Research Project Proposal

Developing a Small Language Model (SLM) for Financial Analysis

G31

E/19/131
Kasuni Hansachapa

Department of Computer Engineering

Faculty of Engineering

University of Peradeniya

2025

Research Project Proposal

Developing a Small Language Model (SLM) for Financial Analysis

G31

E/19/131
Kasuni Hansachapa

Department of Computer Engineering

Faculty of Engineering

University of Peradeniya

2025

Abstract

This study develops a Small Language Model (SLM) which specifically processes financial data at Lanka Education and Research Network (LEARN). The model is trained on LEARN's financial data, including five years of financial statements and audit reports, to enhance decision-making and analysis.The SLM achieves both efficiency and accuracy through several approaches including quantization techniques along with pruning methods, knowledge distillation methods and retrieval-augmented generation (RAG). The model's reliability depends on its adoption of QLoRA for quick model optimization[28] and the inclusion of hallucination reduction mechanisms.The system provides support to LEARN staff members performing financial analysis in an accurate and reliable manner.

The implementation involves data preprocessing, model training, fine-tuning on domain-specific documents, and integration into a user-accessible platform.The input consists of structured and unstructured financial data, which the model processes to generate insights and predictions. The financial data protection is ensured through local hosting of the entire system on LEARN's server. This research outlines how SLMs function as an efficient domain-specific solution against LLMs for financial use cases with regulatory compliant and secure functionality that users can handle easily.

.

Table of Contents

List of Figures

List of Tables

Chapter 1 - Introduction

1.1 Introduction

Large Language Models (LLMs) have transformed Natural Language Processing (NLP) through their advanced capability to understand and reason alongside generating human-level text throughout different general-domain operations LLMs achieve high performance levels across various applications which include code writing [3], math problem solving [4],dialogue [7], common sense reasoning [1], and symbolic reasoning [11]. The ability of LLMs to support question-answering chatbots and automation applications has turned into one of the main use cases over the past several years[1].The financial industry traditionally uses Large Language Models (LLMs) including GPT-3 and BloombergGPT to carry out financial applications(automated decision making,risk assessments,fraud prevention and financial forecasting etc) with their superior natural language processing (NLP) features. However, the high computational cost, large-scale infrastructure requirements, and risk of generating hallucinated outputs limit their practical use, especially for organizations operating with constrained resources.

Small Language Models (SLMs) stand as a practical substitute that combines effective resource utilization with domain-specific accuracy. The combination of quantization with pruning together with knowledge distillation SLMs to match LLM performance levels yet function using much reduced computing resources. Unlike general-purpose LLMs, this SLM is fine-tuned on financial documents using QLoRA for efficient training,and RAG enables with a strong emphasis on hallucination control .Additionally, financial AI applications require high precision, as errors in predictions can lead to serious financial consequences.Financial insights demand reliable results which make data security and hallucination reduction necessary for achieving them.Financial institutions are increasingly adopting AI, but regulatory frameworks and ethical standards for their use are still developing.[2]

The research project creates a SLM prototype which targets the Lanka Education And Research Network (LEARN) through training on five years of financial reports and audit reports. This financial model supports LEARN staff in an accurate and reliable manner. Security of the system rests on its deployment to LEARN local server because this approach eliminates the data risks found in cloud-based AI models.

1.2 Aims and objectives

Aim : The main objective of this study investigates Small Language Modelling (SLM) framework development for financial data modeling at LEARN while prioritizing computational efficiency optimization and hallucination prevention alongside security and reliability improvements.

Objectives

The specific objectives of this research are:

1. **Optimization of Small Language Models for Financial Tasks:** The project will investigate optimization methods including quantization, pruning, knowledge distillation, QLoRA and Retrieval-Augmented Generation (RAG) to improve Small Language Model performance when used in financial applications.By refining these techniques, the research seeks to demonstrate how smaller models can effectively balance efficiency and accuracy.
2. **Hallucination Mitigation and Accuracy Improvement:** The main goal involves developing methods which reduce hallucinations alongside accuracy enhancement in financial SLMs. Research works to find out how adversarial training methods with self-supervised learning techniques and hybrid system implementations help enhance the factual accuracy within model outputs.Additionally, the research will explore how fine-tuning domain-specific question models can further enhance prediction reliability.
3. **Data Security and Bias Mitigation:** The model deployment on LEARN's local servers ensures financial information privacy through data security measures. Bias mitigation techniques will be developed to make financial models deliver fair transparent results.
4. **Development of Multimodal Data Integration Methods:** Integrating multimodal data;such as numerical financial data and textual reports into SLMs. This will enhance the predictive capabilities of the models, providing a more holistic approach to financial forecasting, risk analysis, and decision-making.

1.3 Solution Overview

The proposed platform enables LEARN staff members to access AI-driven tools which improve financial assessment while maximizing decision-making efficiency. The system uses optimized financial data-driven Small Language Models to provide solutions with accurate and secure processing capabilities. The system includes the following components as explained in detail below:

a.Users

The primary users of this system are LEARN finance and administrative staff.The AI-driven tools enable these users to perform financial record management while generating insights and analyzing data together with detecting financial data inconsistencies. Through this system the staff can accomplish their financial analysis task efficiently.

b.Input

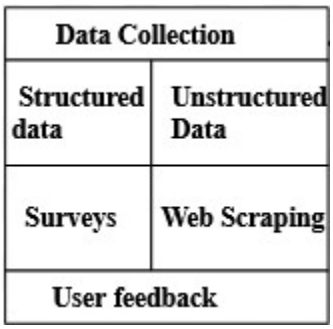

Figure 1.1: Data collection methods

The system primarily retrieves PDF financial statement reports from five years of financial data collection, and the structured data can further enhance model training.

- **Structured Data:** Financial documents such as balance sheets and income statements and cash flow statements and budgetary spreadsheets provided in CSV and Excel documents format fall under structured data category.
- **Unstructured Data:** The system analyzes unstructured financial information from financial audit reports together with bank statements and receipts and contracts toward providing textual content in PDF or different file types.
- **Surveys:** Senior LEARN employees along with stakeholders and clients who receive surveys will give valuable qualitative data concerning financial issues and priorities and concerns.
- **Web Scraping Data**: The system also scrapes financial data from various sources to ensure comprehensive data collection and analysis.

c.Process

The data processing starts with multiple sequential steps to achieve precision in financial domain-specific model development and data handling:

- **Data Preprocessing**: Data cleaning operations begin after which the system applies structural organization to the data during the preprocessing phase. The conversion process for unstructured PDF information depends on OCR technology and alternative text extraction systems to make the data readable by machines. During this phase tokenization becomes part of the process together with normalization and various other NLP techniques.
- **Fine-tuning with QLoRA[28]:** QLoRA functions as a fine-tuning method for pre-trained language models to adapt the models using the financial dataset. With this technique the model achieves high performance levels with reduced computational load that leads to accelerated training and faster inference execution.
- **Integration of RAG**: **Retrieval-augmented Generation (RAG):** The model uses Retrieval-augmented Generation (RAG) for enhancing its capacity to generate precise and contextually suitable replies. The model achieves better performance through RAG because it retrieves significant financial information to handle intricate queries and financial irregularities.
- **Hallucination Reduction**: Financial AI models must deal with the critical issue of hallucinations because they produce incorrect and fabricated results through their generated outputs. The system implements various methods to reduce hallucinations to produce outputs which can be used confidently for decision-making.

d.Output

The analysis generates various output types through the system.

