Automated Financial Document Summarization using GPT

Ricardo Mercado Cavazos

Master of Science

School of Informatics

University of Edinburgh

2024

Abstract

Recently, Large Language Models (LLMs) have demonstrated unparalleled capabilities in understanding, generating and manipulating human language, which has opened the way for new application possibilities across diverse domains. The financial domain has received multiple LLM applications, including specialized models. However, research is still being done to address many other possibilities. One of these potential applications is the usage of LLMs for automatic financial document summarization with the purpose of assisting the financial analysis process. Summarizing financial documents is not a trivial task since documents are usually large and require specialized knowledge to be accurately understood. This dissertation explores the performance of the GPT model `GPT-4o-mini` in summarizing financial documents with focus on the Financial Strength and the Management Team of a company. A dataset comprised of 14,767 financial documents was collected, which includes annual reports, interim reports, fact sheets, and five-year financial result tables. To facilitate the model querying and response evaluation processes, a graphical user interface (GUI) was developed. The experimentation results show that OpenAI's `GPT-4o-mini` model has a promising ability to understand and summarize financial data from annual reports. However, a more extensive experimentation is required to fully assess this. A solid base for such experimentation and other future work has been set through this work's outcomes.

Research Ethics Approval

This project was planned in accordance with the Informatics Research Ethics policy. It did not involve any aspects that required approval from the Informatics Research Ethics committee.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*Ricardo Mercado Cavazos*)

Acknowledgements

Thanks be to God for all the goodness He has had over me. Your blessings, oh Lord, have allowed me to pursue my goals and be resilient to achieve them.

I would like to express my sincere gratitude to my parents, Daniel Mercado González and Sylvia Guadalupe Cavazos Garza for constantly supporting my dreams and passions.

I am deeply grateful to all my friends, but especially to Mr. David Noel Ramírez Padilla and Mr. Raymundo Said Zamora Pequeño, for encouraging me to study a master's degree abroad and supported me through the process.

I would like to thank the University of Edinburgh and The School of Informatics for providing the environment and resources required for achieving this project.

Lastly, I would like to thank Dr. Tiejun Ma, Mengyu Wang, Sunnie Li, and Shobhit Maheshwari for the collaborative work and enriching discussions carried through the process of this dissertation.

Table of Contents

Chapter 1

Introduction

In recent years, Natural Language Processing (NLP) has seen an impressive growth mainly due to the development of Large Language Models (LLMs). These language models have demonstrated unparalleled capabilities in understanding, generating and manipulating human language, leading to a surge in new application opportunities across diverse domains. One of these domains is the field of finance, which has witnessed multiple LLM related developments that even include specialized LLMs, such as FinBERT[1] or BloombergGPT[2]. This dissertation investigates the potential use of foundation LLMs (general purpose models such as GPT, Gemini, Llama, etc.) for financial document summarization to help financial analysis.

Natural Language Processing is a subfield of Artificial Intelligence that aims to provide machines with human language intelligence. It is composed of a set of computational techniques for automatic analysis and representation of human languages[3]. According to Chowdhary [3], NLP is focused on the tasks of natural language translation, information retrieval, information extraction, text summarization, question answering, topic modeling and opinion mining.

One of the computational techniques that compose NLP is Language Modeling which aims to model the generative likelihood of word sequences[4]. Language Modeling has been researched since the 1990s and, according to Zhao et al. [4], it can be divided into four stages: Statistical Language Models, Neural Language Models, Pre-Trained Language Models (PLMs), and Large Language Models (LLMs). In essence, LLMs are scaled PLMs, in model size or volume of training data, that due to the higher volume of parameters and training information, present emergent abilities that smaller PLMs do not have. These abilities include in-context learning, instruction following, and step-by-step reasoning[4]. LLMs can be used for performing language-related tasks,

such as information retrieval, question answering, and text summarizing. Recently, various industries, including the financial, have been leveraging the potential of LLMs to automate diverse processes and tasks.

Financial analysis is a set of principles, procedures and tools that help organize financial, economic, and other business data. This is done to get information about a business to make better and more informed decisions[5]. It typically aims to determine if a company is stable, solvent, liquid or profitable enough to warrant a monetary investment[6]. It is usually a time-consuming process as it involves gathering information from long financial reports such as annual and interim reports. Large Language Models have the potential to revolutionize financial analysis by leveraging their summarizing capabilities to extract key financial information from financial reports.

1.1 Motivation

The ongoing growth of LLMs opens the possibilities for diverse applications, and multiple fields have already been leveraging LLMs to develop solutions to assist their processes. The financial field is not the exception, since it has captured the attention of researchers and various works have been done to use LLMs for financial purposes which have even led to specialized models.

The financial field is attractive for the implementation of NLP processes since it has large amounts of unstructured text data such as companies' financial reports. In addition to this, achieving high quality results with financial information could lead to real-world applications.

Amongst other areas of the financial field, the process of Financial Analysis could be benefited from using LLMs. This process could be partly automated by using LLMs to summarize and extract financial information contained in public documents such as companies' annual and interim reports. This automation could save large amounts of time by allowing financial analysts to focus on decision-making and not in reading large financial reports. However, for this to be achievable, the results of the automated process require to be factually correct and reliable while also including the most relevant information for decision-making. This situation has motivated this research to focus on the potential application of LLMs to summarize large Financial documents.

The main goal of this research is to assess the LLMs capacity to do or answer the following financial-information related tasks or questions, which are related to the Financial Analysis process, more specifically to the financial strength and the

management team aspects of a company:

- **Financial Strength**
  - Summarize financial metrics of the company.
  - Summarizing earnings updates/earnings transcripts.
  - Scoring the company for credit positivity/negativity or earnings upgrades/-downgrades for equities
- **Management Team**
  - How is the management team's ability to allocate capital effectively?
  - How well does the management team identify and manage key risks to the business?
  - How well does the management team respond to changes in the business landscape?

To achieve this, this dissertation aims to produce a corpus of collected publicly available financial documents relevant to the firms that belong to the FTSE ALL-SHARE share index. Furthermore, the corpus is leveraged to test ChatGPT-4o's capacity to summarize and extract financial information. Since the collected documents are usually larger than the context window of the models, LangChain's Refine Chain for large document summarization was implemented. To facilitate the process of querying the model, a Graphical User Interface (GUI) was developed for easy tracing the summarizing chain's intermediate steps as well as keep track of multiple queries done to the model.

1.2 Contributions

The main contribution of this research is assessing the quality of the responses provided by the LLM regarding the financial aspects queried. However, some other contributions have been developed in the process of achieving the main goal. These contributions include:

- A document corpus containing 14,767 financial documents from FTSE ALL-SHARE firms collected from publicly available sources.

- A web-based Graphical User Interface (GUI) for visualizing the corpus and querying the LLM.

The financial documents corpus provides a large database of financial text information that can be leveraged for diverse NLP tasks, including LLM experimentation and fine-tuning. On the other hand, the GUI offers users with a user-friendly interface to navigate the corpus and query the models while also allowing to keep track of the queries made, facilitating the interactions and experiments made with the LLMs. Its organized modular architecture enables easy implementation of external python libraries to extend its functionality besides the Refine Chain summarization pipeline, which introduces potential use cases besides those of this research.

1.3 Dissertation Organization

This document is divided in seven chapters. This chapter (Chapter 1) introduces the work by setting some required background context and stating the motivations and contributions of the research. Chapter 2 presents the researched related work regarding LLMs applications in the Finance field and Long Document Summarization with LLMs. The following chapter, Chapter 3, describes the process through which the dataset for this research was collected, preprocessed and formatted. The methodology used for the experimentation is described in Chapter 4. This chapter includes the details about the LLM used, the large document summarization process, the interaction with LLMs (prompting), the result evaluation process, and the selection of a sample dataset for experimentation. Chapter 5 details the implementation process of the previously described methodology, including the development of the GUI used during the experimentation process. Finally, Experimentation Results are displayed and discussed on Chapter 6 followed by the research's conclusions and a commentary of potential future work on Chapter 7.

Chapter 2

Related Work

The recent advancements of LLMs has revolutionized the field of Natural Language Processing (NLP). The models' capacity to generate text, translate, or summarize has gained the attention of many fields that aim to leverage them for various real-world applications. The field of Finance is not the exception for this due to the ever-growing volume of textual data, which includes financial reports, news articles, and regulatory filings.

LLMs have emerged as a promising tool for multiple financial-realm applications, including information extraction and summarization. Such has been the attention to the application of LLMs into the financial realm that multiple surveys have been done to keep track of the research on the field (e.g. [2, 7, 8, 9, 4]).

Financial documents, specifically annual reports, are often large and contain much textual information. All this information is valuable for NLP tasks, but processing it becomes a challenge. Document summarization is not excepted from this challenge, and thus there is relevant research devoted specifically to the task of large document summarization.

This Chapter delves into existing research relevant to this work. This includes works related to financial datasets for NLP and financial documents summarization.

