Large Financial Documents QA using LLMs

Shobhit Maheshwari, s2534699@ed.ac.uk
Under the supervision of Prof. Tiejun Ma and his PhD. students Mengyu Wang and Sunnie Li

ABSTRACT

Financial asset insights can steer businesses in making informed decisions by distilling vast data into actionable information. The primary goal of this project, thus, is to develop a system that can feed on the information from the financial report provided and perform Question-answering using the information assimilated by the Large Language Model (LLM). Directly processing these documents through LLMs is not very effective because of the magnitude of these financial reports. Two plausible approaches are discussed in the work to tackle this problem of Long documents in the domain of finance namely RAG and MapReduce. An optimised version of MapReduce is built which makes the processing fast and efficient.

THE UNIVERSITY of EDINBURGH informatics

1 INTRODUCTION

The extraction of key business highlights from financial reports can be pivotal in shaping the investment decisions, risk management and planning strategies around businesses. The volume of the financial data in terms of annual reports, balance sheets, etc. is huge and thus the manual analysis of all these becomes an extremely time-consuming task prone to human errors. Multiple approaches to perform QA on financial documents using LLM exist but one of the major concerns related to these documents is the long context sizes needed. Financial PDFs like annual reports can be more than 200 pages long. Most LLMs lack the ability to process these long documents in one pass. Even the ones which can process such documents suffer the problem of missing the correct answers if the answers to the query are mentioned in the middle of a very long context [1].

This work delves deeper into two approaches which can be leveraged to perform QA on such large documents without compromising on the quality of results. Analysis is done in terms of the trade-offs that exist between accuracy and computational costs to query long PDFs. Manual evaluation for the approaches is done and the issues related to BLEU, ROUGE and other embedding based evaluation like BERTScore are discussed. The best performing models are wrapped in an interface to easily query the annual reports. This work paves the way for future work that can be done to make a QA engine using the entire scraped corpus of financial reports of all the companies listed on LSE.

2 CONTRIBUTIONS

The key contributions in this work include the following:

- Scraped financial reports data from two different resources (Annual Reports and Hargreaves Lansdown) for all listed companies on LSE. A total of 70 GB of data was scraped with approximately 18k PDF documents. The scraping was done with the help of fault-tolerant pipeline with retry-mechanisms to ensure data completeness.
- Build and test multiple approaches to perform QA on long documents in the financial domain. Analysis was done using RAG, Stuffing, Refine and MapReduce. RAG and MapReduce were chosen as ideal options considering processing times and efficiency. Experiments were designed to test the viability of the two methods. MapReduce with re-ranking performed the best and was used as the basis for Case-studies for Business Insight Extraction on PDF.
- Wrapping the MapReduce implementation in a GUI. This was done with the help of streamlit. The GUI lets the user perform QA using the MapReduce approach with tweakable parameters for temperature, max tokens, chunk size, etc. as shown in Figure 1. It also logs intermediate results after the Map Phase for debugging.

