Final Project Report

Group 27

Introduction

We focused on applying advanced AI techniques, specifically, Retrieval-Augmented Generation (RAG) and Large Language Models (LLM), to AI CUP 2024 Yushan Artificial Intelligence Open Challenge to address financial question answering. The primary objective of our project was to develop a system capable of identifying and retrieving the most relevant document numbers in response to specific queries from a given set of PDF documents. The documents were categorized into three main types: insurance, finance, and FAQ.

Through careful analysis, we observed that the finance documents presented unique challenges compared to the other categories. Many of these documents were not composed of plain text but contained various financial symbols, complex tables, and embedded explanatory text. This made it difficult to extract meaningful information from them using conventional text-processing methods.

Consequently, we decided to focus on two key areas in our project: first, improving text understanding for the insurance and FAQ categories, which primarily involved working with plain text, and second, enhancing the text recognition capabilities for the finance category, which involved dealing with financial symbols, complex tables, and mixed-format content.

To address these challenges, we leveraged RAG, deep learning techniques, and various open-source resources. By combining these methods, our goal was to enhance the accuracy of text retrieval and the relevance of document identification for financial queries, ultimately improving the performance of our AI-based solution in the context of competition.

Related Works

In understanding the challenges we faced in financial statement analysis, we focused on improving the reading and interpretation of financial reports. In addition to enhancing text recognition, we incorporated regularization techniques and word substitution rules to streamline the process and improve accuracy. Below are some of the key references we consulted in our research.

I. BM25, LLMs, RAG, and BERT in Financial Statement Analysis

A. BM25

In the realm of information retrieval, BM25 has emerged as a standard ranking algorithm due to its ability to effectively estimate the relevance of documents to a query. It builds upon three key concepts: Term Frequency (TF), which measures the occurrence of a term in a document and applies adjustments to prevent over-scoring documents with high term repetition; Inverse Document Frequency (IDF), which highlights the importance of rarer terms by penalizing commonly used ones across the document

corpus; and Document Length Normalization, which ensures fair scoring by mitigating biases toward longer documents that naturally contain more terms. These features make BM25 a foundational method widely applied in search engines and text mining, setting a benchmark for relevance estimation in related studies.(Sourabh Mehta, 2024)

B. LLMs and BERT

Large language models (LLMs), such as GPT-3.5, demonstrate strong natural language processing capabilities but face challenges with deep numerical reasoning. Rather than engaging in explicit numerical analysis, LLMs primarily understand numerical data through its narrative context. This limitation is particularly evident in financial document analysis, where interpreting complex relationships within financial statements requires a nuanced understanding of figures, categories, and financial terms. Without specialized models or explicit feature extraction, LLMs struggle to handle structured numerical data effectively (Kim, Muhn, & Nikolaev, 2024).

Given the limitations of large language models (LLMs) such as GPT-3.5 in handling structured numerical data, we ultimately decided to adopt BERT as the primary model for our task. BERT's strength lies in its capability for deep bidirectional text representation learning, which enables it to excel in diverse natural language processing tasks, particularly those requiring detailed and nuanced understanding of textual elements.

BERT is intended for deep preliminary learning of bidirectional text representation for subsequent use in machine learning models. The advantage of this model is its ease of use, which involves adding just one output layer to the existing neural architecture to obtain text models that surpass the inaccuracy of all existing ones in several natural text processing problems.

There are two categories of natural text processing tasks: holistic, operating with text at the sentence level, and tokenized ones, such as answering a question and attribution of entities, which produce more detailed output at the level of individual text elements. Both categories of problems have recently been using pretrained models, which can significantly reduce the time for designing and training private models while maintaining a high level of efficiency. (Koroteev, 2021)

C. RAG

The rise of LLMs has also sparked interest in their application to specific tasks, prompting the emergence of an approach called retrieval-augmented generation (RAG). RAG was devised to extend the capabilities of LLMs beyond conventional training data. By integrating a specialized body of knowledge, RAG enables LLMs to provide more accurate responses to user queries. In essence, RAG comprises two distinct phases: retrieval and generation. During the retrieval phase, defined sources (e.g., indexed documents) are analyzed to extract relevant information that is aligned with the user’s prompt or question. The retrieved information is then seamlessly integrated with the

user's prompt and forwarded to the language model. In the generative phase, the LLM leverages the augmented prompt and internal understanding to craft a tailored response that addresses the user's query effectively(Iaroshev, Pillai, Vaglietti, & Hanne, 2024).

