AI Agents for Financial Analysis

Vamshika Lekkala[1], **Aditya Varshney**[1]

[1]Northeastern University
lekkala.v@northeastern.edu, varshney.ad@northeastern.edu

Abstract

Financial analysis is a critical task in investment research and corporate strategy, but it remains time-consuming and cognitively intensive due to the volume and complexity of financial data and reports. Analysts often spend hours extracting key metrics from filings, and interpreting financial concepts—tasks that require both domain expertise and technical proficiency in data manipulation. Large Language models have shown great capabilities in understanding and extracting key information from huge texts. We aim to leverage these capabilities to help analysts do their job quicker and with lesser errors. To address this, we present a modular AI agent-based system that leverages recent advances in large language models (LLMs) to automate and streamline the financial analysis workflow.

Code — https://github.com/adityav1810/llm-project

Introduction

Financial analysts are generally considered as the workhorses of the finance industry as they are responsible for interpreting large volumes of financial information to derive insights that inform investment decisions, risk assessments, and corporate strategies. However, the process of financial analysis is often time-consuming, repetitive, and cognitively demanding. Analysts are required to read through lengthy financial statements, extract data from structured tables, and possess a solid understanding of financial concepts. This challenge is compounded by the fact that many domain experts may not be proficient in technical tasks such as data querying or coding, which are increasingly necessary in modern analytical workflows.

In recent years, Large Language Models(LLMs) (Vaswani et al. 2023; Devlin et al. 2019) have undergone a generational transformation in scale, capability and applicability. From early transformer based models like the BERT and GPT-2 which were able to generate good coherent text and have memory to a certain extent, modern day LLMs such as GPT-4, Claude Opus and Mistral now have become more accessible to the public through platforms such as HunggingFace and have shown sparks of abilities well beyond text generation (Zhang et al. 2025). These models are able to not only have coherent memory, but also are great at doing tasks that involve reasoning, multi-step decision-making, and even making decision based on other interactions as well. The progress in LLM capabilities has been driven by scaling laws, which demonstrate that increasing model size, data volume and compute leads to predictable improvement in performance (Kaplan et al. 2020). At larger scales, LLMs exhibit emergent behaviors such as few-shot learning, chain-of-thought reasoning (Wei et al. 2023).

In more recent developments, research has suggested that a way to optimize an LLMs output is to provide external domain knowledge in order to improve the model's understanding of a specific task. This idea of Retreival Augemented Generation(RAG) (Lewis et al. 2021) has changed the process of tuning LLMs wihtout explicit fine-tuning or changing weights of the model. Finally, An AI agent is an LLM-powered component that is assigned a specific role, toolset, and behavior. Instead of relying on one monolithic model, agent-based architectures coordinate multiple specialized LLMs or modules that handle subtasks e.g., extracting KPIs, summarizing filings, or answering conceptual queries. These agents have proven to be of great use when we want an AI system to perform a task with reasoning and reference to multiple sources(Petrovic 2018; Chan et al. 2024). By stacking multiple AI Agents in a single workflow, we can have multiple models which are individually trained for a specialized task and the collective output of these agents result in a more robust result by the AI Agent.

This project aims to combine RAG with AI Agents and software development to help analysts working with large amounts of financial data to quickly extract information from documents, understand complex terminologies present in some texts or even get SQL queries to extract stock data.

Related work

The emergence of large language models (LLMs) has inspired a surge of research and development in augmenting LLMs with retrieval capabilities and structured tools to improve reliability and specialization. A significant body of work has focused on Retrieval-Augmented Generation (RAG), a method that enhances LLM responses by grounding them in retrieved context from relevant corpora. RAG methods such as REALM (Guu et al. 2020) and DPR (Karpukhin et al. 2020) have been widely used in open-

domain question answering, customer support, and document summarization tasks. RAG helps mitigate hallucination and provides traceability, which is especially valuable in sensitive domains like finance.

The concept of AI agents just like RAG has grown rapidly, with research focusing on using LLMs not just as generators, but as planners and actors. Approaches like ReAct (Yao et al. 2023) combine reasoning and tool use by prompting the model to choose actions based on intermediate thoughts. Similarly, Toolformer (Schick et al. 2023) enables LLMs to learn how to use tools from unlabeled data. Frameworks such as HuggingFace, LangChain and AutoGPT have helped make these ideas accessible to a wide variety of users and has enabled users to construct modular systems where agents can autonomously invoke APIs, search engines, or databases. These systems show promise for building real-world applications that blend structured workflows with natural language interfaces.