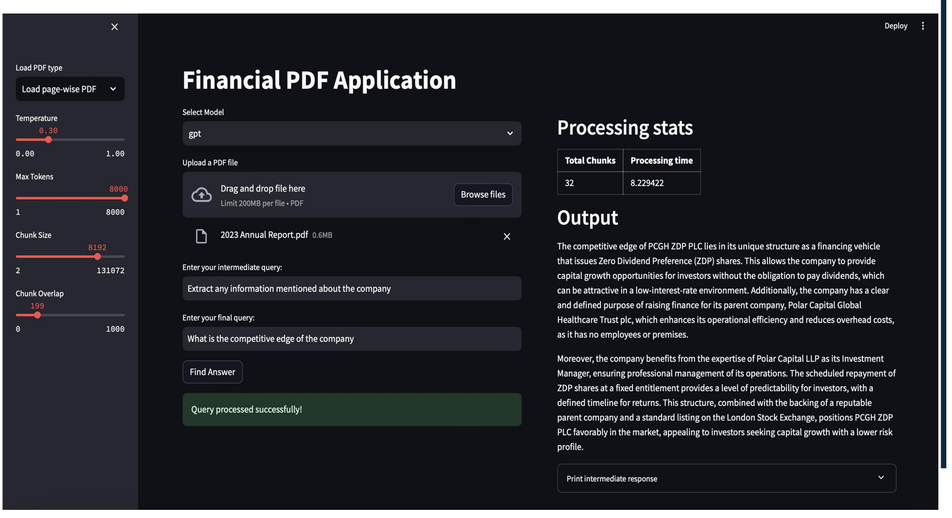

Figure 1: GUI to query and process PDFs using MapReduce

3 METHODOLOGY

Retrieval Augmented Generation

- RAG operates in two phases: Retriever and Generator.
- Retriever component retrieves the top-k relevant chunks from the long PDF using similarity search of vector representations of query and the chunks.
- The retrieved chunks are processed through the Generator to generate an answer which is coherent.
- RAG Generation can be done in two ways: RAG-Token and RAG-Sequence. Considering resource limitations, RAG-Sequence was used.
- The retriever component can limit the ability for a RAG pipeline if relevant documents are not present in top-k retrieved documents.

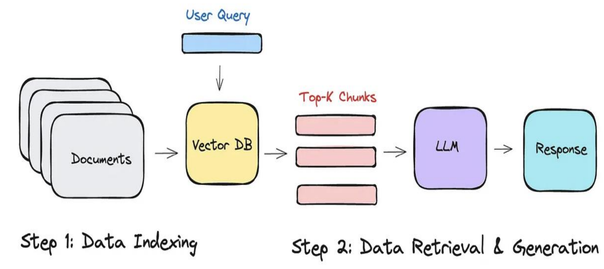

Figure 2: Overview of the usual RAG pipeline

MapReduce with Re-Ranking

- MapReduce gets rid of the dependency on retrieval component.
- We move away from the LangChain abstraction of MapReduce to processes document chunks in parallel with the help of basic LLMChain.
- Few-shot learning is used in the prompts to generate output along with a score of the relevancy of the answer. Filtering is done using this score.
- Filtered results are sent to the Reduce prompt to generate the final output.
- GPT-4o-mini is used for MapReduce since it performed the best in testing RAG.

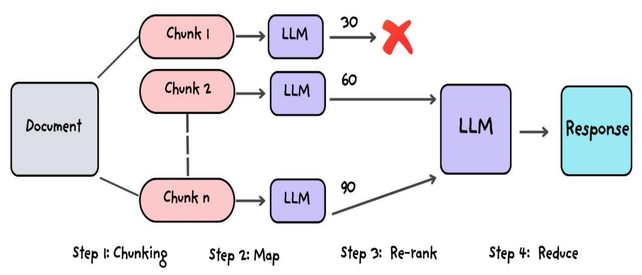

Figure 3: MapReduce with Re-ranking

4 RESULTS

Limitations of Retriever in RAG

Retriever is implemented using Facebook AI Similarity Search (FAISS) and the sentence-transformer model MPNet is used to extract 768-dimensional hidden representations. K is increased to see if the Retriever limits the RAG applications accuracy. For a small-data of 150 Questions (FinanceBench dataset [2]), at k=5, model is able to capture all relevant pieces of text only 83 times. K=5 is used for comparison of Generator models due to an inflection point as shown in Figure 3.

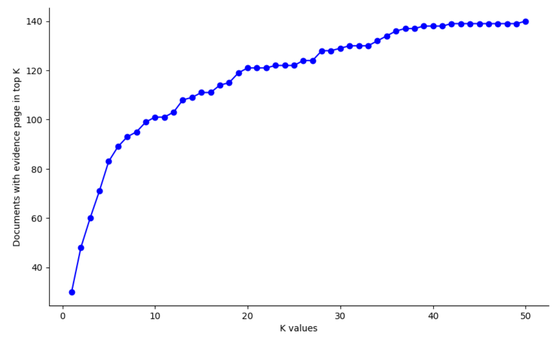

Figure 4: Overview of the usual RAG pipeline

Llama vs Mistral vs GPT for RAG on FinanceBench

RAG application is tested for three different models. For Llama and Mistral, we use 4-bit quantized models to run the Generator component of RAG due to resource constraints. GPT results are generated using OpenAI API. From the results, GPT model outperformed Llama and Mistral. Thus, only GPT-4o-mini was used for the MapReduce application. Running MapReduce with Llama and Mistral with parallelisation would have required extremely resource-heavy machines.