II. OCR technology

OCR technologies play a crucial role in converting scanned or printed financial documents into machine-readable text, making it possible to analyze and retrieve information. OCR systems, such as PyTesseract, are used for extracting textual content from scanned images, and their performance depends significantly on the quality of the input image and the effectiveness of preprocessing techniques (Bennett, 2008). In financial document analysis, OCR's ability to correctly identify characters and symbols—such as currency values, percentage signs, and financial terms—affects the quality of the data that can be processed by downstream models. (Kim, Muhn, & Nikolaev, 2024).

III. Challenges with Tabular Data in Financial Documents

Financial documents often contain tabular data that presents unique challenges for AI models. Traditional deep learning techniques, particularly those used for homogeneous data (like image recognition), struggle with tabular data because they lack inherent spatial relationships between the variables (Yang et al., 2020). In financial statements, tables are common and often contain crucial information about transactions, balances, and other financial metrics. However, the structure of tabular data can vary significantly, and inconsistencies in data quality (e.g., missing values, outliers) are frequent (Yang et al., 2020).

The complexity of working with tabular data in financial documents is compounded by the need for preprocessing steps that may include normalization, handling missing values, and encoding categorical features. In some cases, incorrect preprocessing may lead to information loss, which can significantly reduce the predictive performance of machine learning models. Furthermore, the challenge of learning complex, irregular spatial dependencies in tabular data makes it difficult for standard deep neural networks to effectively process financial tables (Yang et al., 2020).

IV. Preprocessing Challenges and the Role of Regular Expressions

To prepare text for machine learning models, a variety of preprocessing steps are required. Regular expression (regex) substitutions are commonly used to normalize text and address common OCR errors such as impossible character combinations (Bennett, 2008). This preprocessing step ensures that the text is structured appropriately before feeding it into machine learning models for line-level or string-level analysis. However, challenges arise when pattern matching for categorization purposes, particularly in financial documents, where "hinting" phrases may be ambiguous or inconsistent, leading to occasional misclassifications (Bennett, 2008). This highlights the need for further refinement in OCR systems and preprocessing pipelines to enhance accuracy in document categorization and information retrieval.

Based on the literature reviewed, it is evident that OCR technology for converting financial statements faces challenges due to the lack of a standardized format and the need for flexible, human-like interpretation. To improve the accuracy of interpretation, additional conditions and learning mechanisms must be integrated. Therefore, in our approach, we introduced specific conditions and compared the text recognition results of different OCR techniques to enhance the understanding and interpretation of financial documents.

Methodology

I. Pre-Processing and Feature Engineering

A. ChatGPT summarizing

After analyzing the results from pdfplumber, we noticed numerous errors in the extracted text due to irregular table layouts, mixed languages, and scattered numbers or symbols. To address this, we decided to use ChatGPT to refine the extracted content. Our approach involves feeding the raw text extracted by pdfplumber into ChatGPT, accompanied by a prompt. The prompt specifies that the input text may contain irregular structures, such as unexpected combinations of Chinese, English, or numbers, and requests ChatGPT to reorganize the text into clear, concise sentences. Additionally, the prompt emphasizes retaining essential details, such as financial terms, company names, and other specific nouns, while avoiding any output beyond the summarized content. This method leverages ChatGPT's capabilities to transform disorganized OCR results into structured and readable text, ensuring accuracy and preserving critical information for further analysis.

B. Google OCR

Next, we use Google OCR to extract text from a PDF. We initialize an empty string, pdf_text, to store the text. Each PDF page is sent to the Google OCR API with the TEXT_DETECTION feature. The API processes the images, extracts the text, and we add it to pdf_text. Finally, the function returns the complete text, making the PDF content editable and searchable.

```
Function read_pdf(pdf_loc):
    Initialize pdf_text as an empty string
    Convert all pages of the PDF located at pdf_loc to JPEG images

    For each page_jpg in the extracted JPEG pages:
        Encode the current page_jpg image as base64
        Create an OCR API request with TEXT_DETECTION feature
        Send the OCR request and retrieve the detected text
        Append the detected text to pdf_text

    Return pdf_text
```

