

NeuralFinGPT: Leveraging NLP and Generative Models for Financial Analysis in MENA, and Overcoming Data Challenges.

Rany Stephan¹

¹American University of Beirut

Abstract

This paper presents NeuralFinGPT, a project designed to enhance financial analysis for companies in the MENA region through advanced Natural Language Processing (NLP) techniques. This paper outlines the comprehensive process of downloading, preprocessing, and analyzing thousands of financial reports from MENA companies. Given the poor quality of many of these reports, significant effort was dedicated to cleaning and processing the data using regex and other techniques. Sentiment analysis was conducted using tokenizers and Hugging Face packages, with a focus on identifying Forward Looking Statements. The processed data was stored in a PostgreSQL database, alongside relevant financial metrics. To improve analytical capabilities, the Mistral7b large language model (LLM) was fine-tuned using QLoRA, allowing for effective training on limited computing resources. Retrieval-Augmented Generation (RAG) algorithms were employed to create context for user queries, ensuring accurate and contextually relevant responses. Additionally, extensive prompt engineering was performed to refine the model's outputs. Custom functions were developed to handle specific queries, enhancing the system's efficiency and user experience. Despite challenges with data quality, NeuralFinGPT demonstrates significant potential for improving financial analysis in the MENA region. Future work will focus on further refining data preprocessing and expanding the dataset to enhance model performance.

Keywords: Machine Learning, Fine-Tuning, Natural Language Processing (NLP), MENA Region, Sentiment Analysis, Large Language Models (LLMs), QLoRA, Retrieval-Augmented Generation (RAG), Prompt Engineering

Supervising professor: Rida Assaf *E-mail address:* ra278@aub.edu.lb

Received: **Revised:** **Accepted:** **Published:**

■ INTRODUCTION

The financial landscape of the Middle East and North Africa (MENA) region is marked by unique challenges and opportunities, making it a fertile ground for innovative financial analysis tools. Accurate and timely analysis of financial reports is crucial for investors, analysts, and decision-makers. However, the quality of these reports often varies significantly, complicating the extraction of meaningful insights. NeuralFinGPT, an advanced Natural Language Processing (NLP) project, addresses these challenges by leveraging state-of-the-art technologies to enhance the analysis of financial reports from MENA companies.

The primary objective of NeuralFinGPT is to automate and improve the process of extracting and analyzing data from financial reports, which are often poorly formatted and difficult to process. This project involves several key steps, starting with the downloading and preprocessing of thousands of financial reports. Given the generally low quality of Optical Character Recognition (OCR) results on these reports, extensive cleaning and preprocessing using regular expressions were necessary to convert them into usable text files.

Once the reports were preprocessed, sentiment analysis was conducted using tokenizers and Hugging Face packages, focusing on the presence of Forward Looking Statements to gauge the sentiment. This sentiment data, along with other relevant financial metrics such as closing prices, was stored in a PostgreSQL database, providing a structured repository for analysis.

To enhance the analytical capabilities of NeuralFinGPT, the Mistral7b large language model (LLM) was fine-tuned using the Quantized Low-Rank Adaptation (QLoRA) technique. This approach enabled effective training on limited computational resources, making advanced NLP techniques accessible for this project. Retrieval-Augmented Generation (RAG) algorithms were then employed to create context for user queries, ensuring that the model could provide accurate and contextually relevant responses.

Furthermore, prompt engineering was utilized to refine the model's outputs, improving the quality of responses. Custom functions were developed to handle specific queries, enhancing the system's efficiency and user experience. These functions enable the model to

identify the appropriate method for answering a query and provide the necessary inputs for that method.

Despite the challenges posed by the quality of the financial reports, NeuralFinGPT demonstrates significant potential for improving financial analysis in the MENA region. This paper details the methodology, implementation, and outcomes of the NeuralFinGPT project, highlighting its contributions to the field of financial analysis and its potential for future development.

■ Literature Review

The integration of Large Language Models (LLMs) into the financial sector has emerged as a transformative approach to enhancing the analysis and understanding of financial data. This literature review examines the development and application of financial-specific NLP models, with a focus on their impact, methodologies, and the unique challenges they address. By situating the NeuralFinGPT project within this context, we highlight the significance of advancements in financial NLP, particularly for the MENA region.

Large Language Models in Finance

LLMs have shown remarkable capabilities in various natural language processing tasks, and their application in finance has opened new avenues for data analysis and decision-making. BloombergGPT, for example, is a proprietary model designed specifically for financial tasks, leveraging exclusive access to high-quality financial data (Wu et al., 2023) [14]. While BloombergGPT's performance is impressive, its restricted accessibility and high training costs underscore the need for more open and inclusive alternatives.

0.1. FinGPT: Open-Source Financial Large Language Models

FinGPT, introduced by Yang et al. (2023) [15], addresses the challenges of proprietary models by adopting a data-centric approach and emphasizing the democratization of financial data. FinGPT's framework includes layers for data sourcing, engineering, model fine-tuning, and application, showcasing the transformative potential of LLMs in the financial sector. The open-source nature of FinGPT fosters collaboration and innovation, enabling researchers and practitioners to build and fine-tune models using accessible resources.

0.2. Data-Centric Approaches

The quality and relevance of training data are critical to the success of financial LLMs. FinGPT, for instance, prioritizes the collection, cleaning, and preprocessing of high-quality financial data from diverse sources, including news websites, social media, financial statements, and market trends (Yang et al., 2023). This data-centric approach ensures that the models are trained on reliable and relevant data, enhancing their accuracy and effectiveness.

Handling financial data presents unique challenges due to its high temporal sensitivity, dynamism, and low signal-to-noise ratio. These challenges necessitate sophisticated data engineering techniques to filter out noise and highlight valuable information. Real-time processing capabilities are essential for financial NLP to respond promptly to market changes and news updates.

