Master Thesis

IU University of Applied Sciences

Study program: Master of Science Artificial Intelligence

In-Depth Comparative Analysis of Function-Calling in LLMs and Advancing Precision in Function-Calling for LLMs through Focused Fine-Tuning

Date of submission: 07.07.2024

I. Abstract

This thesis investigates the function-calling capabilities of Large Language Models and the impact of fine-tuning, with a particular focus on the QLoRA (Quantized Low-Rank Adaptation) technique. The primary objectives are to conduct a comprehensive review of existing research on LLM function-calling, improve function-calling accuracy through fine-tuning, and evaluate the empirical results of these improvements. An exhaustive literature review reveals that fine-tuning significantly improves LLMs' accuracy and reliability in generating API-compliant arguments, with models like GPT-4, Mistral-7B, and Llama-2-70B showing notable performance gains. However, it also highlights trade-offs, such as increased computational costs. The experimental results confirm these findings, showing substantial improvements in accuracy and precision for fine-tuned models like Llama 2 and Mistral Instruct, particularly through the application of QLoRA. The thesis concludes that while fine-tuning, is essential for optimizing LLMs for specific tasks, careful consideration of resource allocation is crucial. Future research should explore more efficient fine-tuning methods to increase performance while managing computational demands.

Keywords: Large Language Models, Function-Calling, Fine-Tuning, QLoRA, API Compliance, Computational Efficiency

II. Table of Contents

III. Table of Figures

IV. Table of Tables

V. Table of Abbreviations

AI	Artificial Intelligence
API	Application Programming Interface
BART	Bidirectional and Auto-Regressive Transformers
BERT	Bidirectional Encoder Representations from
BFCL	Berkeley Function-Calling Leaderboard
GPU	Graphics Processing Units
GPT	Generative Pre-trained Transformer
LLM	Large Language Model
LoRA	Low-Rank Adaptation
LSTM	Long Short-Term Memory networks
NLP	Natural Language Processing
PEFT	Parameter-Efficient Fine-Tuning.
QLoRA	Quantized Low Rank Adaptation
RLHF	Reinforcement Learning with Human Feedback
RNN	Recurrent Neural Networks
SFT	Supervised Fine-Tuning
T5	Text-to-Text Transfer
TCE	Text Coherence Evaluation

1. Introduction

1.1. Problem Definition

Function-calling in Large Language Models (LLMs) represents a significant advancement in natural language processing (NLP) and artificial intelligence (AI). Despite these advancements, LLMs still face challenges in accurately integrating and utilizing external functions (Endpoints Team, 2023).

A primary challenge is the semantic interpretation of user requests for function-calls. Even advanced LLMs can struggle to precisely understand user requests, especially when language is ambiguous or lacks specificity (Ahmed et al., 2024). This probabilistic nature of natural language understanding can lead to misinterpretation and incorrect function execution (Capelle, 2024a; Capelle, 2024b).

Context management is another critical issue. LLMs must maintain an understanding of past interactions to process function-calls effectively, which is essential for tasks that require multiple steps and a coherent understanding of task evolution (Dettmers et al., 2023).

Interoperability with diverse systems and APIs also poses a significant challenge. Ensuring that an LLM can communicate effectively across various external services, each with unique API structures and protocols, is complex. This complexity is compounded by the need to address security and privacy concerns (Doerrfeld, 2024; McGregor et al., 2017).

The hallucination of knowledge, where LLMs generate plausible but incorrect content, remains a significant concern. This issue is critical when LLMs call functions to retrieve information, as they might fabricate responses rather than execute the function accurately (fireworks.ai, 2023).

Error handling and diagnostics are also challenging. When a function-call fails or returns unexpected results, LLMs must diagnose the issue and provide informative feedback, a task that current models often struggle with (Greyling, 2023a; Greyling, 2023b).

Fine-tuning LLMs for specific function-calling tasks is complex and resource-intensive, requiring large amounts of annotated data and substantial computational resources (Hakhamaneshi & Ahmad, 2023). Ethical and responsible use of function-calling capabilities is another concern. The ability to execute functions that interact with real-world systems necessitates governance to prevent misuse and ensure ethical behavior (Ouyang et al., 2022).

Addressing these challenges requires concentrated research and development efforts to improve LLM capabilities, refine their understanding of context and semantics, improve interoperability with external systems, and ensure ethical use (Ouyang et al., 2022).

Fine-tuning LLMs is crucial for adapting general models to specific tasks or domains. This process is not merely a marginal improvement but central to the practical deployment and utility of LLMs across various industries (Capelle, 2024a; Hakhamaneshi & Ahmad, 2023). Fine-tuning adjusts a

model's weights to better reflect the context and specifics of a target task, leading to improved performance crucial for real-world applications (Hakhamaneshi & Ahmad, 2023).

Fine-tuning allows models to understand and generate domain-specific language. For example, an LLM fine-tuned on medical literature can better use medical terminology, improving its value to healthcare professionals (Dettmers et al., 2023). It also improves efficiency by reducing the need to train models from scratch, conserving computational resources, and being more cost-effective and environmentally sustainable (Ahmed et al., 2024).

Function-call precision, critical for technical applications, can be significantly improved through fine-tuning, allowing models to understand programming languages better and interact with APIs (Endpoints Team, 2023). Additionally, fine-tuning personalizes models to user preferences, delivering a more tailored user experience (fireworks.ai, 2023).

Ethically, fine-tuning can mitigate biases in base models by using more balanced and inclusive datasets, steering outputs away from harmful stereotypes and inaccuracies (Ouyang et al., 2022). It also facilitates the integration of LLMs with tool-augmented capabilities, ensuring accurate and secure function-calls, and improving the applicability of models in complex environments (Greyling, 2023a; Greyling, 2023b).

Fine-tuned LLMs have wide-ranging practical applications in fields like healthcare, finance, customer service, and education. By adapting LLMs to specific domains, AI becomes a powerful tool, assisting professionals with tasks such as diagnostics, legal research, customer interactions, and coding (Hakhamaneshi & Ahmad, 2023).

1.2. Objectives and Research Questions

The first objective of this thesis is to perform an exhaustive review of the existing academic and industry research related to the function-calling capabilities of LLMs. This involves an analysis of scholarly articles, technical reports, and case studies that examine the intricacies of how LLMs generate API-compliant arguments. The literature review aims to synthesize key findings, critically evaluate diverse methodological approaches, and identify significant research gaps. The intent is to collate a comprehensive understanding of the current capabilities, limitations, and areas of improvement within the field of function-calling LLMs.

The second objective focuses on the refinement of function-calling accuracy in LLMs. This empirical research with quantitative methods will be achieved by fine-tuning a compact model with the specific goal of improving its precision and compliance with API standards. The fine-tuning process will be guided by the outcomes of the literature analysis, applying identified best practices, and addressing known limitations. The aim is to advance the function-calling precision of LLMs to a level where they can be reliably used for generating API calls that align with real-world standards and expectations.

