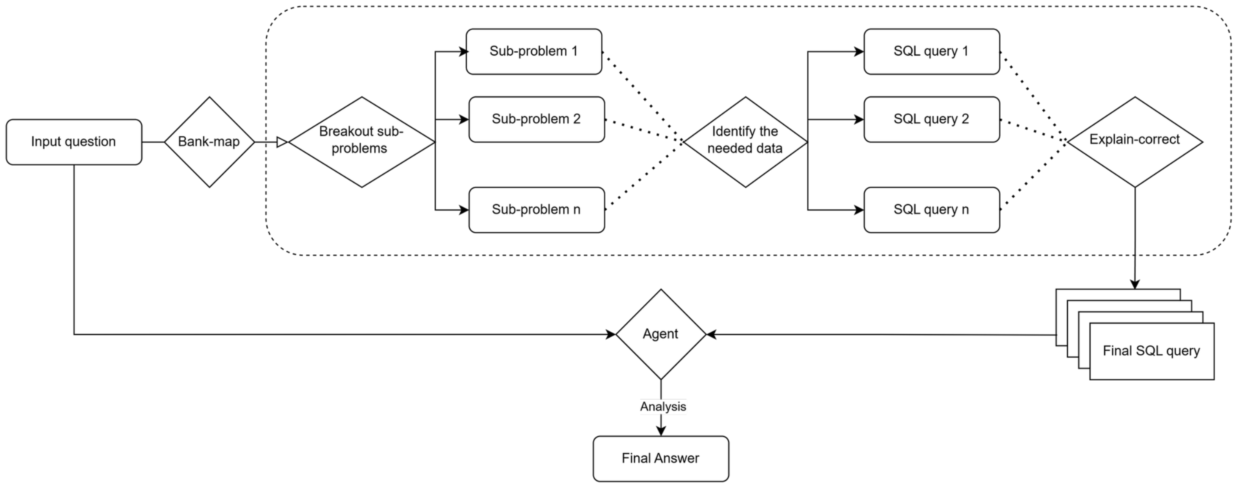


Figure 4: Text2SQL pineline

3.4 Deployment

For the deployment of the database and server, we leverage Azure services, configuring the server in the East US 2 region to meet our specific needs. The chosen configuration is Burstable B1ms, which includes 1 vCore, 2 GiB of RAM, and 32 GiB of storage. This setup is optimal for managing our database and processing workloads while maintaining flexibility in resource allocation. The server ensures that the application can scale dynamically based on demand, providing the necessary performance for storing and retrieving financial data.

For the frontend, we utilize Streamlit Community Cloud, an efficient and easy-to-use platform for deploying interactive web applications. Streamlit's integration capabilities allow us to connect the frontend directly with the backend, creating a seamless and responsive user experience. This deployment method ensures that users can interact with financial reports in real-time, facilitating smooth data querying and visualization. By using Streamlit Cloud, we can focus on building and improving the application without worrying about infrastructure management, making it a cost-effective and scalable solution for the frontend.