2.1 Financial NLP Datasets

Data is a crucial element when working with NLP tasks since it is a central component for understanding language. Multiple datasets for NLP have been created leveraging the vast amounts of public text information available on the internet. Some of these are financial domain-specific datasets that were built from publicly available information.

Some of these datasets are Financial PhraseBank [10], Financial Question Answering and Opinion Mining [11], FinQA [12], ECTSum [13], FiNER [14], FinRED [15], REFinD [16], and FinSBD [17]. These datasets are built mostly from publicly available financial news and filings and have been designed and used for diverse purposes such as sentiment analysis or question and answering.

Other relevant datasets are those provided by the Financial Narrative Processing Workshops (e.g., FNS-2020) [18]. These datasets primarily consist of public financial reports of UK companies listed on the London Stock Exchange and at least three gold-standard summaries for each document. However, its believed that these datasets have limited access only granted to the workshop participants since no public access was found.

2.2 Financial Document Summarization

The attention drawn by financial applications of LLMs has provided a large number of research documents, ranging from the development of datasets, to specialized LLMs. For this research, the related works lie in the available datasets and the financial documents summarization and information extraction.

In recent years, diverse works regarding the summarization and extraction of financial information have been published. Works [19] and [20] are examples of the application of LLMs for financial information summarization and extraction. Both of these works focus on fine-tuning the pretrained BERT model.

Another related work is the research presented on [21], which explores the applications of ChatGPT, Bard, and BingAI for financial tasks. In contrast to the methodology proposed in this dissertation, [21] uses chat models and is not focused on financial document summarization.

The research presented in [22] and [23] delves into the evaluation of the `GPT-3.5-turbo` model's capabilities to summarize financial information for financial analysis. The documents used in [22] are earnings conference calls and extracted management discussion and analysis sections from firms' annual reports. On the other hand, [23] uses earnings call transcripts. Both works report the implementation of chunking techniques to split the documents into smaller texts that would fit in the model's limited context window. It is to be noted that both works report positive outcomes after experimentation. The research presented in this dissertation differs from these works by using a newer version of the GPT model (`GPT-4o-mini`) and by leveraging complete annual reports as context

for summarization.

Some of the research works focused on financial summarization focus on handling large documents. For instance, [24] proposes chunking financial texts by structural element components resulting in larger chunks that better preserve the contexts than the traditional approach of segmenting documents by paragraphs. This approach is similar to the one used in this dissertation when handling large documents.

Chapter 3

Data Collection

A large quantity of financial documents is required to be able to assess the LLMs capacity to retrieve or summarize financial information. Publicly listed shares provide financial documents for investors to analyze the company and decide whether to invest in them. These publicly available financial documents can be found on the internet and consist of annual reports, interim reports, fact sheets, and the financial results of past years. A dataset was built by collecting these financial documents using web scraping to download the documents, process them, and store the information in a JSON[1] file. That information was later used to build a database to integrate it with the developed user interface.

3.1 Web Scraping

Web scraping is the process of automatically retrieving and organizing unstructured or semi-structured data from the web for future analysis[25]. This process is usually done by using either ready-made or custom-made solutions. For this research, a custom tool was built using the Python programming language (v3.11.5)[2] leveraging libraries as `httpx`[3] and `lxml`[4].

According to Krotov and Johnson [25], web scraping is divided in three steps: website analysis, website crawling, and data organization. Website analysis consists of manually analyzing the source websites and understanding the information in them. Website crawling when the scraping tool requests and retrieves information from the

[1] https://www.json.org/json-en.html
[2] https://www.python.org/downloads/release/python-3115/
[3] https://pypi.org/project/httpx/
[4] https://pypi.org/project/lxml/

websites. Finally, data organization is the process of cleaning and organizing the collected data. The result of this process is a clean and organized dataset.

3.2 Sources

The data collected in this research is made up of information from the shares publicly listed on the FTSE All-Share index. The FTSE All-Share is a British stock index that represents the performance of all eligible companies listed on the London Stock Exchange (LSE)[5] main market. It covers 98% of the UK's market capitalization[26].

Two websites were analyzed for collecting the data for the firms listed on the FTSE All-Share index. These are Hargreaves Lansdown[6] and AnnualReports.com[7]. The scraping of these two websites allowed building a dataset containing different financial document types that include annual reports, interim reports, fact sheets, and past 5 years' financial results.

3.2.1 Hargreaves Lansdown

Hargreaves Lansdown (HL) is a UK-based platform for private investors and, as such, it provides information on world and UK stock markets, including the FTSE All-Share index. HL's website displays financial information for each of the listed firms including documents as annual reports, interim reports, fact sheets and financial results for the last 5 years. The annual and interim reports, as well as the fact sheets, are provided in a PDF file format while the financial results for the past 5 years are presented in HTML format. The documents collected from HL's service are, according to availability, the most recent annual and interim reports, fact sheets, and financial results table for the past 5 years.

The crawling of the pages and documents from HL's service was done according to the site's `robots.txt`[8] file. At the moment of crawling, the file defined no limitations to any of the requested pages and had no crawl-delay[9] directive.

[5] https://www.londonstockexchange.com/indices/ftse-all-share
[6] https://www.hl.co.uk
[7] https://www.annualreports.com
[8] https://developers.google.com/search/docs/crawling-indexing/robots/intro
[9] https://www.cloudflare.com/learning/bots/what-is-robots-txt/

3.2.2 AnnualReports.com

AnnualReports.com is a free internet service that provides an online updated listing of annual reports. This service provides annual reports for multiple stock exchanges, including the LSE . Unlike HL's service, AnnualReports.com provides, if available, the most recent annual report along with previous year's reports. All the reports are provided in PDF file format. The documents collected from AnnualReports' service are, according to availability, annual reports prior to the most recent. This was decided as the most recent annual report was being collected from HL's service.

Similar to HL's service, the crawling of AnnualReports.com's pages was done in accordance with the `robots.txt` file of the site. The file presented no limitations to any of the crawled sites at the moment of crawling. However, a `Crawl-delay` directive was present limiting the number of requests to one every 10 seconds.

3.3 Structuring the Data

The data collected from both HL and AnnualReports.com was presented mostly in PDF format except for the past 5 years' financial results that were available in HTML format. Storing such a large number of files required a large amount of storage capacity. To reduce the storage required, the text content of the files was extracted while being downloaded before saving them into storage. This decision was made considering that LLMs only process text information, so the text information is always extracted from the PDFs before querying the LLMs. By doing this beforehand, not only is storage space reduced but also the LLM querying algorithm gets simplified by avoiding the extraction of text from PDFs.

Having done the decision to extract text from PDFs during the scraping process, it was necessary to choose the format in which the text will be stored. It was considered the option of extracting plain text, but this meant that formatting information such as section headers and tables will be lost, and in large documents this information might be highly valuable. Therefore, the Markdown text format was considered.

3.3.1 The Markdown Text Format

Markdown is a markup language that allows to format plain text with simple syntax[27]. A commonly accepted specification of Markdown is CommonMark[28]. Markdown is commonly used for structuring documents in platforms such as GitHub or Jupyter

notebooks, and it can provide multiple benefits when working with LLMs or Retrieval-Augmented Generation (RAG) systems. Using Markdown allows structuring the text into headers, lists, tables and other structured elements, allowing to preserve context. It also allows enriching the text using bold, italics, links and code blocks, enhancing the context for the models. Using Markdown for input documents allows *chunking* (splitting a text into smaller texts) large documents while keeping text with common context together[28]. Another benefit of using Markdown is that the text is human-readable and easily rendered into HTML facilitating the user interaction with the text.

Considering the previously listed benefits, the CommonMark Markdown format was chosen for extracting the PDF documents' information. To achieve this, assistance from the `PyMuPDF4LLM`[10] python library was used. `PyMuPDF4LLM` is a tool for extracting PDF text data. It allows extracting PDF text information in Markdown or LlamaIndex format. This approach covered most of the collected information excepting the past 5 years' financial results which were collected in HTML format. To standardize the format of the data, these HTML documents were also parsed into Markdown format leveraging the `Markdownify`[11] library, which is a tool for converting HTML into Markdown.

3.3.2 The JSONLines file format

Once the format for the data collection was decided, a storing strategy was required. For this purpose, the JSONLines[12] file format, also known as newline-delimited JSON (NDJSON) was chosen. The JSONLines format is a file format in which every line is a valid JSON object[29]. This format was chosen because it allows storing the documents and some metadata in a structured way while also keeping all the files together. Listing 3.1 is an example of how a JSONLines file looks like.

```
{"title": "1Spatial PLC","ticker": "SPA","year": "2019", ...}
{"title": "1Spatial PLC","ticker": "SPA","year": "2018", ...}
{"title": "1Spatial PLC","ticker": "SPA","year": "2017", ...}
```

Listing 3.1: Example .jsonl file content

[10]https://pypi.org/project/pymupdf4llm/
[11]https://pypi.org/project/markdownify/
[12]https://jsonlines.org/

3.4 Cleaning the Data

The process of collecting the documents from both sources took around a week and a half due to the `Crawl-delay` directive of AnnualReports.com and the processing time required to extract the Markdown formatted text from the PDF files. The error log of the collection script reported that 736 documents were not collected due to a problem either requesting them or extracting the content. The resulting dataset contained 15,020 documents; however, some documents were found to contain little to no content and some other metadata fields like the company ticker required some extra processing to clean the collected data.