0.3. Practical Applications of Financial LLMs

Financial LLMs have a wide range of applications, from sentiment analysis to algorithmic trading and robo-advising. Sentiment analysis using social media data, for instance, has been shown to predict stock price movements effectively. Kolasani and Assaf (2020) [5] demonstrated that sentiment analysis of Twitter feeds could predict stock prices with high accuracy, utilizing models such as Support Vector Machines (SVM) and neural networks for better prediction outcomes. This research highlights the potential of social media sentiment as a predictive tool in financial markets and the superiority of neural networks over traditional models for this task.

Conversational AI, like GPTQuant, has also emerged as a valuable tool for investment research. GPTQuant leverages prompt templates and LangChain's integration to activate few-shot learning capabilities, generating Python code for quantitative analysis (Yue et al., 2023). This approach simplifies the investment research process, making it more accessible to retail investors and reducing the technical barriers associated with traditional backtesting methods.

0.4. Fine-Tuning Techniques

Fine-tuning pre-trained LLMs for specific domains, such as finance, offers a cost-effective alternative to training models from scratch. Techniques like Quantized Low-Rank Adaptation (QLoRA) allow for efficient adaptation with limited computational resources (Dettmers et al., 2023). FinGPT employs these techniques to maintain model relevance and accuracy in dynamic financial environments, ensuring timely updates and adaptability.

0.5. Previous Works on Financial NLP Models

The development of financial-specific NLP models has been an area of growing interest, particularly in enhancing the accuracy and relevance of financial data analysis. Prior works have focused on various aspects of financial text processing, including sentiment analysis, forward-looking information, and the creation of financial language models tailored to specific languages and markets.

0.6. FinAraT5: A Financial Arabic Language Model

FinAraT5 represents a significant advancement in the field of financial NLP, specifically tailored for the Arabic language. Zmandar, El-Haj, and Rayson (2023) [16] developed FinAraT5 by continuing the pre-training of AraT5 on a specialized corpus of Arabic financial texts. This model addresses the lack of pre-trained finance-specific Arabic language models and demonstrates superior performance in both generative and discriminative tasks compared to existing models like mT5, AraBART, and BERT. The comprehensive corpus used includes financial reports from various Middle Eastern stock exchanges, highlighting the model's applicability to real-world financial data (Zmandar et al., 2023).

0.7. Forward-Looking Information in UAE Annual Reports

Aljifri and Hussainey (2007) [1] explored the determinants of forward-looking information in the annual reports of UAE companies. Their study identified significant factors influencing the disclosure of such information, including debt ratio and profitability, while factors like sector type, firm size, and auditor size were found to be less influential. This research underscores the importance of forward-looking information for investors and the varying levels of its disclosure across different firms and sectors (Aljifri Hussainey, 2007).

0.8. Alpha-GPT: Interactive Alpha Mining for Quantitative Investment

Alpha-GPT, developed by Liu et al. (2023), is an innovative approach that integrates human and AI capabilities for quantitative investment. The model leverages a conversational AI framework to interactively assist investors in mining alpha from financial data. By combining prompt engineering with LangChain's integration, Alpha-GPT enables few-shot learning and the generation of Python code for quantitative analysis, making it a versatile tool for investment research and decision-making (Liu et al., 2023) [13].

0.9. Contribution of NeuralFinGPT

NeuralFinGPT builds upon the foundational works discussed in the literature review by incorporating advanced NLP techniques and large language models for financial analysis. While previous research has primarily focused on analyzing financial data from well-documented and standardized sources, NeuralFinGPT addresses the unique challenges presented by the MENA region. The lack of clean, readable, and standardized data in this region, especially for public company filings, necessitates a robust preprocessing pipeline to extract and normalize financial information effectively.

NeuralFinGPT distinguishes itself by not only tackling these data quality issues but also integrating support for interactive conversations with relevant filings, sentiment analysis, and other computational finance functions. This conversational capability allows users to query specific financial reports and receive detailed insights, making the model a versatile tool for financial analysts. Additionally, the fine-tuning of the Mistral 7B model using QLoRA on resource-limited environments such as Google Colab's T4 GPU demonstrates the practical applicability of the model for real-world scenarios in the MENA region. This combination of addressing regional data challenges and providing advanced interactive features positions NeuralFinGPT as a significant advancement in the field of financial NLP.

■ Data Collection

Sources of Financial Reports

For our analysis, we focused on collecting Interim filings, namely 'Prelim Q1', 'Prelim Q2', 'Prelim Q3' as well as 'FullYear', which are comprehensive reports of a company's performance that include relevant financial information. The data was obtained using Refinitiv's Filings Service, a robust platform that provides access to global filings documents for thousands of companies spanning over 50 years of history. This service allows for the extraction of specific sections within these documents, such as the Management Discussion and Analysis (MDA) sections, which are particularly relevant for our analysis.

We configured the API and retrieved the necessary API key to access the filings.

Using this setup, we queried the Interim filings for MENA companies, more specifically Algeria, Bahrain, Egypt, Iran, Iraq, Jordan, Kuwait, Lebanon, Libya, Morocco, Oman, Qatar, Saudi Arabia, Syria, Tunisia, United Arab Emirates, Palestine, and Yemen, capturing the relevant sections for further analysis.

Challenges

Collecting high-quality financial reports posed several challenges. One significant issue was the inconsistency in the formatting and

structure of the filings. Different companies may present their financial data in various formats, making it challenging to standardize the extraction process. Additionally, the presence of non-ASCII characters and malformed symbols in the text required extensive preprocessing to ensure the data was in a suitable format for analysis.

In fact, on a broader note, we have found that data collection in the MENA region presents significant challenges that complicate the process of obtaining clean, readable, and standardized financial information. Unlike regions with established frameworks for electronic filings, such as the EDGAR system in the United States, many public company filings in the MENA region lack a unified format or template. These documents often include handwritten sections, inconsistent formatting, and varying levels of detail, which hinder automated extraction and analysis. Additionally, the availability of digital archives is limited, requiring extensive manual effort to locate, scan, and digitize paper documents. This lack of standardized reporting and digital accessibility exacerbates the difficulty of creating comprehensive datasets necessary for accurate financial analysis. Consequently, substantial preprocessing is required to clean and normalize the data, adding layers of complexity and time to the data collection process. NeuralFinGPT addresses these challenges by developing a robust pipeline capable of handling the inconsistencies and poor quality of financial data in the MENA region, ensuring that the extracted information is suitable for advanced NLP techniques.