- **Financial Summaries**: The system creates automated financial summarization of LEARN's financial data including statements and performance indicators together with financial measurement factors for high-level financial status overview.
- **Financial Classification Tasks:** The system classifies financial data into relevant categories, such as expense types, income sources, and asset classes. This structured classification supports improved data organization and analysis.
- **Financial Prediction**: The system generates predictions related to future financial performance, such as revenue forecasting, market trends, or asset valuations. These predictions help in strategic decision-making and long-term financial planning.
- **Sentiment Analysis:** The system conducts sentiment analysis on financial documents, reports, and external market communications. This helps to gauge

the tone and sentiment of financial content, providing insights into the market's or organization's emotional response to financial events

e.System Requirements

The system demands this infrastructure for efficient processing along with system operation:

- The system needs a high-performance local server with GPU acceleration because financial data modeling and inference demands quick processing throughout model training as well as fine-tuning sessions.
- The financial documents require sufficient storage capacity to accommodate large historical datasets along with reports as well as documents.
- Secure authentication mechanisms need to be implemented because the system handles sensitive data therefore only authorized personnel must access and interact with the system.

1.4 Structure and Dissertation

The dissertation is structured as follows:

Chapter 2 provides a detailed review of related works, comparing Small Language Models (SLMs) with Large Language Models (LLMs) and examining previous research on approaches and techniques in financial NLP.

Chapter 3 introduces the proposed approach, focusing on the use of Optimized Retrieval-Augmented Generation (RAG) combined with LoRA to create a more efficient and effective question-answering system.

Chapter 4 explores the technology adapted for the project, including the techniques used to optimize small language models and the technologies employed to facilitate efficient development.

Chapter 5 delves into the analysis and design of the system, covering aspects such as data collection, preprocessing, system development, experiments, and evaluation of the model’s performance.

Chapter 6 details the implementation phase, including the practical aspects of data collection, preprocessing, system development, experiments, and final evaluation of the system.

Chapter 7 provides a comprehensive discussion on the findings, implications of the research, and possible future directions in financial AI applications.

1.5 Summary

This introductory section presents the development of a Small Language Model (SLM) for financial data in a resource-constrained environment. The study highlights challenges such as limited computational power, accuracy concerns, and hallucination risks in financial applications.[9],[10] By integrating quantization, pruning, LoRA fine-tuning, and Retrieval-Augmented Generation (RAG), the model achieves efficiency while maintaining precision.

Test reference [3]

Chapter 2 – Related work

2.1 Introduction

This section provides a concise overview of key technologies and methodologies explored in the literature for optimizing Small Language Models (SLMs). The focus is on various techniques such as Quantization, Pruning, Knowledge Distillation, Retrieval-Augmented Generation (RAG), and LoRA, each aimed at enhancing model efficiency and performance while minimizing computational demands.

2.2 SLM vs LLM

Table 1: Comparison between Small Language models(SLM) & Large Language models(LLM)

Feature	**SLM**	**LLM**
Size & Complexity	Smaller in size, with fewer parameters	Massive in size, billions of parameters
Computational Requirements	Low computational power; optimized for efficiency	Requires high-end GPUs/TPUs and large-scale infrastructure
Training Cost	Lower cost due to smaller datasets and fewer resources	Extremely expensive due to vast datasets and high training complexity
Performance in General Tasks	Specialized performance, optimized for specific domains	Strong general-purpose capabilities, excelling in diverse NLP tasks
Accuracy & Precision	High accuracy within a specific domain (e.g., finance)	High accuracy in general NLP but prone to hallucinations in specialized fields
Data Efficiency	Requires fewer data for training and fine-tuning	Needs massive datasets for training and generalization

2.3 Approaches and Techniques in Previous Research

The first part of review covers preliminary knowledge in several approaches and techniques in previous research:

1. **Optimization Techniques:** Methods such as quantization, pruning, imitation learning, progressive learning, knowledge distillation, and reasoning distillation are explored as strategies to enhance the efficiency of SLMs. These techniques allow smaller models to achieve comparable performance to LLMs while significantly reducing computational demands.
2. **Hallucination Mitigation:** One of the primary concerns in financial AI applications is the generation of hallucinated or incorrect data. The paper discusses mitigation strategies, including Retrieval-Augmented Generation (RAG) and explanation tuning, which help improve the factual accuracy of model outputs. Additionally, hybrid approaches combining fine-tuned question models and RAG are examined.[21]
3. **Evaluation Metrics:** The review details various evaluation metrics applicable to SLMs, including perplexity, BLEU, TER, ROUGE, hallucination score[21], and domain-specific financial metrics like Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE). These metrics are essential for assessing model performance in text generation, prediction, and information retrieval tasks.Also this describes financial benchmarks for accurate evaluation.
4. **Challenges in Financial SLMs:** The review highlights critical challenges, such as data security, hallucination, and ethical concerns in deploying SLMs in financial applications. Financial data security is particularly emphasized, with discussions on private hosting solutions, federated learning, and differential privacy as potential safeguards.
5. **Domain-Specific Applications:** The paper explores how SLMs are tailored for industry-specific applications, including medical, legal, retail, and financial domains. In finance, SLMs support market prediction, financial report analysis, and automated customer interactions. The importance of domain-specific fine-tuning to enhance model accuracy is underscored.

2.4 Summary of the lit review

This paper provides a comprehensive literature review on Small Language Models (SLMs) and their application in the financial domain. The motivation behind SLMs stems from the high computational cost and inefficiency of Large Language Models (LLMs), making smaller models a viable alternative for specialized applications.

Related Works

The second part of the review paper focuses on the advancements in the finance domain, specifically highlighting the integration of language models (LMs) into financial tasks. The financial industry is defined by its vast numerical data, complex terminology, abbreviations, and specialized language, which makes it a challenging field for language models. Recent breakthroughs in LMs have resulted in major improvements in areas such as automated financial statement analysis, personalized report generation, financial forecasting, risk management, and compliance.

Prominent models like FINBERT (2022), BloombergGPT, and FinGPT are designed specifically for the financial domain. BloombergGPT, for instance, is one of the largest finance-focused models, with an impressive 50 billion parameters. Meanwhile, FinGPT serves as an open-source tool aimed at enhancing the development of financial LMs. These models have been adapted to a variety of tasks, including sentiment analysis, entity recognition, and financial classification, which are vital for extracting insights from financial data.

