PROGRAMMING II REPORT
Università della Svizzera Italiana

Group members

Matteo dell'Acqua, Alessandro Fosselard, Rolands Repetto, Eleonora Salcuni

Introduction

The financial world is often at the forefront of technological advancements, significantly impacting various aspects of trading and data analysis. High-Frequency Trading (HFT), for instance, has revolutionized market dynamics by enabling trades to be executed in microseconds[1], thereby increasing market liquidity and reducing bid-ask spreads. This technological leap is driven by sophisticated algorithms and high-speed data networks, allowing for immediate price adjustments and improved market efficiency.

Artificial Intelligence (AI) has also been pivotal in finance for over a decade. A notable example is BlackRock's Aladdin[2], an AI-driven platform managing over $21.6 trillion in assets as of 2020. Aladdin provides tools for risk management, investment analytics, and economic data analysis, supporting asset managers in making informed decisions in a complex global market.

The advent of Large Language Models (LLMs) marks a recent significant development. These models generate extensive texts from brief prompts and are particularly useful in analyzing complex financial documents. Our project leverages information from the SEC's EDGAR database, which has been publicly accessible since October 1, 1994. This tool allows for the scrutiny of internal financial statements, such as the 10-K annual report. By employing AI, we can quickly interpret and respond to queries about these extensive reports, addressing the need for qualitative and clean data.

Our goal is to develop a mobile app that automates access and analysis of financial statements, focusing on extracting and converting income statements from 10-K forms. Using various Python libraries, we extract and structure financial quantitative and qualitative data for analysis and fast consultation. Although we are currently focused on ensuring the functionality of our application, our project aims for broader future applications.

[1] Zhang, S.S. and Riordan, R., 2011. Technology and market quality: The case of high frequency trading. In: 19th European Conference on Information Systems (ECIS 2011), Helsinki, Finland, 9-11 June 2011.

[2] Thomas Jr., L., 2017. At BlackRock, machines are rising over managers to pick stocks. Updated 29 March 2017, first published at 10.20am.

1. The Code (Jupyter Notebook)

Our project is divided into three main parts to effectively achieve our objectives. The initial phase involves backend development, which encompasses web-scraping income statements from the EDGAR database and utilizing a Large Language Model (LLM) to analyze the text and respond to user queries. The outputs from these processes are intended to be displayed through a user-friendly mobile app developed with Adalo[3]. To connect the back and front end we developed a server, implying FileZilla[4], through which securely transferred files and configured NGINX[5] as a reverse proxy for external access. Then the creation of a new URL was integrated into Adalo for direct application access, enhanced by an advanced Large Language Model (LLM) for precise investor queries.

We chose to focus on the income statement because it provides a quick view of a company's financial health. To retrieve these statements, we used API keys from SEC-API[6] and imported essential libraries for handling SEC filings and data conversion. The key steps in this process include:

Downloading Filings

By specifying the ticker symbol (e.g., Ticker = 'AAPL'), we utilized the sec_edgar_downloader to fetch the primary document URL of the latest 10-K filing.

```
python
from sec_edgar_downloader import Downloader
dl = Downloader("path/to/store")
metadatas = dl.get_filing_metadatas(RequestedFilings(ticker_or_cik=Ticker,
form_type="10-K", limit=1))
if metadatas:
    primary_doc_url = metadatas[0].primary_doc_url
```

Converting XBRL to JSON

Using the XbrlApi, we converted the 10-K filing from XBRL to JSON format for easier data manipulation.

```
python
from sec_api import XbrlApi
xbrlApi = XbrlApi(API_KEY)
xbrl_json = xbrlApi.xbrl_to_json(htm_url=primary_doc_url)
```