Reviewing the content of each document manually resulted in a colossal task. To simplify the process of identifying empty documents, the number of tokens per document was calculated. This process was done by leveraging the `Tiktoken`[13] python library. The calculation was done with the `cl100k_base` encoding, which is the encoding used for OpenAI's `gpt-3.5` and `gpt-4` models[30]. The number of tokens per document was leveraged to identify issues with the data quality.

It was identified that documents with low number of tokens presented low quality data. A regular expression was executed to identify documents that contained no words, which returned a total of 188 documents. The collected information in these documents was blank space or unknown characters due to bad performance of the information extraction process through the `PyMuPDF4LLM` library. These documents were removed from the final dataset. After this, some documents remained with a significant small number of tokens. These documents where examined manually to identify if they presented any issues in data quality. A total of 62 documents presented empty content with the tokens corresponding to page breaks or to isolated words or numbers with not enough context to express any meaning. These documents included 1 fact sheet, 57 annual reports, and 4 interim reports. They were removed from the final dataset. After this, three duplicated instances of documents were identified. The duplicates were removed from the final dataset. Finally, it was identified that 703 documents extracted from AnnualReports.com presented an issue with the ticker value. The issue is that AnnualReports.com adds the suffix `.L` to the ticker values of some LSE stocks to identify them from those of the american market. Because the collected dataset contains only LSE shares, this substring was removed from the ticker values that had it. The resulting dataset after these operations comprised 14,767 financial documents.

[13] https://pypi.org/project/tiktoken/

Document Type	Number of Collected Documents
Interim Report	437
Annual Report	14,150
Fact Sheet	147
Financial Results	33
Total	14,767

Table 3.1: Total Collected Documents

3.5 The Collected Dataset

The final collected dataset is a corpus of 14,767 financial documents. The Table 3.1 shows the number of documents per document type. The dataset is stored in a JSONLines file to facilitate portability. This format also eases the process of loading the data into databases or dataframes for further analysis. Each line of the file is a valid JSON object that contains the properties described in the Table 3.2. An example of a document entry on the JSONLines file is displayed on Listing 3.2

Some statistics were calculated on the number of tokens per document through the dataset. Tables 3.3 and 3.4 display these statistics. The information was leveraged to build Figure 3.1 to have a better insight about the size in tokens of the collected documents. This insight is valuable as the number of input tokens for LLMs is restricted to the model's context window size. It is also important because some LLM providers, such as OpenAI, charge the usage of their models according to the number of input/output tokens.

Property	Description
`document_id`	Generated document identifier (UUID)
`title`	Company's title or name
`ticker`	Company's ticker symbol identifier
`document_type`	Type of the document, could be `annual_report`, `interim_report`, `view_factsheet`, or `financial_results`
`year`	Year to which the document refers to
`tokens`	Number fo tokens in the document (Calculated using the `tiktoken` library with the `cl100k_base` encoding)
`src_url`	The URL of the original document
`doc`	The Markdown formatted document

Table 3.2: Dataset Structure

```
{
   "document_id": "41198fc9-abe4-4e80-acdd-2b58451e6825",
   "title": "Zenith Energy Ltd.",
   "ticker": "ZEN",
   "year": "2019",
   "document_type": "annual_report",
   "tokens": "47418",
   "src_url": "https://www.annualreports.com/HostedData/
      AnnualReportArchive/z/TSX-V_ZEE_2019.pdf",
   "doc": "# ZENITH ENERGY LTD.\n#### ANNUAL REPORT AND FINANCIAL
      STATEMENTS\n..."
}
```

Listing 3.2: Example document

Document Type	Min	Max	Mean	Median	Std. Deviation	Variance
Annual Report	2602	463609	72464.8	56425	52233.7	2.72836e+09
Financial Results	635	2042	1785.45	1811	229.572	52703.1
Interim Report	1667	145583	24345.2	21300	17597.5	3.09672e+08
Fact Sheet	1119	10367	3724.24	3328	1507.7	2.27317e+06

Table 3.3: Document file size statistics by document type

Min	Max	Mean	Median	Standard Deviation	Variance
1	43	9.48426	8	6.39141	40.8502

Table 3.4: Collected documents by firm

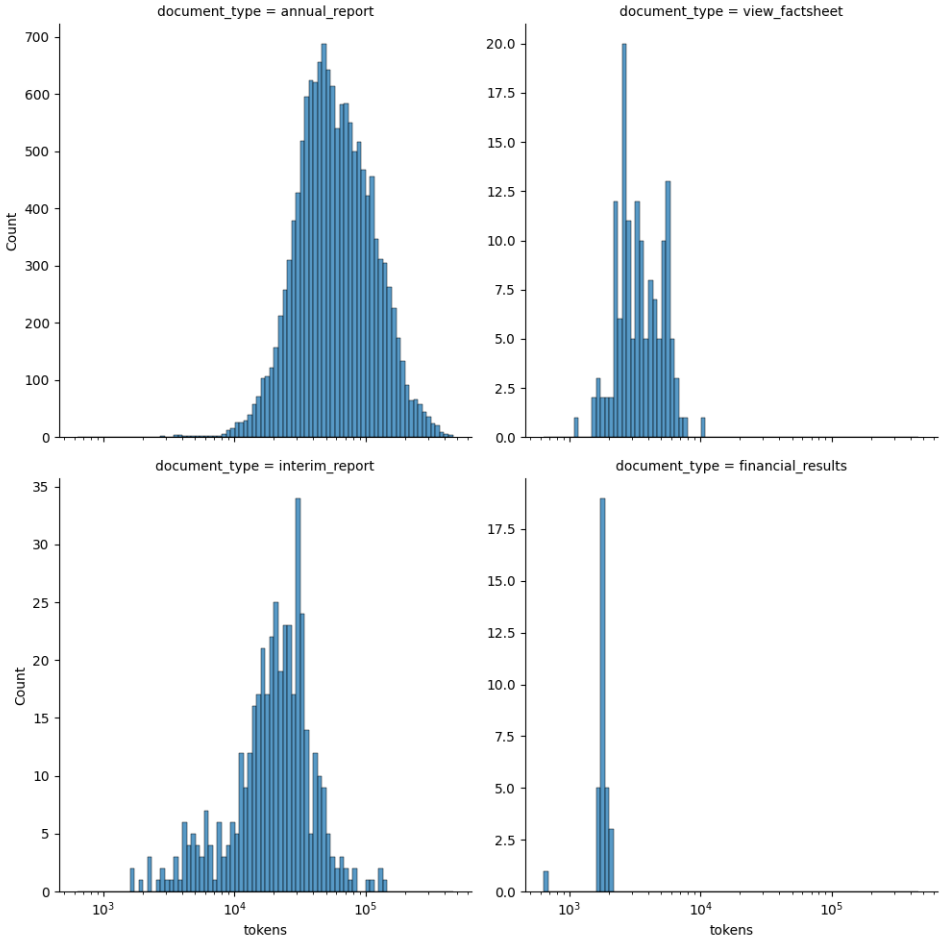

Figure 3.1: Distribution of number of tokens in document by document type

Chapter 4

Methodoogy

Assessing the Foundational LLMs' capacity to summarize or extract financial information effectively is a challenging task. To address this, it is required to appropriately define the LLMs to be queried, the input prompts, a process for querying large documents and a response evaluation process. It is also important to mention that, for the experiments done in this research, a sample dataset was chosen from the larger collected dataset. This was decided based on the limitations of the project, maily because evaluations were done manually. These key aspects are discussed in the sections of this chapter.

4.1 Choosing the LLMs

Multiple LLMs were considered for the experimentation phase of this research. However, access to LLMs is limited to the availability of resources, both computational and economical. The available models for this research were Llama 3, Mistral 7B, and the OpenAI GPT models.

Access to the models `meta-llama/Meta-Llama-3-8B-Instruct`[1] and `mistralai/Mistral-7B-Instruct-v0.3`[2] was obtained through the HuggingFace[3] services. HuggingFace is an AI and Machine Learning service provider. Amongst other services, it provides a service that allows to interact with hosted instances of LLMs, allowing to test models without the need to host them. This service provides limited free access to some models, including the mentioned Llama 3 and Mistral.