Prompt Engineering is another concept which has shown great improvement in results without training or changing the model weights. Prompting essentially means steering the behavior of LLMs, in order to align their responses with desired goals or formats. Since its inception, Prompt Engineering has seen constant updated on its methodologies and has evolved into a very effective way of getting robust outputs from LLMs. (Wei et al. 2022) initially showed that zero-shot prompting was effective, this was later extended by (Touvron et al. 2023) where they observed that showing the model an example input output query and then asking the model to replicate this for a new unseen query improved the generation. Finally, (Brown et al. 2020) showed that giving the model not only one but a bunch of examples further extended this idea and resulting in even better outputs. Recent attempts on improving this further have resulted in ideas such as Chain of Thought prompting (Wei et al. 2023) where the model is provided with an explanation along with the correct answer for an example query and is then expected to recreate a similar chain of thought reasoning for an unseen query.

In the financial domain, recent research has demonstrated how LLMs can support tasks like earnings call summarization, market sentiment analysis, and financial document QA. FinGPT (Yang, Liu, and Wang 2023) provides an open-source framework that integrates RAG, data curation, and fine-tuned financial LLMs. Other proprietary systems like BloombergGPT (Wu et al. 2023) leverage massive financial text corpora to improve domain-specific performance, achieving state-of-the-art results on financial NLP benchmarks. Tools like FinBERT (Araci 2019) also offer pre-trained financial sentiment models, though they are often limited in flexibility compared to more general-purpose LLMs combined with retrieval.

Despite the strength of these monolithic models, this project embraces a modular, multi-agent architecture. Rather than training or fine-tuning a single domain-specific model, we leverage prompt-driven agents and tool-based reasoning to deliver targeted answers. This enables flexibility, scalability, and better user alignment allowing agents to specialize in data querying, concept explanation, or filing analysis.

Background

This project combines Prompt Engineering, Retrieval Augmented Generation (RAG), and AI Agents to build a multi-agent system tailored for financial analysis. These methods, when integrated, allow large language models (LLMs) to operate more effectively across a complex domain like finance, where interpretability, accuracy, and context-awareness are critical. Below, we describe the theoretical and practical foundations for each technique.

Prompt Engineering

Prompt Engineering is the practice of crafting input prompts to elicit desired behaviors from language models. This can include zero-shot prompts (Wei et al. 2022), few-shot examples (Touvron et al. 2023) , chain-of-thought reasoning (Wei et al. 2023). Prompt design has become increasingly crucial in leveraging the latent capabilities of large-scale models like GPT-4, especially in multi-agent or tool-using environments.

Work such as (Wei et al. 2023) demonstrates how prompt design can substitute for task-specific fine-tuning, while LangChain and similar frameworks rely heavily on engineered prompts to drive tool use and control flow.

Retrieval Augmented Generation (RAG)

Retrieval Augmented Generation (RAG) enhances LLM capabilities by supplementing their static training data with dynamic access to a knowledge base. Instead of relying solely on internal parameters, RAG pipelines retrieve top-k relevant documents or snippets from a corpus using vector similarity or keyword search, and then pass this context to the LLM for conditioned generation.

Inspired by systems like REALM (Guu et al. 2020) and Dense Passage Retrieval (Karpukhin et al. 2020), we utilize document embeddings and retrieval APIs to fetch context before passing it to the generation module.

AI Agents

AI agents represent a paradigm shift in how LLMs are deployed: rather than a monolithic model answering all queries, tasks are decomposed across modular agents with tool access, memory, and reasoning loops. ReAct (Yao et al. 2023) and Toolformer (Schick et al. 2023) show how models can interleave reasoning ("think") with acting (e.g., querying a tool or searching a document).

Each agent follows its own set of prompts and tool usage logic. This modular architecture makes the system more interpretable, scalable, and easy to debug or extend.