■ Data Preprocessing

Cleaning Process

The preprocessing of the collected data involved several critical steps to clean and prepare the text for Natural Language Processing (NLP) analysis. The goal was to ensure that the text was in a suitable format for further analysis, which involved removing irrelevant information, normalizing the text, and handling non-ASCII characters.

We first extract the data from the PDF files using the pdfplumber library in Python [17]. The following code snippet demonstrates the function used to extract text from the PDF files:

```
1 import pdfplumber
2
3 def extract_text_from_pdf(pdf_path):
4     text = ''
5     with pdfplumber.open(pdf_path) as pdf:
6         for page in pdf.pages:
7             text += page.extract_text() or '' #
8             Ensuring that None is not returned
9     return text
```

Code 1. Function to extract text from PDF files using pdfplumber.

Handling Poor Quality Data

The financial reports often contained poor-quality data, including non-ASCII characters, malformed symbols, and irrelevant information such as file paths and document headers. These issues required extensive cleaning to ensure the text was suitable for analysis.

We addressed these challenges by implementing robust text cleaning functions, which removed non-ASCII characters, normalized new lines, and eliminated citation-like numbers in brackets. Additionally, we ensured that any irrelevant numerical data that could skew the analysis was carefully handled.

```
1 import re
2
3 def clean_text(text):
4     # Remove file paths and document headers
5     text = re.sub(r'Filings/[\w\.-]+', '', text)
6
7     # Remove malformed characters and symbols
8     text = re.sub(r'[\x00-\x7F]+', '', text) #
9     Remove non-ASCII characters
10    text = re.sub(r'[\r\n]+', '\n', text) # Normalize
11    new lines
```

```
10    text = re.sub(r'[\d+]', '', text) # Remove
11    citation-like numbers in brackets
12    text = re.sub(r'\s{2,}', ' ', text) # Replace
13    multiple spaces with a single space
14
15    return text
16
17 # Example usage
18 with open('Filings/sample_filing.txt', 'r', encoding='
19 utf-8') as file:
20     raw_text = file.read()
21
22 cleaned_text = clean_text(raw_text)
23 print(cleaned_text[:500]) # Print first 500 characters
24 to check the output
```

Code 2. Function to clean and normalize text extracted from filings.

■ Sentiment Analysis

Algorithm Description

For sentiment analysis, we employed pre-trained NLP models specifically designed for financial text. The FinBERT model, fine-tuned on financial data, was used to classify sentences into categories such as Forward-Looking Statements (FLS) and to analyze the sentiment of these statements [2].

We used the transformers library [7] from Hugging Face to load and utilize these models. The following code snippet shows how we set up and used the FinBERT model for sentiment analysis:

- Loading FinBERT Model:
- Performing Sentiment Analysis:

By using these models, we analyzed the text to identify Forward-Looking Statements and their sentiment. The analysis involved breaking down the MDA sections into individual sentences and running each through the models to generate predictions. Here is an example of how we applied the model to a sample sentence:

```
1 from transformers import AutoTokenizer,
2   AutoModelForSequenceClassification, pipeline
3
4 # Load the models
5 finbert = AutoModelForSequenceClassification.
6   from_pretrained('yiyanghkust/finbert-fls')
7 tokenizer = AutoTokenizer.from_pretrained('yiyanghkust/
8   finbert-fls')
9
10 # Create a pipeline for text classification
11 nlp = pipeline("text-classification", model=finbert,
12   tokenizer=tokenizer)
```

Code 3. Loading and setting up the FinBERT model for sentiment analysis.

The output indicates the likelihood of the sentence being a specific or non-specific Forward-Looking Statement, along with the sentiment classification.

Evaluation

Through the detailed processes of data collection, preprocessing, and sentiment analysis, we aimed to explore the potential correlations between the language used in financial reports and the companies' future outlooks. By leveraging advanced NLP models and comprehensive datasets, we provided insights into how financial text can be interpreted to predict business trends and investor sentiment.

```
1 def evaluate(text):
2     # Initialize containers for the results
3     fls_pct = 0
4     total_sentiments = torch.zeros(1, 3)
5
6     # Tokenize the text into individual sentences
7     sentences = sent_tokenize(text)
8     num_sentences = len(sentences)
9     fls_count = 0
10
11    for sentence in sentences:
12        # FLS prediction
```



```

13     prediction = nlp(sentence[:512], top_k=3)[0]['
14         label']
15
16     # Check if the sentence is a Forward Looking
17     # Statement
18     if prediction.startswith("Specific") or
19     prediction.startswith("Non"):
20         fls_count += 1
21
22     # Tokenize for sentiment analysis
23     encoded_input = tokenizer(sentence,
24     return_tensors="pt", truncation=True)
25
26     with torch.no_grad():
27         # Sentiment prediction
28         output = model(**encoded_input)
29         sentiment = torch.nn.functional.softmax
30         (output.logits, dim=-1)
31         total_sentiments += sentiment
32
33     # Calculate the percentage of FLS sentences
34     if num_sentences > 0:
35         fls_pct = (fls_count / num_sentences) * 100
36         average_sentiments = total_sentiments /
37         fls_count # Average the sentiments
38         dominant_sentiment_index = average_sentiments.
39         argmax(dim=1).item()
40         dominant_sentiment = model.config.id2label[
41         dominant_sentiment_index]
42         sentiment_confidence = average_sentiments[0][
43         dominant_sentiment_index].item() * 100
44     else:
45         dominant_sentiment = 'neutral'
46         sentiment_confidence = 0
47
48     return fls_pct, dominant_sentiment,
49     sentiment_confidence

```

Code 4. Function to evaluate the correct sentiment using FLS percentage.