[1] https://huggingface.co/meta-llama/Meta-Llama-3-8B-Instruct
[2] https://huggingface.co/mistralai/Mistral-7B-Instruct-v0.3
[3] https://huggingface.co

Model	**Size category**	**Context Size**	**Cost / 1M input tokens**
`Meta-Llama-3-8B-Instruct`	Small	8k	-
`Mistral-7B-Instruct-v0.3`	Small	32k	-
`gpt-4o-2024-08-06`	Large	128K	$2.500
`gpt-4o-mini-2024-07-18`	Small	128k	$0.150

Table 4.1: Candidate LLMs Characteritics

On the other hand, access to the OpenAI API[4] was kindly provided by the University of Edinburgh's School of Informatics. This API provides access to all OpenAI's models[5]. Two OpenAI models were considered for this research, namely `gpt-4o-2024-08-06` and `gpt-4o-mini-2024-07-18` which, at the moment of research, were the most recent large and small OpenAI models. Table 4.1 lists the main characteristics of the models considered for model selection.

Preliminary tests were carried mainly using the Llama 3 model. However, for the execution of experiments, a careful inspection of the models capabilities was made. As shown in Table 4.1, both GPT models have a large context window of 128k tokens which is significantly larger that the 8k and 32k context windows of `Llama-3-8B` and `Mistral-7B`. A large context window is required for this research since the documents from the dataset are mostly large. Therefore, the context window size was the main factor to discard the usage of the Llama and Mistral models.

The distinction between both GPT models relies on their size and cost. Due to the limited resources allocated for this project, cost efficiency was an important factor. Taking into consideration the number of documents and the number of queries to be done about each document, the approximated cost was calculated as $8.57 USD and $0.51 USD for `gpt-4o-2024-08-06` and `gpt-4o-mini-2024-07-18` respectively. The difference is cost is significant, so the `gpt-4o-mini-2024-07-18` model was chosen for the experimentation. This decision allowed allocating extra resources for intermediate tests that were not considered in the cost approximation.

[4]https://platform.openai.com/docs/api-reference/introduction
[5]https://platform.openai.com/docs/models

Model	Context Window Size	Max Output Tokens
gpt-4o-2024-05-13	128,000 tokens	4,096 tokens
gpt-4o-2024-08-06	128,000 tokens	16,384 tokens
gpt-4o-mini-2024-07-18	128,000 tokens	16,384 tokens
gpt-4-turbo-2024-04-09	128,000 tokens	4,096 tokens
gpt-4-0613	8,192 tokens	8,192 tokens

Table 4.2: GPT Models' Context Window Sizes

4.2 Summarizing Large Documents with LLMs

Automatic Text Summarization is one of the tasks that LLMs can perform. To achieve this, models require a prompt that provides the instruction for summarization and the text to be summarized. However, the input of LLMs is limited by its maximum context length or context window, which indicates the maximum number of tokens allowed in the input plus the generated output for a prompt[31]. Table 4.2 displays the context window sizes and the maximum output tokens for some of the GPT models[6], which is the type of model used for this research.

It is important to note that by June 2024, most foundation models had a small context window size varying from 4,096 to 32,768 tokens[31]. Google DeepMind first introduced a large context window in February 2024 when releasing Gemini 1.5. The Gemini 1.5 standard version has a 128K context window, and the Pro version has a 2M token context window. By July 2024, more models have been released with an increased context window size. Models such as GPT-4o, Llama 3.1, and Mistral Large 2 now include a large context window of 128k tokens.

The context window size limits the size of the input given to the model, making it impossible for LLMs to integrate or use the entire information available in long texts. Even that the trend is to increase the context window size, financial documents, especially annual reports, are large in nature and therefore might exceed the context window size. A total of 1,748 documents from the collected dataset exceed the 128k token limit of the large context window size, which represents almost 12% of the whole dataset. Therefore, a special technique is required to overcome this limitation.

LangChain's[7] refine chain was leveraged to summarize large documents that exceed

[6]https://platform.openai.com/docs/models/gpt-4-turbo-and-gpt-4
[7]https://www.langchain.com/langchain

the model's context window. LangChain is a framework that provides a set of tools to develop applications that interact with LLMs[32]. This framework proposes multiple approaches for document summarization with LLMs, including *Stuffing*, *Map-Reduce* and *Refine*. This research focuses on applying the *Refine* chain for large document summarization.

4.2.1 Refine Chain

The Refine[8] chain for large document summarization consists of splitting the large document into smaller documents to then loop over them while querying the model to iteratively update its answer. Through the process, the document is split into smaller input documents, then the first split is queried with an *Initial Prompt* and the response is gathered. Afterwards, the next split and the previous response are included in a new query, now with a *Refine Prompt*. The process repeats until all the splits get queried [33].

Because of the introduction of the response into the input context provided to the LLMs during the Refine steps of this summarizing chain, it was decided to limit the token length of documents to 1,000 tokens. This maximum document size was adequate because the model used for the experiments is `ggpt-4o-mini-2024-07-18` which has a 128k token context window and a maximum output of 16,384 tokens [9]. The document segmentation was done with the help of LangChain's `MarkdownTextSplitter` [10] which is a python module for splitting text that attempts to split along Markdown-formatted headings. The splits made by this library can have an overlap on the tokens to attempt to preserve the context between the splits. This overlap was configured to 5,000 tokens.

4.2.1.1 The Prompts

The LangChain framework uses *prompt templates* to define the structure of a prompt and later uses them to build each of the prompts queried to the LLM. These templates allow defining variables that would later be replaced with content relevant to the step of the chain (e.g., a document or a previous answer). The description of the variables used in the developed system is included in the Table 4.3.

The Refine Chain uses two different prompts templates, the *Initial Prompt* and the

[8] https://python.langchain.com/v0.2/docs/tutorials/summarization/#refine

[9] https://platform.openai.com/docs/models/gpt-4o-mini

[10] https://api.python.langchain.com/en/v0.1/markdown/langchain_text_splitters.markdown.MarkdownTextSplitter.html

Variable	Description
{text}	Indicates where the input document will be inserted into the prompt.
{task}	Indicates where the task to do will be inserted into the prompt.
{existing_answer}	Indicates where a previous answers will be inserted into the Refine prompt.

Table 4.3: Prompt template variables

Refine Prompt. The *Initial Prompt* is used for the first query made to the LLM. Since there is no previous answer available, only the `text` and `task` variables are included. The purpose of this prompt is to get the model to perform an initial task over an initial document or text. The response to this prompt is to be introduced into the context of the next query in an attempt to preserve the answer through the chain. An example of an Initial Prompt template is displayed in the Listing 4.1.

The *Refine Prompt* is used for all the following steps after the initial query. This prompt template, in addition to the `text` and `task` variables, includes the `existing_answer` variable, which allows introducing the answer of the previous query into the prompt. This variable must be present as its absence would result in a series of unrelated queries. An example of a Refine Prompt template is displayed on Listing 4.2.

4.2.1.2 Refine Chain's Output

The Refine chain can be executed once the prompt templates have been defined and the document to summarize has been split into smaller documents. Once the chain is executed, LangChain will collect the outputs and return two main outputs: the final response and a list of intermediate responses. The latter can be leveraged to analyze the refining process while the former is the final answer to the requested task.

4.3 Financial Document Summarization

To leverage the *Refine Chain* to process large documents, such as financial annual reports, a list of tasks and two input prompts were designed. The purpose of these tasks and prompts is to achieve the goal of addressing the LLMs capacity to summarize

financial information, specifically a company's financial strength and management team.

4.3.1 The Tasks

A set of six tasks was designed to analyze both the financial strength and the management team of a company, with three tasks for each of these key aspects. The financial strength tasks focus on the financial metrics, earning updates, and credit scoring, while the management team tasks focus on a company's ability to allocate capital, manage key risks, and respond to changes in the business landscape. The six final tasks to be used in experimentation are:

Financial Strength

- Summarize the financial metrics of the company
- Summarize the earnings updates and/or earnings transcripts of the company
- Score the company for credit positivity/negativity or earnings upgrades/downgrades for equities; explain and justify the scoring

Management Team

- Describe the company's management team's ability to allocate capital effectively
- Describe the company's management team's ability to identify and manage key risks to the business
- Describe the company's management team's ability to respond to changes in the business landscape

4.3.2 The Prompts templates

The *Refine Chain*, as mentioned in section 4.2, requires two prompt templates, namely the *Initial Prompt Template* and the *Refine Prompt Template*. The *Initial Prompt Template* refers to the prompt used for the initial summarization step while the *Refine Prompt Template* is used for all the following summarization steps, known as Refine steps.