Extracting Income Statement

[3] https://www.adalo.com/

[4] https://filezilla-project.org/

[5] https://nginx.org/en/

[6] https://sec-api.io/

We defined a function get_income_statement that iterates over US GAAP items in the JSON file and converts them into a pandas DataFrame.

```
python
def get_income_statement(xbrl_json):
    income_statement_store = {}
    for usGaapItem in xbrl_json['StatementsOfIncome']:
        values = []
        indices = []
        for fact in xbrl_json['StatementsOfIncome'][usGaapItem]:
            if 'segment' not in fact:
                index = fact['period']['startDate'] + '-' +
fact['period']['endDate']
                if index not in indices:
                    values.append(fact['value'])
                    indices.append(index)
        income_statement_store[usGaapItem] = pd.Series(values,
index=indices)
    return pd.DataFrame(income_statement_store)
```

Developing the Large Language Model (LLM)

The next phase involves creating an LLM to answer specific questions about the 10-K filings. The steps include:

Consolidating Text

We combined text sections from 10-K form into a single text string and saved it in textanalyse.txt.

```
python
texts = [item_1_text, item_1_a_text, ...]  # Add all items
combined_text = ''.join(texts)
with open("textanalyse.txt", "w", encoding="utf-8") as file:
    file.write(combined_text)
```

Setting Up OpenAI API

We configured the OpenAI API key in a .env file for secure access.

```
python
from dotenv import load_dotenv
import os
load_dotenv()
OPENAI_API_KEY = os.getenv("OPENAI_API_KEY")
```

Using Sentence Transformer Embeddings

We used the SentenceTransformerEmbeddings to create embeddings from the consolidated text, which facilitated document analysis through Chroma.

```
python
from langchain_community.embeddings.sentence_transformer import
SentenceTransformerEmbeddings
```

```
embeddings = SentenceTransformerEmbeddings(model_name="all-MiniLM-L12-v2")
```

Creating Chat Model

We employed ChatOpenAI to handle the structured input and output for user queries.

```
python
from langchain_openai.chat_models import ChatOpenAI
model = ChatOpenAI(openai_api_key=OPENAI_API_KEY, model="gpt-4-turbo")
```

Text Splitting and Analysis

We configured settings for Hugging Face Hub to split the text into manageable sections and then used the embeddings to create a vector store for efficient querying.

```
python
from langchain.text_splitter import RecursiveCharacterTextSplitter
text_splitter = RecursiveCharacterTextSplitter(chunk_size=10000,
chunk_overlap=20)
documents = text_splitter.split_documents(text_documents)
from langchain_community.vectorstores import Chroma
db = Chroma.from_documents(documents, embeddings)
```

2. The deployment via Server

To deploy the app for our project, we were inspired by the URL created by our professor[7] and wanted to set up our own. We created a server capable of providing detailed financial information such as balance sheets and responding to investor inquiries, we initiated by provisioning a server instance within Cloudzy, ensuring it provided the required virtual environment for deployment. Once the server configuration was complete, FileZilla facilitated the secure transfer of essential files to the server, organizing them into designated directories optimized for execution. Subsequently, NGINX was specifically configured to function as a reverse proxy, thereby enabling external user access by incoming requests to the appropriate backend services, following predefined rules.

After configuring NGINX, we generated a new URL to enable direct access to the deployed application. This URL was seamlessly integrated into Adalo, enhancing the platform's capability to dynamically retrieve and display the company's income statement. To further enrich user interaction, we integrated an advanced Large Language Model (LLM) into the application framework. This integration empowered investors to pose questions about the company, receiving precise, data-driven responses based on comprehensive document analysis from sources like SEC Edgar, as specified in the Python script.

[7] Plot historical data https://dev01.petergruber.com/demo/plot?ticker=AAPL

Following successful implementation, we refined function names and restarted the Flask application to seamlessly incorporate these updates. Continual enhancement remains paramount, achieved through the ongoing creation of new LLM models. These efforts expand the application's functionality and responsiveness to user inquiries, ensuring a robust and adaptive platform.

3. Application Creation: Tauros App

To create our app, we utilized the intuitive no-code service Adalo[8]. This approach is much simpler than developing an app from scratch using traditional tools like Xcode. Adalo provided all the necessary functions for our app, including the ability to access the web and use Magic Text for dynamic URLs.

The welcome screen of Tauros features a clean and simple design, branded with the app's name. Users can log in or sign up. New users register by providing their email, password, and full name. Registered users can log in with their email and password, with an option to reset the password if it has been forgotten.