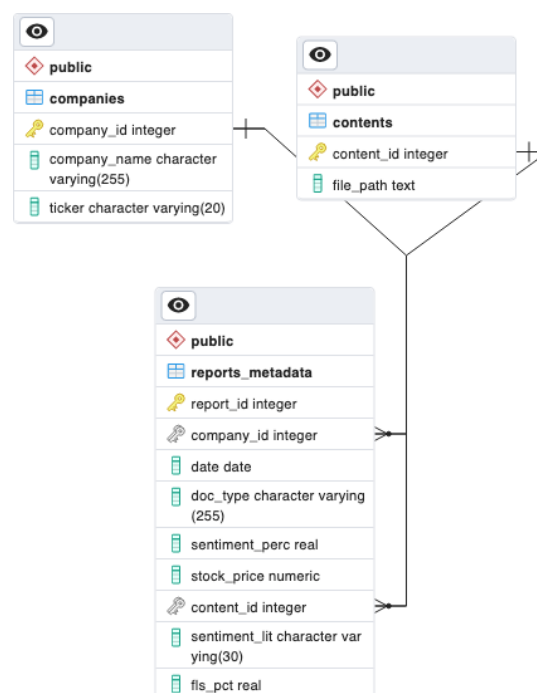


Figure 1. ERP Diagram for MENA Financial Reports Database.

Database Creation

PostgreSQL Database

Creating a PostgreSQL database for storing and managing the collected financial reports is a crucial step in the NeuralFinGPT project. The database serves as a centralized repository that facilitates efficient querying, retrieval, and analysis of the data. Given the variability and volume of financial reports from the MENA region, a robust database management system like PostgreSQL ensures data integrity, consistency, and accessibility.

The database schema includes several tables, such as companies, contents, and reports metadata, each serving a specific purpose. The companies table stores information about the companies, including their names and ticker symbols. The contents table holds the paths to the cleaned and preprocessed text files of the financial reports. Finally, the 'reports metadata' table records detailed metadata about each report, such as the company ID, date, document type, sentiment analysis results, and percentages of Forward-Looking Statements (FLS).

By creating a PostgreSQL database, we ensured that all extracted and processed data are systematically stored and easily accessible for subsequent analysis. This approach not only streamlines the data handling process but also enhances the scalability of the project, allowing for the efficient management of large volumes of financial reports. The database setup is fundamental to achieving the project's goal of providing reliable and accurate financial analysis in the MENA region, despite the inherent challenges associated with data quality and standardization.

Data Schema

This figure shows the Entity Relationship Diagram of the SQL database that was created.

1. Model Fine-Tuning

1.1. Explanation of the Mistral7b Model

Mistral 7B is a 7-billion-parameter language model designed for superior performance and efficiency. It outperforms other large models, such as Llama 2 (13B) and Llama 1 (34B), across various benchmarks including reasoning, mathematics, and code generation. Mistral 7B achieves this by leveraging advanced attention mechanisms like grouped-query attention (GQA) and sliding window attention (SWA), which collectively enhance its performance and efficiency [12].

GQA significantly accelerates inference speed and reduces memory requirements during decoding, allowing for higher batch sizes and increased throughput, crucial for real-time applications. SWA, on the other hand, is designed to handle longer sequences more effectively at a reduced computational cost, thus overcoming common limitations in large language models [12].

The architecture of Mistral 7B is based on a transformer model with key parameters such as 4096 dimensions, 32 layers, 128 head dimensions, 14336 hidden dimensions, 32 heads, and 32000 vocabulary size. These architectural choices, combined with innovative attention mechanisms, enable Mistral 7B to deliver high performance while maintaining efficient inference [12].

1.2. Importance and Benefits of Using QLoRA for Fine-Tuning with Limited Resources

QLoRA (Quantized Low-Rank Adaptation) is an efficient fine-tuning approach that significantly reduces memory usage while preserving full 16-bit fine-tuning task performance. This method allows for the fine-tuning of large models, such as a 65-billion parameter model, on a single 48GB GPU without performance degradation [11].

QLoRA achieves this efficiency through several innovations:

- **4-bit NormalFloat (NF4):** A new data type that is information-theoretically optimal for normally distributed weights, providing better empirical results than 4-bit integers and floats.
- **Double Quantization:** This reduces the average memory footprint by quantizing the quantization constants, thereby saving memory without sacrificing performance.

- **Paged Optimizers:** These manage memory spikes during training, preventing out-of-memory errors and allowing for the efficient use of available GPU memory [11].

The primary advantage of using QLoRA is its ability to make large-scale model fine-tuning accessible to those with limited computational resources. By reducing the memory requirements, QLoRA enables researchers and practitioners to fine-tune state-of-the-art models on consumer-grade hardware, thus democratizing access to advanced NLP capabilities. This approach is particularly beneficial for projects like NeuralFinGPT, where computational resources may be limited, yet high-performance model fine-tuning is essential for analyzing complex financial data.

QLoRA employs the following key techniques for quantization and memory optimization:

- **4-bit Quantization:** The process involves quantizing the model weights to 4-bit precision. Given a tensor $X \in \mathbb{R}^{b \times h}$, the quantization is performed as follows:

$$X_{\text{Int8}} = \text{round} \left(\frac{127}{\text{absmax}(X_{\text{FP32}})} X_{\text{FP32}} \right) = \text{round}(c_{\text{FP32}} \cdot X_{\text{FP32}}),$$

where c_{FP32} is the quantization constant. The dequantization process is:

$$\text{dequant}(c_{\text{FP32}}, X_{\text{Int8}}) = \frac{X_{\text{Int8}}}{c_{\text{FP32}}} = X_{\text{FP32}}.$$

- **Low-Rank Adapters (LoRA):** LoRA reduces memory requirements by using a small set of trainable parameters while keeping the full model parameters fixed. The projection $Y = XW$ is augmented as:

$$Y = XW + s \cdot XL_1L_2,$$

where $L_1 \in \mathbb{R}^{h \times r}$ and $L_2 \in \mathbb{R}^{r \times o}$, and s is a scalar.

