

**Artificial
Intelligence**

Fine Tuning Techniques for large language models (LLMs)

- Full fine-tuning
- Prompt engineering
- PEFT (LORA)
- RAG
- Agentic RAG

Realized by :

Amira khalfi

Islem maiti

GENERATIVE AI AND LARGE LANGUAGE MODELS (LLMs)

Generative AI and Large Language Models (LLMs) are among the most dynamic and engaging areas within artificial intelligence. While Generative AI broadly encompasses a variety of AI systems designed to produce new, original content across different media—such as text, images, music, and code—LLMs represent a more specialized subset of generative AI, focusing primarily on generating text-based data.

Generative AI and LLM: Definition:

- **Generative AI:** This term refers to the broad class of AI models capable of creating new content based on patterns and examples from their training data. These models leverage complex algorithms to understand context, structure, and style, enabling them to generate coherent and meaningful output in various forms, including but not limited to text, images, and audio.
- **Large Language Models (LLMs):** LLMs are a specific category of generative AI that focuses on language modeling. Trained on vast amounts of text, they capture the statistical properties of language, excelling in predicting the next word in a sequence or generating text that flows naturally from a given prompt.

The Role of Generative AI and LLM

- Generative AI models are trained on extensive datasets to learn underlying patterns, allowing them to generate content aligned with their training data's characteristics. For example, a generative AI model trained on images of cats can create entirely new images of cats.
- LLMs undergo training on large volumes of text, from sources like books, articles, and code, making them well-suited for text-specific tasks. Once trained, LLMs can generate language, translate text, create content across genres, and provide informative responses to queries.

Scope of Generative AI and LLM

The primary distinction between Generative AI and LLMs lies in their scope:

- **Generative AI:**
 - Encompasses various types of models capable of generating diverse content types beyond just text, such as images and music.
 - Allows for the creation of multimodal content across multiple media formats.
- **LLMs:**
 - Specialized in language, excelling in generating human-like text based on learned language patterns.
 - Focuses on textual data, equipping them to provide nuanced, contextually appropriate predictions and responses.

The Role of LLMs in Finance :

LLMs help financial institutions by automating important tasks, improving decision-making, and enhancing customer experiences. They are particularly useful for understanding complex financial language and extracting valuable insights from text data. This is especially beneficial in areas like compliance, risk management, customer service, and fraud detection. LLMs enable financial institutions to operate more efficiently and with greater accuracy in a fast-moving market.

Large language model lifecycle:

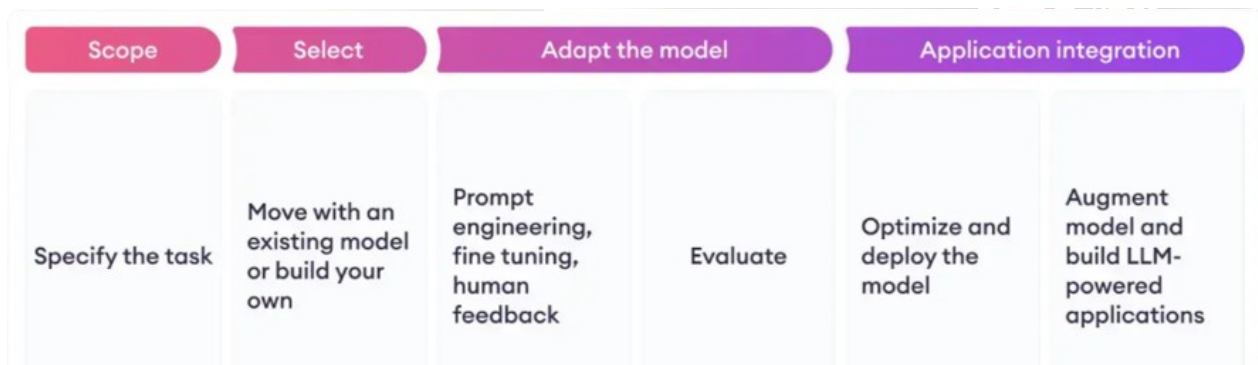


Figure 1: Large language model lifecycle

Key Applications of LLMs in Finance:

- **Personalized Financial Advice:**
LLMs analyze clients' financial data and goals to create personalized investment strategies and financial plans, offering tailored advice to improve client satisfaction.
- **Real-Time Fraud Detection:**
LLMs monitor transaction patterns to spot fraudulent activities in real-time, enhancing the security of payment systems.