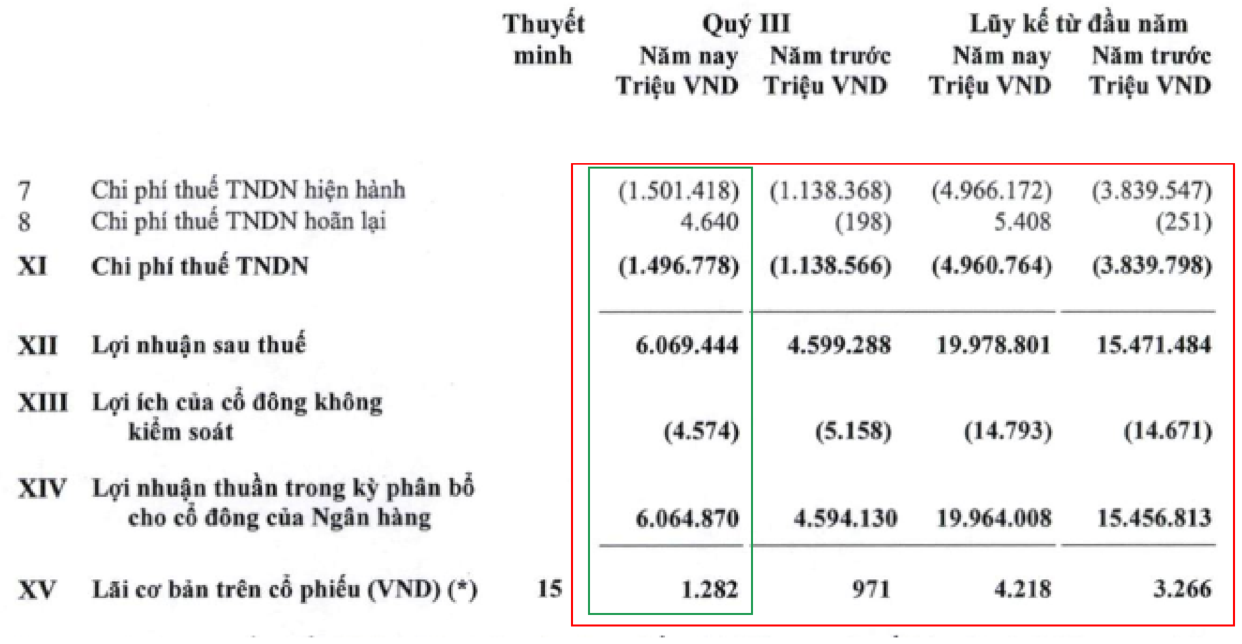
4 Evaluation

4.1 OCR

When working with Vision-Language Models (VLMs) for data extraction, it is crucial to acknowledge their limitations compared to Optical Character Recognition (OCR) systems, particularly in terms of extracting structured data with high accuracy. OCR systems are specialized for such tasks and often deliver more consistent results. However, VLMs require fewer processing steps and are easier to implement compared to OCR systems, making them a more convenient option for complex use cases.

Despite their potential, VLMs require significant manual verification to determine the quality of input data. This manual process is necessary to address the limitations of VLMs in accurately extracting structured data, especially when the input quality varies or the data involves intricate tabular formats.

In our evaluations, we focused on two prominent VLMs: ChatGPT-4o and Gemini-1.5pro. Both models have shown promise, but neither is perfect, and both exhibit notable challenges during data extraction tasks. One of the most frequent issues observed during testing was related to errors in column alignment, where data was mistakenly extracted from the wrong columns. This misalignment can significantly affect the usability and reliability of the extracted information, especially when dealing with financial statements containing complex tabular data **Figure 5**.