- **Double Quantization:** To save memory, the quantization constants are further quantized. The second quantization uses an 8-bit float for constants:

$$\text{doubleDequant}(c_{\text{FP32}}, c_{\text{k-bit}}, W_{\text{4-bit}}) = \text{dequant}(\text{dequant}(c_{\text{FP32}}, c_{\text{k-bit}}), W_{\text{4-bit}}).$$

- **Paged Optimizers:** These use NVIDIA unified memory to handle memory spikes, avoiding out-of-memory errors during gradient checkpointing.

■ Context Creation with RAG Algorithms

Retrieval-Augmented Generation (RAG)

Retrieval-Augmented Generation (RAG) represents a significant advancement in the field of natural language processing (NLP), particularly for knowledge-intensive tasks. The core idea behind RAG is to augment the generative capabilities of large language models (LLMs) with external knowledge sources, thereby enhancing their ability to produce accurate and contextually relevant responses.

RAG models combine two main components: a retriever and a generator. The retriever is responsible for fetching relevant documents from a large corpus, typically using dense vector representations and similarity search methods. The generator then uses these retrieved documents as additional context to generate more informed and accurate responses. This approach addresses some of the key limitations of purely generative models, such as hallucinations and the inability to update knowledge dynamically [6] [8] [9].

Algorithm Description

In our project, we utilized the RAG framework to tackle the specific challenges posed by financial report analysis in the MENA region. The financial data from this region often lacks the structured format found in Western counterparts, making the retrieval and generation process particularly beneficial.

The retriever component in our RAG implementation employs a bi-encoder architecture, as described in the Dense Passage Retrieval (DPR) model. The bi-encoder processes the input query and candidate documents separately, encoding them into dense vectors. These vectors are then used to compute similarity scores, with the top-k documents being selected as the most relevant [4] [6] [9].

The generator component is a fine-tuned sequence-to-sequence transformer model, which generates responses based on the input query and the retrieved documents. This model is capable of marginalizing over the latent documents to produce a coherent and contextually enriched output. We explored both RAG-Sequence and RAG-Token models, finding that the ability to condition on multiple documents at different stages of the generation process significantly improves the accuracy and diversity of the generated responses [6] [9].

Importance for Our Project

Integrating RAG into our project has several crucial benefits:

- **Enhanced Accuracy:** By leveraging external documents, RAG reduces the likelihood of generating factually incorrect content, which is particularly important for financial report analysis where accuracy is paramount [8] [3].
- **Dynamic Knowledge Update:** Unlike static models, RAG can dynamically incorporate the latest information from updated financial reports, making it more adaptable to real-world changes and ensuring that the generated insights are based on the most current data available [9] [10].
- **Handling Unstructured Data:** The financial reports from the MENA region often lack standardization. RAG's ability to retrieve relevant information from a large and diverse set of documents helps in synthesizing useful insights from such unstructured data [4] [6].
- **Improved Sentiment Analysis:** The retrieval mechanism helps in gathering contextual information that enhances the sentiment analysis of financial reports, making it possible to understand the sentiment in a more nuanced manner [4] [6].

In conclusion, RAG plays a pivotal role in our project by addressing the unique challenges associated with financial report analysis in the MENA region. Its integration into our NLP pipeline enhances the accuracy, relevance, and timeliness of the generated insights, thereby significantly improving the overall efficacy of our analysis tool.

■ Implementation

The implementation of Retrieval-Augmented Generation (RAG) in our project significantly enhanced the accuracy and relevance of the responses generated by our model. By integrating external knowledge sources into the generative process, RAG effectively addresses the challenges posed by the unstructured and often incomplete financial data from the MENA region.

We implemented the RAG framework using a combination of a retriever and a generator. The retriever is based on a bi-encoder architecture that processes input queries and candidate documents separately, encoding them into dense vectors. These vectors are then used to compute similarity scores, and the top-k documents are selected as the most relevant.

The following code snippet illustrates the setup of the retriever and the process of querying the relevant documents:

```
1
2 from llama_index.embeddings.huggingface import
   HuggingFaceEmbedding
3 from llama_index.core import Settings,
   SimpleDirectoryReader, VectorStoreIndex
4 from llama_index.core.retrievers import
   VectorIndexRetriever
5 from llama_index.core.query_engine import
   RetrieverQueryEngine
```

```

6 from llama_index.core.postprocessor import
  SimilarityPostprocessor
7
8 # import any embedding model on HF hub
9 Settings.embed_model = HuggingFaceEmbedding(model_name=
  "BAAI/bge-small-en-v1.5")
10 Settings.llm = None # we won't use LlamaIndex to set up
  LLM
11 Settings.chunk_size = 256
12 Settings.chunk_overlap = 25
13
14 documents = SimpleDirectoryReader("filings").load_data
  ()
15 index = VectorStoreIndex.from_documents(documents)
16
17 # set number of docs to retrieve
18 top_k = 3
19
20 # configure retriever
21 retriever = VectorIndexRetriever(
22     index=index,
23     similarity_top_k=top_k,
24 )
25
26 # assemble query engine
27 query_engine = RetrieverQueryEngine(
28     retriever=retriever,
29     node_postprocessors=[SimilarityPostprocessor(
30         similarity_cutoff=0.5)],
31 )
32 query = "Tell me about Eshraqs residential
33         apartments in Marina Rise registered record
34         occupancy rates in Q3 2023?"
35 response = query_engine.query(query)
36
37 # reformat response
38 context = "Context:\n"
39 for i in range(top_k):
40     context = context + response.source_nodes[i].text +
41         "\n\n"
42 print(context)

```

Code 5. Setting up the retriever and querying relevant documents.