Together, these objectives drive the research towards contributing meaningful insights into the capabilities of LLMs within the domain of function-calling and offering pragmatic solutions to improve their precision. By focusing on these dual aims, the thesis aspires to not only advance academic understanding but also to provide tangible improvements in the application of LLMs for automated API interactions.

The key research question is

> "How does fine-tuning impact the performance of LLMs specifically for function-calls?"

The pursuit of deepening our understanding of these capabilities and improving their precision through fine-tuning is guided by a set of research questions that target the core aspects of this current LLM topic:

1. What are the principal outcomes and investigative approaches in the current scholarly landscape concerning LLMs' capabilities in API-compliant argument generation?
2. What are the prevalent research gaps and challenges identified in the literature regarding function-calling by LLMs?
3. How has the evolution of LLMs influenced their ability to generate API-compliant function-calls in recent years?
4. How can fine-tuning a compact LLM model improve its accuracy and adherence to API standards in function-calling tasks?
5. What fine-tuning strategies are most effective in improving the precision of LLMs for API-compliant function-calls?
6. What is the impact of model fine-tuning on the performance of LLMs in various function-calling scenarios, and how does it compare to baseline models?

These research questions aim to find nuanced insights into the complex mechanics of LLMs and establish a benchmark for evaluating the impact of fine-tuning techniques on their functional competencies.

The interplay between fine-tuning and the robustness of LLMs is another area that requires further exploration. As LLMs are fine-tuned for specific tasks, they may become more susceptible to adversarial attacks or generate biased outputs. There's a need for research that specifically investigates how fine-tuning impacts the security and fairness of these models (Endpoints Team, 2023).

These contributions are intended to significantly advance the understanding and capabilities of LLMs in the context of API-compliant function-calling, thereby driving progress in the field of artificial intelligence.

While this thesis provides a comprehensive analysis of fine-tuning techniques and their effects on LLM performance, it does not examine several other specific areas. The thesis does not aim to create

new algorithms or methods for fine-tuning. Instead, it focuses on evaluating existing techniques. The study is limited to specific tasks such as function-calling and factual accuracy, and broader applications of LLMs, such as in creative writing or open-domain dialogue, are outside the scope of this research. Although ethical considerations are discussed, developing a complete ethical framework for the deployment of LLMs is beyond the scope of this thesis. Furthermore, the practical integration of fine-tuned LLMs into real-time systems or commercial applications is not covered, as the research remains focused on theoretical and empirical evaluation. Although the thesis compares several LLMs, it does not include an exhaustive comparison of all available models in the market. The focus is on a select few that are representative of the state-of-the-art. By defining these boundaries, the thesis ensures a focused investigation into the specific aspects of fine-tuning LLMs.

1.3. Structure of the Thesis

The thesis is structured to provide a logical and comprehensive exploration of the function-calling capabilities of LLMs, focusing on their current state, challenges, and opportunities for improvement through fine-tuning. Here is an overview of the chapters that compose this thesis:

Chapter 1 sets the stage for the research by outlining the context, relevance, and scope of the study. It introduces the reader to the concept of function-calling in LLMs and the importance of fine-tuning these models for improved interaction with APIs. It also lays out the thesis objectives and the research questions that will guide the inquiry.

An extensive review of the scholarly literature forms the core of chapter 2. It examines existing research on LLMs, with a particular emphasis on function-calling and fine-tuning techniques. The chapter aims to synthesize key findings, identify methodological trends, and highlight the gaps that the current study seeks to address.

Following the literature review is Chapter 3. It is dedicated to the fine-tuning experiment. This chapter elaborates on the preparation of the dataset, the used function, the fine-tuning techniques employed, including the used Python code, and how the results will be evaluated.

Next, we have the methodology chapter. Chapter 4 details the systematic approach taken to conduct this research, including the criteria for source identification and the analysis of findings. It explains the choice of the LLM for fine-tuning and the development of a tailored API task designed to test and improve the model's function-calling proficiency.

Chapter 5 presents the results of the literature analysis and the fine-tuning process. It provides an assessment of the LLMs' initial capabilities in function-calling and a detailed account of the fine-tuning's impact on accuracy, precision, and correctness of API-compliant argument generation.

The concluding chapter 6 synthesizes the insights gained throughout the research, reflecting on the thesis objectives and research questions. It discusses the theoretical and practical implications of the findings and suggests areas for future research. The thesis finishes with recommendations based

on the research findings. It also discusses the limitations encountered during the study, providing a critical lens on the research conducted and its implications for future work.

The fine-tuning process constitutes the heart of the methodological flow. The chosen LLMs are fine-tuned using advanced techniques that align with the findings from the literature review.

The chapters are designed to cumulatively build a narrative that not only charts the progress made in LLM function-calling capabilities but also critically examines the role of fine-tuning in advancing these capabilities to meet real-world application standards.

The sequence of chapters in this thesis is crafted to facilitate a coherent and logical progression of research, from conceptualization to conclusion, ensuring a thorough understanding and exploration of the fine-tuning and function-calling capabilities of LLMs.

This intentional ordering of chapters is designed to guide the reader through a methodical sequence of understanding, critically examining, and applying the concepts and techniques related to the fine-tuning and function-calling of LLMs, resulting in a holistic and impactful study.

2. Literature Review

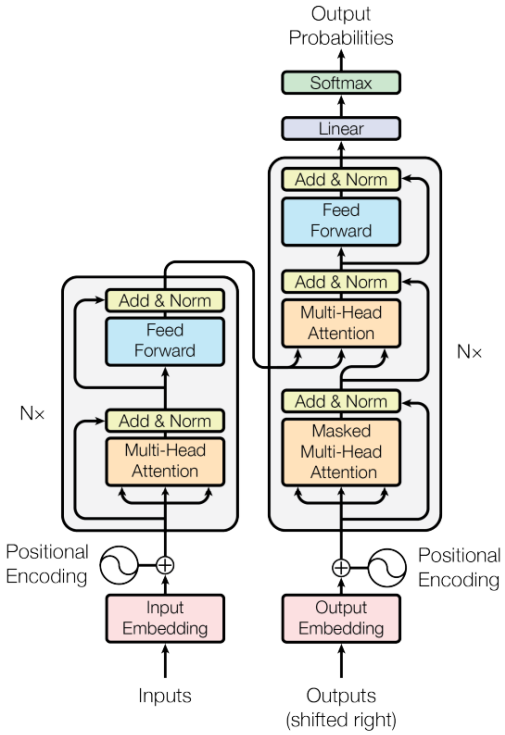
2.1. Fundamentals of LLMs

2.1.1. Historical Development and Current State

In the examination of the evolution of LLMs, the path from foundational concepts to modern state-of-the-art implementations is marked by significant advancements. Key contributions to this progression are attributed to a range of academic papers and research projects that explored various architectural paradigms and methodologies. The literature review serves to critically analyze these developments, highlighting both achievements and areas for further research.