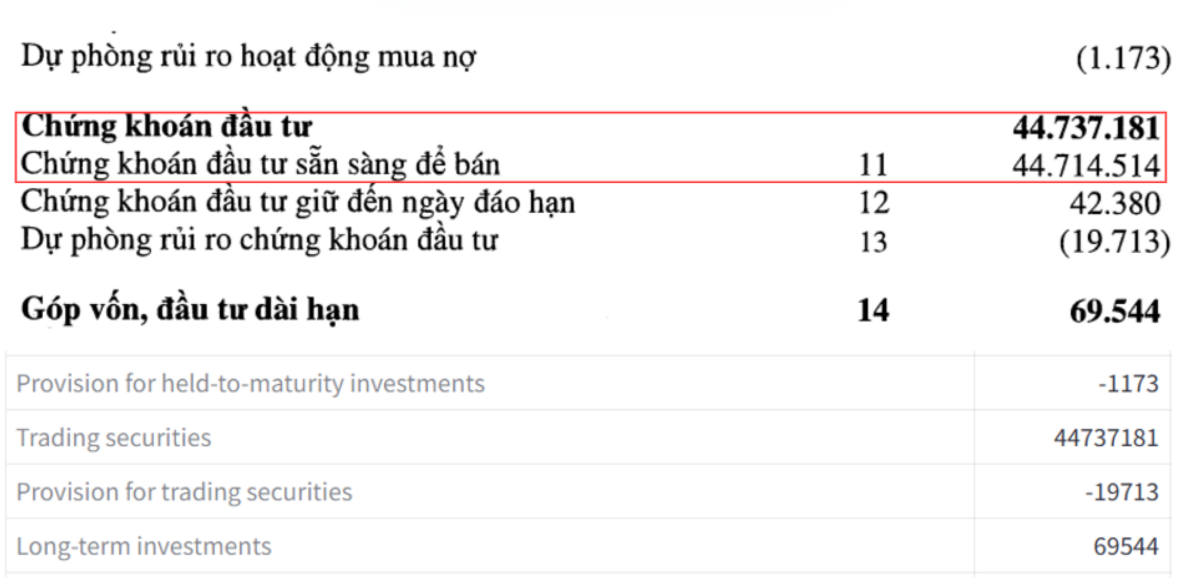


		Thuyết minh	Quý III		Lũy kế từ đầu năm	
			Năm nay Triệu VND	Năm trước Triệu VND	Năm nay Triệu VND	Năm trước Triệu VND
7	Chi phí thuế TNDN hiện hành		(1.501.418)	(1.138.368)	(4.966.172)	(3.839.547)
8	Chi phí thuế TNDN hoãn lại		4.640	(198)	5.408	(251)
XI	**Chi phí thuế TNDN**		**(1.496.778)**	**(1.138.566)**	**(4.960.764)**	**(3.839.798)**
XII	**Lợi nhuận sau thuế**		**6.069.444**	**4.599.288**	**19.978.801**	**15.471.484**
XIII	**Lợi ích của cổ đông không kiểm soát**		**(4.574)**	**(5.158)**	**(14.793)**	**(14.671)**
XIV	**Lợi nhuận thuần trong kỳ phân bổ cho cổ đông của Ngân hàng**		**6.064.870**	**4.594.130**	**19.964.008**	**15.456.813**
XV	**Lãi cơ bản trên cổ phiếu (VND) (*)**	**15**	**1.282**	**971**	**4.218**	**3.266**

Figure 5: The green rectangle highlights the correct column, while the red rectangle shows the columns extracted by the VLMs.

Another common issue was the extraction of incorrect data values, which occurred more frequently in ChatGPT-4o. While this model performed slightly better overall when assessed across a wide variety of datasets, its tendency to produce value extraction errors is a notable drawback, especially for applications demanding high precision in financial computations.

While ChatGPT-4o faces challenges with the accuracy of extracted data values, Gemini-1.5pro encounters a different issue, specifically related to the extraction of missing data. This means that Gemini-1.5pro occasionally fails to capture certain information present in the input, which can lead to incomplete datasets **Figure 6**.



Dự phòng rủi ro hoạt động mua nợ		(1.173)
Chứng khoán đầu tư		**44.737.181**
Chứng khoán đầu tư sẵn sàng để bán	11	44.714.514
Chứng khoán đầu tư giữ đến ngày đáo hạn	12	42.380
Dự phòng rủi ro chứng khoán đầu tư	13	(19.713)
Góp vốn, đầu tư dài hạn	**14**	**69.544**

Provision for held-to-maturity investments	-1173
Trading securities	44737181
Provision for trading securities	-19713
Long-term investments	69544

Figure 6: An example of extracting missing data. The lines in the red rectangle are not extracted.

To evaluate the performance of these models, we analyzed a sample of 10 randomly selected financial statements, summarizing the results in **Table 1** and **Table 2**. The comparison revealed key differences in their strengths and weaknesses. ChatGPT-4o excels in extracting a complete dataset, capturing more information overall. However, it is more prone to extracting incorrect values, which is a concern when accuracy is crucial. On the other hand, Gemini-1.5pro is better at extracting correct values with fewer mistakes, though it sometimes misses data, leading to incomplete results. Additionally, Gemini-1.5pro is more cost-effective, making it a suitable choice for processing large volumes of data.

VLM	Table	Accurate	Total	Accuracy	Avg.
GPT-4o	Balance Sheet	514	532	0.966	**0.969**
	Income Statement	216	221	0.977*	
	Cash Flow	321	332	0.967	
Gemini-1.5 -pro	Balance Sheet	490	494	0.992*	**0.964**
	Income Statement	207	230	0.9	
	Cash Flow	338	349	0.968	

Table 1: Tabular Reader Performance Metrics Using VLMs.

VLM	Table	Prompt tokens	Output tokens	Total	Cost (vnd)
GPT-4o	Balance Sheet	1732	634	2366	266.75
	Income Statement	883	252	1135	118.19
	Cash Flow	1710	483	2193	218.52
					603.46
Gemini-1.5 -pro	Balance Sheet	862	1090	1952	163.18
	Income Statement	405	417	822	64.78
	Cash Flow	834	713	1547	115.18
					343.14

Table 2: VLMs Token count and Cost Evaluation for OCR task.

The **Accuracy** column represents the ratio of the number of correctly extracted data points (**Accurate**) to the total number of extractions (**Total**). Based on these findings, we selected ChatGPT-4o due to its superior overall performance.