While these large-scale models exhibit impressive performance, research has shown that smaller models tend to have lower accuracy, particularly for more complex financial tasks. Models like Gemma-2B and OpenELM-270M have been evaluated in zero-shot and few-shot learning scenarios, and while few-shot models perform better in specific metrics like ROUGE scores, they still lag behind larger models, such as BloombergGPT, in overall performance.

In Table 2, the performance of different language models in terms of resource allocation and inference time is shown. For instance, models with fewer parameters, like Apple-OpenELM-270M, require lower GPU utilization compared to larger models, which can be more resource-intensive. This highlights the importance of balancing resource efficiency with the performance capabilities of financial language models.

Recent innovations in large language models (LLMs) have enhanced insights, trend identification, and assessment capabilities, with notable models like FINBERT, BloombergGPT, and FinGPT tailored specifically for financial tasks. A comparison of these models is conducted in Table 3.

Key Models:

1. **FINBERT (2022)**: Specialized in sentiment analysis, financial entity recognition, and classification.
2. **BloombergGPT**: A large-scale 50-billion-parameter model trained on comprehensive financial data, designed for robust financial analysis tasks.This is not an open source language model.

3. **FinGPT**: An open-source model focusing on financial document summarization and question-answering, fostering financial decision-making automation.

Small language models (SLMs), such as FinBERT, BloombergGPT, and FinMA, have been optimized for financial NLP tasks like sentiment analysis, document classification, and text summarization. These models serve various roles, from sentiment analysis to enhancing financial predictions, summarizing documents, and answering complex questions.

Model Comparison:

- **InvestLM**: Trained using CFA exam questions and SEC filings, with a model size of 658 million parameters.
- **FinGPT**: Focuses on document summarization, available in both smaller and larger versions.
- **BloombergGPT and FLANG**: Known for sentiment analysis and named entity recognition.

Additionally, models such as TinyLlama, Google-Gemma, and Microsoft-Phi vary in size and dataset usage, reflecting the trade-off between computational efficiency and task performance.

Challenges and Future Directions: Despite their successes, SLMs still face challenges, including issues with data security, model bias, and hallucination control. The future development of financial language models should focus on enhancing their accuracy, integrating multiple data sources, and incorporating ethical guidelines to ensure responsible use in the financial sector. Additionally, advances in technologies like Retrieval-Augmented Generation (RAG) and optimization techniques such as quantization and knowledge distillation hold potential for improving model reliability and performance in real-world financial applications.

Table 2 : Hardware Requirements for language models

Model	GPU (GiB)	RAM (MB)	Avg. Inf Time (sec)

(1)Apple-OpenELM-270M	2.2	642.2977	5.64
(2)Apple-OpenELM-450M	3.7	588.7348	7.32
(3)Apple-OpenELM-1.1B	8.2	765.3945	9.89
(4)Apple-OpenELM-3B	13.6	473.3031	14.60
(5)Microsoft-phi-1B	8.2	759.8051	7.28
(6)Microsoft-phi-1.5B	8.2	670.2625	7.30
(7)Microsoft-Phi-2B	10.3	410.8238	7.07
(8)Google-gemma-2B	9.5	792.9766	6.68
(9)TinyLlama-1.1B	8.3	721.0668	5.65

Table 3 : Model parameters count and capabilities of language models

Finance-Specific Language Models	Model Parameters	Model Capabilities
BloombergGPT 50B [62] Dataset:FinPile	50B [62]	● Sentiment analysis ● Named entity recognition ● Question answering
FinBERT(open)	110M [62]	● Sentiment analysis

Dataset:Financial PhraseBank		• Financial entity recognition • Financial classification tasks
FLANG (open)	110M [62]	• Sentiment analysis • Named entity recognition • Document classification
InvestLM(fine-tuned LLaMA-open) Dataset:CFA,SEC	65B [61]	• Sentiment analysis • Financial text classification
FinMA(fine-tuned LLaMA-open) Dataset:PIXIU	7B and 13B [61]	• Sentiment analysis • Financial document summarization • Question answering
FinGPT(open) Dataset:FinQA,FinRed	7B and 13B [61]	• Financial document summarization • Question answering
Google-gemma 2B [61]	2B [61]	• Financial Text Classification • Financial Document Summarization • Question Answering
TinyLlama(fine-tuned LLaMA)	1.1B [61]	• Financial Text Classification • Financial Document Summarization • Question Answering
Apple-OpenELM Dataset:RefinedWeb,Pile	270M - 3B [61]	• Sentiment Analysis • Named Entity Recognition (NER) • Document Classification
Microsoft-phi	1B - 3B [61]	• Sentiment Analysis • Named Entity Recognition (NER) • Document Classification • Question Answering

2.5 Summary

The literature review highlights recent advancements in language models (LMs) tailored for the finance domain, where large amounts of numerical data, specialized terminology, and complex transformations are prevalent. Models like FINBERT, BloombergGPT, and FinGPT have shown significant progress in financial applications such as automated financial analysis, forecasting, and risk management. While large models offer superior performance, smaller models tend to be more

resource-efficient but face accuracy challenges. The review suggests further development of small, resource-efficient models, integration of techniques like knowledge graphs and Retrieval-Augmented Generation (RAG), and the creation of more comprehensive financial datasets for future research.

Chapter 3 – Proposed Approach

3.1 Introduction

This chapter presents the proposed approach for the small language model on financial data. The workflow of the system is shown here. It follows a resource-efficient and accurate approach for developing a small language model for financial data analysis.