Figure (1) Google OCR pseudocode

C. Context cleaning and feature engineering

Then, we will further process the text by addressing potential issues arising from

synonymous expressions represented by different terms, such as the conversion between quarters and months(months 1-3 are replaced with “第一季,” 4-6 with “第二季,” 7-9 with “第三季,” and 10-12 with “第四季.”) or between Minguo and Gregorian years(a 3- or 4-digit number before ‘年’).

```
Function Convert_to_Chinese_Year(text):
    Define a mapping from digits (0-9) to Chinese numerals
    For each match of "3 or 4 digit number followed by '年'":
        If number > 1911: Convert to Minguo year by subtracting 1911
        Replace year with Chinese numeral using the mapping
    Return modified text

Function Convert_Quarter(text):
    For each match of "number followed by '月'":
        If 1-3: Replace with "第一季"
        Else if 4-6: Replace with "第二季"
        Else if 7-9: Replace with "第三季"
        Else if 10-12: Replace with "第四季"
    Return modified text
```

Figure (2) context cleaning and feature engineering pseudocode

Additionally, we will account for punctuation marks or special financial symbols that could lead to errors, removing them to ensure accurate interpretation. This approach aims to enhance the clarity and precision of the organized data, ensuring it is suitable for subsequent analysis and understanding, the following is the detail what we done:

- Remove all digits: Matches and removes all numeric characters using the pattern \d+.
- Remove all non-full-width punctuation marks: Matches and removes ASCII punctuation and symbols (e.g., ,, ., ?, !, $, etc.) using the pattern [!-~], which covers all printable characters in the ASCII range except for whitespace.
- remove all useless spaces with replace (“ “,” ”)
- Companies name: The company names in the financial reports are usually full name. However, we found out that most of the questions ask with an abbreviated name of the company. Therefore, we found some common short names of the company and replaced them with their full name while they appear in the query sentence.

D. Word segmentation and NER and POS tagging (using ckip-tagger)

We selected CKIP Tagger for word segmentation, named entity recognition (NER), and part-of-speech (POS) tagging due to its strong localization advantages and high-quality linguistic models tailored for Traditional Chinese. Developed in Taiwan by the Chinese Knowledge and Information Processing (CKIP) group, CKIP Tagger is uniquely equipped to handle the minor difference and terminology commonly used in Taiwan.

Our textual data is primarily sourced from Taiwanese documents, making CKIP

Tagger an ideal choice. Unlike tools like jieba, which is more focused on Simplified Chinese and generalized Chinese text processing, CKIP Tagger integrates linguistic insights specifically for Taiwan's Traditional Chinese context. This enables it to accurately process Taiwan-specific vocabulary, idiomatic expressions, and grammar. Its alignment with the linguistic characteristics of our data ensures higher accuracy and better relevance compared to other tools.

E. N-gram

Observing the separation made by ckip, we found out that some common financial report words that have been separated by ckip. For example, "應付帳款" has been cut into "應付" and "帳款". Therefore, we use the bigram method to fill this gap.

II. Model choosing and training

In the model selection process, we adopted a progression from text-based statistical representation to deep learning for textual comprehension, experimenting with four different modeling approaches. Below is a detailed explanation of each method:

A. BM25

Using term frequency and TF-IDF, we ranked and filtered data based on its relevance to the query. Before ranking, we preprocessed the text by removing less significant parts of speech and applied the bi-gram method to combine adjacent terms into unified phrases, enhancing the semantic coherence of the retrieved content. When processing queries, we segmented the text, tagged it with part-of-speech annotations, and converted numeric expressions into their Chinese equivalents to ensure consistency across synonymous expressions.

BM25 is efficient and effective for basic relevance computation, leveraging the proportion and relevance of recognized terms to produce accurate results. However, through our evaluation and analysis of incorrect outputs, we observed that while BM25 currently achieves high accuracy, its inability to comprehend the meaning of terms limits its applicability in more semantically complex tasks. To address these limitations, we explored alternative methods that focus on deeper textual understanding.

```
# Process each PDF text
For each PDF_text in PDF_texts:
    Filter words by selected POS tags
    Combine the remaining words (and bi-grams at the same time)
    into a single sentence separated by spaces

# Process each query and retrieve answers
For each query in queries:
    Segment the query with CKIP
    Tag the query with POS using CKIP
    Convert all numeric digits in the query to Chinese numbers
    Filter the tagged words by selected POS tags
    Combine the filtered words into a processed query sentence
    Extract candidate texts from the PDFs
    Send the processed query and candidate texts to BM250kapi
    Retrieve and return the top-ranked candidate text as the answer
```

