Fall 2023 Capstone Project
Progress Report 1

Reinforcement Learning for Generative AI LLMs

Oct 21, 2023

Group 13

Yi Lu (yl5118)
Xiaolin Sima (xs2483)
Yanni Chen (yc4179)
Junyuan Huang (jh4608)
Michelle Sun (ms6514)

Table of Contents

1. Introduction of Context and Project Goal

This capstone project is a collaboration between Columbia University Data Science Institute and Accenture. The objective of this Capstone Project is to unlock significant insights into the Oil and Gas Industry, emphasizing the critical role of data-driven decision-making through the use of AI such as Large Language Models (LLMs). Company documentation such as annual reports and sustainability reports can range from 50 to 500 pages and contain large amounts of both numeric and textual information. With the help of LLMs, the process of understanding how the company performs can be really efficient.

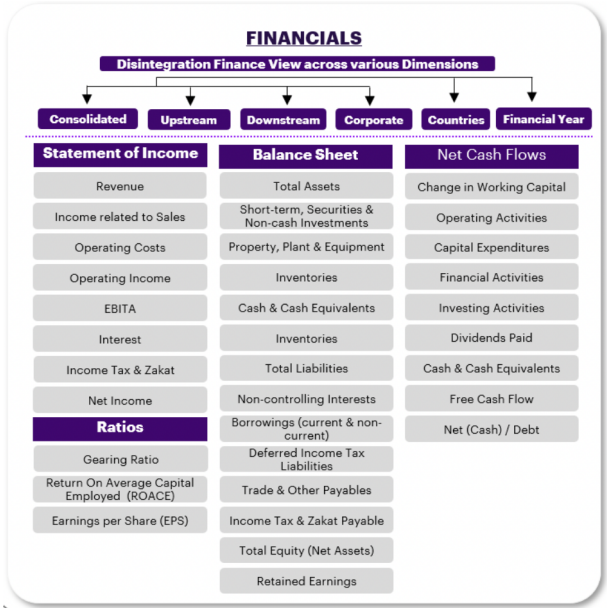
By extracting and analyzing data from some important Key Performance Indicators (KPIs) of the industry through LLMs, we hope to drive valuable insights into how a company performs. With LLMs and methodologies such as Reinforcement Learning with Human Feedback (RLHF) and Retrieval-Augmented-Generation (RAG), we are able to perform data extraction, document summarization, and conversational question-answering, allowing us to gain deep insights from large amounts of company reports.

2. Research

2.1. KPIs

Below are the illustrations of key KPIs that our LLM model is expected and recommended to identify. We will break the KPIs into two big categories: Financial KPIs and Non-Financial KPIs. As for the specific field of the oil industry, we are also interested in upstream, midstream, and downstream productions. For the Financial KPIs, we will mainly extract them from the three financial statements: the Income Statement, the Balance Sheet, and the Statement of Cash Flows, since they mainly cover all of them. Also, we are interested in the non-financial metrics that break

into three fields: Operations, Safety, and ESG (Environmental Social, and Governance) which we can mainly extract from text and small tables.



2.2. Vector Databases

As we are dealing with pdf data, embedding is needed to represent data in a lower-dimensional vector space, making it easier to work with and analyze using machine learning algorithms. After we have our embedded chunks from the raw pdf data, we need to index (store) them somewhere so that we can retrieve them quickly for inference, where the vector database is needed. In this section, we investigate different vector databases, and compare the vector libraries and vector databases, in order to find an optimal solution to easily store and retrieve the data.

2.2.1. Comparison between Vector Libraries and Vector Database

Vector Libraries

Vector libraries store vector embeddings in in-memory indexes, in order to perform similarity-search. Most vector libraries share the following characteristics:

1. Store vectors only
2. Index data is immutable
3. Query during import limitation

Vector Databases

One of the core features that sets vector databases apart from libraries is the ability to store and update your data. Vector databases have full CRUD (create, read, update, and delete) support that solves the limitations of a vector library. Additionally, databases are more focused on enterprise-level production deployments.