3.2 Optimized RAG with LoRA for Smarter QA

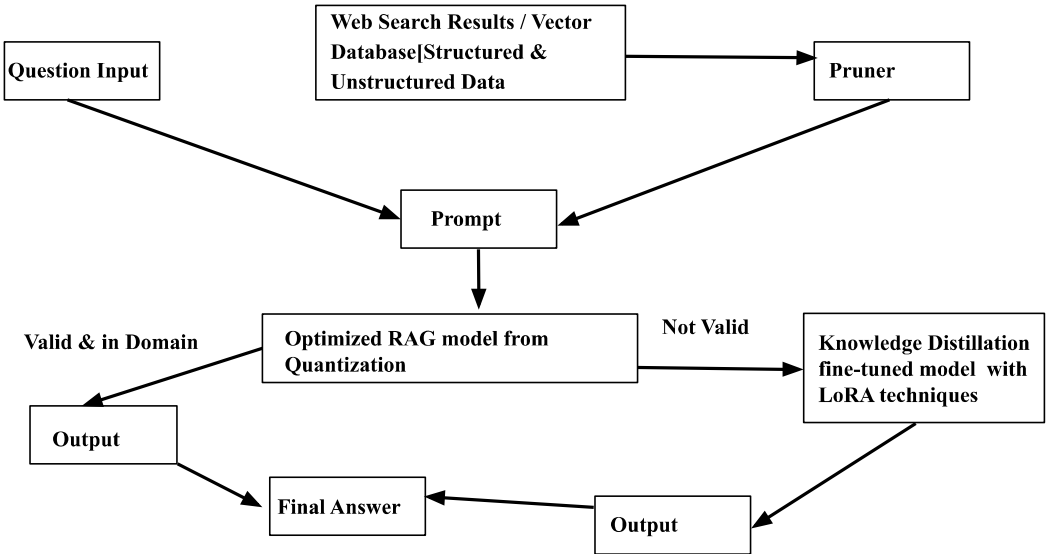

Figure 3.1: High level architecture of proposed approach

To address the challenge of efficient and accurate question answering, we adopt an optimized Retrieval-Augmented Generation (RAG) approach that integrates retrieval techniques with lightweight and fine-tuned language models. Users submit a question input, which is processed by retrieving relevant information from web search results or a vector database containing structured and unstructured data. A pruner filters out irrelevant content to retain only high-quality data, which is then formatted into a prompt for response generation. The system leverages an Optimized RAG Model with Quantization to reduce computational overhead while maintaining performance. Additionally, a Knowledge Distillation Fine-Tuned Model with LoRA Techniques enhances accuracy by refining the model's knowledge while reducing its size. The system generates initial outputs, which are further refined to produce the final answer, ensuring precision, relevance, and computational efficiency. By leveraging quantization, knowledge distillation, and LoRA-based fine-tuning, this approach balances high-quality responses with optimized performance, making it a robust solution for real-world question-answering applications.

3.3 Language model selection

For our task, which primarily focuses on analyzing financial statement PDFs, models like FinBERT (with 110 million parameters) and FinGPT (with parameters ranging from 2 billion to 8 billion) would be more suitable. These models are specifically trained on financial data and are effective for tasks like extracting key information, sentiment analysis, and classification from financial documents. FinBERT, with its efficient architecture, can be particularly useful if hardware resources are limited, while FinGPT can handle more complex financial data processing. Both models are designed to understand the unique terminology and structures present in financial statements, making them ideal choices for this task, and both are open source language models.

In addition to these, we could consider training the following language models within our own server:

1. TinyLlama (1.1 billion parameters) – A smaller model that could offer a balance between performance and resource usage.
2. Apple-OpenELM (with parameter sizes ranging from 270 million to 3 billion) – Known for its efficiency in training and inference, making it a viable option for resource-constrained environments.
3. InvestLM (658 million parameters) – Trained on financial datasets like CFA and SEC filings, this model specializes in sentiment analysis and financial text classification.

3.4 Summary

This optimized RAG approach enhances question answering by retrieving and filtering relevant data, using a quantized model and a fine-tuned LoRA-based model for efficiency. It ensures accurate, refined, and computationally efficient responses, making it ideal for real-world applications. Models like FinBERT and FinGPT are well-suited for financial statement analysis, leveraging their specialized training on financial data, while alternative models like TinyLlama, Apple-OpenELM, and InvestLM offer additional flexibility for various financial tasks.

Chapter 4 – Technology Adapted

4.1 Introduction

This section discusses optimization techniques and mitigation strategies for developing efficient Small Language Models (SLMs). The selected approaches focus on enhancing performance while minimizing resource consumption, ensuring interpretability, and supporting smooth deployment.

4.2 Optimizing Small Language Models: Techniques and Technologies for Efficient Development

Developing efficient Small Language Models (SLMs) requires a series of mitigation strategies and optimization techniques to enhance performance while reducing computational demands. The following methodologies have been selected specifically to address the challenges of resource efficiency, model interpretability, and deployment feasibility.

Table 4 : Technology adopted and how they solve the problems

Techniques	**Technologies**	**How it solves the problem**
Quantization(Optimization Techniques)	PyTorch(torch.quantization),TensorFlow Lite	To reduce memory usage and computational overhead, making SLMs more feasible for deployment in resource-constrained environments.
Pruning(Optimization Techniques)	TensorFlow Model OptimizationToolkit,PyTorchSparseML	To streamline models by removing redundant parameters, and reducing inference time
Knowledge Distillation (KD)(Optimization Techniques)	DistilBERT (Hugging Face)	To optimize efficiency while maintaining reasoning ability
Retrieval-Augmented Generation (RAG)(Mitigation Strategies)	LangChain,FAISS ,ChromaDB	To reduce hallucinations and improve contextual responses.
LoRA Techniques(Optimization Techniques)	Hugging Face PEFT LoRA by PyTorch	To reduce trainable parameters while preserving performance

Table 5: Summary of technologies for implementing the proposed design

Component	**Technology**
Frontend	React.js
Backend API	FastAPI(Python)
Model Serving	Hugging Face
Database	PostgreSQL(Structured data),MongoDB(Unstructured data),ChromaDB(VectorDB for RAG)
Authentication	JWT/OAuth2
Deployment	Docker,nginx,Gunicorn
CI/CD for Model Updates	GitHub Actions
Server & Infra	Ubuntu
Monitoring	Prometheus

4.3 Summary

This section highlights optimization techniques for Small Language Models (SLMs), including Quantization (PyTorch, TensorFlow Lite) to reduce memory usage, Pruning (TensorFlow, PyTorch) to streamline models, Knowledge Distillation (DistilBERT) to maintain reasoning ability, RAG (LangChain, FAISS, ChromaDB) for improved contextual accuracy, and LoRA (Hugging Face PEFT, PyTorch) to reduce trainable parameters while maintaining performance. These techniques enhance efficiency while addressing resource and deployment challenges.

Chapter 5 – Analysis and Design

5.1 Introduction

This research follows a resource-efficient and accurate approach for developing a small language model for facial data analysis. The methodology consists of several key stages: data collection, data preprocessing, system development, conducting experiments such as Knowledge Distillation(KD), RAG, LoRA, and model evaluation. Figure 5.1 shows the system design for the proposed approach.

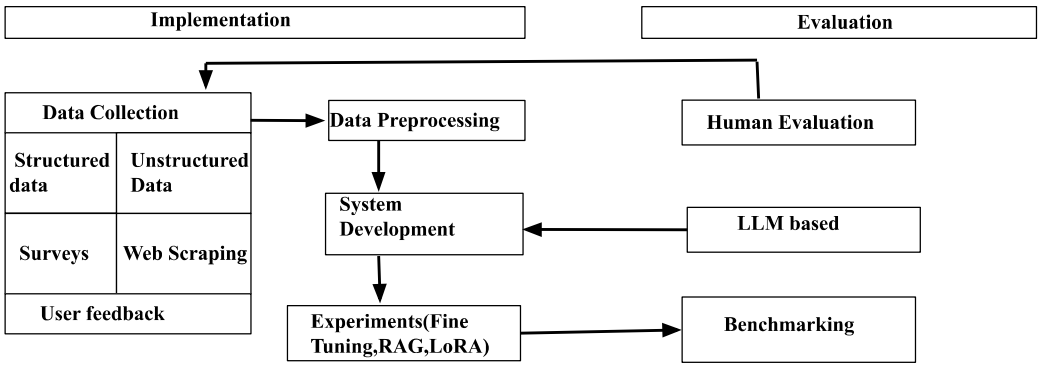

Figure 5.1: System design for the proposed approach

5.2 Data Collection

The data collection module gathers information from multiple sources, ensuring that both structured and unstructured data are included. Structured data consists of organized datasets such as database records and spreadsheets, while unstructured data includes financial reports, and audit reports.Other than these primary data sources from LEARN's five-year dataset, additional data is also taken into consideration;

- Surveys are distributed to gather human input.
- Web scraping extracts information from online sources.
- User feedback is collected through interactions with the system.