Both the *Initial Prompt Template* and the *Refine Prompt Template* were designed through an iterative process, in which the prompt was constantly modified until the

response was closer to the desired output. The final prompt templates are displayed in Listings 4.1 and 4.2. It is worth noting that both prompts use special character delimiters to distinguish the `text` and `existing_answer` variables' contents from the instructions (e.g., the use of `%%%%%` on the prompt templates to delimit the input document content). This delimiter technique showed to help the model properly to identify the context to use for the specified task. The decision to include the `task` variable facilitated the design of two generic prompts that could be leveraged for the six different tasks defined in the previous subsection (4.3.1).

```
Context information is below delimited by %%%%%.

%%%%%
{text}
%%%%%

Given the context information and no prior knowledge, {task}:
```

Listing 4.1: Prompt template for *Initial Prompt*

```
The task is to {task}.
A previous response is provided below delimited by &&&&&.
Context information is below delimited by %%%%%.

&&&&&
{existing_answer}
&&&&&

%%%%%
{text}
%%%%%

Given the context information and the previous response, {task}:
```

Listing 4.2: Prompt template for *Refine Prompt*

4.3.3 Evaluation

Evaluating automatic summarization is a complex task that has been highly researched. Measures such as ROUGE[34] and BLEU[35] have been developed for automatic summarization evaluation. However, these measures are limited and their results do not always align to those of human evaluation. Despite this, ROUGE remains as the primary evaluation metric. In recent years, research has been done regarding summarization evaluation metrics assisted by Natural Language Understanding models, including the usage of QA models for summarization evaluation.

Wang et al. [36] Propose the Question Answering and Generation for Summarization (QAGS) metric for evaluating the factual consistency of summaries, which, according to their research, shows higher correlations with human judgments of factuality in comparison to other metrics such as ROUGE. Inspired by this work, ConfidentAI[11] developed their summarization evaluation metric included in their open-sourced AI evaluation framework *DeepEval* [12].

DeepEval's Summarization evaluation metric[13] aims to evaluate a summary's coverage of the original text details and the factuality of the generated summary by generating closed-ended questions (questions answered by 'yes' or 'no') from either the original document or the summary and using a LLM to answer the question using the other text. This measure uses two scores, the coverage score and the alignment score.

The Coverage Score represents the amount of information from the original text that is included in the summary. It is calculated by generating questions from the original text and asking a LLM to answer them with the summary as reference. All the generated questions should be answered with a 'yes'. The Coverage Score represents the percentage of questions answered correctly. The Alignment Score measures the factuality of the information included in the summary. It is measured by generating questions from the summary and asking a LLM to answer them with the original text as context. As with the coverage score, all questions should be answered with a 'yes'. The Alignment Score is interpreted as the percentage of information in the summary that is true.

This measure was implemented in an attempt to achieve automatic summary evaluation, however, the results showed to be inaccurate, and therefore unreliable, which led to the rejection of this evaluation approach. The fact that the measure proposed by

[11] https://www.confident-ai.com/
[12] https://docs.confident-ai.com/
[13] https://docs.confident-ai.com/docs/metrics-summarization

ConfidentAI showed to have a poor performance for this research might be due to the big difference in size of the original documents and the generated summaries. Another potential reason for this low performance could be the nature of specific information summarization that the previously defined tasks (Section 4.3.1) attempt to achieve.

Even tho this automated approach was discarded, the main idea of the process was leveraged to choose an evaluation measure for this research. Both Coverage and Alignment Scores were considered for the process of manual evaluation. However, implementation of the Coverage Score's manual evaluation was identified as a highly complex task as selecting a number of relevant facts to be included in the summary requires a significant amount of time and specialized knowledge. In contrast, the Alignment Score shows to be more possible for manual evaluation as the whole summary can be fact checked by leveraging the original document. Due to these considerations, the Alignment Score was chosen as the evaluation metric to be used to manually evaluate the generated summaries.

The process applied for measuring the Alignment Score is as follows:

1. Split the generated summaries as a list of claims.
2. Fact-check the list of claims leveraging the original document.
3. Calculate the percentage of true claims from the total of claims. This percentage represents the Alignment Score.

It is important to mention that a pattern was identified where the response structure consisted of an introduction, a body, and a conclusion or summary. In this response structure, the body contained facts extracted from the documents while the introduction and conclusion parts were mainly opinion commentary generated by the model about the information contained in the body. Because the Alignment Score aims to evaluate the factual consistency of the generated response, only the body was considered for evaluation.

For this research, the splitting of the responses into claims considers every sentence to be a claim, excluding the introduction and conclusion or summary sections of the response, as well as any markdown formatted text that indicates content sections.

4.4 The sample dataset

The collected dataset contains a total of 14,767 documents. Analyzing all of these documents through the manual process described in the previous sections of the Chapter

Document ID	Company	Year	Tokens
cc38482d-ae22-42e5-826b-9926bf594722	Pets at Home Group (PETS)	2021	165,237
057b5bf3-61ea-45fb-b2e9-db683745d6b4	Entain Plc (ENT)	2018	123,567
85f11fa0-d70a-497a-b9d7-e3df6344be35	International Distribution Services Plc (IDS)	2023	142,078

Table 4.4: Selected documents for experimentation

4 would require a huge amount of time. Due to this, it is required to select a small sample that accurately represents the data across the dataset.

To find a small set of documents that best represent the data through the collected corpus, a Latent Dirichlet Allocation (LDA) model was trained on the whole dataset. LDA is a generative probabilistic model for collections of discrete data[37]. LDA allows modeling a set of topics present in a corpora, where each of the documents in the corpora has a probability of belonging to each of the topics. After training the model with a total of 15 topics and getting a categorization for each document, topic relevance was calculated by counting the number of documents classified for each topic. The three most relevant topics were selected as a sample from the set of topics. From these three topics, the most relevant document (the document with the highest probability of belonging to the topic) was chosen. This process provided a sample dataset of three documents to use for experimentation. The list of selected documents is displayed on the Table 4.4. It was decided to limit the scope to three documents because each document would be queried six times, one for each of the tasks described in the Section 4.3.1, totaling up to 18 queries that require manual evaluation.

All three selected documents are annual reports because the majority of the dataset consists of this type of document (14,150 annual report documents out of 14,767 total documents). It is important to note that the number of tokens for the Pets at Home Group and International Distribution Services Plc documents are larger than the 128k token context window of the GPT models. Similarly, the Entain Plc document is slightly smaller than the 128k tokens, but still higher than the 100k token limit set for the document spliter described in Section 4.2.1. It is important for this to be highlighted as it provides justification for the usage of the *Refine Chain* for large document summarization.

Chapter 5

Implementation

The refine chain was implemented through a python script with a command line user interface. Since its outputs are a list of long texts, it is challenging to analyze such output. A Graphical User Interface (GUI) was developed with the purpose of facilitating the process of visualizing and analyzing the Refine chain outputs. The features of the developed GUI are:

- Display and search the financial document dataset.
- Preview documents from the dataset (original PDF and Markdown-formatted text).
- Query an LLM about a document allowing to choose a LLM and define the prompt templates.
- Track, display and search the query history.
- Visualize the LLM response, including
  - Input documents (document splits).
  - Individual step responses.
  - Final response.
  - Prompt Templates.
  - Queried document (original PDF and Markdown-formatted text).
- Allow adding comments when visualizing a response.

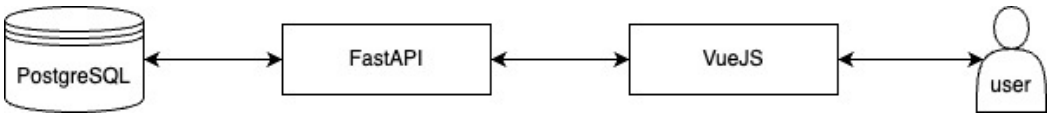

Figure 5.1: Architechture of Web Application

5.1 Web Application

It was decided to build the GUI as a web application for portability and for leveraging the familiarity with the technologies for a fast development. The development of the application was split into two sections: *Frontend* and *Backend*. The former is composed of a Vue.js[1] application while the latter is made up of a PostgreSQL[2] database and a FastAPI[3] RESTfull[4] Application Programming Interface (API). The solution is orchestrated using Docker's[5] `docker compose`[6] tool, making it a portable application. A diagram of the application's architecture is displayed on the Figure 5.1.

5.1.1 Backend

The server-side aspect of web development is referred to as Backend. It is focused on creating and managing server logic, databases, and APIs[38]. The GUI application developed for this research requires a backend composed of data storage and an API for accessing the data on such storage.

The storage used for this application consists of a PostgreSQL database. This database contains the information collected through the process described in the Chapter 3, the query results, and the comments. The reason to store these data in a relational database is that all information was initially handled in jsonlines files because the collected dataset was in such a format. However, the performance for retrieving documents was poor, so the information was migrated to the database.

The FastAPI framework was chosen to develop the backend main service because it is a fast python web framework. The fact that the framework works with the python

[1] https://vuejs.org/
[2] https://www.postgresql.org/
[3] https://fastapi.tiangolo.com/
[4] https://www.ibm.com/topics/rest-apis
[5] https://www.docker.com/
[6] https://docs.docker.com/compose/

language allows seamlessly integrating the previously developed scripts for querying LLMs to the REST API. The API allows the frontend application to interact with the data as required, including

- Fetching the available documents list and query responses list.
- Fetching a document's detailed information.
- Fetching the query response history.
- Fetching a query response's detailed information.
- Requesting the execution of a query.
- Adding comments to query responses.

5.1.2 The Frontend Application

Frontend or Front-end development is the development of the user interactive part of a website [39]. The frontend application developed for this research provides users with a GUI that allows interaction with the collected data.