2.2.2. Pros and Cons of Different Vector Database/Libraries

Name	Pros	Cons
ChromaDB	- Easy application and integration - Hands-on experience - Embedding model can be customized	- May struggle with a large-scale dataset - Limited customization for data modeling and querying (may not be the main concern)
Milvus	- Optimized for similarity search with advanced indexing techniques to speed up search operations - Allow large datasets and high query loads - Can leverage GPU resources, leading to significant speed improvements - Specialized in similarity search and handle embedding vectors converted from unstructured data (built to pair with FAISS)	- Setting up and configuring Milvus can be complex - May need to fine-tune Milvus, select appropriate indexing methods, and configure hardware resources like GPUs. This optimization process can be challenging and time-consuming. - Data is often assumed to be static once fed into existing systems, complicating processing for dynamic data
pgvector	- Easy to integrate with PostgreSQL	- Most cloud offerings of PostgreSQL have not yet integrated pgvector - Not that convenient if we are not going to use PostgreSQL

Qdrant	- Fast and reliable even under high load - Supports metadata filtering like ChromaDB, and integrates into technologies like Cohere (embeddings), LangChain, and LlamaIndex	- Open-source vector database written in Rust - Not entirely free
Pinecone	- Offers blazing-fast search capabilities, allowing users to retrieve similar vectors in real-time, making it well-suited for content-based searching - Excellent choice to deal with vast amount of data due to its architecture design - Relatively easy to apply with automatically indexes vectors	- Might lack some advanced querying capabilities that certain projects require - Not open-source but has a free version that search through roughly a million vectors in around 100ms, or through 100K vectors in around 20ms
FAISS	- Allows developers to quickly search for embeddings of multimedia documents that are similar to each other - Contains supporting code for evaluation and parameter tuning	- Not a database, but a library

According to our research, we think ChromaDB and FAISS can be two possible options that are worth trying and comparing when we store and retrieve data with designed queries. They are both easy to implement and are able to perform the essential functions we need.

2.3. Open-source Large Language Models

Name	Introduction	Pros	Cons
T5	The T5 series of models open-sourced by Google is available in various sizes of parameters. It is a Sequence to Sequence model that	- A great model to fine-tune at a relatively low cost - Wide variety of model sizes	- Requires a large amount of computational power and memory - May generate

	follows an Encoder and Decoder architecture.	- Well suited for translation and summarization	unreliable results with new inputs - Long training time
Falcon	Falcon is a causal decoder-only model built by TII and trained on tokens of RefinedWeb enhanced with curated corpora to use on summarization, text generation, chatbot, etc.	- Strong conversational capabilities and performance optimization, well-suited for conversational experiences	- Falcon contains fewer parameters than GPT so less complexity and capacity to capture and generate human-like text
Llama	Developed by Meta, Llama is an auto-regressive language model that uses an optimized transformer architecture. The tuned versions use supervised fine-tuning and RLHF to align with human preference for helpfulness and safety.	- LLaMA 2 can generate high-quality texts in a matter of seconds. - It uses less computational resources than other LLMs of similar size and complexity	- Fewer parameters, data, context length, and modalities than GPT giving Llama an edge in terms of accuracy, complexity, diversity, and generality of its outputs.
Mistral	The Mistral LLM developed by Mistral AI is a 7-billion-parameter language model engineered for superior performance and efficiency.	- Mistral 7B is better than Llama 2 13B on all benchmarks, has natural coding abilities, and 8k sequence length	- Mistral 7B is a pre-trained base model and therefore does not have any moderation mechanisms
Vicuna	Vicuna is a chat assistant trained by fine-tuning Llama 2 on user-shared conversations collected from ShareGPT by The large model systems organization (LMSYS).	- It achieves more than 90% quality of GhatGPT while outperforming other models like LLaMA in more than 90% of cases according to a non-scientific evaluation	- Vicuna is not good at tasks involving reasoning or mathematics and may have limitations in ensuring the factual accuracy of its outputs

3. Data Preparation

3.1. Source

We systematically gathered our raw pdf data by collecting recent 5-year annual reports, most recent sustainability reports, and 2-3 years quarterly reports directly from the official websites of ten select companies[1]. This data collection process allowed us to access the most up-to-date and accurate data, providing a comprehensive overview of each company's financial performance, strategic initiatives, and commitment to sustainable practices. By sourcing these critical reports directly from the company's own digital platforms, we ensure the authenticity and reliability of the information, enabling us to perform in-depth analyses and benefit from further informed decisions in our ongoing business and investment strategies.

3.2. Load and Split

The process of PDF loading and splitting plays a crucial role in the workflow of PDF information retrieval. Such documents often contain rich and structured content, including text, images, tables, and graphs. The ability to load and parse PDFs enables the Q&A bot to extract valuable insights from various types of data structures embedded. In addition to the loading process, data splitting is equivalently essential for breaking down large documents into smaller, more manageable sections, also called chunks. The splitted chunks are then used as a comprehensive knowledge base for LLM. Different configurations of chunk size, chunk overlap, and separators will form a distinct representation of the original PDF. A successful segmentation makes it easier for the vectorstore to retrieve relevant information quickly and efficiently, improving the overall performance and response time of the system.