The Transformer architecture, introduced by Vaswani et al. (2017), revolutionized the landscape of sequential data processing in NLP. This architecture as depicted in Figure 1, characterized by its self-attention mechanisms, laid the groundwork for models that could understand and generate human-like text. It allowed for parallel processing of sequences, laying a robust foundation for subsequent LLMs. The Transformer model's encoder and decoder structures, with their unique combination of multi-head attention and feed-forward networks, form the backbone of many contemporary LLMs (Vaswani et al., 2017). However, the complexity of the Transformer architecture also raises questions about computational efficiency and scalability. As LLMs increase in size, the implications for resource utilization and hardware requirements become significant, meaning more and better hardware is needed, as well as energy and training data (Vaswani et al., 2017; Wu et al., 2024).

Figure 1 Transformer Architecture

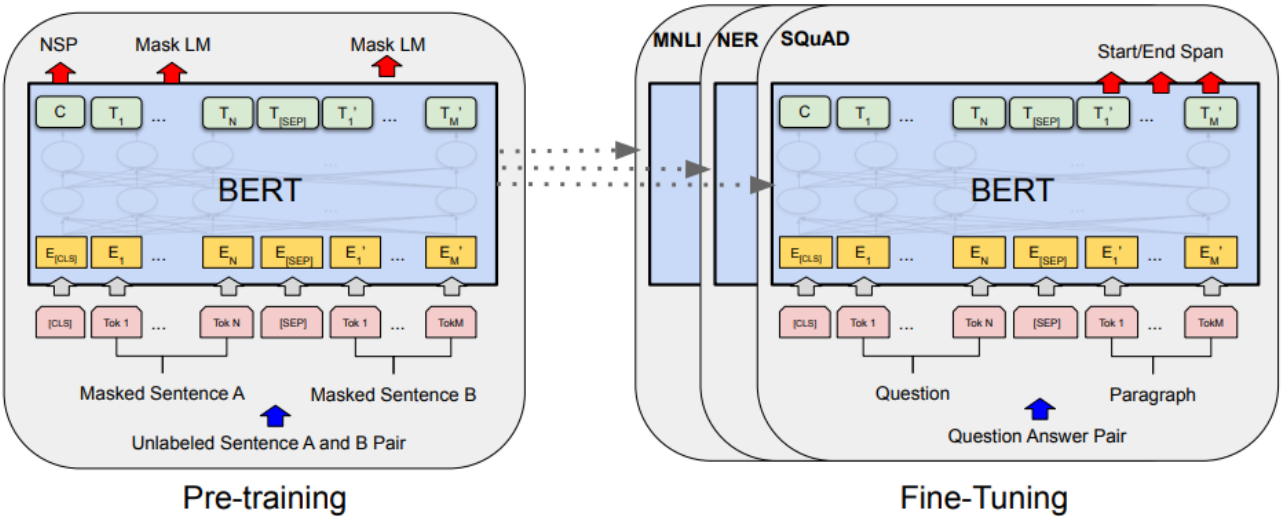


Source: Vaswani et al., 2017, p. 3

Building on the Transformer architecture, OpenAI's GPT series (Generative Pre-trained Transformer), particularly GPT-3 and GPT-4, demonstrated considerable improvements in generating coherent and contextually relevant text (Brown et al., 2020). GPT-3's massive scale allowed it to handle a wide variety of tasks without extensive fine-tuning, indicating a move towards more general-purpose AI systems. However, GPT-3's vast parameter count raises concerns about energy consumption, model interpretability, and the environmental impact of training such large models (Raman, 2023). These considerations suggest the need for more efficient training methods and model designs that reduce computational costs without compromising performance.

Similarly, models like BERT (Bidirectional Encoder Representations from Transformers) by Devlin et al. (2019) and its derivatives, focused on understanding context and nuance in text, demonstrating superior capabilities in tasks like question answering and sentiment analysis, as depicted in Figure 2. The critical review of BERT and its variations uncovers a potential over-reliance on large datasets for pre-training, which might not always be sustainable or available in specialized domains (Devlin et al., 2019). This reliance on pre-trained models such as BERT raises questions specifically about gender biases inherent in the training data, potentially affecting the models' outputs and posing ethical concerns, as has been shown by the work of Jentzsch & Turan (2023).

Figure 2 BERT Architecture



Source: Devlin et al., 2019, p. 3

Further diversification in LLMs is noted with models like T5 (Text-to-Text Transfer) and BART (Bidirectional and Auto-Regressive Transformers), each bringing unique contributions to the generation and interpretation of text (Raffel et al., 2020; Lewis et al., 2019). These models underscore the versatility of LLMs but also illustrate the ongoing challenges of balancing model complexity with training efficiency.

The contemporary state of LLMs is represented by models like GPT-4 and Meta’s open-source LLM Llama 2, which continue to push the boundaries of natural language understanding and generation

(Touvron et al., 2023). These models have advanced in terms of parameter count and efficiency, yet their increasing size brings about challenges related to scalability and ethical considerations. The capability for "few-shot" or "zero-shot" learning exhibited by these models introduces an element of adaptability that lowers the barrier to AI applications across various domains, but it also raises concerns about generalization and reliability when applied to specific tasks (Brown et al., 2020).

A noteworthy development is the integration of LLMs with external data sources through API calls, enabling these models to function as intelligent agents capable of performing specific tasks (Wu et al., 2024). While this extends the utility of LLMs beyond simple text generation, it also necessitates a rigorous examination of the security and robustness of these interactions. The ability to execute specific tasks, such as booking appointments or summarizing research articles, relies heavily on the accuracy and security of function-calling mechanisms.

2.1.2. Key Architectures and Models

The progression from rudimentary rule-based models to advanced neural network designs laid the foundation for modern LLMs. A critical examination of key architectures and models reveals a transformative tour characterized by pivotal shifts in NLP paradigms.

The transition to Recurrent Neural Networks (RNNs) and their improved variant, Long Short-Term Memory networks (LSTMs), revolutionized the processing of sequential data (Hochreiter & Schmidhuber, 1997). Although RNNs and LSTMs addressed sequence processing with some success, they faced challenges like vanishing gradients, which limited their effectiveness in capturing long-range dependencies.

The introduction of the Transformer architecture by Vaswani et al. (2017) marked a concept shift in NLP. The self-attention mechanism allowed Transformers to process sequences in parallel, significantly improving efficiency and effectiveness. This architectural leap opened the door to new levels of scalability and parallelism, enabling models to manage complex NLP tasks without the constraints of sequential processing inherent in RNNs and LSTMs.