Vue.js is a versatile framework for building web user interfaces. It was chosen as the main tool for developing the user interface. The reason for this is that It allows fast development and the integration of libraries such as PrimeVue[7], which allows including prebuilt components such as data-tables (used to display data in tabular format).

The frontend application consists of four main views:

- Document list (Figure 5.2)
- Document detail and query model form (Figures 5.3 and 5.4)
- Query response list (Figure 5.5)
- Query response detail (Figure 5.6)

[7] https://primevue.org/

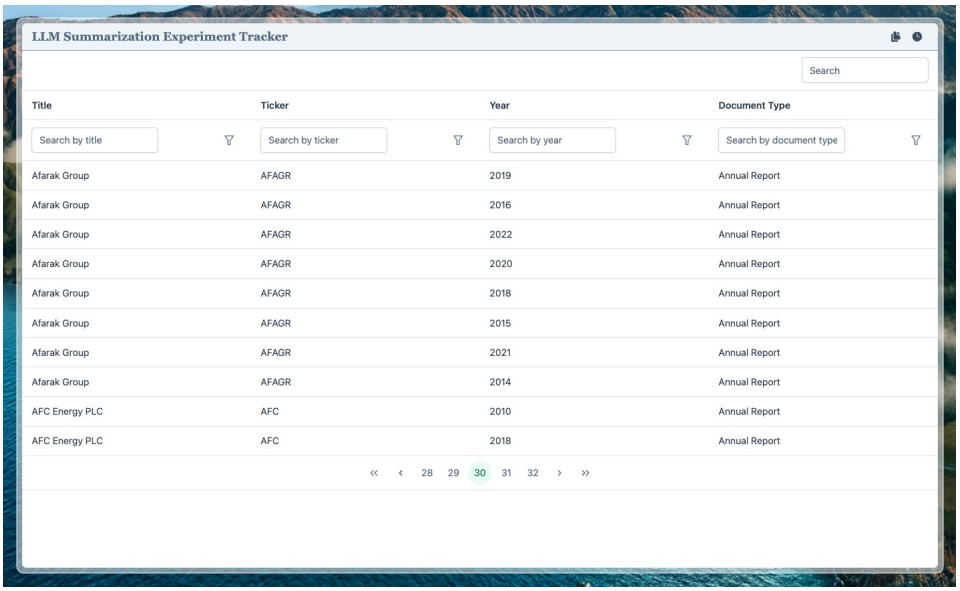

Figure 5.2: Document List

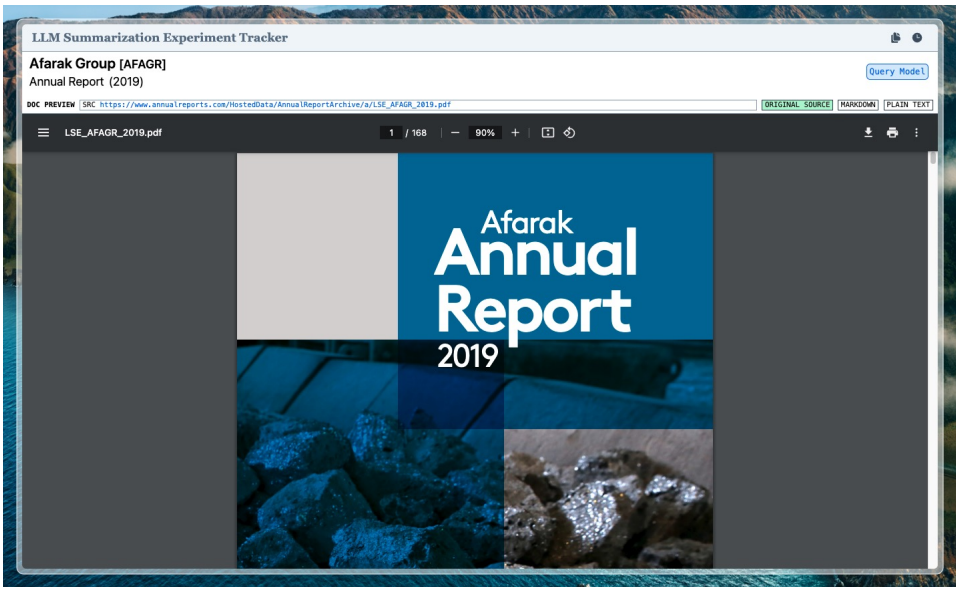

Figure 5.3: Document detail (Original PDF preview)