Figure (3) bm25 pseudocode

B. TF-IDF

TF-IDF assigns higher weights to terms that appear frequently in a document but are less common across the entire corpus, effectively capturing more distinctive words. Similar to BM25, we filtered specific parts of speech and ensured consistency in term representation during preprocessing. However, like BM25, TF-IDF is unable to comprehend the semantic meaning of terms within the text. After testing, we found that TF-IDF's accuracy was less precise compared to BM25, highlighting its limitations in tasks requiring nuanced understanding of textual content.

```
# Process each PDF text
For each PDF_text in PDF_texts:
    Filter words by selected POS tags
    Fit and transform the tfidf matrix to all the PDFs

# Process each query and retrieve answers
For each query in queries:
    Segment the query with CKIP
    Tag the query with POS using CKIP
    Convert all numeric digits in the query to Chinese numbers
    Filter the tagged words by selected POS tags
    Transform the processed query into the category's tfidf vector
    Use cosine similarity to find the top-ranked candidate
```

Figure (4) TF-IDF pseudocode

C. Retrieval-Augmented Generation (RAG)

In our project, Retrieval-Augmented Generation (RAG) plays a critical role in improving the accuracy and relevance of document retrieval for financial question answering. In this method, we leverage Retrieval-Augmented Generation (RAG) to process and store document embeddings for efficient similarity-based retrieval.

i. Deep Learning(bert-base-chinese)

Building upon the limitations of statistical models, we explored deep learning models capable of understanding textual semantics to ensure that the correct documents are retrieved based on their actual relevance, rather than relying solely on term similarity. These models are adept at handling subtle semantic differences and more complex syntactic structures, making them highly effective for tasks requiring deeper comprehension. Through our experimentation, we observed significant improvements in performance and understanding, particularly in the insurance and FAQ domains, where the models demonstrated superior accuracy and contextual relevance.

```
# Chunk the PDF texts into chunks
# by the length of the text

For each query in queries:
    Filter the chunks
    using the candidate list

    Compute similarity scores
    between the query and the filtered chunks
    using the BERT model

    Sort candidates by scores
    and select the best match
```

Figure (4) bert pseudocode

ii. Embed with Openai

The approach begins by splitting the documents into manageable chunks using a text splitter. These chunks are then stored for further embedding generation. To ensure proper access to OpenAI's services, the user is prompted to input their API key, which is stored as an environment variable. Any existing vector storage directory is deleted to reset the storage and avoid conflicts. The embedding process uses the OpenAI embeddings model to convert the document chunks into high-dimensional vectors. A vector store is created to persistently store these embeddings, using the specified directory path. This setup ensures fast and accurate document retrieval based on semantic similarity, a crucial step for downstream tasks like query answering.

```
Initialize text_splitter with chunk_size and chunk_overlap
Split finance, insurance, and faq data into chunks using text_splitter
Initialize embeddings using OpenAIEmbeddings
Embed and store the data into chroma

For each query in queries:
    Embed the query
    Use chroma database to find similarity between each query and its candidates
```

Figure (5) embed with Openai pseudocode

Experiments and Results

I. Verse pdfplumber & Google OCR , ckipws & jieba

Initially, we sought to enhance document recognition by evaluating the performance of PDFplumber, Google OCR, and the ChatGPT API (for summarizing content) in combination with the BM25 and TF-IDF methodologies. The results indicated that the use of Google OCR significantly enhanced table recognition, leading to an improvement in accuracy within the finance category. BM25 and TF-IDF demonstrated comparable performance.

Based on the above findings, we decided to adopt Google OCR as the primary tool for text extraction. Additionally, we implemented various approaches for data preprocessing, including tokenization, part-of-speech analysis, and text processing. The test results are summarized in the table below:

Table(1) Comparison of Tokenization and Recognition

	BM25			TF-IDF		
	Insurance	finance	faq	Insurance	finance	faq
pdfplumber + jieba	0.8	0.44	0.9	0.8	0.32	0.88
pdfplumber + ChatGPT summarizing + jieba	0.68	0.72	0.9	0.72	0.74	0.9
Google OCR + jieba	0.8	0.54	0.9	0.84	0.3	0.88
Google OCR + ckip ws	0.76	0.54	0.88	0.84	0.7	0.92