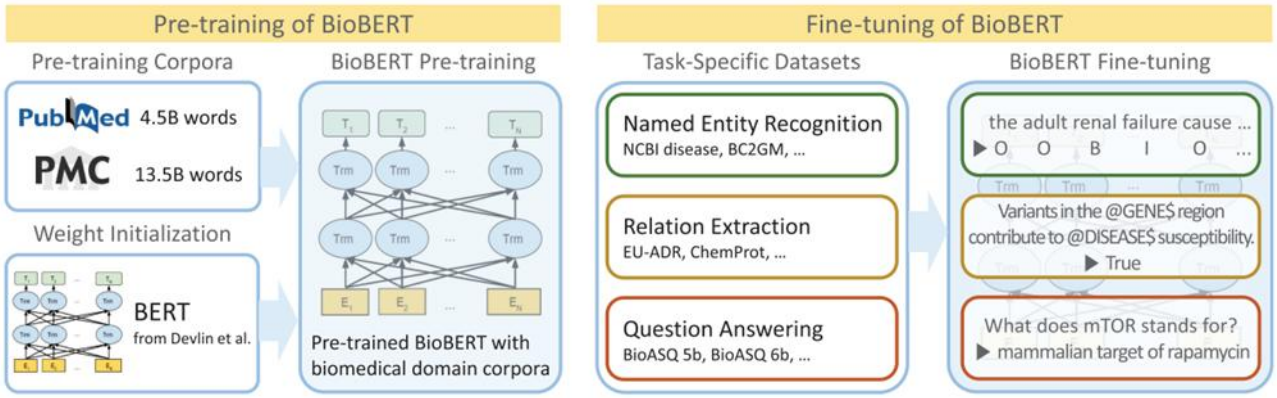
OpenAI's GPT series demonstrated the impact of Transformers when pre-trained on vast datasets (Radford et al., 2018). GPT's success in generalizing knowledge to a wide array of tasks established the concept of transfer learning in NLP (Ruder et al., 2019). This approach of pre-training followed by fine-tuning has become a standard practice, yet it raises concerns about data bias and ethical implications associated with training on large, often uncurated datasets (Jentzsch & Turan, 2023).

BERT introduced by Devlin et al. (2019) advanced the concept of bidirectional training, allowing the model to gather context from both directions. This bidirectional capability led to improvements across various NLP tasks but also highlighted the need for extensive computational resources to manage the increased complexity.

Other architectures like T5 by Raffel et al. (2020) offered a different approach by converting every NLP task into a text-to-text format, streamlining the process and facilitating a broader range of applications. This simplification brought versatility but also necessitated a comprehensive training dataset, emphasizing the importance of data diversity to prevent overfitting to specific patterns.

The expansion of domain-specific models further diversified the LLM landscape. Models like BioBERT (Figure 3) and SciBERT applied pre-training and fine-tuning to specialized datasets, yielding improved performance in specific domains, such as biomedicine or computer science (Lee et al., 2020; Beltagy et al., 2019). These models demonstrated the adaptability of Transformer-based architectures yet underscored the need for domain-specific expertise to effectively fine-tune models for niche applications.

Figure 3 Overview of the pre-training and fine-tuning of BioBERT



Source: Lee et al., 2020, p. 1235

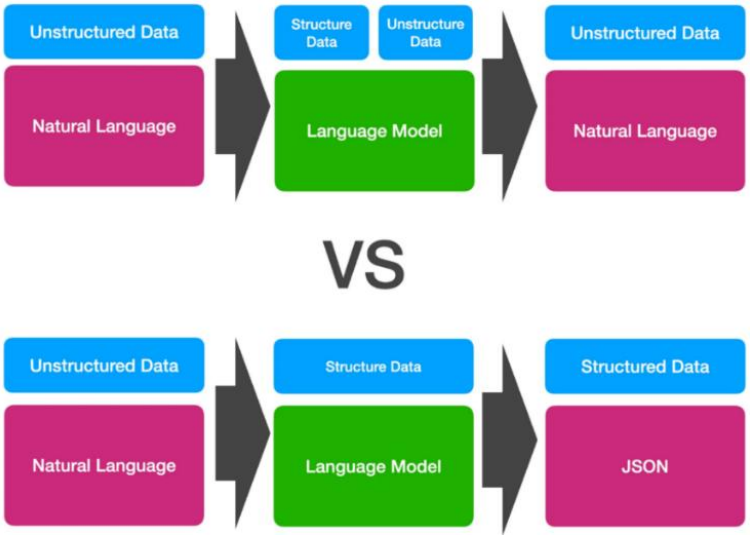
Newer models like GPT-4 and Llama 2 continue to build upon these foundational architectures, exploring tasks involving reasoning, common sense application, and multi-modal integration (Brown et al., 2020; Touvron et al., 2023). These advancements raise questions about scalability, efficiency, and the environmental impact of training large models. The push for multi-modal integration also necessitates the exploration of new architectures that can seamlessly handle text, images, and other data types coherently.

2.2. Function-calling in LLMs

2.2.1. Definition and Significance

Function-calling in LLMs represents an advancement in their application, leveraging their inherent proficiency in handling unstructured conversational data (like chat replies and prose). These models excel at reformatting this data into structured outputs like JSON, typically required for effective interaction with external systems and applications, see Figure 4. (Greyling, 2023).

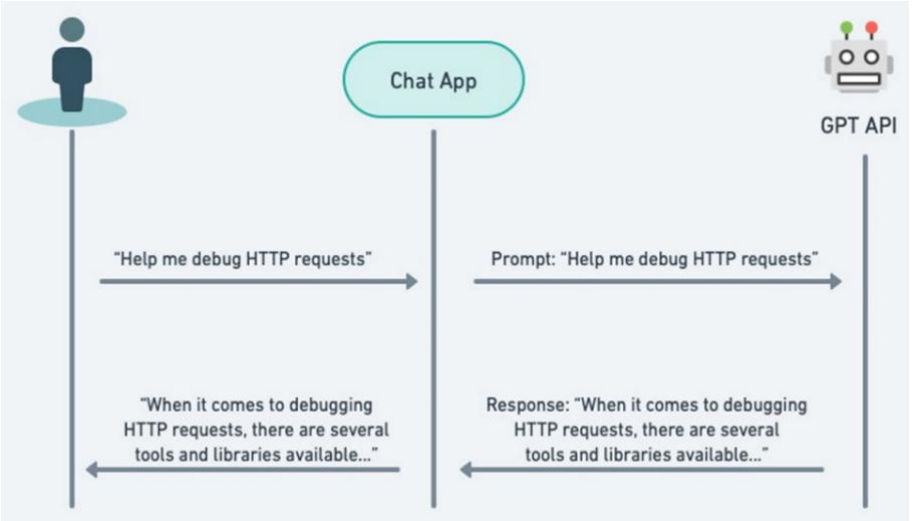
Figure 4 Generating unstructured versus structured data using LLMs



Source: Greyling, 2023b

Fernandez & Ackerson (2023) state that LLMs such as GPT-4-0613 and GPT-3.5 Turbo-0613, developed by OpenAI, represented significant strides in AI by integrating function-calling capabilities directly into their architectures. This innovation marked a departure from traditional models, enabling these systems to generate structured function-calls in response to user inputs, rather than mere text outputs (Fernandez & Ackerson, 2023). This capability enriched the interaction between users and LLMs, providing a more dynamic conversational experience, as depicted in Figure 5.