Project description

In this section, we explain in detail how each agent is designed and connected to each other. We propose a multi-agent AI system that consists of multiple AI agents that perform specific domain related tasks. On a high level, this project consists of a routing agent which is responsible for routing user queries further to the correct agent. This routing agent sends the query to the relevant worker agent namely, the 'stock', 'filing' or 'concept' agents. The outputs from

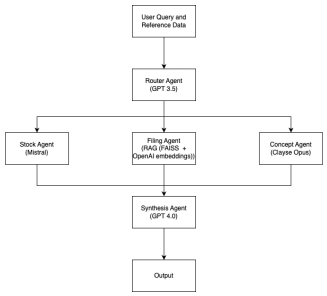

Figure 1: Workflow of multi-agent network

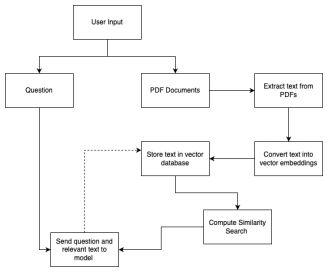

Figure 2: RAG pipeline for Filing Agent

each of these individual agents is then interpreted by a synthesis agent which takes all of the information generated by each of the agents and creates a single answer. We require the user to provide text documents as well as the question they wish to ask the system. The output of our architecture is a single cohesive answer. A more clear description of this entire network is described in figure 1

We now describe in detail how each of the agents are setup.

Router Agent

The routing agent is responsible for deciding which specific downstream agents are to be called based on the user query. OpenAI's `gpt-3.5-turbo` is used as the backbon LLM. This model has been a long standing choice in working as ane executor agent. Since the task is lightweight and does not require advanced reasoning or context retention, `gpt-3.5-turbo` is sufficient and cost-effective. Its low latency also helps in reducing system response time. We engineer the prompt to this model by adding text such as

"`You are a routing agent, Classify the user's question into one of these categories ONLY`"

Stock Agent

This agent generates SQL queries from natural language. The SQL queries are executed using SQLite on the local stock database. We use the `mistral-medium` model due to its strength in structured language generation and compact, cost-efficient architecture. Mistral models are known to perform well in code and SQL generation tasks at a lower inference cost than GPT-4. We engineer the prompt including the schema for this model by adding text such as

`You are an expert SQL assistant. Here is the schema of the table ... Write a valid SQLite query for the question below using these column names exactly. User question: {user_question} Only return the SQL query in a single line with no explanation or notes.`

Filing Agent

This agent performs question answering over long financial documents such as 10-K or 10-Q filings. We selected `claude-3-opus` due to its superior performance on long-context tasks (up to 200K tokens) and its strong summarization and reasoning abilities. Claude Opus is particularly suited for extracting insights from dense legal and financial text with minimal hallucination. Furthermore in order to implement a complete RAG pipeline- as the user attaches reports in pdf format, we also use a small script to efficiently and correctly read the pages and convert the text present in the pdf into a string format which is then sent to the LLM for processing. the extracted text is then converted into vector embeddings and stored in a vector database for later retrieval by other agents and queries. We use the FAISS library to efficiently perform similarity searches on the stored text.

Figure 2 describes this pipeline in more detail

Concept Agent

This agent explains financial concepts to users in a clear and conversational manner. `claude-3-opus` was chosen again for its strong zero-shot explanation ability and its superior ability to maintain clarity, nuance, and educational tone. We use a prompt such as `Explain the following financial concept:{ user_question}`

Synthesis Agent

The synthesis agent aggregates outputs from multiple agents to generate a holistic and context-aware final answer. We use `gpt-4` here due to its top-tier reasoning, integration, and summarization capabilities. `gpt-4` performs well in synthesizing multi-modal inputs into a consistent narrative and is trusted for high-stakes generation. It takes into account the text generated from all the individual agents, summarizes it and returns a single comprehensive response to the user.

Emperical results

We evaluated a multi-agent financial question-answering system composed of three agents: a Stock Agent (SQL-based), a Filing Agent (RAG-based), and a Concept Agent (zero-shot inference). Each agent was assessed on 50 questions aligned with its capabilities. GPT-4 was used as an evaluator to score agent responses on **Relevance (0–10)** and **Hallucination (0–10)**, where 0 is best.

Filing Agent Evaluation

Methodology: The Filing Agent was tested with 50 questions referencing quarterly 10-Q reports of Tesla, Apple, and Microsoft. The agent used Retrieval-Augmented Generation (RAG) via Claude Opus, supported by OpenAI embeddings and FAISS vector search over PDF chunks. GPT-4 evaluated each response using both the generated answer and the retrieved source evidence.

Examples of Questions:

- Did Tesla update its depreciation schedule or accounting methods?
- What are Apple's off-balance-sheet arrangements?
- How does Tesla report on ESG compliance?