The generator component, which uses a fine-tuned sequence-to-sequence transformer model, generates responses based on the input query and the retrieved documents. Here is the implementation of the generator and an example of its usage:

```

1 # load fine-tuned model from hub
2 from peft import PeftModel, PeftConfig
3
4 model_name = "TheBloke/Mistral-7B-Instruct-v0.2-GPTQ"
5 model = AutoModelForCausalLM.from_pretrained(model_name
6 ,
7         device_map="auto",
8         trust_remote_code=False,
9         revision=
10         "main")
11
12 # configure the base model to the fine-tuned version.
13 config = PeftConfig.from_pretrained("ranystephan/
14     NeuralFinGPT-v1")
15 model = PeftModel.from_pretrained(model, "ranystephan/
16     NeuralFinGPT-v1")
17
18 # load tokenizer
19 tokenizer = AutoTokenizer.from_pretrained(model_name,
20     use_fast=True)
21
22 instructions_strings = f"""NeuralFinGPT, functioning as
23     a virtual quantitative \
24     finance consultant, analyst. Communicates in clear
25     language, accessible \
26     language, escalating to technical depth upon requests.
27     It reacts to feedback \
28     and uses its knowledge to comprehensively answer
29     questions. Do not add any \
30     additional text, do not be verbose, just answers the
31     question. The question \
32     that will be asked will be in relation to the data
33     given to the model. The \
34     model will not be expected to find any third-party
35     information.
36     Please respond to the following question.

```

```

26 """
27
28 prompt_template = lambda input: f'''[INST] {
29     instructions_strings} \n{input} \n[/INST]'''
30 question = "Tell me about Eshraqs residential
31         apartments in Marina Rise registered record
32         occupancy rates in Q3 2023?"
33
34 prompt = prompt_template(question)
35 print(prompt)
36
37 model.eval()
38
39 inputs = tokenizer(prompt, return_tensors="pt")
40 outputs = model.generate(input_ids=inputs["input_ids"].
41     to("cuda"),
42                             max_new_tokens=280)
43 print(tokenizer.batch_decode(outputs)[0])

```

Code 6. Generating responses using the fine-tuned model.

Without RAG:

Question: Tell me about Eshraq's residential apartments in Marina Rise registered record occupancy rates in Q3 2023?

Answer: Eshraq's residential apartments in Marina Rise achieved a registered record occupancy rate of 98.5% in Q3 2023. This high occupancy rate underscores the strong demand for Marina Rise's prime waterfront location and high-quality amenities.

With RAG:

Question: Tell me about Eshraq's residential apartments in Marina Rise registered record occupancy rates in Q3 2023?

Context Window: Jacques Fakhoury as a new Board Member, both of whom bring significant portfolio management and investment experience across a range of sectors and will play an important role in guiding our future growth strategy, which we expect to unveil early next year.[...]

Eshraq's residential apartments in Marina Rise registered record occupancy rates in Q3 2023, reaching average occupancy of 94% for the 9-month period, a reflection of the assets' premium standing in the market. Eshraq continues to optimise its real estate portfolio. [...]

Answer: Eshraq's residential apartments in Marina Rise reached record average occupancy of 94% in Q3 2023, reflecting the assets' premium standing in the market.

Comparison and Results

The implementation of RAG in our project significantly improved the quality and relevance of the responses. The RAG-enhanced model provided more accurate and contextually appropriate answers, leveraging the additional information retrieved during the query process. This demonstrates the effectiveness of RAG in handling the complexities of financial data analysis in the MENA region.

■ Prompt Engineering

Prompt engineering is a critical aspect of refining and optimizing the performance of language models. It involves crafting specific and detailed prompts to guide the model's responses, ensuring that the output is relevant, accurate, and contextually appropriate. In the context of NeuralFinGPT, prompt engineering played a pivotal role in enhancing the model's ability to function effectively as a virtual quantitative finance consultant.

The carefully devised prompt for NeuralFinGPT, as shown below, provides clear and precise instructions on how the model should behave. This prompt outlines the model's role, communication style, and the scope of its responses, ensuring that the answers are concise and focused on the given data.

NeuralFinGPT, functioning as a virtual quantitative finance consultant, analyst. Communicates in clear language, accessible language, escalating to technical depth upon requests. It reacts to feedback and uses its knowledge to comprehensively answer questions. Do not add any additional text, do not be verbose, just answers the question. The question that will be asked will be in relation to the data given to the model. The model will not be expected to find any third-party information.

Please respond to the following question.

Engineered prompt for NeuralFinGPT.

Before implementing prompt engineering, the model's responses tended to be overly verbose and occasionally included irrelevant information. For instance, the response to the prompt "What is value at risk in finance?" was detailed but lacked conciseness and focus:

Prompt: What is value at risk in finance?

Answer: Value at Risk (VaR) is a statistical measure used in finance to estimate the maximum potential loss that a portfolio or investment may experience with a given level of confidence over a specified time period, typically one day. It quantifies the risk of losses that could be incurred from market movements under normal circumstances. VaR is calculated using historical market data and statistical methods, such as the standard deviation or the percentile method. It's an important tool for risk managers to assess and manage the market risk of their investments. However, it's important to note that VaR does not account for all types of risk, such as credit risk or liquidity risk.

Model response before prompt engineering.

After implementing the engineered prompt, the model produced more concise and relevant responses. The improved response to the same question is:

Value at Risk, or VaR for short, is a risk management concept used in finance to estimate the potential loss that a portfolio or investment may experience over a given time period, typically one day, with a specified level of confidence.

The VaR calculation provides an estimate of the maximum possible loss under normal market conditions. It's important to note that VaR does not account for extreme market movements, also known as tail risks.

VaR is usually expressed as a dollar amount or as a percentage of the portfolio's value. For example, a VaR of \$1 million for a \$100 million portfolio represents a 1% risk.

Model response after prompt engineering.

This example demonstrates the effectiveness of prompt engineering in guiding the model to deliver more structured and contextually appropriate answers. By clearly defining the model's role and response expectations, we significantly improved the quality of the outputs, making NeuralFinGPT a more reliable and useful tool for financial analysis.

■ Function Automation

Function automation in NeuralFinGPT enhances both the usability and efficiency of the model by enabling it to autonomously determine the appropriate methods to address various financial queries. This section details the implementation of function automation, its usability benefits, and the specific techniques employed to achieve this functionality.