4.2 RAG

We collected a diverse set of 100 questions from various sources to ensure a comprehensive evaluation. These questions cover a wide range of topics and complexities, from simple queries involving individual tables to more complex ones requiring combined table data. This variety provides a solid foundation for assessing the performance of our Retrieval-Augmented Generation (RAG) system. A detailed list of these questions is included in **Appendix 3**.

To evaluate the results, we manually compared the SQL queries generated by different approaches due to the domain-specific nature and complexity of some questions. We defined two key criteria for accepting a query:

- **Completeness**: The SQL query must be executable.
- **Relevance**: Given the domain-specific challenges, perfect accuracy may not be possible, but the results must be relevant and appropriately aligned with the query requirements.

Using these criteria, we evaluated the 100 questions and created a comparison table that includes results from different methods: GPT-4o-mini, GPT-4o-mini with Chain-of-Thought (COT), and our proposed GPT-4o-mini with COT and Explainable. To ensure a thorough assessment, we categorized the questions based on their complexity and nature. This categorization allowed us to evaluate each method's performance across different question types, helping us better understand the strengths and weaknesses of each approach in **Table 3**.

Overall, our solution achieved the highest acceptance rate at 75%, followed by the GPT-4o-mini-CoT model at 42%, and finally, the GPT-4o-mini model at 23%. This demonstrates that the integration of techniques and optimizations in our solution yields significant effectiveness.

Most tables, such as Account, Bank and Metadata, achieved relatively high accuracy. However, there were notable shortcomings in the GPT-4o-mini model, with acceptance rates of only 40% and 66.67%, respectively. For the GPT-4o-mini-CoT model, these values improved to 60% and 66.67%. Notably, our solution excelled with a 100% acceptance rate** for both table types.

Conversely, Explainable table groups, all three models performed poorly. Both GPT-4o-mini and GPT-4o-mini-CoT showed no significant acceptance (0%). Even our solution requires further improvement in these groups, despite some progress.

Regarding the three main tables—Balance Sheet, Income Statement, and Cash Flow—the results reveal a clear trend of improvement. For GPT-4o-mini, acceptance rates ranged from 15% to 35%. With GPT-4o-mini-CoT, the rates increased to 40% to 55%. Particularly, our solution achieved acceptance rates between 70% and 90%, showcasing superior performance compared to the other two models.

VLM	**Prompt tokens**	**Output tokens**	**Total**	**Cost (vnd)**
GPT-4o-mini	169.34	57.76	227.1	24.06
GPT-4o-mini-Cot	4414.06	29.39	4443.45	348.03
Our Proposal	**5409.04**	**114.56**	**5603.6**	**593.87**

Table 4: LLMs Token Count and Cost Evaluation.

The table highlights the average token usage for each model, reflecting differences in computational effort. Our Proposal stands out with the highest average usage of 5409.04 prompt tokens and 114.56 output tokens, totaling 5603.6 tokens per task. In comparison, GPT-4o-mini-CoT averages 4414.06 prompt tokens and 29.39 output tokens, totaling 4443.45 tokens. Meanwhile, GPT-4o-mini is the most resource-efficient, with just 169.34 prompt tokens and 57.76 output tokens, for a total of 227.1 tokens.

Model	Table Type	Executable	Relevance	Acceptability
GPT-4o-mini	Account	3/5	2/5	0.4
	Metadata	5/6	4/6	0.67
	Explainable	5/5	0/5	0
	Bank	4/4	0/4	0
	Balance Sheet	15/20	3/20	0.15
	Income Statement	17/20	4/20	0.2
	Cash Flow	16/20	3/20	0.15
	Compound Tables	16/20	7/20	0.35
	Total	81/100	23/100	0.23
GPT-4o-mini -CoT	Account	5/5	3/5	0.6
	Metadata	6/6	4/6	0.67
	Explainable	5/5	0/5	0
	Bank	3/4	0/4	0
	Balance Sheet	20/20	8/20	0.4
	Income Statement	19/20	8/20	0.40
	Cash Flow	19/20	8/20	0.4
	Compound Tables	20/20	11/20	0.55
	Total	97/100	42/100	0.42
Our proposal	Account	5/5	5/5	1
	Metadata	6/6	6/6	1
	Explainable	3/5	0/5	0
	Bank	4/4	3/4	0.75
	Balance Sheet	20/20	18/20	0.9
	Income Statement	19/20	15/20	0.75
	Cash Flow	20/20	14/20	0.7
	Compound Tables	18/20	14/20	0.2
	Total	**96/100**	**75/10**	**0.75**

Table 3: Evaluation of Models on Completeness, Relevance, and Acceptability for Table Data.