Different LLMs have their own input size constraint, which results in different choices on chunk size and number of chunks retrieved per question. Upon testing 21 combinations from 7 PDF loaders, and 3 splitters, we successfully found an efficient setting of loader and splitter. Detailed analysis of loaders and splitters is included in the Q&A section.

---

[1] Shell plc, BP PLC , Saudi Aramco, Chevron, TotalEnergies, Valero Energy, Marathon Petroleum Corporation, Sinopec and PetroChina

4. Modeling

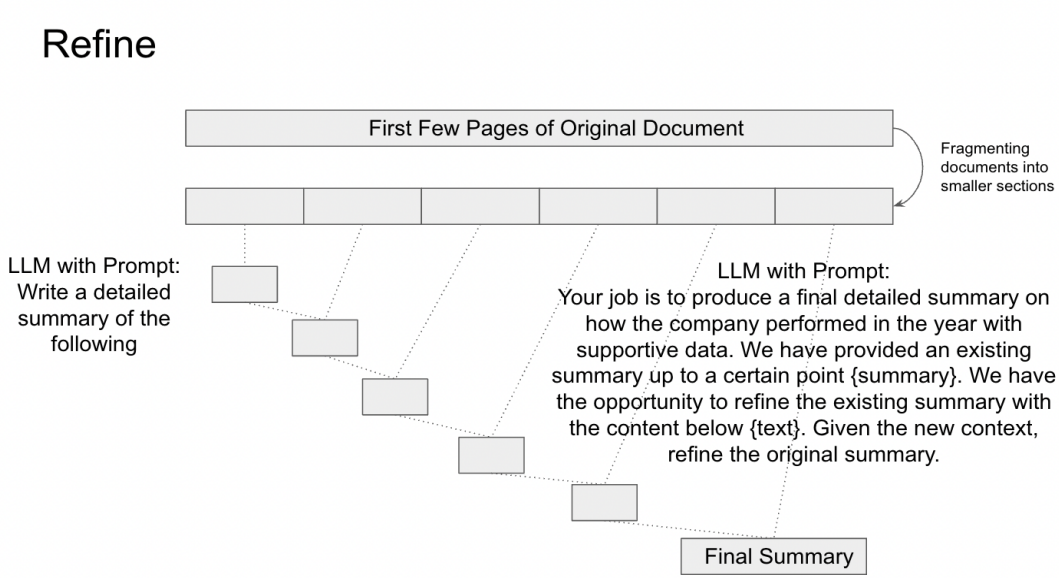
We have two main tasks for this project. One is to create a summarization model, and the other is to create a Q&A bot, which starts with a basic Q&A model. These will serve as useful tools to summarize the report contents and answer questions regarding KPIs.

4.1. Summarization Model

In this section, we will introduce our approaches to creating a summarization model to generate summaries of reports. We tried with different LLMs using both "refine chain" and "map-reduce chain", strategies for creating concise and coherent summaries from lengthy textual content to improve the quality and coherency of generated summaries.

4.1.1. Refine Chain

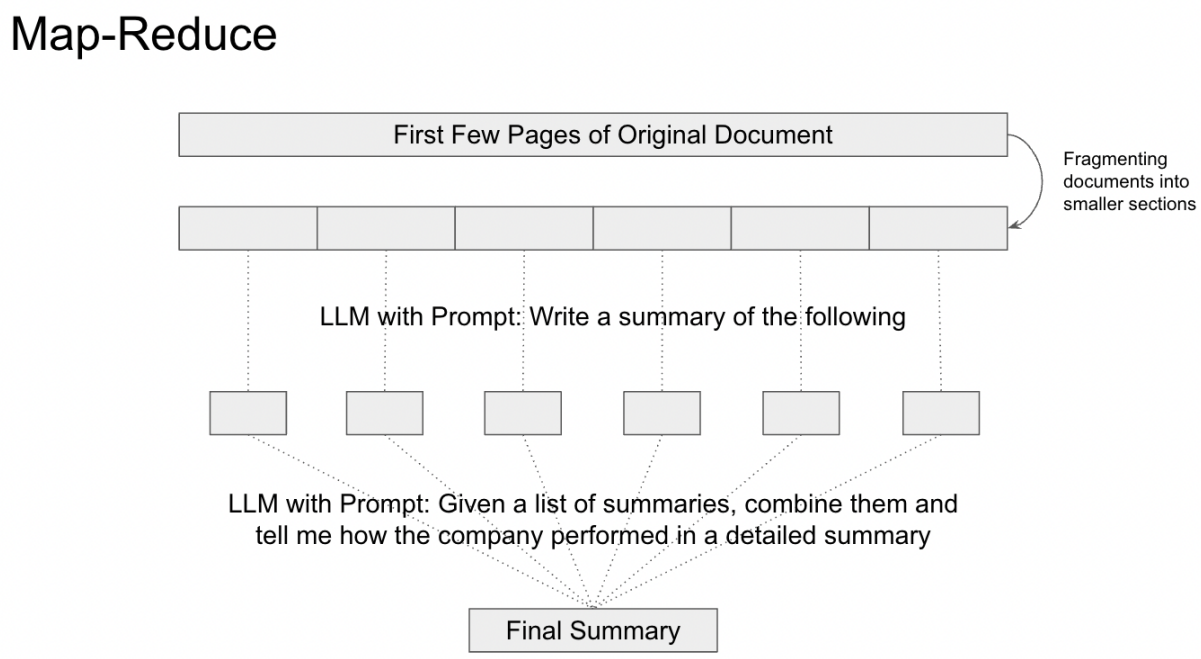
The refine chain is a technique used to enhance the quality and coherence of generated summaries. It involves a series of iterative steps designed to progressively include more and more content and gradually improve the summary's readability and informativeness. The refine chain often consists of multiple refining stages, and an example of the framework that we used is shown in the flowchart below.