Evaluation Metrics:

- **Relevance:** Whether the answer directly and completely addressed the question.
- **Hallucination:** Whether unsupported claims were made beyond the retrieved evidence.

Results:

- **Average Relevance Score:** 0.24
- **Average Hallucination Score:** 2.62

Analysis: Despite leveraging RAG, the agent's relevance was poor due to occasional routing failures and retrieval of weak or irrelevant chunks. However, hallucination was relatively low, indicating that answers—though vague—stayed grounded in the available context.

Stock Agent Evaluation

Methodology: 50 questions targeting structured stock metrics (e.g., sentiment score, PE ratio) were posed. Mistral generated SQL queries executed against a SQLite database containing synthetic stock data. GPT-4 evaluated the SQL query itself against the schema and user intent, not the result.

Examples of Questions:

- What is the average market cap of Tesla in Q1 2022?
- What was the dividend yield across all companies in 2023?

Evaluation Metrics:

- **Relevance:** Whether the SQL matched the question's logic and intent.
- **Hallucination:** Whether the SQL used non-existent fields, incorrect filters, or invented logic.

Results:

- **Average Relevance Score:** 3.26
- **Average Hallucination Score:** 1.52

Analysis: The agent handled simple queries reliably, but struggled with aggregations and temporal filters. Low hallucination scores indicate that the model respected schema constraints.

Concept Agent Evaluation

Methodology: Claude Opus answered 50 finance-related conceptual questions in a zero-shot fashion. GPT-4 evaluated the clarity, accuracy, and factual correctness of the explanation.

Examples of Questions:

- What is discounted cash flow (DCF) analysis?
- How does return on equity (ROE) differ from return on assets (ROA)?

Evaluation Metrics:

- **Relevance:** Completeness and topicality of the answer.
- **Hallucination:** Use of incorrect examples, fabricated figures, or speculative claims.

Results:

- **Average Relevance Score:** 10.0
- **Average Hallucination Score:** 6.0

Analysis: The Concept Agent consistently produced highly relevant explanations. However, hallucination was moderate due to occasional unsupported assumptions or generalizations.

Successes

- **Concept Agent:** Delivered accurate and fluent responses to all questions with perfect relevance scores. Claude Opus was highly effective for definitional or educational tasks.
- **Stock Agent:** Maintained schema adherence and delivered correct queries for straightforward retrieval tasks. Hallucination remained low across all tested queries.
- **Filing Agent:** Generated grounded answers from retrieved content when the query matched well with localized chunks.

Failures

- **Router Errors:** Several questions were routed to the wrong agent, especially when phrased ambiguously (e.g., a stock query interpreted as concept).
- **Prompting Gaps:** The SQL prompt used for Mistral occasionally failed to enforce proper constraints or temporal logic, reducing relevance.
- **Complex SQL Limitations:** The stock agent underperformed for compound queries involving 'GROUP BY', nested aggregations, or time comparisons.

Summary Table

Agent	Avg. Relevance Score	Avg. Hallucination Score
Filing Agent	0.24	2.62
Stock Agent	3.26	1.52
Concept Agent	10.00	6.00

Table 1: Evaluation metrics for each agent based on 50 questions for each agent.

Visual Analysis

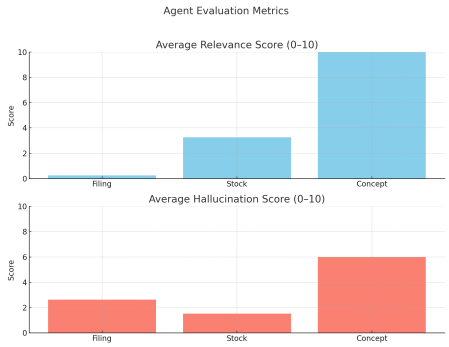

Figure 3: Average Relevance Scores across Agents

Broader Implications

This project demonstrates the potential of multi-agent Large Language Model (LLM) systems to revolutionize financial analysis by reducing the cognitive and operational load on human analysts. By combining structured database querying, unstructured document understanding, and conceptual reasoning under a unified interface, the system enables non-technical users—such as equity analysts, portfolio managers, and journalists—to access high-quality, explainable insights using natural language alone.