Usability of Function Automation

The primary goal of function automation in NeuralFinGPT is to streamline the interaction process for users, allowing them to obtain relevant financial insights quickly and accurately. By automating the selection of methods based on the query's content, NeuralFinGPT can:

- Reduce the need for user intervention in selecting the appropriate analytical method.
- Ensure consistent and accurate application of the methods to queries.
- Enhance the model's efficiency by directly routing queries to the suitable computational function.

This automation is particularly useful in financial analysis, where different types of queries—such as sentiment analysis, stock price tracking, and news summarization—require distinct approaches. By predefining a set of potential prompts and corresponding methods, NeuralFinGPT can effectively categorize and address these queries.

Implementation of Function Automation

The implementation of function automation in NeuralFinGPT involves a combination of rule-based selection and model-based inference. The process is as follows:

Predefined Methods

A set of potential prompts is predefined, each corresponding to a specific method. These prompts include tasks such as "Sentiment Analysis Checker for Report," "Daily Stock Price Tracker," and "Recent News about Company or Index." These predefined methods serve as a guideline for the model to follow when determining the appropriate response to a query.

Rule-Based Method Selector

A rule-based method selector is implemented to evaluate the query's content and select the appropriate method based on predefined rules. This selector checks for specific keywords in the query to determine the suitable method. For example, if the query includes terms like "sentiment" and "report," it selects the sentiment analysis method; if it includes terms like "stock price" or "daily price," it selects the stock price tracker method.

Model-Based Inference

If the rule-based selector cannot determine the method, the model is queried to identify the suitable method and provide the necessary arguments. The model has been fine-tuned using the QLoRA (Quantized Low-Rank Adaptation) technique, which optimizes its performance on limited computational resources while maintaining high accuracy.

The model is provided with a structured prompt that includes instructions and the list of available methods. It then evaluates the query and returns the number corresponding to the most suitable method or indicates that none apply.

Example Dataset for Model Training

The model used for function automation is pretrained to identify the appropriate method for a given query and provide the necessary arguments. Table ?? shows an example dataset used for training this model, which includes a variety of financial queries and their corresponding method types and arguments.

Core Functionality

The core functionality of NeuralFinGPT's function automation lies in its ability to autonomously determine the appropriate method for various financial queries and execute the relevant computational functions. This process involves:

1. **Query Analysis:** The model analyzes the query to identify the task and determine the suitable method.
2. **Method Selection:** Based on predefined rules or model inference, the appropriate method is selected.
3. **Argument Extraction:** The model extracts necessary arguments for the selected method from the query.
4. **Execution:** The selected method is executed with the extracted arguments to provide a precise and relevant response.

By leveraging both rule-based and model-based approaches, NeuralFinGPT can efficiently automate the selection and execution of the most appropriate methods for various financial queries, thereby enhancing its usability and reliability as a financial analysis tool.

■ Discussion

Current Achievements

The NeuralFinGPT project has successfully advanced the application of Natural Language Processing (NLP) and large language models (LLMs) in the realm of financial analysis within the MENA region. One of the primary achievements of this project is the comprehensive collection of a substantial dataset of financial reports from various companies in the region. This was accomplished despite the significant challenges posed by inconsistent formatting and handwritten sections, demonstrating the project's robust data collection capabilities.

Another notable achievement is the effective data preprocessing pipeline developed to transform the collected financial reports into a usable format. This preprocessing involved extensive cleaning techniques to address issues such as non-ASCII characters, malformed symbols, and irrelevant information. The resulting clean and normalized text facilitated accurate sentiment analysis and other computational tasks.

The implementation of sentiment analysis, particularly through the use of the FinBERT model, has proven to be highly effective in identifying sentiment and analyzing forward-looking statements within financial texts. This capability provides valuable insights into company performance and outlook, which are critical for investors and analysts.

Furthermore, the creation of a PostgreSQL database to store and manage the processed data has enhanced the project's efficiency. This database provides a structured and accessible repository, enabling efficient querying and analysis of financial data.

The fine-tuning of the Mistral7b model using Quantized Low-Rank Adaptation (QLoRA) represents a significant technical achievement. This approach allowed the model to be fine-tuned on limited computational resources, such as Google Colab's T4 GPU, without compromising performance. This demonstrates the practicality of deploying advanced NLP techniques in resource-constrained environments.

The integration of Retrieval-Augmented Generation (RAG) has significantly improved the accuracy and relevance of the model's responses. By incorporating external knowledge sources into the generative process, RAG enhances the model's ability to produce contextually enriched and accurate answers, addressing one of the core challenges of analyzing unstructured financial data.

Moreover, detailed prompt engineering has refined the model's outputs, ensuring that responses are concise and relevant. The development of automated functions for specific queries has further enhanced the usability and efficiency of NeuralFinGPT, making it a versatile tool for financial analysts.

Limitations

Despite the considerable achievements, the NeuralFinGPT project faced several limitations. The variability and poor quality of financial reports from the MENA region posed significant challenges. Although extensive preprocessing efforts mitigated these issues to some extent, the lack of standardized formats and the presence of handwritten sections limited the consistency of the data. This variability necessitated robust preprocessing techniques to ensure the data's suitability for analysis, but it also introduced potential inaccuracies.

Another limitation is related to computational constraints. While QLoRA enabled effective fine-tuning on limited resources, more powerful hardware could further enhance model performance and reduce training time. The current computational setup, although sufficient for initial development, may not support the scaling required for larger datasets and more complex analyses.

The training dataset, despite being substantial, could benefit from further expansion. Including more diverse financial reports and historical data would improve the model's robustness and accuracy. This is particularly important for capturing the nuances of financial language and ensuring the model's applicability across different contexts within the MENA region.

Moreover, the current implementation does not fully support real-time data updates, which are crucial for dynamic financial analysis and decision-making. The ability to ingest and process data in real-time would significantly enhance the model's utility, allowing it to provide up-to-date insights and respond promptly to market changes.

Future Work

Future work on the NeuralFinGPT project will focus on addressing these limitations and expanding the project's capabilities. Enhancing data collection and standardization efforts will be a priority. Collaborating with local regulatory bodies and financial institutions to improve the quality and accessibility of public filings can help achieve this goal. Developing a more scalable infrastructure, including upgrading computational resources, will enable more efficient model training and deployment.