- **Automated Regulatory Compliance Reporting:**

LLMs help generate compliance reports automatically and quickly adjust to new regulations, making it easier for financial institutions to meet regulatory requirements.

- **AI Chatbots for Payment Support:**

LLM-powered chatbots provide instant help for payment-related questions, improving customer service and satisfaction.

- **Predictive Market Risk Management:**

LLMs analyze market trends and financial data to predict market changes, helping financial institutions manage risks more effectively.

- **Automated Question Answering:**

LLMs assist in answering complex financial queries, whether from clients or internal teams, by extracting relevant information from large datasets and documents in real-time.

- **Resume Screening and Candidate Evaluation:**

LLMs help financial institutions streamline their hiring processes by analyzing resumes, matching candidates to job requirements, and providing insights on their suitability for roles based on qualifications and experience.

FINE TUNING

Full fine-tuning :

Full fine-tuning, also known as instruction fine-tuning, involves updating all of the model's weights. This process creates a new version of the model with adjusted weights. It's important to note that, like pre-training, full fine-tuning requires substantial memory and computational resources to store and process the gradients, optimizers, and other components that are updated during training.

Advantages:

- **Model Customization:** Full fine-tuning allows the model to be specifically adapted to a particular task or dataset, improving performance on that task.
- **Improved Accuracy:** By updating all the weights, the model can better learn task-specific nuances, often resulting in higher accuracy.

Disadvantages:

- **High Resource Requirements:** Full fine-tuning is computationally expensive, requiring significant memory and processing power to update all the model weights.
- **Time-Consuming:** The training process can be slow, especially with large models and extensive datasets.

Prompt Engineering Techniques

In prompt engineering, different techniques are used to enhance the model's performance based on the structure and content of the prompt.

In-context Learning :

In-context learning improves the prompt by providing specific task examples within the prompt itself, giving the LLM a blueprint of what needs to be accomplished. This allows the model to better understand the task and generate more accurate outputs based on the given context.

– Zero-shot Inference :

Zero-shot inference involves providing input data directly in the prompt without any additional examples. The model is expected to infer the task based on its pre-existing knowledge. While efficient, zero-shot inference may not always yield the desired results.

– One-shot and Few-shot Inference :

If zero-shot inference does not produce the desired outcome, one-shot or few-shot inference can be employed. These methods involve adding one or multiple completed examples within the prompt to guide the model's understanding of the task. These approaches help smaller LLMs perform better by providing more context.

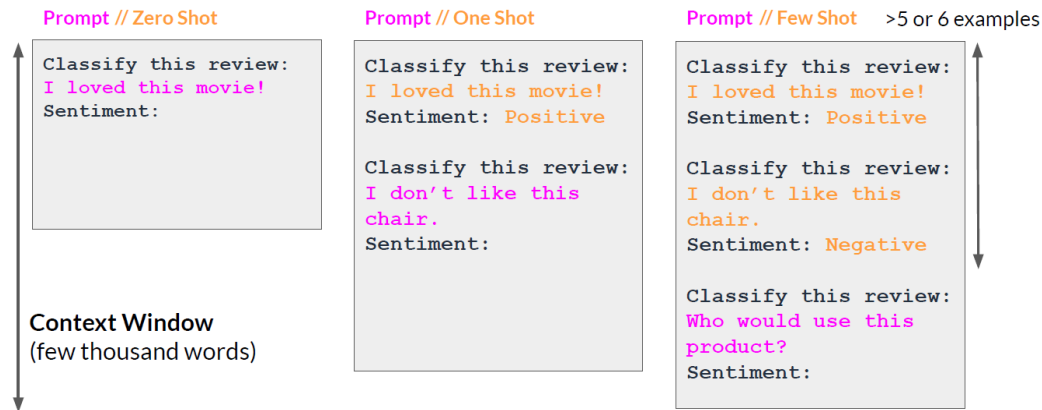


Figure 2: In-context Learning Example

Chain-of-Thought (CoT) Prompting :

Intuitively, we as humans break a larger task or problem into sub-tasks, chaining these sub-tasks together by using the output of one sub-task as the input for the next. Similarly, chain-of-thought (CoT) prompting enables complex reasoning capabilities through intermediate reasoning steps. By combining CoT prompting with few-shot prompting, you can achieve better results on more complex tasks that require reasoning before responding.