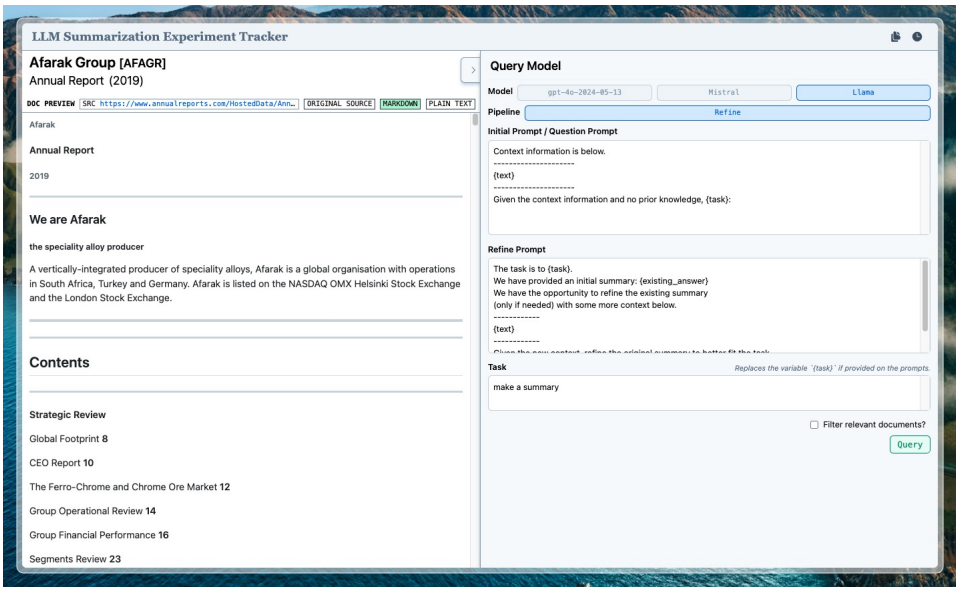

Figure 5.4: Document detail (rendered Markdown preview) and query model form

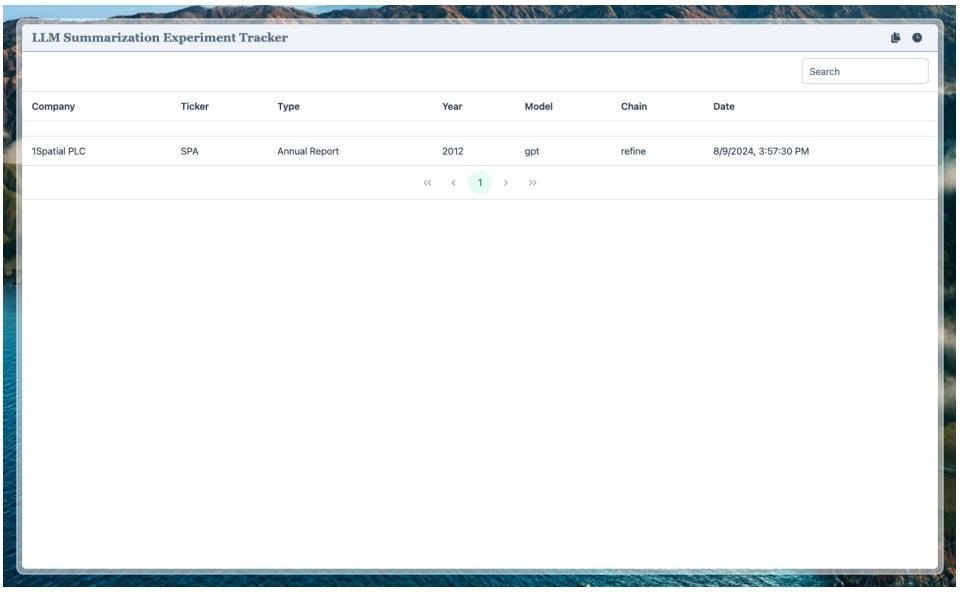

Figure 5.5: Query response list

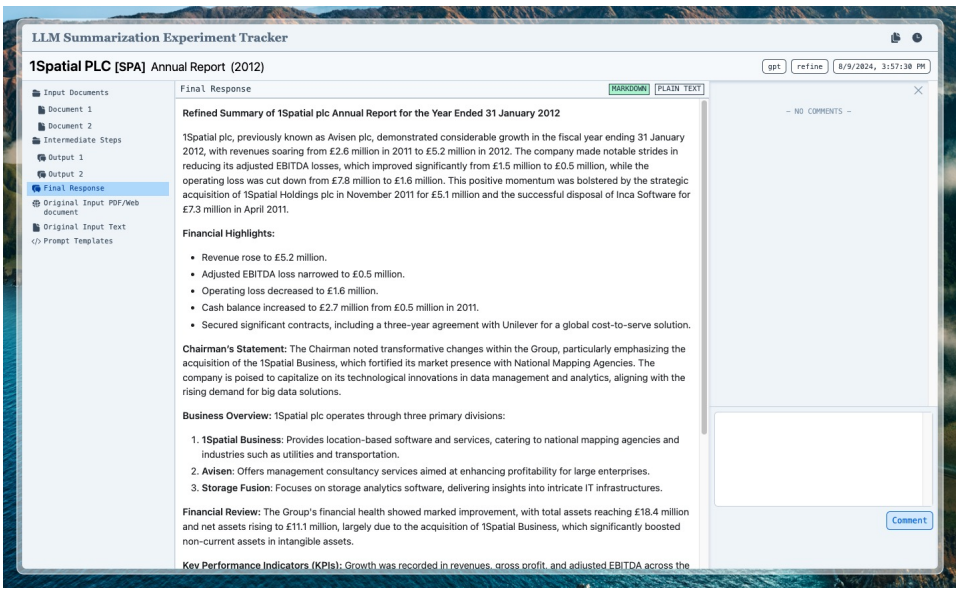

Figure 5.6: Query response detail (displaying final answer)

5.2 Potential Improvements

The GUI application has shown to be useful when navigating the dataset through the document list, querying models, and keeping track of query history. However, there are some areas of opportunity that working on them could potentially benefit the user experience, including

- Integrating other summarization methods/chains like map-reduce for greater versatility when querying the LLMs.
- Integrate more models to experiment with.
- Add feature to export or download the query results for further analysis.

Chapter 6

Results and Discussion

This chapter presents the gathered results from the execution of the experiment described on Chapter 4. Firstly, the alignment score results for the experimentation are introduced. This is followed by the discussion of the responses' quality and relevancy. Finally, other observations are presented.

To facilitate the presentation of results, a task key has been assigned to each of the tasks defined on Section 4.3.1. Table 6.1 lists these keys and their corresponding task.

Task Key	**Task**
FS1	Summarize the financial metrics of the company
FS2	Summarize the earnings updates and/or earnings transcripts of the company
FS3	Score the company for credit positivity/negativity or earnings upgrades/downgrades for equities; explain and justify the scoring
MT1	Describe the company's management team's ability to allocate capital effectively
MT2	Describe the company's management team's ability to identify and manage key risks to the business
MT3	Describe the company's management team's ability to respond to changes in the business landscape

Table 6.1

Pets at Home Group (PETS) Annual Report 2021				
Task	**True claims**	**False Claims**	**Total Claims**	**Alignment Score**
FS1	12	0	12	1.000
FS2	11	2	13	0.846
FS3	12	0	12	1.000
MT1	16	0	16	1.000
MT2	20	0	20	1.000
MT3	17	0	17	1.000

Table 6.2: Alignment scoring for queries regarding the Pets at Home Group Annual Report 2021

Entain Plc (ENT) Annual Report 2018				
Task	**True claims**	**False Claims**	**Total Claims**	**Alignment Score**
FS1	20	0	20	1.000
FS2	7	3	10	0.700
FS3	19	0	19	1.000
MT1	14	1	15	0.933
MT2	16	0	16	1.000
MT3	16	0	16	1.000

Table 6.3: Alignment scoring for queries regarding the Entain Plc Annual Report 2018

International Distribution Services Plc (IDS) Annual Report 2023				
Task	**True claims**	**False Claims**	**Total Claims**	**Alignment Score**
FS1	17	0	17	1.000
FS2	21	0	21	1.000
FS3	19	0	19	1.000
MT1	19	0	19	1.000
MT2	22	0	22	1.000
MT3	16	0	16	1.000

Table 6.4: Alignment scoring for queries regarding the International Distribution Services Plc Annual Report 2023

Task	Average Number of Claims	Average Alignment Score
FS1	15	1.000
FS2	15	0.849
FS3	17	1.00
MT1	17	0.978
MT2	20	1.000
MT3	17	1.000

Table 6.5: Average number of claims and alignment score per task

6.1 Alignment Score Results

Results were manually evaluated after querying the GPT model regarding the six tasks for each of the three documents. Each of the claims presented in the responses was fact checked against the original document by leveraging the GUI. Tables 6.2, 6.3, and 6.4 show the alignment results for each document.

Alignment scores are presented as a 0 to 1 value, where 1 represents a completely true summary. After analyzing the collected results, it can be commented that the overall score is high, with 15 queries out of 18 (0.833%) having a score of 1. The remaining three queries have a score equal or greater than 0.7, which is still a significantly high score. The fact that two out of three FS2 queries present false claims showcases a potential struggle of the model to factually address this task. Further experimentation is encouraged to properly identify a potential pain point of the model.

Table 6.5 displays the average for both the total number of claims and the alignment score, highlighting the overall high score across all tasks. The average for the total number of claims shows that the size of the responses is similar in size, independently of the task or document.

6.2 Quality and relevancy of the responses

After manual evaluation of each response, it can be said that the overall quality of the responses is good. This is supported by the high alignment score across all responses indicating that they are factually correct.

The financial strength task responses display an understanding of key financial figures and the relations between them. On the other hand, the management team task

responses showed understanding of the company's contexts and how it influenced its decisions.

Each response presented relevant and true information in an adequate format according to the task. It is relevant to mention that the spotted false claims were mostly incomplete truths (e.g., missing data in a list) or misinterpreted numbers from information presented in a confusing manner even for humans.

Event tho the responses were presented in an appropriate format, the format was format was not the same for the same task in a different document. For instance, TS3 asks the model to give the company a credit scoring, for which one document got a numeric scoring, while the remaining two only got a tag of positive or negative. This can be potentially enhanced by leveraging few shot prompts to include examples of the expected response format.

6.3 Other Observations

The sample dataset showed different sentiments across the documents. The PETS document was positive as their results promised a bright future after recovering from the effects of the COVID pandemic. In contrast, IDS was a negative document due to a severe decrease in profit due to high losses in one of the inner divisions. The remaining document, ENT, showed to be neutral with uncertainty in the near future due to recent acquisitions that elevated the company's debt but at the same time promised a high growth for the following years. The responses managed to replicate the sentiment expressed in the section of the document that contained the relevant information.

All the responses follow the structure consisting of an introduction, a body and a conclusion or summary. The introduction and conclusion segments of the response were always generated opinions written in paragraphs, while the body contained the facts that supported such opinions. The body of the responses for the financial strength tasks is usually formatted in bullet point lists while for the management team ones is in paragraphs. This showcases an understanding of data content and shows skills for presenting data in a clear and appropriate format according to the needs.

Through the querying process, the documents were split as follows: 8 splits for PETS, 6 splits for ENT, and 7 splits for IDS. The number of splits appears to not have any effect on the quality of the responses, as the intermediate steps show that the model managed to adequately update the response with each step. The number of splits indicates that the whole context window was not used on each query. However, the

larger context window allowed having less number of splits, reducing the number of calls to the model's API.

The Refine Chain for large document summarization showed to have no negative effect on the final response. Even tho must of the response's information seemed to come from the firsts splits, it was identified that key relevant information from the last splits was still being included in the final answers.

Chapter 7

Conclusions and Future Work

7.1 Conclusions

The results presented in the previous chapter showcase OpenAI's `GPT-4o-mini` pre-trained model's capacity for achieving financial document summarization. The overall high alignment score across all queries demonstrates that the model is capable of extracting and summarizing financial information. The gathered responses display the model's rational capacity to understand and reply to specific financial tasks such as identifying a company's risks or financial metrics. Long document summarization was achieved by leveraging the Refine Chain, which showed to have no negative effect on the final responses.

Two side products were developed for this dissertation: a financial document dataset and a graphical user interface for querying LLMs. The dataset consists of 14,767 financial documents, from which 437 are Interim reports, 147 are fact sheets, 33 are financial results from the previous five years, and 14,150 are annual reports. This is an important contribution since most of the available datasets are built over small documents. Having a large document dataset opens the door for experimentation with this type of data. On the other hand, the developed GUI showed to be a useful tool for interacting with the dataset, querying the model, and interacting with the query response. Its user-friendly interface allowed for easy exploration of the original documents while comparing to the responses during the evaluation process. The commenting feature was particularly helpful to keep track of the evaluation process.

7.2 Future Work

Further experimentation is required to better assess the model's capacity to achieve financial document summarization. Gathering a larger number of results can provide a better evaluation of the model's performance regarding the tasks defined in Section 4.3.1. The GUI can be leveraged while executing these experiments and more, since the scope of the tasks can be expanded to more financial aspects other than the financial strength and the management team of a company.

The results of this work could be improved by enhancing the prompts for querying the LLMs by implementing few-shot or chain of though prompting. This has the potential to improve the answers by aiming for standardization of formats and asking for required information.