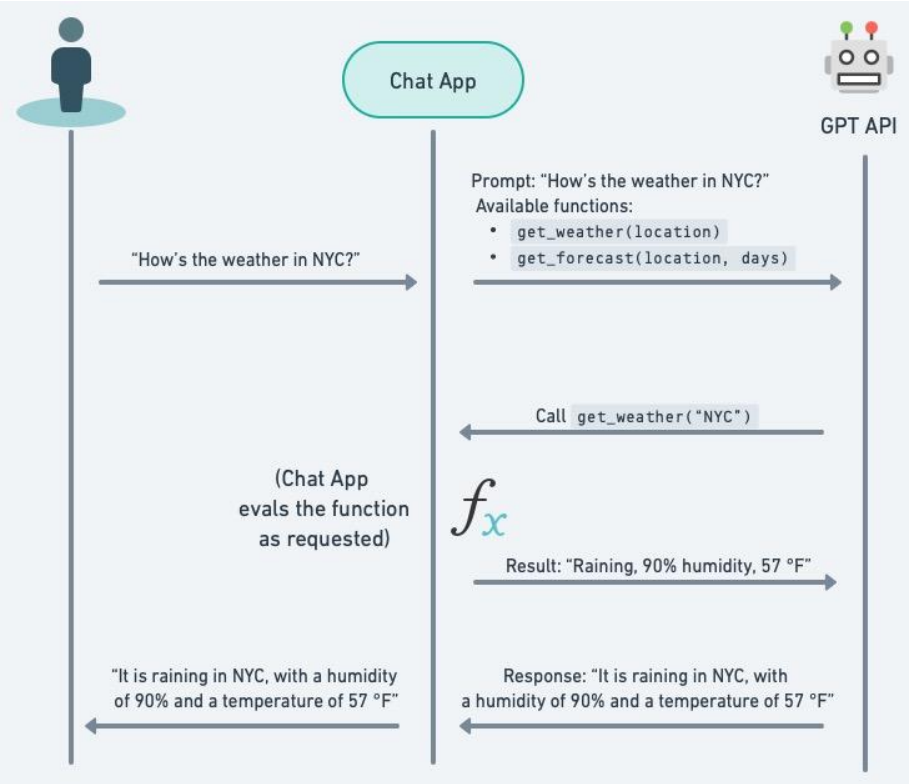
Figure 5 Conversation Flow between User and LLM without Function-Calling



Source: Fernandez & Ackerson, 2023

The inclusion of function-calling allows LLMs to serve as more than just passive conversational agents; they become active participants capable of accessing and manipulating external data. For example, these models can now perform tasks that require real-time data fetching or actions beyond their training datasets, as illustrated in Figure 6 (Fernandez & Ackerson, 2023).

Figure 6 Conversation Flow between User and LLM with Function-Calling



Source: Fernandez & Ackerson, 2023

Function-calling is not merely an incremental improvement, it represents a significant change in how LLMs are perceived and utilized. Traditionally, LLMs operated within the confines of their training data, limited to generating responses based on previously ingested information. With function-calling, LLMs go beyond these limitations, interacting with external APIs to execute actions or retrieve information, thereby acting as semi-autonomous agents (fireworks.ai, 2023).

This capability has profound implications for various industries. In e-commerce, for example, LLMs equipped with function-calling can automatically check inventory levels and update customers in real-time, improving customer service and operational efficiency (Kim et al., 2023). In the travel industry, these models can access up-to-date travel records to offer personalized advice, significantly improving user experience (Roberts, 2023a).

The theoretical underpinning of function-calling in LLMs is grounded in the expansion of their operational scope, from closed systems to open systems that can dynamically interact with external environments. This advancement underscores a shift towards more integrated and versatile AI systems, paving the way for further innovations in automation and decision support systems (Fernandez & Ackerson, 2023; fireworks.ai, 2023).

Practically, function-calling empowers LLMs to perform complex tasks that were previously unattainable, such as coding software or managing e-commerce transactions. These capabilities

demonstrate the models' utility in practical applications, marking them as integral components of technological ecosystems rather than standalone tools (OpenAI, 2023; Greyling, 2023).

While the benefits of function-calling are clear, the integration of this capability introduces new challenges, particularly in terms of security, privacy, and the ethical use of AI. Ensuring that LLMs handle external data securely and respect user privacy becomes paramount as these models become more embedded in critical systems (Doerrfeld, 2024; Touvron et al., 2023).

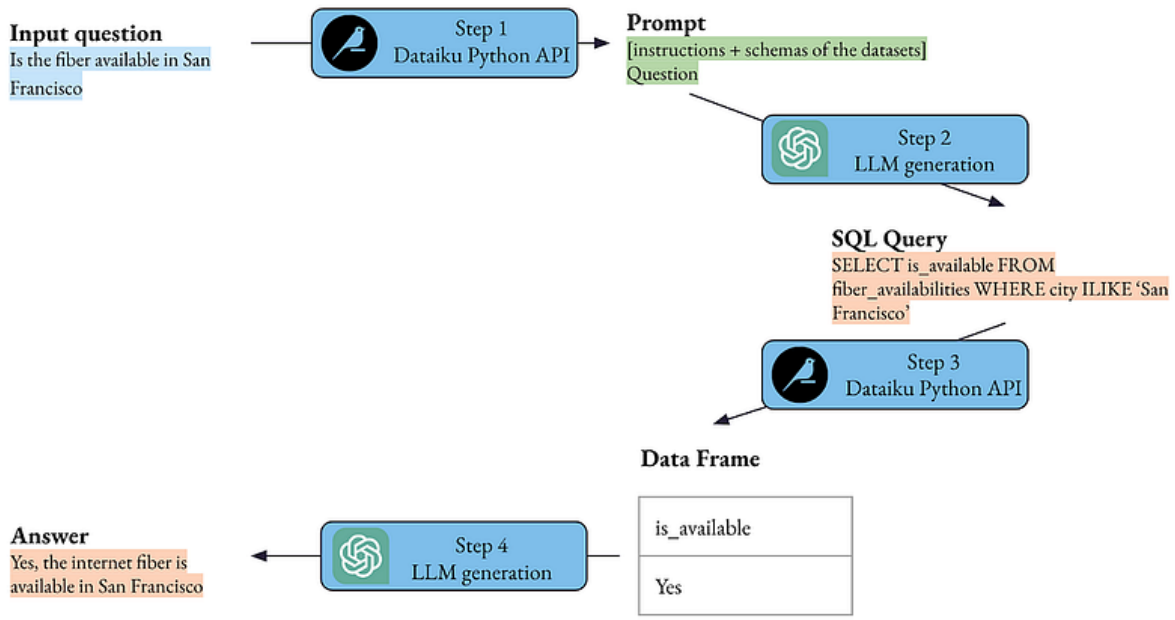
Furthermore, the need for fine-tuning and the use of embeddings highlights the ongoing requirement for substantial resources to train these models effectively. Introducing the direct ability of LLMs to perform direct function-calling, offers a third pathway to improve LLM capabilities, complementing traditional methods by enabling real-time data processing and response generation within ongoing interactions (Fernandez & Ackerson, 2023).