The process typically begins with the creation of an initial summary, which is generated according to the initial prompt. The following steps are guided by the refine prompt, asking the model to iteratively regenerate a new summary, which combines the existing summary and the new content as the model progresses. The primary objective of the refine chain is to produce summaries that closely resemble human-generated content, effectively conveying the main ideas of the source document.

4.1.2. Map-reduce Chain

The map-reduce chain involves a sequence of map-reduce jobs used to process and summarize extensive textual data efficiently. In a map-reduce chain for summarization, each map-reduce job represents a distinct summarization subtask, and an example of the framework that we used is shown in the flowchart.



After each subtask is accomplished by the map prompt, the reduce prompt will ask the model to generate a final summary based on all the generated sub-summaries. The map-reduce approach ensures scalability and fault tolerance, allowing us to efficiently extract valuable insights and summaries, especially from extensive textual data.

4.1.3. Performance

4.1.3.1. Model Selection

In order to save running time and computation cost, we tested three LLM models (Flan-T5, Falcon, and Mistral) with a simple prompt on small financial-related paragraphs before we ran the summarization task on the actual report. The general performances are shown below:

Models	Performance
Flan-T5	Understandable summary capturing key ideas
Falcon	Whole prompt + a single word was provided
Mistral	Short sentences with supportive data

Given the results, we decided to move forward to the annual report of targeted companies with the Flan-T5 model and Mistral model.

Models	Chain Type	Performance
Flan-T5-Large	Refine	The refine chain successfully captures some supportive data but is missing some key financial indicators, for example, net profit.
	Map-Reduce	The map-reduce chain not only generalizes an informative summary but also captures the key financial indicators.
Mistral-7b	Refine	Missing an understandable summary
	Map-Reduce	With some useful information captured, the summary is generally not that informative and may contain inaccurate data with unknown data sources.

4.1.3.2. Model Improvement

We further engineered the prompt and increased the model size to Flan-T5-XXL. A better and more stable performance of the summarization task is achieved on both chains under the prompt shown below:

Refine Chain:

Initial Prompt:
Summarize the key highlights and findings of the company's {year} annual report, including financial performance, strategic initiatives, market position, challenges faced, and future outlook, if available.

Refine Prompt:
Your job is to provide a comprehensive overview of the {year} annual report. We have provided an existing summary up to a certain point: {existing_answer}. We have the opportunity to refine the existing summary with content below {text}. Improve the clarity and coherence of the summary, ensuring it flows logically. Add specific data and figures from the annual report where available.

Map-Reduce Chain:

Map Prompt:
Summarize the key highlights and findings of the company's {year} annual report, including financial performance, strategic initiatives, market position, challenges faced, and future outlook, if available.

Reduce Prompt:
Given a list of summaries, generate a structured summary of the company's {year} annual report, including financial performance, strategic initiatives, market position, challenges faced, and future outlook, if available.