II. Verse types of Text Processing

We have selected Google OCR and ckip ws as the primary tools for converting PDF files into text and performing word segmentation. Subsequently, we evaluated the impact of different text processing techniques, such as part-of-speech (POS) tagging and other preprocessing methods, on improving document retrieval accuracy. Bigram analysis shows the best overall results, especially in the Insurance and FAQ categories. The results are summarized in the table below:

Table(2) Comparison of Text Processing

	BM25			TF-IDF		
	Insurance	finance	faq	Insurance	finance	faq
Google OCR + ckip ws + POS	0.72	0.72	0.9	0.86	0.74	0.94
Google OCR + ckip ws + Text Processing	0.76	0.56	0.9	0.84	0.72	0.92
Google OCR + ckip ws + Text Processing, Bigram	0.84	0.74	0.9	0.9	0.72	0.92

III. Verse types of retrieval-augmented generation (RAG)

Next, we explored deep learning approaches to address the limitations of BM25 mentioned earlier, specifically its reliance on text proportion calculations, which may lead to misjudgments when encountering words with the same spelling but different meanings.

In our experiments with BERT, we evaluated the differences between using chunking and not using chunking. For the chunking approach, we set the value to 500 words per chunk with no overlapping.

For the RAG with Openai embedding component, we experimented FAQ with two approaches to feeding data: "with answer" and "without answer." In this context, "answer" refers to the presence of both questions and corresponding answers within the FAQ dataset, where the questions are relatively concise, and the answers are more detailed and comprehensive. Our experiments aimed to compare the differences between "retrieving questions based on a question" and "retrieving questions based on the complete dataset,

including answers." This comparison allowed us to evaluate the impact of using more context-rich data on the quality and accuracy of the retrieval process.

Table (3) Comparison of deep learning

	Insurance	finance	faq
Google OCR + ckip ws + POS	0.72	0.72	0.9
Google OCR + ckip ws + Text Processing	0.76	0.56	0.9
Google OCR + ckip ws + Text Processing, Bigram	0.84	0.74	0.9

IV. Competition Results：Team 6596

During the competition, Our approach adopts a multi-model output comparison strategy: when at least two models produce the same result, we directly select this consistent output, as it indicates a higher level of confidence among the models. Conversely, if all three models generate different results, we employ an experimental comparison method by individually selecting the output of each model for separate uploads. Specifically, the first upload uses the result from BM25, the second uses the result from TF-IDF, and the third uses the result from RAG. This process design not only embodies the flexibility of a hybrid model approach but also leverages the unique strengths and characteristics of each model. It aligns well with our operational logic and enhances both the accuracy and robustness of the system.

Ultimately, we found that the RAG-based approach delivered better performance, achieving a final accuracy of 0.8836, which show in figure (7).

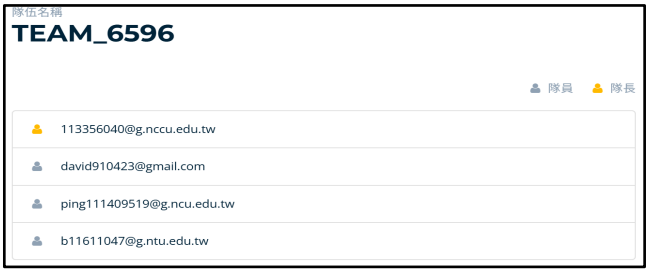

Figure (6) Team member

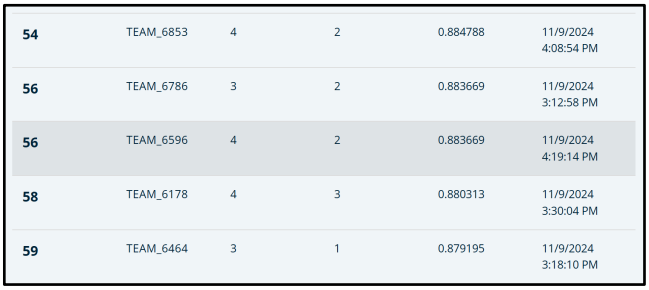

Figure (7) Ranking and Scores

Discussions

I. Preprocessing of text recognition

In the results, we observed that the combination of Google OCR and ckip word

segmentation produced more accurate word segmentation and aligned better with the logic of Traditional Chinese. This is particularly evident in the recognition of specialized terms and proper nouns, where this combination demonstrated superior performance. Additionally, while BM25 and TF-IDF are based on similar computational principles, BM25 and TF-IDF also demonstrated comparable performance.