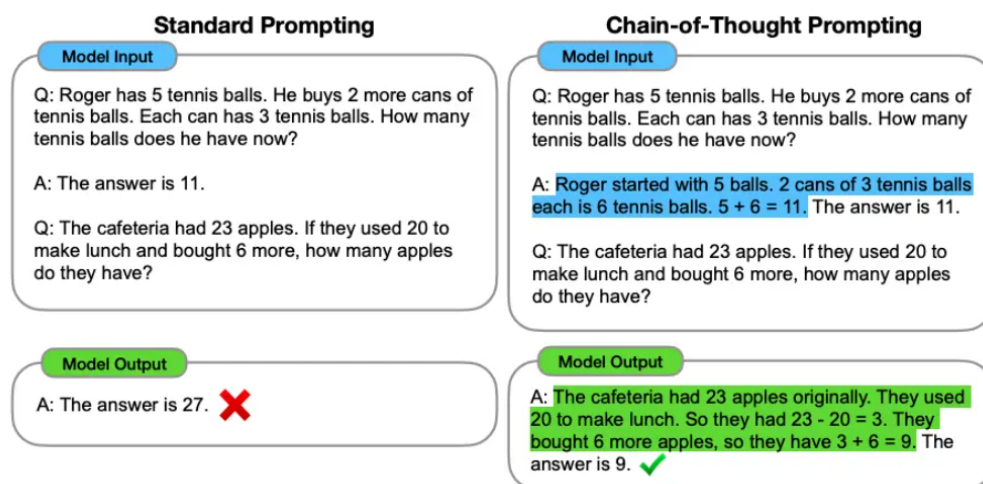


Figure 3: Chain of thoughts Example

Sequential Prompting :

Sequential prompting is a method used in natural language processing (NLP) where a series of prompts are presented to a language model in a specific order to elicit a desired response. This technique involves crafting a sequence of questions, tasks, or instructions that build upon

Prompt:

The odd numbers in this group add up to an even number: 4, 8, 9, 15, 12, 2, 1.
 A: Adding all the odd numbers (9, 15, 1) gives 25. The answer is False.
 The odd numbers in this group add up to an even number: 15, 32, 5, 13, 82, 7, 1.
 A:

Output:

Adding all the odd numbers (15, 5, 13, 7, 1) gives 41. The answer is False.

Figure 4: Chain of thoughts Example of a prompt and it's output

each other to achieve a particular goal or understanding. Sequential prompting is particularly useful in applications where:

- **Complex Tasks:** Breaking down complex tasks into smaller, manageable steps can improve the accuracy and efficiency of the model's responses.
- **Contextual Understanding:** Applications that require a deep understanding of context, such as chatbots or virtual assistants, can benefit from sequential prompting.
- **Knowledge Graph Construction:** Sequential prompting can be used to construct knowledge graphs by presenting a series of prompts that elicit specific information about entities and relationships.

Actually , Sequential Prompting focuses on the ranking stage of recommender systems, since LLMs are more expensive to run on a large-scale candidate set. The ranking performance is sensitive to the retrieved top-ranked candidate items, which is more suitable to exam.

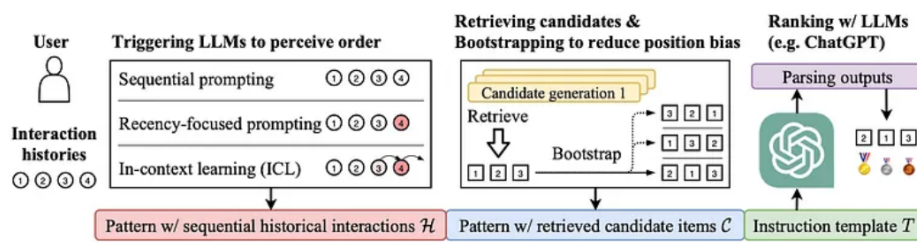


Figure 5: Sequential Prompting Example

ReAct Prompting (Reasoning + Action)

ReAct prompting is a dynamic approach where the AI alternates between reasoning through a problem and taking actions based on that reasoning. This technique is especially valuable for tasks that require iterative decision-making and interactions with external systems or data.