4.1.3.3. Limitation

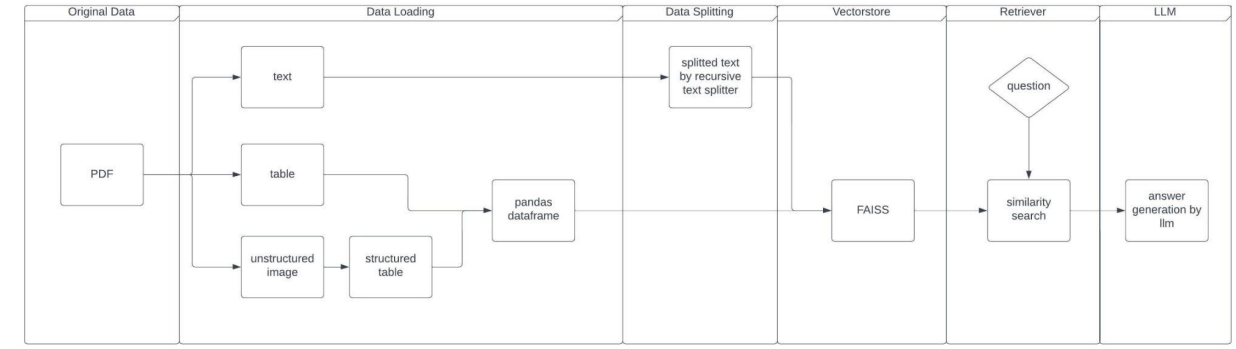
After selecting the best model and engineering the prompt, we figured out that the performance of the model may highly depend on the content of the pages that we select to do the summarization task. If a page selected contains the table of contents or CEO introduction, the model may not be able to perform well to summarize the financial performance of the company as our prompt asked. Therefore, the next step of automating the process of selecting the page that contains useful financial indicators might be helpful to get a more useful and informative financial summary of the annual report.

4.2. Q & A Model

In this section, we will introduce our approaches to the second task, creating a chatbot Q&A. For the purpose of efficiency and accuracy, we have conducted a comparative analysis of various loaders, splitters, and LLMs. Currently, we have found a reliable workflow and combination of tools. However, more research on each stage of the process is needed in the following weeks.

4.2.1. Workflow

The overall workflow has six stages described in the diagram below. Because of the LLM's capability and efficiency, we cannot simply feed the whole document into the model combined with the question from the user. Under such consideration, we store the preprocessed data in the FAISS vectorstore. Then, whenever the system receives a question from user input, only related chunks will be retrieved by FAISS similarity search. This process greatly increases the performance of LLM. To increase the performance of the model, we conducted several comparative analyses on the crucial components of the process.



4.2.2. Comparative Analysis of Loader and Splitter

As we previously mentioned, PDF loading and splitting are essential steps in the workflow of building a Q&A bot using an LLM. They enable the bot to access diverse information, improve contextual understanding, and efficiently retrieve data to provide accurate and valuable responses to user queries.

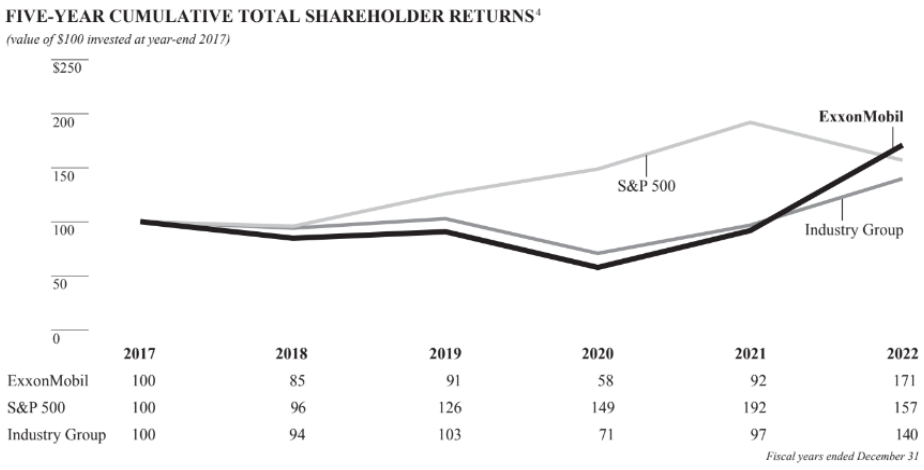
We have tested on 3 splitters and 7 different loaders. Since the splitters' job is relatively simple in the sense that only text is split into chunks, the test conducted is mainly focused on the separators, and the evaluation matrix is chosen to be the running time and the quality of the information contained in chunks. Holding the loader constant, we found an efficient list of separators to be '\n\n', ' ', ' ', '.', which will be the default configuration of the following test.