II. Preprocessing of Text Processing

Focusing on text processing techniques, we noted that POS tagging is effective for extracting standardized semantics, but its impact on improving retrieval performance is limited, likely depending more on the capabilities of the retrieval model. On the other hand, Bigram processing demonstrated more significant improvements by adding continuous phrase information to the context, which notably enhanced retrieval accuracy. This effect was particularly pronounced in categories like insurance and finance, where context support plays a crucial role.

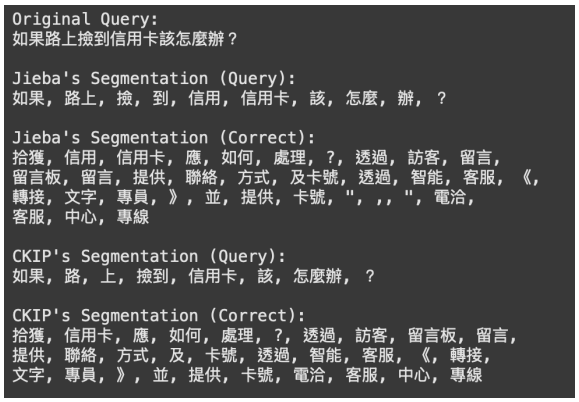

Figure (8) The word segmentation results of Jieba and CKIP

Furthermore, we conducted an in-depth analysis of the differences between Jieba and CKIP, two widely used text segmentation tools. As shown in the figure(8), Jieba tends to repeatedly segment certain uncertain terms, which can result in these terms receiving higher weights during the subsequent weighting process. While this approach may enhance the recognition of specific terms in certain scenarios, it also introduces potential noise that could negatively impact the overall accuracy of the model.

In contrast, CKIP adopts a more straightforward segmentation strategy, performing simple text segmentation without repeating terms. This method allows for clearer identification of key terms during the weighting process, particularly when the weighting strategy focuses on emphasizing critical keywords. Under such conditions, CKIP may offer superior performance. This explains why, in shorter datasets such as FAQs, Jieba tends to achieve higher accuracy.

III. RAG implementation - BERT and OpenAI embedding

By comparing the performance of RAG and BERT, investigating the effects of dataset configurations, and analyzing errors, we aim to identify key factors influencing model

accuracy and propose strategies for improvement.

A. Analysis of Deep Learning Methods: RAG vs. BERT

In our exploration of deep learning methods, we found that RAG exhibited the best overall performance among all approaches. However, when comparing BERT and RAG, BERT demonstrated lower accuracy. One significant factor for this disparity could be the lack of additional domain-specific training or the incorporation of financial terminology into BERT's model configuration. Without such specialized adaptations, BERT, when processing original Chinese text, struggled to accurately interpret certain domain-specific terms or concepts, resulting in reduced performance for tasks requiring specialized knowledge.

B. Chunking vs. Non-Chunking in BERT

Within the context of BERT, we further compared chunking and non-chunking methods. The results showed minor differences in accuracy across the insurance and finance categories. Specifically, chunking achieved scores of 0.58 and 0.68 in these categories, while non-chunking scored 0.34 and 0.38, respectively. These findings suggest that chunking is better suited for longer textual content, such as financial and insurance documents, as it enables a more precise analysis of the content by breaking down the text into manageable pieces.

C. Effects of Including Answers in the Dataset

Additionally, we investigated the impact of including answers in the dataset. When answers were included ("with answer"), the text structure became more complex, as answers tended to be longer and contained more noise. In contrast, questions alone ("without answer") encapsulated the essential information required to address the task and contained minimal noise. These factors contributed to the lower accuracy observed with "with answer" datasets. The additional, often unnecessary information interfered with the model's ability to focus on the core content. Therefore, relying solely on questions ("without answer") may enhance performance and efficiency in certain applications.

D. Analysis of Errors and Causes

To further refine our understanding, we conducted an in-depth investigation into the incorrect results. The majority of inaccuracies were observed in the insurance and FAQ categories. A significant portion of these errors stemmed from inconsistencies between the answers and the retrieval resources. Specifically, during answer selection, official answers were often absent from the provided "resources," leading to discrepancies.

In the finance category, most errors were linked to questions involving quantitative inquiries (e.g., "how much") or issues related to financial statements. These errors underscore the challenges of aligning the system's understanding with the domain-specific requirements of financial data and associated queries.