When combined with the acceptability and relevance evaluation, Our Proposal consistently outperforms the other models, achieving the highest acceptability score (75%), particularly excelling in core table types like Balance Sheet, Income Statement, and Cash Flow, with acceptability rates of 70–90%. While Gemini-4o-mini-CoT demonstrates moderate improvements in token usage efficiency and performance, GPT-4o-mini sacrifices depth and accuracy for minimal computational demands, achieving only 23% acceptability. These results underscore the trade-off between efficiency and effectiveness, with Our Proposal providing the most comprehensive and reliable outputs, albeit at a higher computational cost.

Listing 2: Example prompt for LLM

```
# Example our Chain-of-thought:
'''
You are a financial analyst. Analyze the question below and follow the steps to
    answer it accurately and in detail.

**Question**: "{question}"

### Step 1: Identify the type of problem to be analyzed and break it down

- **Translate the question into English**.

- **Analyze the requirements** and break down the problem: [Identify specific
    problems to be analyzed from the question, such as financial ratios or
    related asset information].

- Create a list of specific problems to solve, for example:

- ['Find the profit after tax and total capital, then calculate the ROE ratio
    and find the bank with the largest ROE in the second quarter of 2024',

- 'Find the total assets of ACB in the second quarter of 2024'].

### Step 2: Determine the basic information to query from the problem in Step 1
- **Convert the bank name to symbol in the BANK table**:
- Use the list that has been mapped from the bank name to symbol:
    {self.query_bank_to_symbol(question)}.
- Consider the approximate symbols to determine the required bank Symbol.
- **Fiscal year**: [Example: 2024].
- **Reporting period**: [Example: Q1, Q2, Q3, Q4].
- **Specific item in the report**: [Based on the subdivided problem, clearly
    identify the item to query, for example: revenue, net profit, ROE].
- **Financial report type**: [Clearly identify the required report type, for
    example: balance sheet, income statement].

### Step 3: Create SQL queries for each problem
- For each problem from Step 1 and the information from Step 2, create a
    corresponding SQL query.
- **Database structure**: {self.db_structure}.
- **List of available account names**: {json.dumps(self.acc_name)}.
- Example query: {self.load_example()}.

### Step 4: Explain and check for SQL query errors
- **Explain each SQL query**:
```

```
- Describe how the query is executed and why it fits the problem requirements.
- **Check again**:
- Compare the query with the question requirements, determine if it is correct.

- If incorrect, correct the query by going back to Step 1.

### Step 5: Record the final SQL query

After ensuring the queries are correct:

Record the SQL query in the format:
‘‘‘sql
[Correct SQL query]
‘‘‘
- **Note**:

- Only use data from the connected database.
- For abbreviated bank names, prioritize ‘symbol‘ or ‘abbreviation‘ if necessary.
- Use the ‘ACCNO‘ table to find a common account number when there is no
    specific time/bank information.
- For listed questions, there is no limit to the number of results.
Please do it sequentially from Step 1 and answer in Vietnamese.

"""
```