Upon successful login, users are directed to the Home Screen. Here, they can enter a specific ticker symbol to access the income statement of a desired company. The income statement is presented in a structured table format, providing all necessary financial information concisely. Users can save their favorite tickers for quick access.

Users input a ticker symbol to retrieve and display the company's income statement. This feature uses a public URL to fetch the latest financial statements and present them in the app. The URL is dynamically generated using Adalo's Magic Text feature, allowing for seamless integration and data retrieval.

The chatbot, accessible from the bottom menu bar, allows users to ask specific questions about financial statements. It uses an AI model to provide detailed answers, saving users time from manually searching through documents. For example, users can ask, "What are Tesla's plans for the commercialization of the Cybertruck?", "What are Microsoft future goals?", and receive an instant, concise response. However, it's important to note that the chatbot's accuracy was around 70%, meaning that while it often provided satisfactory answers, there is room for improvement. In addition there is a limitation for openai api users so we will have to find a way of making multiple users use our chat without limits and this is possible only with the weaker GPT3.5 model[9]. Eventually, we published our project in Github [10].

[8] https://alessandro-fosselards-team.adalo.com/finstatement

[9] https://platform.openai.com/settings/organization/limits

[10] https://github.com/rrepetto99/project2024

4. Future expectations

To enhance Tauros, we plan to integrate additional financial statements such as the balance sheet and quarterly reports. In the future, we could also include data analysis of these statements through Python code, which is well-suited for such tasks. Adding more interactive features and data visualizations will provide deeper insights into financial data. For the chatbot, improving its accuracy and expanding its capabilities to handle more complex queries will enhance user experience. We also aim to ensure that the output is always satisfactory, which can be achieved through better models in sentence transformers and more efficient code. In terms of design, we chose a minimalistic approach, making the app easy to use with only four sections in the sub-menu and hardly any animations to keep it simple and fast. We maintained consistent design elements, such as stable colors, fonts, and layouts. Using color-friendly designs, like contrasting blue with white, enhances user experience. Involving a designer in the future could further improve the app's visual appeal. We are eager to involve professional designers to further refine the app's visual appeal while maintaining a minimalistic and user-friendly interface.

Implementing robust encryption protocols and ensuring compliance with GDPR are crucial steps to safeguard user data and financial information. These measures will establish a strong foundation of privacy and increase user trust. Incorporating customer service features, such as a chatbot for reporting issues and providing feedback, will ensure continuous support for users, creating a comprehensive and secure environment for financial analysis. By focusing on these enhancements, Tauros aims to offer users a unique, efficient tool for analyzing financial insights and executing investment strategies based on the latest data.

Conclusion

The Tauros app is designed to make the process of consulting financial statements textual parts accessible for everyone as they are often lengthy and tedious to read. By integrating a Large Language Model (LLM), users can quickly access the information they need or request summaries, making financial analysis more efficient. This was made possible through RAG which gives additional data for the model to create text. We gave this data in the form by increasing embeddings which are the vector representations of the words. We enriched the vector space of the model but only in the specific niche of the 10-K that is requested. This is effective because ChatGPT is not inherently well-suited for specialized tasks. Overall, this app is designed for individuals who want to save time and gain a competitive advantage through modern AI tools.

Using the no-code platform Adalo, we successfully developed and delivered our app, allowing users to consult company financials and reports on the go, in the palm of our hands. While the app is functional enough to be published as a beta, we acknowledge that it is not perfect and

have outlined several goals for future improvements. We plan to seek assistance from Adalo experts to refine the app further.

The group work and organization were critical to the project's success. All our team members added something to the project, and we often did so specific to our respective areas of expertise (coding, writing, organizing, designing, ideas...). We all listened to each other, which was important to create a project together. We often communicated via WhatsApp groups to stay updated and discuss problems and ideas. We also did many video conferences on Zoom or Google Meet where we could share our screens.

To conclude we are all happy with our final product which can be downloaded and understood and used by anyone who has a phone. We have plans to make the app better so that we can really take this project to the level of a start-up.