Having a well-performed splitter, we are now able to test on loaders: 'PyPDFLoader', 'Unstructured_file', 'Unstructured_pdf', 'Unstructured_pdf2', 'PyPDFium2', 'PDFMiner', 'PyMuPDF'.

```
--------------------------------------------------------------------------------
Performing data loading, text splitting, and vectorstore transforming
7 loaders, 1 splitters
--------------------------------------------------------------------------------
| PyPDFLoader + rec_500_100 |
        loading & splitting time:  300.86 s
        transformation time:       1.45 s
--------------------------------------------------------------------------------
| Unstructured_file + rec_500_100 |
        loading & splitting time:  499.64 s
        transformation time:       1.33 s
--------------------------------------------------------------------------------
| Unstructured_pdf + rec_500_100 |
        loading & splitting time:  497.5 s
        transformation time:       1.33 s
--------------------------------------------------------------------------------
| Unstructured_pdf2 + rec_500_100 |
        loading & splitting time:  496.66 s
        transformation time:       1.96 s
--------------------------------------------------------------------------------
| PyPDFium2 + rec_500_100 |
        loading & splitting time:  6.04 s
        transformation time:       1.37 s
--------------------------------------------------------------------------------
| PDFMiner + rec_500_100 |
        loading & splitting time:  469.49 s
        transformation time:       1.33 s
--------------------------------------------------------------------------------
| PyMuPDF + rec_500_100 |
        loading & splitting time:  7.21 s
        transformation time:       1.34 s
--------------------------------------------------------------------------------
```

The processing time varies depending on the underlying algorithms of the packages. The most productive ones are ‘PyPDFium2’ and ‘PyMuPDF’; the others run a lot slower. Besides running time, the completeness of the answer is also considered. To evaluate the overall performance of all the loaders, we have designed three types of queries: data queries, table queries, and image queries. Then, we have two sample questions for each query. Take an image query and response from ‘PyPDFium2’ as an example.

Question: *What is ExxonMobil’s five-year cumulative total shareholder returns in 2021?*



	2017	2018	2019	2020	2021	2022
ExxonMobil	100	85	91	58	92	171
S&P 500	100	96	126	149	192	157
Industry Group	100	94	103	71	97	140

Fiscal years ended December 31

```
--------------------------------
What is ExxonMobil’s five-year cumulative total shareholder returns in 2021?
-----------------------------------------------------------------------------------------------------------
| 1. Page 147 | Found: True |
--------------------------------
500
2017 2018 2019 2020 2021 2022
ExxonMobil 100 85 91 58 92 171
S&P 500 100 96 126 149 192 157
Industry Group 100 94 103 71 97 140
Fiscal years ended December 31
TEN-YEAR CUMULATIVE TOTAL SHAREHOLDER RETURNS4
$400
300
200
100
0
(value of $100 invested at year-end 2012)
ExxonMobil
Industry Group
S&P 500
2012
ExxonMobil 100 113 118 97 66 195
S&P 500 100 151 171 199 310 327
Industry Group 100 108 116 129 98 192
Fiscal years ended December
-----------------------------------------------------------------------------------------------------------
```

The first figure from the document contains the image requested by the user. The second figure is the generated chunk by the question. Indeed, the information contained in the chunk is accurate and well-formatted as a text table. Thus, it is a successful retrieval for image queries. Repeating the process for the other loaders and query types, we found that ‘PyPDFium2’ performs well on 2 out of 3 tasks (text query and image query), especially on data extraction from images. Unfortunately, 7 loaders failed on the table query, and we have not yet explored an efficient substitute. For now, we will continue using ‘PyPDFium2’ and keep searching for solutions to the table query.