Listing 3: Example prompt for LLM with 2 questions at the same time.

```
# Example our Chain-of-thought:
’’’
### Question:
1. Find the bank with the highest ROE in the second quarter of 2024?
2. Find the total assets of MBB in the fourth quarter of 2023?

### SQL Queries and Results:
1. **First SQL Query**:
   ‘‘‘sql
   SELECT
       "BANK"."bankname",
       ("INCOMESTATEMENT"."amount" / NULLIF("BALANCESHEET"."amount", 0)) * 100 AS
           "ROE"
   FROM
       "METADATA"
   JOIN
       "BANK" ON "METADATA"."bankid" = "BANK"."bankid"
   JOIN
       "INCOMESTATEMENT" ON "METADATA"."reportid" = "INCOMESTATEMENT"."reportid"
   JOIN
```

```
        "BALANCESHEET" ON "METADATA"."reportid" = "BALANCESHEET"."reportid"
    WHERE
        "METADATA"."year" = 2024
        AND "METADATA"."quarter" = 2
        AND "INCOMESTATEMENT"."accountname" = 'Profit after tax'
        AND "BALANCESHEET"."accountname" = 'Total equity'
    ORDER BY
        "ROE" DESC
    LIMIT 1;
    ```
    SQL Result:
    - Result: `[('NHTMCP Ky Thuong Viet Nam', Decimal('17.67024387006810235000'))]`
    - Interpretation: The bank with the highest ROE in the second quarter of 2024
        is "NHTMCP Ky Thuong Viet Nam" with an ROE of approximately 17.67%.

2. **Second SQL Query**:
    ```sql
    SELECT
        "BANK"."bankname",
        "BALANCESHEET"."amount"
    FROM
        "METADATA"
    JOIN
        "BANK" ON "METADATA"."bankid" = "BANK"."bankid"
    JOIN
        "BALANCESHEET" ON "METADATA"."reportid" = "BALANCESHEET"."reportid"
    WHERE
        "METADATA"."year" = 2023
        AND "METADATA"."quarter" = 4
        AND "BANK"."symbol" = 'MBB'
        AND "BALANCESHEET"."accountname" = 'Total assets'
    LIMIT 1;
    ```
    SQL Result:
    - Result: `['']`
    - Interpretation: There is no result returned for the total assets of the bank
        with the symbol 'MBB' for the fourth quarter of 2023. This could indicate
        that either the data does not exist or there was an issue retrieving it.

### Final Answer:
1. The bank with the highest Return on Equity (ROE) in the second quarter of
    2024 is **NHTMCP Ky Thuong Viet Nam** with an ROE of **17.67%**.
2. Unfortunately, there is **no available data** on the total assets of **MBB**
    for the fourth quarter of 2023. This could suggest that the required
    financial data might not have been reported or captured in the database.
'''
```
```

Conclusion and Future Work

In conclusion, the proposed system successfully combines OCR, Vision-Language Models (VLMs), and Large Language Models (LLMs) to streamline financial report analysis and interactive query handling. By dividing the process into document processing and query resolution phases, the system ensures efficient data extraction and accurate responses. Its hybrid use of relational and vector databases enhances its ability to manage structured and semantic data effectively.

The integration of VLMs with tailored prompt engineering addresses challenges in extracting data from complex financial documents, ensuring reliable storage and retrieval. The Retrieval-Augmented Generation (RAG) framework further improves query accuracy by decomposing complex questions and generating precise SQL queries. This modular approach ensures high accuracy and adaptability to financial domain-specific tasks.

The deployment leverages scalable technologies like Azure and Streamlit, delivering a cost-effective, user-friendly solution. Overall, the system represents a significant step forward in automating financial analysis, offering a robust, scalable, and innovative framework for future advancements in the field.

In our future work, we aim to transition from relying on vision-language models (such as GPT-4o) to implementing an OCR-based approach for extracting and understanding financial data. Currently, VLMs have been utilized due to time constraints, allowing us to process Income Statements, Balance Sheets, and Cash Flow Statements efficiently. However, OCR provides a more reliable and accurate method for extracting structured data, which is critical for ensuring precision in financial document analysis.

By integrating OCR into our pipeline, we plan to enhance the robustness of data extraction across all three financial tables. This shift will not only improve the accuracy of the extracted information but also reduce dependency on pre-trained language models, allowing for greater flexibility in handling diverse document formats and layouts. Future efforts will focus on optimizing the OCR process and validating its performance for financial report understanding.