Another potential work is the benchmarking of the Refine Chain against the Map-Reduce Chain. Both have a focus on large document summarization and their results might differ in quality.

The collected dataset has the potential to be leveraged to build task-specific training datasets for training financial domain-specific LLMs. It can not only be used for model training but also for evaluation datasets. By automating the extraction of key fragments from the annual reports, the dataset can be leveraged to build other datasets, like financial statements datasets.

Bibliography

[1] Yi Yang, Mark Christopher Siy UY, and Allen Huang. Finbert: A pretrained language model for financial communications. 6 2020.

[2] Yuqi Nie, Yaxuan Kong, Xiaowen Dong, John M. Mulvey, H. Vincent Poor, Qingsong Wen, and Stefan Zohren. A survey of large language models for financial applications: Progress, prospects and challenges. 6 2024.

[3] K. R. Chowdhary. *Natural Language Processing*, pages 603–649. Springer India, 2020.

[4] Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, Yifan Du, Chen Yang, Yushuo Chen, Zhipeng Chen, Jinhao Jiang, Ruiyang Ren, Yifan Li, Xinyu Tang, Zikang Liu, Peiyu Liu, Jian-Yun Nie, and Ji-Rong Wen. A survey of large language models. 3 2023.

[5] Jerry A Viscione. *Financial analysis : tools and concepts*. National Association of Credit Management, 1st edition, 1984.

[6] Alicia Tuovila, David Kindness, and Suzanne Kvilhaug. Financial analysis: Definition, importance, types, and examples, 6 2024.

[7] Huaqin Zhao, Zhengliang Liu, Zihao Wu, Yiwei Li, Tianze Yang, Peng Shu, Shaochen Xu, Haixing Dai, Lin Zhao, Gengchen Mai, Ninghao Liu, and Tianming Liu. Revolutionizing finance with llms: An overview of applications and insights. 1 2024.

[8] Jean Lee, Nicholas Stevens, Soyeon Caren Han, and Minseok Song. A survey of large language models in finance (finllms). 2 2024.

[9] Yinheng Li, Shaofei Wang, Han Ding, and Hang Chen. Large language models in finance: A survey. In *4th ACM International Conference on AI in Finance*, pages 374–382. ACM, 11 2023. ISBN 9798400702402. doi: 10.1145/3604237.3626869.

[10] Pekka Malo, Ankur Sinha, Pyry Takala, Pekka Korhonen, and Jyrki Wallenius. Good debt or bad debt: Detecting semantic orientations in economic texts. 7 2013.

[11] Macedo Maia, Siegfried Handschuh, André Freitas, Brian Davis, Ross McDermott, Manel Zarrouk, and Alexandra Balahur. Www'18 open challenge. In *Companion of the The Web Conference 2018 on The Web Conference 2018 - WWW '18*, pages 1941–1942. ACM Press, 2018. ISBN 9781450356404. doi: 10.1145/3184558.3192301.

[12] Zhiyu Chen, Wenhu Chen, Charese Smiley, Sameena Shah, Iana Borova, Dylan Langdon, Reema Moussa, Matt Beane, Ting-Hao Huang, Bryan Routledge, and William Yang Wang. Finqa: A dataset of numerical reasoning over financial data. 8 2021.

[13] Rajdeep Mukherjee, Abhinav Bohra, Akash Banerjee, Soumya Sharma, Manjunath Hegde, Afreen Shaikh, Shivani Shrivastava, Koustuv Dasgupta, Niloy Ganguly, Saptarshi Ghosh, and Pawan Goyal. Ectsum: A new benchmark dataset for bullet point summarization of long earnings call transcripts. 10 2022.

[14] Agam Shah, Ruchit Vithani, Abhinav Gullapalli, and Sudheer Chava. Finer: Financial named entity recognition dataset and weak-supervision model. 2 2023.

[15] Soumya Sharma, Tapas Nayak, Arusarka Bose, Ajay Kumar Meena, Koustuv Dasgupta, Niloy Ganguly, and Pawan Goyal. Finred: A dataset for relation extraction in financial domain. 6 2023.

[16] Simerjot Kaur, Charese Smiley, Akshat Gupta, Joy Sain, Dongsheng Wang, Suchetha Siddagangappa, Toyin Aguda, and Sameena Shah. Refind: Relation extraction financial dataset. 5 2023. doi: 10.1145/3539618.3591911.

[17] Willy Au, Abderrahim Ait-Azzi, and Juyeon Kang. Finsbd-2021: The 3rd shared task on structure boundary detection in unstructured text in the financial domain. In *Companion Proceedings of the Web Conference 2021*, pages 276–279. ACM, 4 2021. ISBN 9781450383134. doi: 10.1145/3442442.3451378.

[18] Mahmoud El-Haj, Ahmed AbuRa'ed, Marina Litvak, Nikiforos Pittaras, and George Giannakopoulos. The financial narrative summarisation shared task (fns 2020). In Dr Mahmoud El-Haj, Dr Vasiliki Athanasakou, Dr Sira Ferradans, Dr Catherine Salzedo, Dr Ans Elhag, Dr Houda Bouamor, Dr Marina Litvak, Dr Paul Rayson, Dr George Giannakopoulos, and Nikiforos Pittaras, editors, *Proceedings of the 1st Joint Workshop on Financial Narrative Processing and MultiLing Financial Summarisation*, pages 1–12. COLING, 12 2020. URL https://aclanthology.org/2020.fnp-1.1.

[19] Samir Abdaljalil and Houda Bouamor. An exploration of automatic text summarization of financial reports. In Chung-Chi Chen, Hen-Hsen Huang, Hiroya Takamura, and Hsin-Hsi Chen, editors, *Proceedings of the Third Workshop on Financial Technology and Natural Language Processing*, pages 1–7. -, 8 2021. URL https://aclanthology.org/2021.finnlp-1.1.

[20] Moreno La Quatra and Luca Cagliero. End-to-end training for financial report summarization. In Dr Mahmoud El-Haj, Dr Vasiliki Athanasakou, Dr Sira Ferradans, Dr Catherine Salzedo, Dr Ans Elhag, Dr Houda Bouamor, Dr Marina Litvak, Dr Paul Rayson, Dr George Giannakopoulos, and Nikiforos Pittaras, editors, *Proceedings of the 1st Joint Workshop on Financial Narrative Processing and MultiLing Financial Summarisation*, pages 118–123. COLING, 12 2020. URL https://aclanthology.org/2020.fnp-1.20.

[21] David Krause. Large language models and generative ai in finance: An analysis of chatgpt, bard, and bing ai. *SSRN Electronic Journal*, 2023. ISSN 1556-5068. doi: 10.2139/ssrn.4511540.

[22] Alex Kim, Maximilian Muhn, and Valeri Nikolaev. Bloated disclosures: Can chatgpt help investors process information? 6 2023.

[23] Alex Kim, Maximilian Muhn, and Valeri Nikolaev. From transcripts to insights: Uncovering corporate risks using generative ai. 10 2023.

[24] Antonio Jimeno Yepes, Yao You, Jan Milczek, Sebastian Laverde, and Renyu Li. Financial report chunking for effective retrieval augmented generation. 2 2024.

[25] Vlad Krotov and Leigh Johnson. Big web data: Challenges related to data, technology, legality, and ethics. *Business Horizons*, 66:481–491, 7 2023. ISSN 00076813. doi: 10.1016/j.bushor.2022.10.001.

[26] FTSE Factsheet. Ftse all-share indices, 2008.

[27] Harald Lieder. Rag/llm and pdf: Conversion to markdown text with pymupdf, 4 2024.

[28] John MacFarlane, Martin Woodward, and Jeff Atwood. Commonmark a strongly defined, highly compatible specification of markdown, 8 2024.

[29] Ian Ward. Json lines, 8 2024.

[30] Ted Sanders. How to count tokens with tiktoken, 12 2022. URL `https://cookbook.openai.com/examples/how_to_count_tokens_with_tiktoken`.

[31] IBM. Techniques for overcoming context length limitations, 6 2024. URL `https://dataplatform.cloud.ibm.com/docs/content/wsj/analyze-data/fm-context`

[32] LangChain. Langchain, 7 2024. URL `https://github.com/langchain-ai/langchain`.

[33] LangChain. Summarize text, 2024. URL `https://python.langchain.com/v0.2/docs/tutorials/summarization/#refine`.

[34] Chin-Yew Lin. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81, 2004.

[35] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318, 2002.

[36] Alex Wang, Kyunghyun Cho, and Mike Lewis. Asking and answering questions to evaluate the factual consistency of summaries. 4 2020.

[37] David M Blei, Andrew Y Ng, and Michael I Jordan. Latent dirichlet allocation. *Journal of machine Learning research*, 3:993–1022, 2003.

[38] roadmap.sh. Backend developer, . URL `https://roadmap.sh/backend`.

[39] roadmap.sh. Frontend developer, . URL `https://roadmap.sh/frontend`.