4.2.3. Comparative Analysis of LLMs

Having a trustworthy data preprocessing algorithm, we are now ready to compare different LLMs. The tests are again query-oriented. We just finished the analysis on two models, ‘Flan-t5’ and ‘Mistral-7b’, and we will keep on trying other models in the future.

```
---------------------------------------------------------------------------------------------------------
What is the upstream earnings after income tax in 2017?
---------------------------------------------------------------------------------------------------------
| Flan-t5 |
---------------------------------------------------------------------------------------------------------
Vectorstore + similarity search:
    - Chroma: 14,079
         retriver time:  0.04 s
         model time:     3.18 s
    - FAISS: 14,079
         retriver time:  0.02 s
         model time:     2.81 s
---------------------------------------------------------------------------------------------------------
Vectorstore + retriver:
    - Chroma: 14,079
         retriver time:  0.02 s
         model time:     2.06 s
    - FAISS: 14,079
         retriver time:  0.02 s
         model time:     3.32 s
---------------------------------------------------------------------------------------------------------
Retriver only:
    - SVM: 14,079
         retriver time:  0.06 s
         model time:     1.8 s
---------------------------------------------------------------------------------------------------------
| Mistral-7b |
---------------------------------------------------------------------------------------------------------
Vectorstore + similarity search:
    - Chroma:  The upstream earnings after income tax in 2017 is 13,355 million dollars.
         retriver time:  0.02 s
         model time:     71.42 s
    - FAISS:  The upstream earnings after income tax in 2017 is 13,355 million dollars.
         retriver time:  0.02 s
         model time:     74.19 s
---------------------------------------------------------------------------------------------------------
Vectorstore + retriver:
    - Chroma:  The upstream earnings after income tax in 2017 is 13,355 million dollars.
         retriver time:  0.05 s
         model time:     68.88 s
    - FAISS:  The upstream earnings after income tax in 2017 is 13,355 million dollars.
         retriver time:  0.02 s
         model time:     80.99 s
---------------------------------------------------------------------------------------------------------
Retriver only:
    - SVM:  The upstream earnings after income tax in 2017 is $13,355 million.
         retriver time:  0.07 s
         model time:     46.27 s
---------------------------------------------------------------------------------------------------------
```

```
---------------------------------------------------------------------------------------------
What is the upstream earnings minus income tax in 2017?
---------------------------------------------------------------------------------------------
| Flan-t5 |
---------------------------------------------------------------------------------------------
Vectorstore + similarity search:
    - Chroma: 0
         retriver time:  0.03 s
         model time:     3.54 s
    - FAISS: 0
         retriver time:  0.03 s
         model time:     2.35 s
---------------------------------------------------------------------------------------------
Vectorstore + retriver:
    - Chroma: 0
         retriver time:  0.02 s
         model time:     2.25 s
    - FAISS: 0
         retriver time:  0.02 s
         model time:     2.36 s
---------------------------------------------------------------------------------------------
Retriver only:
    - SVM: 14,079
         retriver time:  0.05 s
         model time:     2.4 s
---------------------------------------------------------------------------------------------
| Mistral-7b |
---------------------------------------------------------------------------------------------
Vectorstore + similarity search:
    - Chroma:  The upstream earnings in 2017 were $13,355 million. The income tax in 2017 was $1,174 million. Therefore, the upstream earnings minus income tax in 2017 is $13,355 million - $1,174 million = $12,181 million.
         retriver time:  0.02 s
         model time:     117.94 s
    - FAISS:  The upstream earnings in 2017 were $13,355 million. The income tax in 2017 was $1,174 million. Therefore, the upstream earnings minus income tax in 2017 is $13,355 million - $1,174 million = $12,181 million.
         retriver time:  0.02 s
         model time:     120.34 s
---------------------------------------------------------------------------------------------
Vectorstore + retriver:
    - Chroma:  The upstream earnings in 2017 were $13,355 million. The income tax in 2017 was $1,174 million. Therefore, the upstream earnings minus income tax in 2017 is $13,355 million - $1,174 million = $12,181 million.
         retriver time:  0.04 s
         model time:     120.69 s
    - FAISS:  The upstream earnings in 2017 were $13,355 million. The income tax in 2017 was $1,174 million. Therefore, the upstream earnings minus income tax in 2017 is $13,355 million - $1,174 million = $12,181 million.
         retriver time:  0.02 s
         model time:     127.01 s
---------------------------------------------------------------------------------------------
Retriver only:
    - SVM:  The upstream earnings in 2017 were $13,355 million. Income tax is not mentioned in the table, so we don't know what it is. Therefore, the upstream earnings minus income tax in 2017 is $13,355 million.
         retriver time:  0.05 s
         model time:     75.72 s
---------------------------------------------------------------------------------------------
```

Two sample tests are shown above. The correct answer is *$12,181 million*. For the original question, both models failed to generate the right answer. ‘14,079’ from ‘Flan-t5’ is the upstream earning for 2018. ‘13,355’ from ‘Mistral-7b’ is the one for 2017. However, both models did not capture the context of the question which asked for the earnings after income tax. Prompt engineering was performed to address this problem. On one hand, ‘Mistral-7b’ successfully understood what the question was asking, grabbed the correct information from related chunks, and did calculations; on the other hand, ‘Flan-t5’ was totally lost generating 0. While ‘Flan-t5’ has a lower time complexity, ‘Mistral-7b’ generates a more precise, calculated, well-formatted answer. The trade-off here between time and accuracy is obvious.

As we continue developing and updating the algorithms, we will try other LLMs and do similar tests as well. We expect to have a thorough, well-organized, and time and space-saving system before fine-tuning and applying RLHF to LLMs.