In future work, we aim to improve both the database structure and the text-to-SQL conversion to better handle complex user queries. While the current text-to-SQL system performs well for simpler queries, it struggles with more intricate questions, potentially due to limitations in the existing database design. Enhancing the structure of the database will ensure better alignment with the query logic, enabling more accurate and context-aware SQL generation.

Moreover, as we integrate OCR into the pipeline for financial document understanding, we will leverage the extracted structured data to refine query processing further. This integration will allow for more precise mapping between extracted information and database schemas, enhancing the system's capability to process complex queries. These improvements will ultimately provide a more robust and reliable solution for interacting with financial data.

Acknowledgements

We express our deepest gratitude to those who contributed to the success of this project. This endeavor would not have been possible without the support, guidance, and encouragement from various individuals and organizations.

Firstly, we sincerely thank Dr. Luong Van Thien for his invaluable guidance, insightful feedback, and unwavering support throughout this project. His expertise and dedication significantly enriched our understanding and contributed to the completion of our intensive project.

We would like to express our heartfelt gratitude to Quang Hung Nguyen, Phuong Anh Trinh for their invaluable support and encouragement throughout the course of this project. Their assistance greatly contributed to the quality and clarity of our work. The willingness of Hung and Phuong Anh to lend a helping hand, share insights, and motivate us was truly inspiring and deeply appreciated.

Additionally, we are grateful to the developers and maintainers of the tools and frameworks we relied upon, including but not limited to OCR technologies, Large Language Models, and Retrieval-Augmented Generation systems. These technological advancements enabled us to realize the innovative solutions outlined in our report.

Lastly, we thank our families and friends for their encouragement and understanding during this project. Their support kept us motivated through the demanding periods of research and development.

To all who played a role, however small, in the success of this project, we extend our heartfelt appreciation.

References

[1] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

[2] Srikar Appalaraju, Bhavan Jasani, Bhargava Urala Kota, Yusheng Xie, and R Manmatha. Docformer: End-to-end transformer for document understanding. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 993–1003, 2021.

[3] Anurag Arnab, Mostafa Dehghani, Georg Heigold, Chen Sun, Mario Lučić, and Cordelia Schmid. Vivit: A video vision transformer. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 6836–6846, 2021.

[4] Lucas Beyer, Andreas Steiner, André Susano Pinto, Alexander Kolesnikov, Xiao Wang, Daniel Salz, Maxim Neumann, Ibrahim Alabdulmohsin, Michael Tschannen, Emanuele Bugliarello, et al. Paligemma: A versatile 3b vlm for transfer. *arXiv preprint arXiv:2407.07726*, 2024.

[5] Zhe Chen, Weiyun Wang, Hao Tian, Shenglong Ye, Zhangwei Gao, Erfei Cui, Wenwen Tong, Kongzhi Hu, Jiapeng Luo, Zheng Ma, et al. How far are we to gpt-4v? closing the gap to commercial multimodal models with open-source suites. *arXiv preprint arXiv:2404.16821*, 2024.

[6] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale, 2021.

[7] Mats Finsås and Joachim Maksim. Optimizing rag systems for technical support with llm-based relevance feedback and multi-agent patterns. Master's thesis, NTNU, 2024.

[8] Kai Han, Yunhe Wang, Hanting Chen, Xinghao Chen, Jianyuan Guo, Zhenhua Liu, Yehui Tang, An Xiao, Chunjing Xu, Yixing Xu, et al. A survey on vision transformer. *IEEE transactions on pattern analysis and machine intelligence*, 45(1):87–110, 2022.

[9] Teakgyu Hong, Donghyun Kim, Mingi Ji, Wonseok Hwang, Daehyun Nam, and Sungrae Park. Bros: A pre-trained language model focusing on text and layout for better key information extraction from documents. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 10767–10775, 2022.

[10] Allen H Huang, Hui Wang, and Yi Yang. Finbert: A large language model for extracting information from financial text. *Contemporary Accounting Research*, 40(2):806–841, 2023.

[11] Alex Kim, Maximilian Muhn, and Valeri Nikolaev. Financial statement analysis with large language models. *arXiv preprint arXiv:2407.17866*, 2024.

[12] Chenxia Li, Weiwei Liu, Ruoyu Guo, Xiaoting Yin, Kaitao Jiang, Yongkun Du, Yuning Du, Lingfeng Zhu, Baohua Lai, Xiaoguang Hu, et al. Pp-ocrv3: More attempts for the improvement of ultra lightweight ocr system. *arXiv preprint arXiv:2206.03001*, 2022.