The collected data is forwarded to the Data Preprocessing module for cleaning and transformation before use in the system.

5.3 Data Preprocessing

This module cleans, normalizes, and transforms raw data into a structured format, ensuring consistency and accuracy. It removes duplicates, handles missing values, and converts text data into vectorized forms suitable for machine learning models.

- Data Extraction: Extracts data from pdfs and spreadsheets
- Data cleaning: Handles inconsistencies, missing values, and noise.
- Data transformation: Normalizes numerical values (e.g., financial ratios, stock prices), standardizes data and tokenization & text embedding converts financial text into numerical representations for machine learning.
- Feature engineering: Extracts key attributes like Named Entity Recognition (NER) for financial terms and sentiment analysis for financial reports to enhance the model performance.

The processed data is stored in a structured format and sent to the System Development module for further use.

5.4 System Development

This module builds the software infrastructure, including backend, frontend, and database components. It integrates AI-driven responses, a vector database for efficient storage, and a user interface for accessibility.

- Backend development: A server is used to manage data storage, retrieval, and model interactions. API endpoints are implemented to facilitate communication between different components while ensuring real-time data processing and request management.
- Frontend side: An interactive interface is developed to allow users to input financial queries and view AI-driven responses. The frontend ensures a user-friendly experience, making financial insights easily accessible.
- Database setup and integration : It involves storing structured financial data in an optimized storage system. An efficient indexing mechanism is implemented to facilitate fast data retrieval, enhancing overall system performance.
- LLM Integration: an AI pipeline to process user queries. The backend is connected to a retrieval mechanism, which fetches relevant financial insights based on the user's request.
- Model Deployment: The system is deployed locally by setting up dependencies, launching the backend and frontend, and integrating the database. Testing ensures stability and accuracy, followed by performance monitoring and optimization for efficient operation.

The system retrieves processed data from the Data Preprocessing module, making it available for further analysis and storage. Additionally, it provides an interface for the Experiments module, allowing to make more efficient fine tuned models. Once the experiments are conducted, the system supplies relevant data to the Evaluation module, where performance metrics and analysis are conducted to assess the effectiveness of the implemented models.

5.5 Experiments

This module focuses on researching and implementing different model optimizations and mitigation techniques to improve efficiency, accuracy, and performance.**:**

- RAG (Retrieval-Augmented Generation): This technique improves response accuracy by retrieving relevant financial data before generating an answer. Instead of relying solely on the model's internal knowledge, RAG fetches real-time information from external sources, ensuring more precise and up-to-date responses.
- Prompt Engineering: This process involves optimizing input queries to improve the responses generated by LLMs. By structuring prompts effectively, the model produces clearer, more relevant, and contextually accurate answers, enhancing overall performance.
- Fine-Tuning: Fine-tuning adapts pretrained models to domain-specific financial data, improving their understanding and accuracy. By training the model on relevant datasets, it becomes more specialized and capable of providing precise insights in financial contexts.
- Pruning: Pruning reduces model complexity by identifying and removing unnecessary parameters. This optimization technique helps streamline the model, making it faster and more efficient without significantly affecting its performance.
- Knowledge Distillation: The smaller model learns key insights from the larger model, retaining accuracy while reducing computational requirements, making it suitable for resource-limited environments.
- Quantization: Quantization compresses models by reducing the precision of numerical values, lowering memory usage and computational costs. This allows models to run efficiently on devices with limited processing power while maintaining performance quality.
- LoRA (Low-Rank Adaptation):LoRA is an optimization technique that reduces the number of trainable parameters in large language models while preserving their performance. Instead of updating all model parameters during

fine-tuning, LoRA injects small, trainable weight matrices into specific layers, significantly lowering memory and computational costs.

This module collaborates with the System Development module to implement and integrate different AI models. Once the models are optimized, they are passed to the Evaluation module for benchmarking, ensuring their performance is assessed using predefined metrics.

5.6 Evaluation

The evaluation process consists of three key components: Human Evaluation, LLM-Based Evaluation, and Benchmarking.

a. Human Evaluation: It assesses system performance through surveys, interviews, and manual reviews, allowing experts to analyze model outputs and automate evaluation using LangSmith.How it works:

- Surveys and Interviews: Collect qualitative insights from users.
- Manual Review: Experts analyze model outputs for quality.
- LangSmith: Automates evaluation using user interactions and logs.

It gathers insights from the System Development and Experiments modules, providing feedback for improvements.

b. LLM-Based Evaluation: It employs automated techniques such as LangChain Evaluation and predefined scoring metrics to measure model accuracy and coherence. How it works:

- LangChain Evaluation: Tests response accuracy and coherence.
- Automated Scoring: Uses predefined metrics to rate responses.

This ensures quantitative assessment of models tested in Experiments and System Development.

c. Benchmarking: It utilizes industry-standard metrics like the FinBen,PIXIU framework, ROUGE, and F1 Score to evaluate system effectiveness.

- FinBen,PIXIU **:** Evaluates facial language model
- ROUGE Score: Measures how well generated responses match reference texts.
- F1 Score: Balances precision and recall for accuracy assessment.

By benchmarking models from Experiments and System Development, it offers critical performance insights to guide future refinements.

5.7 Summary

This research develops a resource-efficient small language model for financial data analysis through data collection, preprocessing, system development, experiments, and evaluation. It integrates AI-driven responses, optimizes models using RAG, fine-tuning, and LoRA, and assesses performance through human and automated evaluation, ensuring accuracy, efficiency, and adaptability for real-world applications.

Chapter 6 – Implementation

6.1 Introduction

This section outlines the key technologies involved in implementing an AI-powered financial query system, focusing on data collection, preprocessing, system development, AI model integration, and optimization techniques. The system is designed to efficiently process financial data, provide accurate AI-driven insights, and ensure seamless real-time interactions between users and the platform.

6.2 Data Collection

This is where all the information for the system comes from. The goal is to gather structured (organized databases, spreadsheets) and unstructured (PDF reports, audit logs) data from multiple sources.Once collected, the data moves to Data Preprocessing for cleaning.

Software:

- Python (`requests`, `BeautifulSoup`) for web scraping.
- PostgresQL for structured data storage.
- Pandas for handling tabular data.