2.2.2. Review of Current Function-Calling Capabilities

Originally, LLMs like GPT-3 were designed primarily for generating textual content based on vast training datasets. The integration of function-calling capabilities, however, has enabled these models to execute external tasks through API requests, significantly broadening their applicability (Brown et al., 2020). For instance, LLMs can now automate interactions with software, databases, and internet services, transforming how they interact with the digital world (Utemuratov, 2024).

Figure 7 provides a practical illustration of this capability. It depicts an LLM formulating a customer inquiry into an SQL query, which is then sent to a database via an API. The retrieved information is used to generate responsive communication with the customer, showcasing the LLM's ability to bridge conversational input with structured database interaction (Utemuratov, 2024).

Figure 7 Step-by-step Telecom LLM API usage example

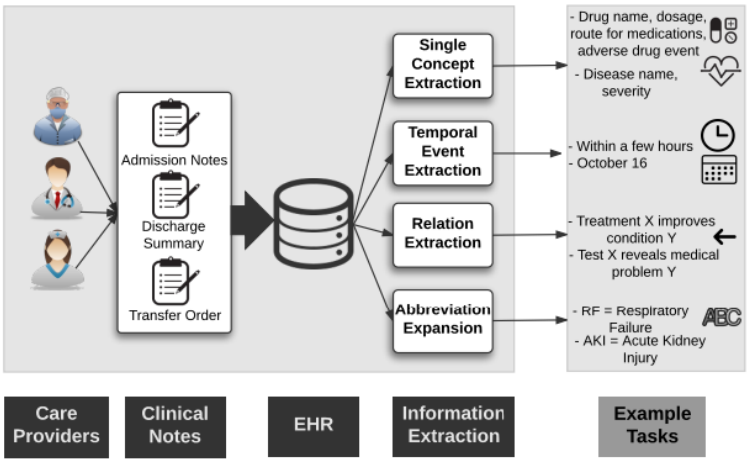


Source: Utemuratov, 2024

One of the most notable applications of function-calling in LLMs is in the domain of software development. The Codex model, a derivative of GPT-3, exemplifies this by generating syntactically correct code snippets in response to programming prompts, thereby aiding in software development (Chen et al., 2021). Similarly, in the business sector, companies like Salesforce have integrated LLMs to automate customer service operations, increasing efficiency and personalization (Dey, 2023).

In the financial industry, LLMs have proven capable of parsing complex documents, extracting crucial data points, and interfacing with analytical tools to provide insights into market trends (Choi et al., 2023). Healthcare is another field benefiting from this technology, where LLMs assist in diagnosing by accessing and analyzing patient data (Shickel et al., 2018), as illustrated in Figure 8.

Figure 8 LLM-assisted diagnostic



Source: Shickel et al., 2018, p. 6

The academic sector has also leveraged the function-calling capabilities of LLMs. Researchers utilize these models to automate literature reviews, data collection, and hypothesis testing, significantly reducing the manual effort involved in these tasks (Williams, 2023).

Furthermore, the practical scenarios highlighted, such as using an LLM to determine the hottest city from a given list, underscore the model's ability to not only fetch and analyze data but also perform comparative analyses. This capability is particularly advantageous in scenarios where quick, informed decisions are based on real-time data (Utemuratov, 2024).

The Berkeley Function-Calling Leaderboard provides a competitive arena for benchmarking the performance of function-calling in LLMs. This platform is instrumental in fostering a landscape of continuous improvement and innovation, driving models to achieve higher accuracy and efficiency in function-calling (Yan et al., 2024).

Despite these advancements, the field of function-calling in LLMs is still evolving. Challenges related to the accuracy, security, and ethical use of AI in accessing and manipulating external data remain.

The rapid pace of development in this area suggests that current capabilities are just the beginning, with potential for significant growth and refinement in the coming years.

2.2.3. Challenges and Limitations in Current Implementations

One of the main challenges is maintaining stability across different versions of LLMs. The rapid pace of advancements in LLM technologies necessitates frequent updates to incorporate new improvements. However, these updates can lead to inconsistencies and unpredictability in function-calling behaviors. Developing robust versioning and update management strategies is essential to ensure that applications remain stable and perform reliably over time (Weis, 2024).

The computational demands of state-of-the-art LLMs, such as GPT-4, pose significant barriers. These models require extensive computational resources for training and inference, making them less accessible to smaller organizations or independent researchers. There is a pressing need to optimize these models for greater efficiency and lower resource consumption to broaden access and encourage innovation across different sectors (Greyling, 2023a; Greyling, 2023b).

The effectiveness of function-calling significantly depends on the user's ability to craft precise prompts. Prompt engineering requires a deep understanding of the model's capabilities and limitations, which represents a steep learning curve for many users. Developing comprehensive training programs and resources to improve proficiency in prompt design is critical for maximizing the utility of LLMs in practical applications (fireworks.ai, 2023).

The operational and computational costs associated with deploying LLMs for function-calling tasks are substantial. Yan et al. (2024) highlight these costs which can vary widely among different LLMs, as illustrated in Figure 9, which details the cost per 1,000 function-calls across various models. This economic barrier restricts the accessibility of advanced LLM capabilities to entities with significant funding, potentially limiting diversity and innovation within the AI ecosystem (Semaphore CI, 2024).

Figure 9 Berkeley Function-Calling Leaderboard

					Abstract Syntax Tree (AST) Evaluation				Evaluation by Executing APIs						Latency (s)	
Rank	Overall Acc	Model	Organization	License	Simple Function	Multiple Functions	Parallel Functions	Parallel Multiple	Simple Function	Multiple Functions	Parallel Functions	Parallel Multiple	Relevance Detection	Cost ($ Per 1k Function Calls)	Mean	Standard Deviation
1	79.35	Claude-3-Opus-20240229 (Prompt)	Anthropic	Proprietary	83.09	90.5	80	62.5	83.53	74	70	45	80.83	10.81	5.67	1.43
2	78.71	GPT-4-0125-Preview (Prompt)	OpenAI	Proprietary	83.64	90	86.5	64.5	82.94	76	68	42.5	69.17	5.22	2.1	1.4
3	78.35	Gorilla-OpenFunctions-v2 (FC)	Gorilla LLM	Apache 2.0	83.27	93	85.5	66	87.06	76	68	47.5	60.83	1.53	2.38	2.05
4	78.18	GPT-4-0125-Preview (FC)	OpenAI	Proprietary	78.91	91	88.5	66.5	70	68	70	47.5	81.67	4.76	4.37	5.31
5	77.71	GPT-4-1106-Preview (FC)	OpenAI	Proprietary	77.45	88	88	62	79.41	74	70	45	80.83	4.95	6.45	6.45
6	73.82	Claude-3-Sonnet-20240229 (Prompt)	Anthropic	Proprietary	79.45	85.5	85	65.5	75.29	74	70	45	53.33	2.13	2.38	0.66
7	72.24	Mistral-Medium-2312 (Prompt)	Mistral AI	Proprietary	74.73	73.5	79.5	53.5	67.65	68	64	27.5	88.33	1.75	3.41	4.65
8	70.71	Functionary-Small (FC)	MeetKai	MIT	72.36	86.5	76.5	55.5	60	72	66	45	74.17	1.94	3.02	3.2
9	69.18	Claude-instant-1.2 (Prompt)	Anthropic	Proprietary	76.18	84	78.5	46	76.47	66	58	45	54.17	0.95	1.75	1

Source: Yan, 2024

Maintaining the accuracy and reliability of function-calls is a critical challenge. LLMs often rely on pattern recognition to generate outputs, which can result in errors in command execution, misunderstandings of user intentions, or inappropriate responses. To address these issues, there is a need for improved training methodologies that can improve LLMs' understanding of the context and implications of their actions. Additionally, robust error-checking and feedback mechanisms are required to monitor and correct function-calling behaviors (Roberts, 2023b; Weis, 2023).