[13] Huaxia Li, Haoyun Gao, Chengzhang Wu, and Miklos A Vasarhelyi. Extracting financial data from unstructured sources: Leveraging large language models. *Journal of Information Systems*, pages 1–22, 2023.

[14] Yinheng Li, Shaofei Wang, Han Ding, and Hang Chen. Large language models in finance: A survey. In *Proceedings of the fourth ACM international conference on AI in finance*, pages 374–382, 2023.

[15] Xinyu Liu, Houwen Peng, Ningxin Zheng, Yuqing Yang, Han Hu, and Yixuan Yuan. Efficientvit: Memory efficient vision transformer with cascaded group attention. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14420–14430, 2023.

[16] Zhuang Liu, Degen Huang, Kaiyu Huang, Zhuang Li, and Jun Zhao. Finbert: A pre-trained financial language representation model for financial text mining. In *Proceedings of the twenty-ninth international conference on international joint conferences on artificial intelligence*, pages 4513–4519, 2021.

[17] Haoyu Lu, Wen Liu, Bo Zhang, Bingxuan Wang, Kai Dong, Bo Liu, Jingxiang Sun, Tongzheng Ren, Zhuoshu Li, Hao Yang, et al. Deepseek-vl: towards real-world vision-language understanding. *arXiv preprint arXiv:2403.05525*, 2024.

[18] Qing Lyu, Shreya Havaldar, Adam Stein, Li Zhang, Delip Rao, Eric Wong, Marianna Apidianaki, and Chris Callison-Burch. Faithful chain-of-thought reasoning. *arXiv preprint arXiv:2301.13379*, 2023.

[19] Aman Madaan and Amir Yazdanbakhsh. Text and patterns: For effective chain of thought, it takes two to tango. *arXiv preprint arXiv:2209.07686*, 2022.

[20] Ahmed Masry and Amir Hajian. Longfin: A multimodal document understanding model for long financial domain documents. *arXiv preprint arXiv:2401.15050*, 2024.

[21] Rishabh Mittal and Anchal Garg. Text extraction using ocr: a systematic review. In *2020 second international conference on inventive research in computing applications (ICIRCA)*, pages 357–362. IEEE, 2020.

[22] Ahmed Nassar, Nikolaos Livathinos, Maksym Lysak, and Peter Staar. Tableformer: Table structure understanding with transformers, 2022.

[23] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR, 2021.

[24] Pawan Kumar Rajpoot and Ankur Parikh. Gpt-finre: In-context learning for financial relation extraction using large language models. *arXiv preprint arXiv:2306.17519*, 2023.

[25] Machel Reid, Nikolay Savinov, Denis Teplyashin, Dmitry Lepikhin, Timothy Lillicrap, Jean-baptiste Alayrac, Radu Soricut, Angeliki Lazaridou, Orhan Firat, Julian Schrittwieser, et al. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *arXiv preprint arXiv:2403.05530*, 2024.

[26] Spurthi Setty, Harsh Thakkar, Alyssa Lee, Eden Chung, and Natan Vidra. Improving retrieval for rag based question answering models on financial documents. *arXiv preprint arXiv:2404.07221*, 2024.

[27] Ray Smith. An overview of the tesseract ocr engine. In *Ninth international conference on document analysis and recognition (ICDAR 2007)*, volume 2, pages 629–633. IEEE, 2007.

[28] Zineng Tang, Ziyi Yang, Guoxin Wang, Yuwei Fang, Yang Liu, Chenguang Zhu, Michael Zeng, Cha Zhang, and Mohit Bansal. Unifying vision, text, and layout for universal document processing. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 19254–19264, 2023.

[29] Sinh Van Nguyen, Dung Anh Nguyen, and Lam Son Quoc Pham. Digitalization of administrative documents a digital transformation step in practice. In *2021 8th NAFOSTED Conference on Information and Computer Science (NICS)*, pages 519–524. IEEE, 2021.

[30] Paruchuri Vik. Surya. `https://github.com/VikParuchuri/surya.git`, 2024.

[31] Bin Wang, Chao Xu, Xiaomeng Zhao, Linke Ouyang, Fan Wu, Zhiyuan Zhao, Rui Xu, Kaiwen Liu, Yuan Qu, Fukai Shang, et al. Mineru: An open-source solution for precise document content extraction. *arXiv preprint arXiv:2409.18839*, 2024.

[32] Haifeng Wang, Jiwei Li, Hua Wu, Eduard Hovy, and Yu Sun. Pre-trained language models and their applications. *Engineering*, 25:51–65, 2023.

[33] Jiaqi Wang, Zhengliang Liu, Lin Zhao, Zihao Wu, Chong Ma, Sigang Yu, Haixing Dai, Qiushi Yang, Yiheng Liu, Songyao Zhang, et al. Review of large vision models and visual prompt engineering. *Meta-Radiology*, page 100047, 2023.

[34] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

[35] Chongjian Yue, Xinrun Xu, Xiaojun Ma, Lun Du, Hengyu Liu, Zhiming Ding, Yanbing Jiang, Shi Han, and Dongmei Zhang. Leveraging llms for kpis retrieval from hybrid long-document: A comprehensive framework and dataset. *arXiv preprint arXiv:2305.16344*, 2023.

[36] Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, et al. A survey of large language models. *arXiv preprint arXiv:2303.18223*, 2023.