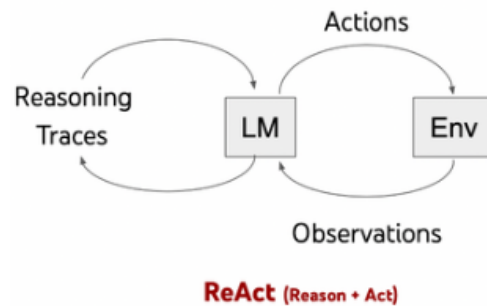


Figure 6: *ReACT*

- The process of ReAct prompting typically revolves around three interconnected components: Thought, Act, and Observation. These components enable the AI to reason, act, and learn iteratively, ensuring more accurate and context-aware outputs. The iterative cycle of Thought → Act → Observation equips the AI with a structured process to solve problems effectively:

- **Thought** fosters reasoning and planning.
- **Act** translates reasoning into actionable steps.
- **Observation** provides feedback for continuous improvement.

Question: Aside from the Apple Remote, what other device can control the program Apple Remote was originally designed to interact with?

ReAct

Thought 1: I need to search Apple Remote and find the program it was originally designed to interact with.

Act 1: `Search[Apple Remote]`

Obs 1: The Apple Remote is a remote control introduced in October 2005 by Apple ... originally designed to control the `Front Row media center` program ...

Thought 2: Apple Remote was originally designed to control the Front Row media center program. I need to search `Front Row` next and find what other device can control it.

Act 2: `Search[Front Row]`

Obs 2: Could not find [Front Row]. Similar: ['Front Row Seat to Earth', 'Front Row Motorsports', '`Front Row (software)`', ...]

Thought 3: Front Row is not found. I need to search `Front Row (software)` .

Act 3: `Search[Front Row (software)]`

Obs 3: Front Row is a discontinued media center software ...

Thought 4: `Front Row (software)` is controlled by an Apple Remote or the keyboard function keys. So the answer is keyboard function keys.

Act 4: `Finish[keyboard function keys]` ✓

Figure 7: *ReAct Example*

Parameter Efficient Fine Tuning (PEFT):

Parameter-Efficient Fine-Tuning (PEFT) is an advanced technique aimed at optimizing the fine-tuning process of machine learning models by reducing resource requirements and associated costs. This approach is particularly effective for domain-specific tasks that demand model adaptation. PEFT enables a balanced approach, retaining the valuable knowledge of pre-trained models while efficiently adapting them to target tasks with a minimal increase in parameters. Several methodologies exist for implementing PEFT, including the Low-Rank Adaptation (LoRA) technique, which has proven highly effective in achieving these goals.

Low-Rank Adaptation (LoRA):

LoRA (Low-Rank Adaptation) is a technique used for efficiently adapting large pre-trained models by reducing the number of parameters that need to be adjusted during fine-tuning. Instead of retraining all the model's parameters, LoRA introduces low-rank matrices (A and B) that decompose the weight matrix (W) into a sum of $W + A \times B$. This approach only modifies the matrices A and B , which are much smaller than the original weight matrix W , significantly reducing the resources required for model adaptation.

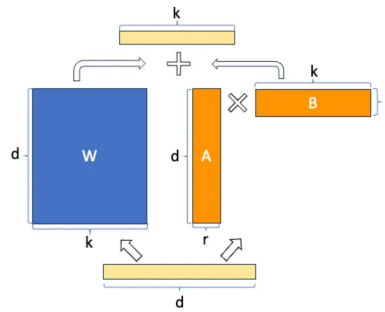


Figure 8: LoRa

Main Benefits of LoRA:

- **Efficiency:** Significantly reduces memory and computational costs by adjusting only a small subset of parameters.
- **Scalability:** Allows large models, such as those used in Natural Language Processing (e.g., GPT-3 or BERT), to be adapted without retraining the entire model.
- **Flexibility:** Can be applied to various tasks while preserving the original pre-trained weights, making it suitable for environments with limited resources.

Typical Application:

LoRA is particularly useful in Natural Language Processing (NLP) tasks, such as text classification or domain adaptation (e.g., medical, legal), where a pre-trained model needs to be fine-tuned for a specific context without high computational costs.

In summary, LoRA enables efficient adaptation of pre-trained models to specific tasks with a low computational cost while preserving the knowledge acquired during initial training.

RAG :

Retrieval Augmented Generation (RAG) is an innovative approach that combines the best of two worlds in AI: information retrieval (which does not generate original responses) and content generation (which relies only on the data learned during training). Traditionally, large language models (LLMs) generate content based solely on the information learned during their training phase. RAG, on the other hand, allows the model to “consult” an external database or corpus of documents in real-time to enrich its text generation. This retrieval capability significantly enhances the accuracy, relevance, and richness of the generated content.