Recent initiatives, such as the studies conducted by Berkeley's AI Research Lab and their function-calling leaderboard, highlight the importance of benchmarking LLM capabilities to identify performance limitations and develop standards for function-calling (Yan, 2024). The field continues to explore a variety of solutions, ranging from technical innovations in model architecture to policy and economic strategies aimed at ensuring equitable access to function-calling capabilities (Roberts, 2023a; Yan, 2024).

2.3. Fine-Tuning Techniques, Approaches and Strategies

The fine-tuning of LLMs such as GPT-3, GPT-4, and Llama 2 has witnessed significant advancements, driven by the need to adapt these models more effectively to diverse applications. This section investigates the sophisticated strategies employed in fine-tuning, highlighting the essential techniques that have shaped the current practices in model optimization.

Fine-tuning LLMs involves precise adjustments to key training parameters to tailor the models to specific tasks. This customization includes modifying the learning rate, batch size, and number of epochs. Each of these parameters plays an important role (Run.AI, n.D.): the learning rate determines the adjustment speed of model weights to minimize errors; batch size impacts the model's generalization capabilities and computational efficiency; and the number of epochs affects the depth of learning and the risk of overfitting (Run.AI, n.D.). Proper calibration of these parameters is critical for optimizing model performance to the nuances of targeted applications (Li & Liang, 2021; Niederfahrenhorst et al., 2023).

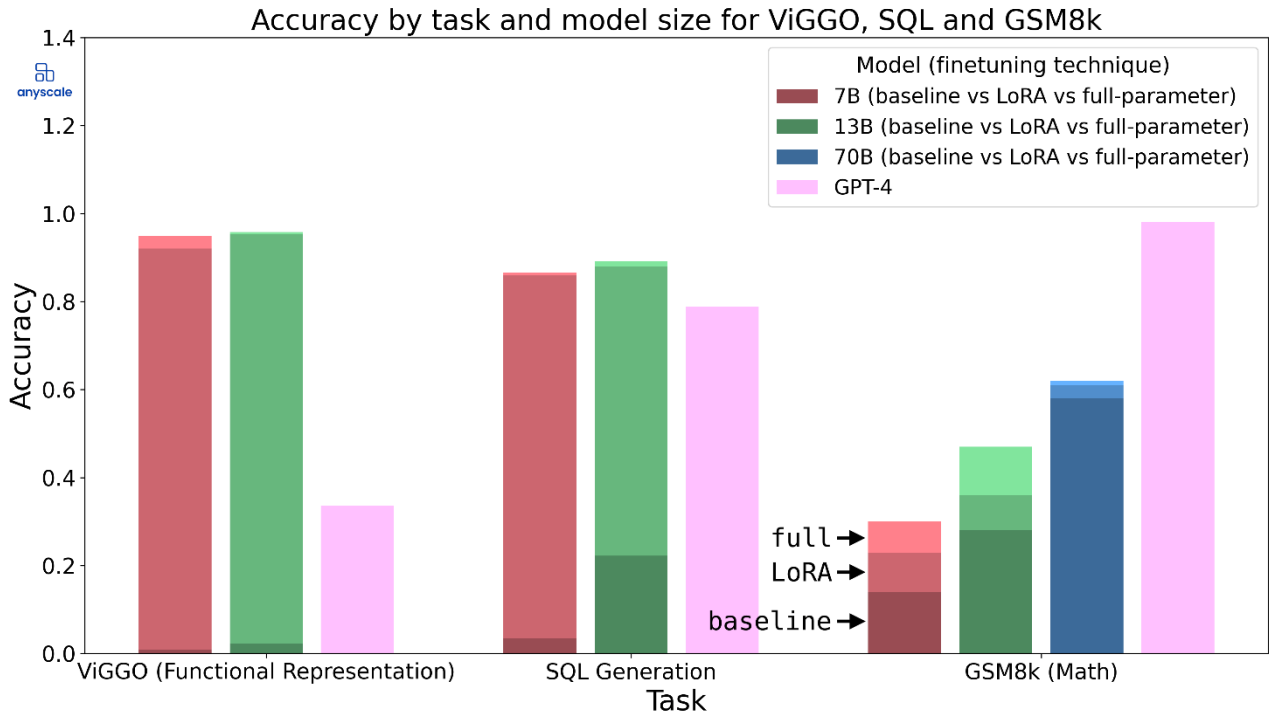
In their paper Houlsby et al. (2019) introduced PEFT (Parameter-Efficient Fine-Tuning) techniques, which have become a cornerstone in the fine-tuning of LLMs, enabling models to adapt to new tasks with minimal modifications to their existing parameters. This approach retains the model's pre-trained strengths while integrating new functionalities, offering a balance between resource efficiency and task-specific adaptability. PEFT addresses the challenge of catastrophic forgetting and reduces the resource demands typically associated with comprehensive retraining (Houlsby et al., 2019; Li & Liang, 2021).

Given the hardware limitations often encountered in practice, gradient accumulation has surfaced as a new technique. It allows for effective training under resource constraints by accumulating gradients over multiple iterations before updating the model weights and simulating larger batch sizes (Raj et al., 2024). This method improves training stability and model generalization by providing

more accurate gradient estimates, which is essential for training robust models (Y. Wang et al., 2024; Zhang et al., 2024; Raschka, 2023).

Low-Rank Adaptation (LoRA) offers a novel approach to fine-tuning by integrating low-rank matrices into the model's architecture, targeting selective updates that improve specific functionalities without overhauling the entire model structure (E. Hu et al., 2021). This method optimizes the computational efficiency by limiting the number of parameters updated during fine-tuning, which not only conserves resources but also maintains the model's core accuracy and reduces the risk of overfitting (E. Hu et al., 2021; Niederfahrenhorst et al., 2023). Figure 10 provides a comparative analysis of model performance with LoRA against baseline and full-parameter models.