Democratizing Access to Financial Intelligence

The system lowers the barrier to data-driven financial analysis. Traditionally, effective querying of structured data (e.g., SQL databases) or comprehension of unstructured text (e.g., SEC filings) requires domain knowledge and technical fluency. Our architecture bridges this gap through LLM agents, allowing users to ask nuanced questions without programming skills. This has the potential to make high-level financial reasoning more accessible to small firms, educators, and even individual investors.

Scalability and Efficiency

Multi-agent systems enable modular and parallelizable processing of financial queries. As the volume of financial documents continues to increase with global digitization and regulatory transparency, such architectures offer a scalable solution for continuous monitoring, summarization, and contextualization of financial data. This paves the way for integration into decision-support platforms and real-time financial assistants.

Risks and Societal Concerns

Despite the advantages, several risks must be addressed:

- **Hallucination and Misinformation:** Even low-hallucination agents (e.g., Stock Agent) occasionally produced subtly incorrect outputs. In sensitive financial contexts, such errors could mislead decisions and erode trust.
- **Bias and Fairness:** LLMs trained on web-scale data may inherit and amplify biases in how financial performance is framed or interpreted. Care must be taken to ensure neutrality and transparency in outputs.
- **Automation and Job Displacement:** While these systems augment human decision-making, they could also reduce the need for entry-level roles in equity research or data extraction, necessitating a shift in skill development.

Ethical Use and Governance

To ensure responsible deployment, outputs from such systems should include transparency mechanisms such as source attribution (via RAG) and confidence scoring. Human-in-the-loop designs remain essential, especially in regulated industries. Further research is needed to formalize robustness guarantees, mitigate model drift, and design evaluation frameworks aligned with real-world impact.

Overall, this work showcases the potential of intelligent agents to enhance financial literacy, accelerate analysis workflows, and promote data accessibility—provided their limitations are understood and mitigated responsibly.

Conclusion

We developed a multi-agent financial QA system integrating specialized agents for structured data querying (Stock Agent), retrieval-augmented generation (Filing Agent), and conceptual explanation (Concept Agent). Three of them were evaluated on 50 targeted questions using GPT-4 for relevance and hallucination scoring.

- **Concept Agent:** Achieved perfect relevance (10.0), excelling at zero-shot explanation of finance terms. Moderate hallucination (6.0) due to unverified assumptions.
- **Stock Agent:** Performed moderately with relevance 3.26 and low hallucination 1.52. Struggles on complex aggregations or time filters.
- **Filing Agent:** Low relevance (0.24), limited by chunking and retrieval accuracy, but maintained a low hallucination score (2.62). The system shows strong potential for scalable financial analysis and education, with future improvements needed in agent coordination and complex query reasoning.

In order to make this application more efficient and robust, we would look into ways of improving the individual AI Agents by applying fine-tuning methods such as PEFT and LoRA (Xu et al. 2023; Hu et al. 2021). Furthermore, we also would like to add more agents to the workflow in order to have more informed outputs. We would also like to improve the router accuracy by refining our prompts to the routing tool, introduce a feedback loop which would allow the user to improve misclassified / wrong outputs from agents. Finally, we would also like to implement relevance filters to validate if retrieved data actually supports the user's question.

Advice to future students?

This course has been fun, rigorous and challenging all at the same time. We would like to suggest all future students to

start planning the project from day 1 so that you are in a better shape during the final leg of the semester. We would also like propagate the idea of making AI tools which actually solve a real world problem!

Acknowledgments

We would like to thank Prof. Yehoshua Roi for his insightful lectures and course content that helped us not only to understand how Large Language Models work but also gave us great hands on experience. We would also like to thank the Teaching Assistants Nina Shamsi and Tongfei Guo for their guidance throughout the semester.

References

Araci, D. 2019. FinBERT: Financial Sentiment Analysis with Pre-trained Language Models. arXiv:1908.10063.

Brown, T. B.; Mann, B.; Ryder, N.; Subbiah, M.; Kaplan, J.; Dhariwal, P.; Neelakantan, A.; Shyam, P.; Sastry, G.; Askell, A.; Agarwal, S.; Herbert-Voss, A.; Krueger, G.; Henighan, T.; Child, R.; Ramesh, A.; Ziegler, D. M.; Wu, J.; Winter, C.; Hesse, C.; Chen, M.; Sigler, E.; Litwin, M.; Gray, S.; Chess, B.; Clark, J.; Berner, C.; McCandlish, S.; Radford, A.; Sutskever, I.; and Amodei, D. 2020. Language Models are Few-Shot Learners. arXiv:2005.14165.