Expanding the training dataset to include a broader range of financial reports and historical data is also a key area of focus. This will help improve the model's performance and ensure its relevance in diverse financial contexts. Additional fine-tuning on specific financial tasks and regional data will further enhance the model's accuracy and applicability.

Implementing real-time processing capabilities is another critical objective. Developing mechanisms for real-time data ingestion and processing will allow NeuralFinGPT to provide timely insights, which are essential for financial decision-making.

Lastly, creating a user-friendly interface and integrating NeuralFinGPT with existing financial analysis tools will improve accessibility and usability for financial analysts and decision-makers. This will help translate the model's technical capabilities into practical applications, making it a valuable tool for the financial community in the MENA region.

In conclusion, the NeuralFinGPT project has demonstrated significant potential for improving financial analysis in the MENA region through the application of advanced NLP techniques and LLMs. Despite facing challenges related to data quality and computational resources, the project has achieved notable successes and laid a strong foundation for future enhancements. By addressing current limitations and expanding its capabilities, NeuralFinGPT can become an indispensable tool for financial analysis and decision-making in the region.

■ REFERENCES

- [1] K. Aljifri and K. Hussainey, “The determinants of forward-looking information in annual reports of UAE companies”, *Managerial Auditing Journal*, vol. 22, no. 9, pp. 881–894, 2007. DOI: <http://dx.doi.org/10.1108/02686900710829390>.
- [2] D. C. Araci, “FinBERT: A Pretrained Language Model for Financial Communications”, *arXiv preprint*, 2019. [Online]. Available: <https://arxiv.org/abs/1908.10063>.
- [3] K. Guu, K. Lee, Z. Tung, P. Pasupat, and M.-W. Chang, “Retrieval as Generation: Improving Open-Domain Question Answering by Combining Dense Retrieval and Language Generation”, *arXiv preprint*, 2020. [Online]. Available: <https://arxiv.org/abs/2002.08909>.
- [4] V. Karpukhin, B. Oğuz, S. Min, *et al.*, “Dense Passage Retrieval for Open-Domain Question Answering”, *arXiv preprint*, 2020. [Online]. Available: <https://arxiv.org/abs/2004.04906>.
- [5] S. V. Kolasani and R. Assaf, “Predicting stock movement using sentiment analysis of Twitter feed with neural networks”, *Journal of Data Analysis and Information Processing*, vol. 8, no. 4, pp. 309–319, 2020. DOI: <http://dx.doi.org/10.4236/jdaip.2020.84018>.
- [6] P. Lewis, E. Perez, A. Piktus, *et al.*, “Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks”, *arXiv preprint*, 2020. [Online]. Available: <https://arxiv.org/abs/2005.11401>.
- [7] T. Wolf, L. Debut, V. Sanh, *et al.*, “Transformers: State-of-the-Art Natural Language Processing”, *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pp. 38–45, 2020. DOI: [10.18653/v1/2020.emnlp-demos.6](https://doi.org/10.18653/v1/2020.emnlp-demos.6).
- [8] J. Ni, G. Hahn-Powell, S. W.-t. Yih, and S. Riedel, “RAG for LLMs: A Survey”, *arXiv preprint*, 2021. [Online]. Available: <https://arxiv.org/abs/2103.01207>.
- [9] D. Yogatama, C. de Masson d’Autume, L. Kong, J. Berant, A. Lazaridou, and W. Ling, “Efficient Use of Retrieval in RAG Models”, *arXiv preprint*, 2021. [Online]. Available: <https://arxiv.org/abs/2104.08115>.
- [10] S. Borgeaud, A. Mensch, J. Hoffmann, *et al.*, “Improving Language Models by Retrieving from Trillions of Tokens”, *arXiv preprint*, 2022. [Online]. Available: <https://arxiv.org/abs/2112.04426>.
- [11] T. Detrmers, A. Pagnoni, A. Holtzman, and L. Zettlemoyer, “QLORA: Efficient Finetuning of Quantized LLMs”, *arXiv preprint*, 2023. DOI: <https://arxiv.org/abs/2305.14314>.
- [12] A. Q. Jiang, A. Sablayrolles, A. Mensch, *et al.*, “Mistral 7B: A 7-billion-parameter language model engineered for superior performance and efficiency”, *arXiv preprint*, 2023. DOI: <https://arxiv.org/abs/2310.06825>.
- [13] H. Liu, J. Wang, J. Zhu, and B. Chen, “Alpha-GPT: Human-AI interactive alpha mining for quantitative investment”, *arXiv preprint*, 2023. DOI: <http://arxiv.org/abs/2302.03771>.
- [14] S. Wu, O. Irsoy, S. Subramanian, *et al.*, “BloombergGPT: A large language model for finance”, *arXiv preprint*, 2023. DOI: <http://arxiv.org/abs/2303.17564>.
- [15] H. B. Yang, X. Liu, and C.-D. Wang, “FinGPT: Open-Source Financial Large Language Models”, *Columbia University, New York University (Shanghai)*, 2023. DOI: <https://github.com/AI4Finance-Foundation/FinGPT>.
- [16] N. Zmandar, M. El-Haj, and P. Rayson, “FinAraT5: A text to text model for financial Arabic text understanding and generation”, *UCREL NLP Group, Lancaster University*, 2023.
- [17] J. SV, “pdfplumber (Version 0.8.0)”, *GitHub Repository*, 2024. [Online]. Available: <https://github.com/jsvine/pdfplumber>.

Table 1. Example dataset for training the function automation model.

Query	Type	Arguments
What was the closing price of Google stock yesterday?	2	{'company': 'Google', 'date': '2024-05-12'}
Find the price of Shopify shares three days ago.	2	{'company': 'Shopify', 'date': '2024-05-10'}
Do we have any fresh news regarding the S&P 500?	3	{'index': 'S&P 500'}
What was the last recorded price for the AMD stock?	2	{'company': 'AMD', 'date': 'last recorded'}

Note: This dataset includes various financial queries with their corresponding method types and arguments.