Future Works

I. Trying different PDF OCR method – PyTesseract

Through our literature review, we also found references suggesting that the performance of PyTesseract in OCR tasks can be improved through adjustment and training. In the future, efforts could focus on training OCR models, particularly for handling less distinguishable content, such as tables in financial statements. By accurately identifying table locations and further enhancing interpretation capabilities, it would become possible to answer questions more precisely and improve the accuracy of information retrieval.

II. Finishing the Generation part of RAG

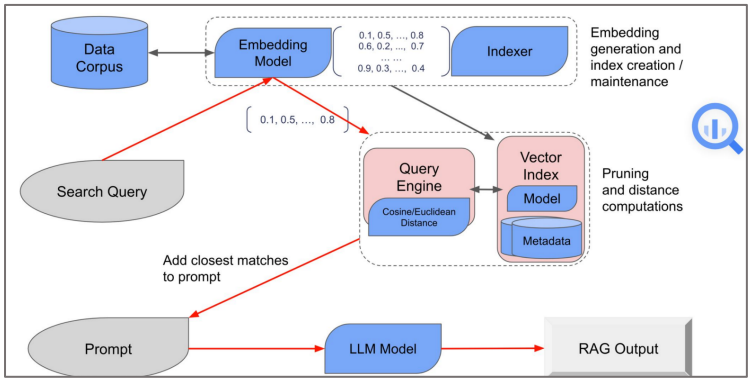

Figure (9) The process of RAG

A. Retrieval

Another future work related to improving retrieval methods in RAG (Retrieval-Augmented Generation) involves several approaches that can enhance retrieval accuracy based on literature review:

i. Multi-Vector Retriever

This approach divides the original chunk into smaller sub-chunks, and these smaller chunks are used to build the vector database. During retrieval, multiple small chunks are retrieved and then mapped back to their corresponding original chunks. These original chunks are considered the retrieved documents.

ii. Multi-Query Retriever

In some cases, it may be necessary to try multiple queries to obtain the most suitable results for generating answers. This method allows the LLM (Large Language Model) to generate multiple queries from different perspectives after receiving the initial query. These queries are then used to retrieve a richer set of documents.

iii. Re-ranking

Comparing vector similarity alone may not always yield the best results. This method involves re-evaluating the similarity results using techniques such as LLM, MRR (Mean Reciprocal Rank), or MAP (Mean Average Precision) to score and re-rank the retrieved documents.

B. Generation

Our implementation focuses on improving retrieval accuracy, specifically on the steps prior to the prompt shown in the diagram. Therefore, one of the future works would be to complete the entire RAG (Retrieval-Augmented Generation) process. The simplest way to implement this would be to combine the retrieved document contents with the user's initial query to form a new prompt. This prompt would then be input into a large language model, allowing the model to generate the corresponding answer.

Site

Bennett, F. G., Jr. (2008). *Reconstructing financial statements*. Nagoya University, Nagoya 464-8601, Japan. Retrieved from https://discovery.ucl.ac.uk/id/eprint/9137

Kim, A., Muhn, M., & Nikolaev, V. (2024). Financial statement analysis with large language models. *arXiv*. https://doi.org/10.48550/arXiv.2407.17866

Yang, L., et al. (2020). Tabular data understanding with deep learning: A survey. *IEEE Access*. https://doi.org/10.1109/ACCESS.2020.2994557

Iaroshev, I., Pillai, R., Vaglietti, L., & Hanne, T. (2024). Evaluating retrieval-augmented generation models for financial report question and answering. *Applied Sciences*, *14*(20), 9318. https://doi.org/10.3390/app14209318

Koroteev, M. V. (2021). BERT: A review of applications in natural language processing and understanding. *arXiv*. https://doi.org/10.48550/arXiv.2103.11943

Mehta, S. (2024, July 11). *Understanding Okapi BM25: A guide to modern information retrieval*. AdaSci. Retrieved from https://adasci.org/understanding-okapi-bm25-a-guide-to-modern-information-retrieval/?utm_source=chatgpt.com

Group Contribution

- **documentation(text and related work studying) writing:** 黃麗蘋(65%)、李承諭(35%)
- **slide and video filming :** 黃麗蘋(35%)、李承諭(65%)
- **data preprocessing:** 楊閎凱（ckip, google ocr, chatGPT summarizing 100%）
- **feature engineering:** 楊閎凱(50%)、 林靖淵(50%)
- **model training:** RAG-openAI: 林靖淵 TFIDF, BM25, RAG-BERT: 楊閎凱