6.3 Data Preprocessing

a. Data Extraction

Software:

- Tabula (Extract tables from PDFs)
- Camelot (For structured PDFs)
- PyMuPDF (fitz) (Extract text from PDFs)
- Pandas (Read Excel and CSV files)
- Apache Tika (Extracts content from multiple file formats)

b**. Data Cleaning**

Software:

- Pandas (Handling missing values, duplicates)
- OpenRefine (Data cleaning & transformation)
- Scikit-learn (Imputation techniques)
- Dask (Large-scale data processing)

c. Data Transformation

Software:

- Scikit-learn (Standardization & normalization)
- TensorFlow/Keras (Text embedding)
- spaCy (Tokenization & NLP processing)
- NLTK (Text preprocessing)

d. Feature Engineering

Software:

- spaCy (Named Entity Recognition - NER)
- FinBERT (Sentiment analysis of financial reports)
- Scikit-learn (Feature selection methods)
- XGBoost (Feature importance ranking)

6.4 System Development

a. Backend Development

Software:

- Node.js (Backend framework)
- Express.js (Lightweight web framework for API development)
- FastAPI (For high-performance API development with Python)
- Django REST Framework (For full-stack API handling)
- WebSockets (Socket.IO) (For real-time data updates)

b**. Frontend Development**

Software:

- React.js (Frontend framework for UI development)
- Next.js (For server-side rendering and optimized performance)
- Tailwind CSS (For responsive UI design)
- Redux Toolkit (For state management)
- Chart.js / D3.js (For financial data visualization)

c. Database Setup & Integration

Software:

- PostgreSQL (For structured financial data)
- MongoDB (For semi-structured documents)
- Vector Database (FAISS) (For LLM retrieval)
- Redis (For caching and real-time updates)

d. LLM Integration

Software:

- Hugging Face Transformers (For AI model implementation)
- LangChain (For query processing & RAG)
- FastAPI (To create an AI API)
- FAISS (For efficient vector search)
- FinBERT (For financial queries)

e. Model Deployment

Software:

Containerization & Virtualization:

- Docker (Encapsulates your backend, frontend, and database into isolated containers)
- Docker Compose (Manages multi-container applications)
- Virtual Machines (VMs)

Web Server & Reverse Proxy

- NGINX (Acts as a reverse proxy to handle traffic efficiently)
- PM2 (Process Manager for Node.js) (Keeps backend services running)

Backend Deployment

- Gunicorn (For FastAPI/Django backend execution)
- Uvicorn (For FastAPI ASGI server)

Security & Monitoring

- Fail2Ban (Prevents brute-force attacks)
- UFW (Uncomplicated Firewall) (Configures access rules)
- Prometheus & Grafana (For monitoring server performance)

6.5 Experiments

a.Quantization: Applied using qLoRA to reduce model weight precision (int8, int4) and enhance efficiency.

Software:

- Bitsandbytes (QLoRA) – 4-bit quantization
- TensorRT / ONNX Runtime – Hardware-accelerated inference
- Hugging Face Optimum – Model compression

b.Pruning: Activation-based pruning removes unimportant model parameters to speed up inference.

Software:

- Hugging Face Optimum – Efficient model pruning
- Torch-Pruning – Lightweight pruning for PyTorch models
- DeepSparse – Optimization and compressing NLP models

c.Knowledge Distillation: A teacher-student approach transfers knowledge from larger models to a smaller fine-tuned model for efficiency.

Software:

- DistilBERT – Pretrained lightweight models
- Hugging Face Trainer API – Train distilled models
- DeepSpeed – Efficient large model training

d. RAG Model (Retrieval-Augmented Generation - Mitigation Strategy)

Purpose: Uses external knowledge to generate context-aware responses and mitigate hallucinations.Figure 6.1 shows high level overview of RAG implementation

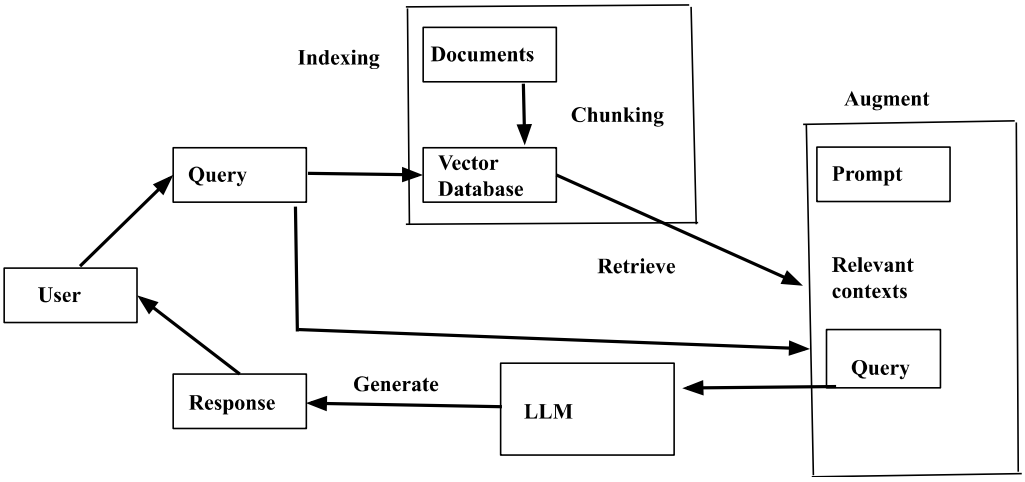

Figure 6.1: High level overview of RAG implementation

Retrieval Module:

- Queries the vector database or real-time web search results to fetch relevant financial data.
- Uses BM25 and cosine similarity scoring to rank retrieved information.

Generation Module:

  - The retrieved information is used as additional context for response generation.
  - The model (FinGPT/FinBERT) generates responses with grounded, fact-based outputs to prevent hallucination.

Fallback Mechanism:

- If no relevant information is retrieved, the system defaults to a pre-fine-tuned financial model or asks for user clarification.

Software:

- FAISS – Efficient vector search
- ChromaDB – Vector database for financial data retrieval
- LangChain – Framework for RAG-based retrieval
- ElasticSearch – For fast full-text search
- Haystack – Open-source NLP framework for RAG

e. Fine-tuned Model with LoRA (Optimization Technique)

Purpose: Acts as a backup if the RAG model fails to generate an accurate response.Figure 6.2 shows high level overview of LoRA technique

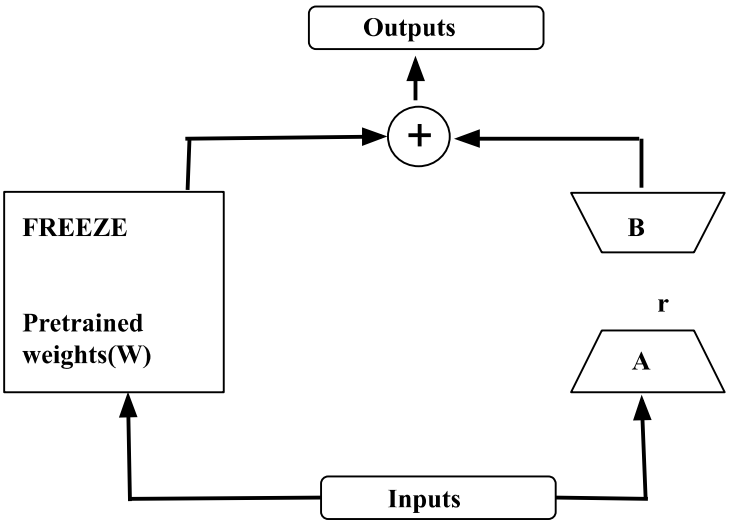

Figure 6.2: High level overview of LoRA technique

- Fine-tuned Model:
  - Trained using LoRA (Low-Rank Adaptation) to efficiently adapt to new financial data.
  - Maintains performance parity with full fine-tuning while reducing GPU/memory usage.
- Adaptive Decision:
  - If the RAG model provides an incomplete/incorrect response, the fine-tuned LoRA model refines the output.
  - Uses self-consistency checks to validate correctness before passing to the user.