How does RAG work?

The RAG process can be divided into two main steps:

1. **Retrieval:** When the model receives a query, it performs a search within a predefined set of documents or data to find the most relevant information related to the query. This search is often facilitated by sophisticated indexing and information retrieval techniques.
2. **Generation:** Once the relevant information is retrieved, the model uses it, in addition to its internal knowledge, to generate a response or content that not only answers the original query but does so in a more informed and accurate manner.

Use Case Example: RAG in Financial Decision Support

Problem: A financial analyst needs to generate a report summarizing the impact of recent market events (such as interest rate changes or stock market trends) on a specific company’s financial performance. The analyst requires up-to-date information, historical data, and detailed analysis of how these events affect the company’s stock price, profitability, and market position.

Traditional Approach: The analyst would typically rely on historical financial data and reports to generate the analysis. However, this method might not be fully accurate or up-to-date, especially for real-time market events and their implications.

RAG-Based Approach:

- **Retrieval:** The financial model receives a query asking for a report on the recent impact of interest rate hikes on Company X’s stock price. The model searches through up-to-date financial news sources, market reports, and the company’s latest financial statements in real-time to retrieve relevant information.
- **Generation:** Once the most pertinent documents are retrieved (such as news articles on recent interest rate hikes, market analysis, and the company’s quarterly reports), the model combines this external information with its internal knowledge (e.g., past trends, economic models) to generate a comprehensive, accurate, and timely financial report. The report would include insights on how the interest rate hike is likely to affect Company X’s revenue, stock performance, and overall market outlook.

Benefits in Finance:

- **Real-Time Data:** RAG allows the model to pull in the most up-to-date financial news, regulatory changes, or economic reports, making the analysis more relevant and accurate.

- **Comprehensive Insights:** By combining external knowledge with the model's internal expertise, the output can include nuanced insights and data-backed predictions that wouldn't be possible with a model relying solely on its training data.
- **Improved Decision Making:** Financial analysts and decision-makers can make more informed, data-driven decisions based on RAG-generated reports that incorporate both historical knowledge and real-time data.

In this images , we will show most important steps to achieve this :