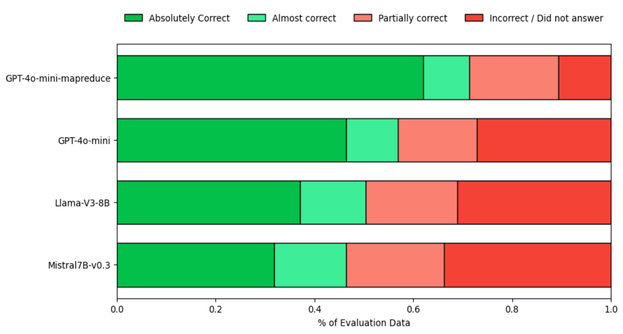

Figure 5: Output Evaluation on FinanceBench

Optimised MapReduce with Re-ranking

In order to optimize the MapReduce with parallelisation, LangChains' LLMChain was used with ThreadPoolExecutor. The reason to not use LangChain abstraction was better processing time and ability to tweak results at an intermediate stage. The prompt was modified to generate and output answer along with the score for the output. This score was filtered to take only scores above 50 for the reduce phase so that the context length at the reduce stage does not become too big. Bigger context windows are known to have issues if the right answer is in the middle of the context as demonstrated by Liu et. al in "Lost in the middle" [1]. Human Evaluation was conducted to compare the models shown in Figure 5.

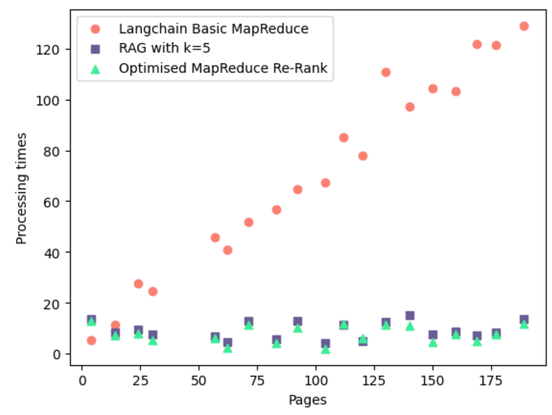

Figure 6: Processing times vs PDF pages

Processing times analysis for PDF

The processing time for the LangChain Version of MapReduce chain, self-implemented version of MapReduce with re-rank and RAG application was done using GPT as the Generator model. The optimized version of MapReduce with re-ranking was able to give similar processing times to RAG despite having much more visibility into the PDF since it scans all chunks. The LangChain version though increases linearly as the PDF size increases.

5 CONCLUSIONS

Financial QA from annual reports is a significantly challenging task due to the PDFs being extremely long. Two approaches to chunk the long PDF and process it were tried i.e. RAG and MapReduce with Re-ranking. RAG applications limitations were analysed from the retriever module. Experiments were conducted to identify the best Generator Model for the RAG application from three competing models i.e. Llama, Mistral and GPT-4o-mini. GPT-4o-mini performed significantly better by outperforming Llama and Mistral. The major limitation with the RAG application was the retriever bottleneck. The retriever was not able to correctly identify the relevant chunks from the long PDF documents which the Generator needed to generate an apt answer. Considering the weakness of RAG, MapReduce was considered, and an optimized version of MapReduce was implemented. The model increased the proportion of Almost correct and Absolutely correct class by 16 percent. The number of calls to the Generator network in case of MapReduce is significantly higher (of the order of the number of chunks in the PDF), but the processing times were found to be in the same bounds as the RAG. Thus, MapReduce was proposed as the ideal approach to extract insights in the form of Question-Answering from Financial PDF documents.

6 REFERENCES

[1] Nelson F Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. Lost in the middle: How language models use long contexts. Transactions of the Association for Computational Linguistics, 12:157–173, 2024

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