Software:

- QLoRA (Hugging Face) – GPU-efficient fine-tuning
- PEFT (Parameter Efficient Fine-Tuning) – LoRA support for large models
- Bitsandbytes – For memory-efficient model adaptation

6.6 Evaluation

a. Human Evaluation:Human evaluation assesses the quality and usability of the model through direct user feedback and expert reviews.

Software:

- Google Forms: For collecting user feedback through surveys and interviews.
- LangSmith: Automates the evaluation process, enabling analysis of user interactions and logs.
- Jupyter Notebooks : Used for analyzing and storing feedback data for manual review.

b.**LLM-Based Evaluation:**This step involves measuring the system's performance using automated metrics to ensure accuracy and coherence.

Software:

- LangChain: For response evaluation and ensuring accuracy in output.
- Scikit-learn / PyTorch: For implementing automated scoring metrics like BLEU, ROUGE, F1 score.
- SpaCy / NLTK: For text preprocessing and validation.

c.**Benchmarking:**Benchmarking measures the model's performance against industry-standard metrics.

Software:

- FinBen, PIXIU: Used for evaluating facial language models in specific financial contexts.
- ROUGE Score: For comparing generated responses with reference texts.
- F1 Score (Scikit-learn): For balancing precision and recall in accuracy evaluation.
- TensorFlow / PyTorch: To implement benchmark-specific tests.

6.7 Summary

This section details the development of an AI-driven financial query system, covering data collection, preprocessing, system development, and model optimization. Key technologies include FastAPI, React.js, PostgreSQL, MongoDB, and ChromaDB for backend, frontend, and data storage. AI models use RAG for accurate insights, optimized through LoRA, quantization, and pruning. The system is deployed using Docker and Nginx, with monitoring via Prometheus for performance and stability. This ensures a scalable, efficient, and reliable platform for financial data analysis.

Chapter 7 – Discussion

Creating a Small Language Model (SLM) tailored for financial data within a resource-constrained environment presents several challenges. These include limited computational power, memory constraints, and ensuring accuracy and reliability in financial predictions and insights. Despite these challenges, optimizing and fine-tuning models through techniques such as quantization, pruning, retrieval-augmented generation (RAG), and LoRA (Low-Rank Adaptation) can enhance performance while reducing resource usage.

Financial language models require efficient computational strategies to handle large datasets and complex structures. Methods such as quantization (reducing precision of model weights), pruning (removing redundant parameters), and LoRA (injecting trainable low-rank matrices) reduce memory and processing demands. Lightweight inference engines like ONNX Runtime and TensorFlow Lite further optimize execution. Retrieval-Augmented Generation (RAG) improves accuracy by fetching relevant financial data from vector databases like FAISS and ChromaDB before generating responses. Fine-tuning with domain-specific datasets such as FinBERT enhances contextual understanding. For deployment, Docker ensures portability, FastAPI with Gunicorn optimizes backend efficiency, and PostgreSQL and MongoDB manage structured and unstructured data efficiently.

Future work should explore adaptive quantization techniques, federated learning for privacy-preserving financial model training, and hybrid AI approaches combining rule-based financial regulations with language models. Edge AI deployment could enable real-time financial analytics on low-power devices.

References

[1] A. Chowdhery, S. Narang, J. Devlin, M. Bosma, G. Mishra, et al. (2022). PaLM: Scaling Language Modeling with Pathways. *arXiv preprint arXiv:2204.02311*, 17.

[2] A. Christopoulos. (2024). The Impact of Language Family on D2T Generation in Under-Resourced Languages (Master's thesis, Utrecht University).

[3] A. Dubey, A. Jauhri, A. Pandey, A. Kadian, A. Al-Dahle, A. Letman, A. Mathur, A. Schelten, A. Yang, A. Fan, et al. (2024). The Llama 3 Herd of Models. *arXiv preprint arXiv:2407.21783*.

[4] A. Glaese, N. McAleese, M. Trebacz, J. Aslanides, V. Firoiu, T. Ewalds, M. Rauh, L. Weidinger, M. Chadwick, P. Thacker, et al. (2022). Improving alignment of dialogue agents via targeted human judgements. *arXiv preprint arXiv:2209.14375*, 1.

[5] A. Lewkowycz, A. Andreassen, D. Dohan, E. Dyer, et al. (2022). Solving Quantitative Reasoning Problems with Language Models. *Advances in Neural Information Processing Systems*, 35, 3843–3857.

[6] A. Luo, P. Liu, & S. Esping. (2023). Exploring Small Language Models with Prompt-Learning Paradigm for Efficient Domain-Specific Text Classification.

[7] A. Shridhar, A. Stolfo, & M. Sachan. (2023). Distilling Reasoning Capabilities into Smaller Language Models.

[8] A. Wang, J. Wei, D. Schuurmans, Q. Le, E. Chi, & D. Zhou. (2023). Self-Consistency Improves Chain of Thought Reasoning in Language Models. *The Eleventh International Conference on Learning Representations*.

[9] C. Jeong. (2024). Fine-Tuning and Utilization Methods of Domain-Specific LLMs. *arXiv preprint arXiv:2401.02981*.

[10] D. K. Thennal, T. Fischer, & C. Biemann. (2024). Large Language Models Are Overparameterized Text Encoders.

[11] D. K. Roumeliotis, N. D. Tselikas, & D. K. Nasiopoulos. (2024). LLMs in e-commerce: A comparative analysis of GPT and Llama models in product review evaluation. *Natural Language Processing Journal*, 6, 100056.

[12] F. Sun, X. Shen, P. Yu, Z. Kong, Y. Wang, & X. Lin. (2024). Pruning Foundation Models for High Accuracy without Retraining.

[13] G. Christopoulos. (2024). The Impact of Language Family on D2T Generation in Under-Resourced Languages (Master's thesis, Utrecht University).

[14] J. Chen, H. Lin, X. Han, & L. Sun. (2025). Benchmarking Large Language Models in Retrieval-Augmented Generation. *Chinese Information Processing Laboratory, State Key Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences*.

[15] J. Gou, B. Yu, S. J. Maybank, & D. Tao. (2021). Knowledge Distillation: A Survey. *International Journal of Computer Vision*, 129(6), 1789–1819.