Chan, A.; Ezell, C.; Kaufmann, M.; Wei, K.; Hammond, L.; Bradley, H.; Bluemke, E.; Rajkumar, N.; Krueger, D.; Kolt, N.; Heim, L.; and Anderljung, M. 2024. Visibility into AI Agents. arXiv:2401.13138.

Devlin, J.; Chang, M.-W.; Lee, K.; and Toutanova, K. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. arXiv:1810.04805.

Guu, K.; Lee, K.; Tung, Z.; Pasupat, P.; and Chang, M.-W. 2020. REALM: Retrieval-Augmented Language Model Pre-Training. arXiv:2002.08909.

Hu, E. J.; Shen, Y.; Wallis, P.; Allen-Zhu, Z.; Li, Y.; Wang, S.; Wang, L.; and Chen, W. 2021. LoRA: Low-Rank Adaptation of Large Language Models. arXiv:2106.09685.

Kaplan, J.; McCandlish, S.; Henighan, T.; Brown, T. B.; Chess, B.; Child, R.; Gray, S.; Radford, A.; Wu, J.; and Amodei, D. 2020. Scaling Laws for Neural Language Models. arXiv:2001.08361.

Karpukhin, V.; Oğuz, B.; Min, S.; Lewis, P.; Wu, L.; Edunov, S.; Chen, D.; and tau Yih, W. 2020. Dense Passage Retrieval for Open-Domain Question Answering. arXiv:2004.04906.

Lewis, P.; Perez, E.; Piktus, A.; Petroni, F.; Karpukhin, V.; Goyal, N.; Küttler, H.; Lewis, M.; tau Yih, W.; Rocktäschel, T.; Riedel, S.; and Kiela, D. 2021. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. arXiv:2005.11401.

Petrovic, V. 2018. Artificial Intelligence and Virtual Worlds – Toward Human-Level AI Agents. *IEEE Access*, 6: 1–1.

Schick, T.; Dwivedi-Yu, J.; Dessì, R.; Raileanu, R.; Lomeli, M.; Zettlemoyer, L.; Cancedda, N.; and Scialom, T. 2023. Toolformer: Language Models Can Teach Themselves to Use Tools. arXiv:2302.04761.

Touvron, H.; Lavril, T.; Izacard, G.; Martinet, X.; Lachaux, M.-A.; Lacroix, T.; Rozière, B.; Goyal, N.; Hambro, E.; Azhar, F.; Rodriguez, A.; Joulin, A.; Grave, E.; and Lample, G. 2023. LLaMA: Open and Efficient Foundation Language Models. arXiv:2302.13971.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, L.; and Polosukhin, I. 2023. Attention Is All You Need. arXiv:1706.03762.

Wei, J.; Bosma, M.; Zhao, V. Y.; Guu, K.; Yu, A. W.; Lester, B.; Du, N.; Dai, A. M.; and Le, Q. V. 2022. Finetuned Language Models Are Zero-Shot Learners. arXiv:2109.01652.

Wei, J.; Wang, X.; Schuurmans, D.; Bosma, M.; Ichter, B.; Xia, F.; Chi, E.; Le, Q.; and Zhou, D. 2023. Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. arXiv:2201.11903.

Wu, S.; Irsoy, O.; Lu, S.; Dabravolski, V.; Dredze, M.; Gehrmann, S.; Kambadur, P.; Rosenberg, D.; and Mann, G. 2023. BloombergGPT: A Large Language Model for Finance. arXiv:2303.17564.

Xu, L.; Xie, H.; Qin, S.-Z. J.; Tao, X.; and Wang, F. L. 2023. Parameter-Efficient Fine-Tuning Methods for Pretrained Language Models: A Critical Review and Assessment. arXiv:2312.12148.

Yang, H.; Liu, X.-Y.; and Wang, C. D. 2023. FinGPT: Open-Source Financial Large Language Models. arXiv:2306.06031.

Yao, S.; Zhao, J.; Yu, D.; Du, N.; Shafran, I.; Narasimhan, K.; and Cao, Y. 2023. ReAct: Synergizing Reasoning and Acting in Language Models. arXiv:2210.03629.

Zhang, K.; Yang, J.; Inala, J. P.; Singh, C.; Gao, J.; Su, Y.; and Wang, C. 2025. Towards Understanding Graphical Perception in Large Multimodal Models. arXiv:2503.10857.