4.3. Conversation Model

We also explored a conversation model integrated with our Q&A system. In this iteration, all reports were ingested into the model. The setup and procedure mirrored those detailed in previous sections. Our construction leveraged the `ConversationalRetrievalChain` from the `langchain.chains` package.

To guide the AI, we crafted prompts that ensured the generated content was beneficial and devoid of harmful, unethical, or toxic information. Crucially, the prompt incorporated a `chat_history` parameter, capturing the user's queries. These were then stored in the `ConversationBufferWindowMemory`. As such, when a user posed a question, the AI not only provided a response but also saved that interaction in its history. This retained information was referenced in subsequent dialogues, incrementally refining the conversation model.

For user convenience, we integrated two commands: "Clear" and "Exit". The "Clear" command purges the chat history, freeing associated memory, whereas "Exit" terminates the conversation.

It's pertinent to note that this model is a preliminary version of the final conversational Q&A system we aim to present. Enhancements in accuracy and efficiency are anticipated, focusing on refining both the language learning model (LLM) and the retriever chain components. Additionally, we plan to overlay a user-friendly interface to augment the user experience.

5. Tuning

Since the open-source LLM model itself is not tuned for financial purposes, we need to tune the LLM model so that it can understand and answer the financial questions. To tune the model, we will first generate 5000 Q&A pairs from the financial reports using GPT 3.5. By feeding appropriate prompts, we asked the AI to generate questions in these three categories: Data Query, Data Statistical Analysis, and Open Question. Below is an example of the output:


```
{"content": "What is the total number of employees at BP as of December 2018?",
"summary": "BP had a total of 73,000 employees as of December 2018."}

{"content": "What was the percentage increase in BP's profit attributable to
shareholders in 2018 compared to 2017?", "summary": "BP's profit attributable to
shareholders increased by 176% in 2018 compared to 2017."}

{"content": "How does BP contribute to reducing emissions while meeting the
increasing energy demand?", "summary": "BP contributes to reducing emissions by
investing in renewable energy sources, implementing energy efficiency measures, and
developing low carbon businesses. They also promote the use of natural gas, which
has lower carbon emissions compared to other fossil fuels."}

{"content": "How many retail sites does BP operate globally?", "summary": "BP
operates a total of 18,700 retail sites globally."}

{"content": "What was the replacement ratio of BP's proved reserves in 2018?",
"summary": "BP's replacement ratio of proved reserves in 2018 was 100%."}
```

Next, we will take a sample portion from these Q&A pairs and evaluate the accuracy using a set of evaluation metrics. Theoretically, this is a better output from a better model than the open-source LLMs, so we can use it to fine-tune our model. Also, we will use these generated 5000 Q&A pairs from GPT 3.5 to evaluate the accuracy and the performance of the open-source LLM models in the next step by taking the samples and evaluating the accuracy of each model and comparing the overall performance. Moreover, we may explore and use the loRA package to further fine-tune our model.

6. Next Steps

6.1. Summarization

Acknowledging the limitations of our current summarization output, which is overly reliant on input and struggles with varied report formats across different companies, we consulted our mentors for guidance. They proposed an innovative solution: to first use the Q&A model to extract major KPIs from the reports, and following this extraction, perform a summarization of all the extracted data. We believe that this is a good solution, as it aims to focus on the most crucial information in the reports, ensuring that the summary is relevant and concise. Hence, we plan to test it out as our next step.

6.2. Q & A

We will continue to pick our best model through a systematic and comprehensive approach. Firstly, we will continue developing and updating our algorithms, exploring a variety of LLMs, and conducting similar tests to compare their performances to achieve a well-organized, time-efficient, and space-efficient system before we apply RLHF to fine-tune the model. Secondly, we will focus on enhancing our data representation strategies, concerning complex data structures found in PDFs, such as applying semantic indexing using FAISS to boost model efficiency. This multi-pronged approach ensures that we not only choose the best Q&A model but also optimize the entire system for superior performance.

6.3. RLHF

After we pick our best model, the next steps in our research project on Reinforcement Learning from Human Feedback (RLHF) involve a detailed approach aimed at optimizing the performance of our model. As we mentioned, we plan to select a representative sample from the 5000 Q&A pairs generated by GPT 3.5 to get an initial benchmark of the model's accuracy and quality. Then we will use these questions to fine-tune our model further to enhance the performance and surpass the capabilities of our current best model. Ideally, by integrating LoRA, we hope to boost the performance of our RLHF system, ensuring it delivers highly accurate and effective natural language understanding and generation.