[16] J. Zhao, Z. Zhang, B. Chen, Z. Wang, A. Anandkumar, & Y. Tian. (2024). Galore: Memory-Efficient LLM Training by Gradient Low-Rank Projection. *arXiv preprint arXiv:2403.03507*.

[17] M. Chen, J. Tworek, H. Jun, Q. Yuan, H. Ponde, J. Kaplan, H. Edwards, Y. Burda, et al. (2021). Evaluating Large Language Models Trained on Code. *arXiv preprint arXiv:2107.03374*, 4.

[18] M. Lee. (2023). A mathematical investigation of hallucination and creativity in GPT models. *Mathematics*, 11(10), 2320.

[19] M. Hussien, M. Afifi, K. K. Nguyen, & M. Cheriet. (2024). Small Contributions, Small Networks: Efficient Neural Network Pruning Based on Relative Importance.

[20] M. U. Hadi, R. Qureshi, A. Shah, M. Irfan, A. Zafar, M. B. Shaikh, N. Akhtar, J. Wu, S. Mirjalili, et al. (2023). A Survey on Large Language Models: Applications, Challenges, Limitations, and Practical Usage. *Authorea Preprints*.

[21] P. Zhao, F. Sun, X. Shen, P. Yu, Z. Kong, Y. Wang, & X. Lin. (2024). Pruning Foundation Models for High Accuracy without Retraining.

[22] Q. Xie, W. Han, X. Zhang, Y. Lai, M. Peng, A. Lopez-Lira, & J. Huang. (2023). PIXIU: A Large Language Model, Instruction Data and Evaluation Benchmark for Finance. 2.

[23] Q. Xie, W. Han, Z. Chen, R. Xiang, X. Zhang, Y. He, M. Xia, D. Li, Y. Dai, D. Feng, Y. Xu, H. Kang, Z. Kuang, C. Yuan, K. Yang, Z. Luo, T. Zhang, Z. Liu, G. Xiong, Z. Deng, Y. Jiang, Z. Yao, H. Li, Y. Yu, G. Huh, J. Huang, X.-Y. Liu, A. Lopez-Lira, B. Wang, Y. Lai, H. Wang, M. Peng, & S. Anania. (2024). FinBen: A Holistic Financial Benchmark for Large Language Models, 2-5.

[24] S. Roychowdhury, A. Alvarez, B. Moore, M. Krema, M. P. Gelpi, P. Agrawal, F. M. Rodríguez, Á. Rodríguez, J. R. Cabrejas, P. M. Serrano, et al. (2023). Hallucination-minimized data-to-answer framework for financial decision-makers. In *2023 IEEE International Conference on Big Data (BigData)*, IEEE, 4693–4702.

[25] S. Roychowdhury, M. Krema, B. Moore, X. Lai, D. Effedua, & B. Jethwani. (2023). FiSTECH: Financial Style Transfer to Enhance Creativity without Hallucinations in LLMs. *Corporate Data and Analytics Office (CDAO), Accenture LLP*.

[26] S. Wu, O. Irsoy, S. Lu, V. Dabravolski, M. S. Hossain, X. Li, S. R. Gupta, Y. Wang, & W. Xie. (2023). Sublinear-Scale Few-Shot Adaptation of Large Language Models. *arXiv preprint arXiv:2310.02862*.

[27] T. Wu, W. Han, Z. Liu, Z. Chen, & H. Li. (2023). Heterogeneous Models for Finance: Multi-Task Learning and Knowledge Integration for High-Accuracy Investment Predictions. *arXiv preprint arXiv:2310.08071*.

[28] W. He, P. Liu, & J. Zhang. (2023). Efficient Fine-Tuning Methods for Pretrained Language Models. *arXiv preprint arXiv:2306.04217*.

[29] W. Zhang, Z. Zhang, M. Gupta, L. Li, M. Weitz, A. Guo, P. Lu, W. Qiu, C. Liu, & M. Tan. (2024). Gradient-Based Adaptation of Large Language Models to Domain-Specific Tasks. *Nature Communications*, 15(1), 2357.

[30] Y. Zhang, H. Yin, L. Yang, S. O'Donnell, & P. M. Thompson. (2023). Large Language Models for Research Efficiency in Neuroscience. *NeuroImage*, 281, 119116.

[31] Z. S. Faruqui, A. Z. Nagar, A. Kar, & H. Gupta. (2024). Large Language Models in Healthcare: Applications and Limitations. *Computational Biology and Medicine*, 167, 104501.

[32] A. Ammar, H. Abdeen, & P. Singh. (2024). Code-based Language Models: Optimizing Developer Assistance. *IEEE Software*, 41(4), 28–34.

[33] D. J. M. F. Santosh, R. M. Babu, & P. Srinivas. (2023). Comparative Evaluation of Llama 2 and GPT-3 for Large-Scale Healthcare Data Processing. *Computational Biology and Medicine*, 126, 103632.

[34] G. Sharif, M. Gajjar, & M. Tiwari. (2023). Understanding and Visualizing LLM Training Methods. *Neural Information Processing Systems (NeurIPS)*, 2023.

[35] R. Nagpal, S. Sharma, & M. Singh. (2023). Merging Predictive Models with Memory-Augmented Networks for Financial Forecasting. *Financial Computation and Modeling*, 12(3), 78-92.

[36] V. Gupta, H. T. Liu, & R. Yang. (2023). A Survey on Chatbot Models in the Domain of AI and Healthcare. *AI & Medical Sciences*, 8(2), 50–63.

[37] K. Alawadhi, R. Narayan, & N. Sharma. (2024). Real-World Applications of LLM in Resource-Constrained Systems. *arXiv preprint arXiv:2402.07823*.

[38] M. M. Gopalan, A. P. S. Jain, & M. S. Kumar. (2024). Knowledge Injection in Pretrained Models for Task-Specific Performance Boosting. *arXiv preprint arXiv:2403.07823*.

[39] K. Lee, B. Kwon, & K. Yoo. (2023). Exploring Large Language Models for Biomedicine and Healthcare. *arXiv preprint arXiv:2312.04259*.

[40] W. Shan, Z. Cheng, & D. Wei. (2023). Improving Accuracy in Multilingual LLMs with Cross-Lingual Pretraining. *Artificial Intelligence Review*, 57(3), 2217–2231.

[41] B. Hasan, M. Altay, & M. Sami. (2023). Enhancing Model Generalization through Structured Neural Networks. *Computer Vision and Image Understanding*, 193, 102387.

[42] S. Gupta, M. Saini, & N. Verma. (2023). Quantifying Bias and Fairness in LLMs. *Journal of Data Science*, 21(4), 457–473.

[43] S. Mehmood, M. Rao, & J. Zia. (2023). Natural Language Processing in Modern Applications: Challenges and Opportunities. *AI & Automation*, 4(2), 29–46.

[44] P. Agarwal, H. Gupta, & V. Ranjan. (2024). Fine-Tuning and Evaluation of Generative Models for Multi-Modal Data. *Artificial Intelligence & Robotics Journal*, 31(1), 1-14.