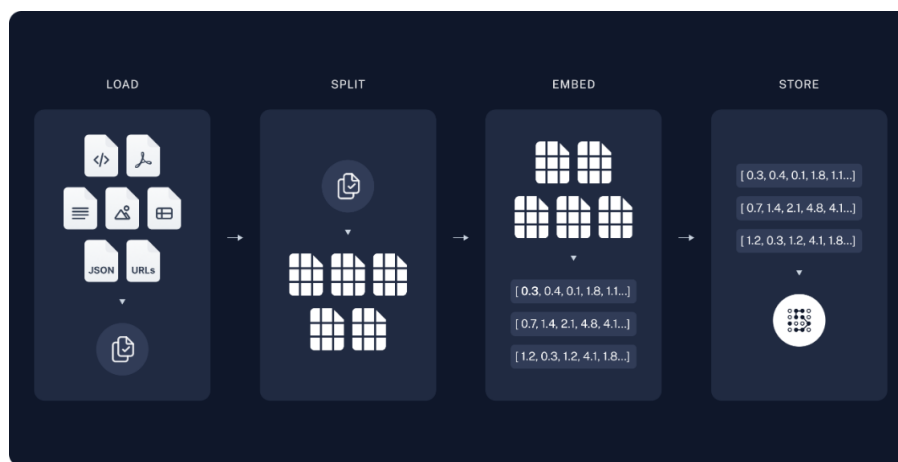


Figure 9: Indexing

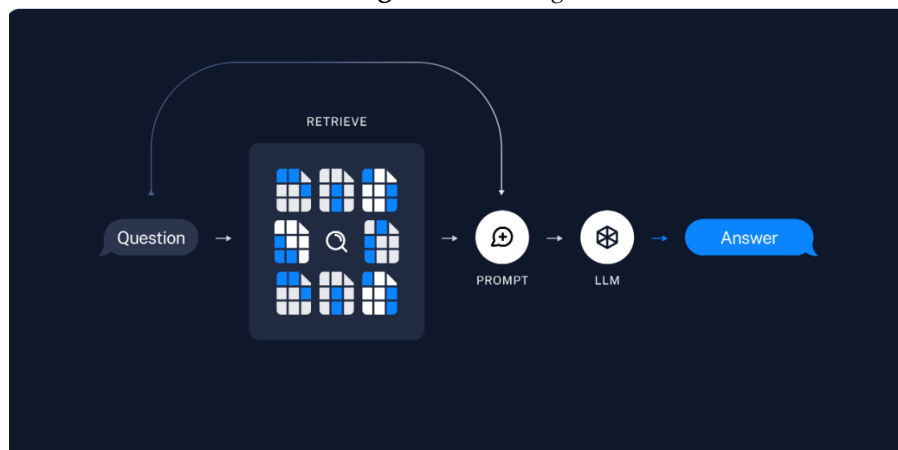


Figure 10: Retrieval and generation

Goodbye, RAG—hello, Agentic RAG ! Time to level up and let the agents handle the heavy lifting.

Agentic RAG: An Advanced Approach to Question Answering

While standard RAG excels at simple queries across a few documents, agentic RAG takes it a step further and emerges as a potent solution for question answering. It introduces a layer of intelligence by employing AI agents. These agents act as autonomous decision-makers, analyzing initial findings and strategically selecting the most effective tools for further data retrieval. This multi-step reasoning capability empowers agentic RAG to tackle intricate research tasks, like summarizing, comparing information across multiple documents, and even formulating follow-up questions – all in an orchestrated and efficient manner. These newfound agents transform the LLM from a passive responder to an active investigator, capable of delving deep into complex information and delivering comprehensive, well-reasoned answers. Agentic RAG holds immense potential for such applications, empowering users to understand complex topics comprehensively, gain profound insights, and make informed decisions.

Agentic RAG Implementation

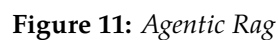
Agentic RAG transforms how we approach question answering by introducing an innovative agent-based framework. Unlike traditional methods that rely solely on large language models (LLMs), agentic RAG employs intelligent agents to tackle complex questions requiring intricate planning, multi-step reasoning, and utilization of external tools. These agents act as skilled researchers, adeptly navigating multiple documents, comparing information, generating summaries, and delivering comprehensive and accurate answers. Agentic RAG creates an implementation that easily scales. New documents can be added, and each new set is managed by a sub-agent.

Think of it as having a team of expert researchers at your disposal, each with unique skills and capabilities, working collaboratively to address your information needs. Whether you need to compare perspectives across different documents, delve into the intricacies of a specific document, or synthesize information from various summaries, agentic RAG agents are equipped to handle the task with precision and efficiency.

Key Features and Benefits of Agentic RAG

- **Orchestrated question answering:** Agentic RAG orchestrates the question-answering process by breaking it down into manageable steps, assigning appropriate agents to each task, and ensuring seamless coordination for optimal results.
- **Goal-driven:** These agents can understand and pursue specific goals, allowing for more complex and meaningful interactions.
- **Planning and reasoning:** The agents within the framework are capable of sophisticated planning and multi-step reasoning. They can determine the best strategies for information retrieval, analysis, and synthesis to answer complex questions effectively.
- **Tool use and adaptability:** Agentic RAG agents can leverage external tools and resources, such as search engines, databases, and specialized APIs, to enhance their information-gathering and processing capabilities.
- **Context-aware:** Agentic RAG systems consider the current situation, past interactions, and user preferences to make informed decisions and take appropriate actions.

- In essence, agentic RAG presents a powerful and adaptable approach to question-answering. It harnesses the collective intelligence of agents to tackle intricate information challenges. Its ability to plan, reason, utilize tools, and learn makes it a game-changer in the quest for comprehensive and reliable knowledge acquisition.

Figure 11: *Agentic Rag*

CONCLUSION

In this article, we explored the pivotal role of large language models (LLMs) in the financial sector, examining their applications and potential for enhancing decision-making. We then delved into the fine-tuning of LLMs, covering various techniques such as full fine-tuning, advanced prompt engineering strategies tailored to specific domains, and parameter-efficient fine-tuning (PEFT) with a focus on LoRA. Lastly, we highlighted the emergence of Retrieval-Augmented Generation (RAG) and Agentic RAG, cutting-edge techniques that are revolutionizing the way LLMs handle complex tasks in real-time, providing enhanced accuracy, adaptability, and decision support in financial applications. These advancements offer significant potential for further improving the application of LLMs in finance.