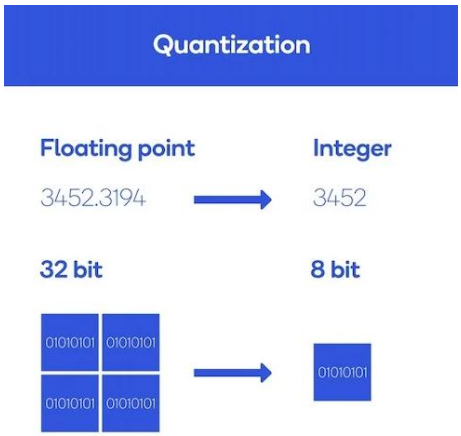
Figure 10 Model Performance Comparison: Baseline vs LoRA vs Full-Parameter models



Source: Niederfahrenhorst et al., 2023

Building on LoRA, Quantized Low-Rank Adaptation (QLoRA) incorporates quantization to further reduce the computational demand and memory footprint, facilitating deployment in resource-limited settings. This technique allows the model to maintain high performance while operating more efficiently (Dettmers et al., 2023). Figure 11 depicts the memory savings achieved through quantization from 32-bit to 8-bit precision.

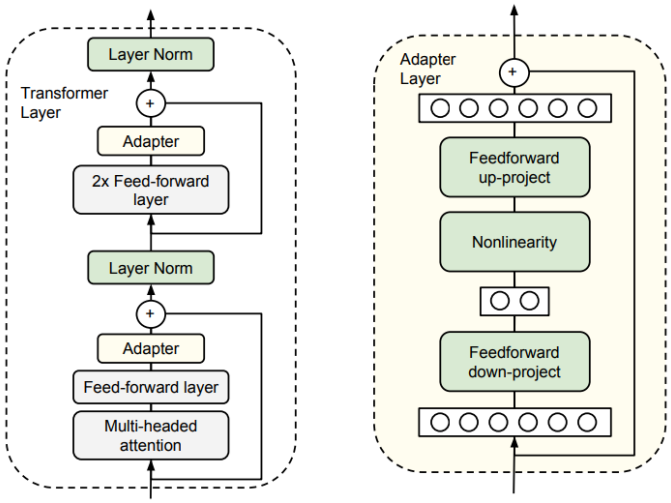
Figure 11 Memory Savings with Quantization, 32-bit floating point to 8-bit integer



Source: John, 2023

Adapter modules represent another advancement in fine-tuning LLMs, allowing for efficient model adaptation to new tasks by inserting small, modular neural networks into the existing model architecture. This strategy preserves the integrity of the original model while enabling quick adaptation to new data, significantly reducing the associated computational costs (Houlsby et al., 2019). Figure 12 shows how an adapter layer integrates into the Transformer architecture.

Figure 12 Adapter Layer integrated into Transformer Architecture



Source: Houlsby et al., 2019, p. 3

Prompt tuning and tool augmentation are complementary strategies that refine the model's interaction with specific tasks and external data sources. Prompt tuning involves crafting input prompts to guide the model's responses, improving its relevance and accuracy for particular applications. Tool augmentation extends the model's capabilities by integrating external tools or APIs, allowing real-time data retrieval and interaction, which broadens the scope of LLM applications in dynamic environments (Li et al., 2023; Wang et al., 2024).

2.4. Multilingual and Cross-Domain Considerations

2.4.1. Multilingual Function-Calling

The implementation of multilingual function-calling in LLMs requires comprehensive linguistic understanding, achievable through training on diverse multilingual datasets. Such training equips LLMs to accurately process inputs and execute function-calls in multiple languages, ensuring effective communication regardless of linguistic diversity (Conneau et al., 2020). Transformer-based models, exemplified by Vaswani et al. (2017), have been instrumental in this regard due to their efficient handling of sequential data across languages.

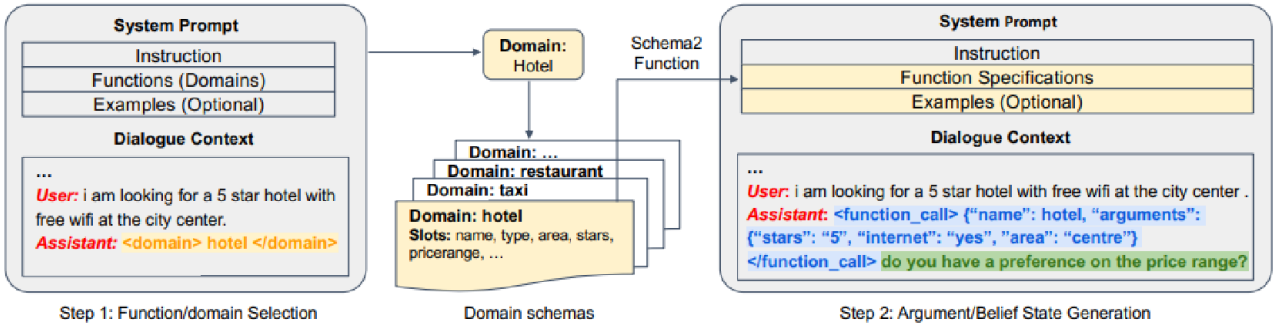
The cross-domain challenge in multilingual function-calling is ensuring that LLMs not only translate or process language but also adapt to domain-specific contexts in different languages. For example, a multilingual LLM in the healthcare sector must accurately understand and generate medical function-calls in various languages, applying domain-specific knowledge universally (Devlin et al., 2019).

Significant strides have been made with models like mBERT and XLM-R, which improve multilingual capabilities and provide frameworks for implementing effective multilingual function-calling across various domains (Conneau et al., 2020).

2.4.2. Domain-Specific Function-Calling

Achieving domain-specific function-calling involves training models on specialized datasets to grasp industry-specific terminologies and processes. This dual capability of general language understanding coupled with specialized knowledge allows LLMs to perform precise and contextually appropriate functions within various domains (Vaswani et al., 2017; Devlin et al., 2019), which is highlighted in Figure 13.

Figure 13 Example of Domain-Specific Function-Calling



Source: Li et al., 2024, p. 4

The foundational technology facilitating this capability remains the Transformer architecture, which supports sequential data processing and adapts to varied domain requirements through advanced fine-tuning methods like those used in BERT and its variants (Conneau et al., 2020).

In healthcare, for instance, domain-specific LLMs can interface with medical databases to aid diagnostic processes, improving the decision-making capabilities of medical professionals (Shickel et al., 2018). In academia, models like BioBERT and SciBERT demonstrate the effectiveness of LLMs in navigating complex scientific terminologies and data, underscoring their adaptability across specialized fields (Lee et al., 2020; Beltagy et al., 2019).

2.5. Ethical Considerations and Societal Impact

This section critically examines these dimensions, particularly focusing on issues related to bias, fairness, privacy, security, and the broader implications for employment and industry practices.

The ability of LLMs to understand and generate human-like text, coupled with their function-calling capabilities, poses substantial risks related to bias propagation. Biases in LLMs may reflect the inherent prejudices in their training data, potentially leading to discriminatory outcomes across various demographic groups when these models are employed in decision-making processes or content generation (Bender et al., 2021; Devlin et al., 2019). Addressing these biases requires a multifaceted approach, including the diversification of training datasets and the implementation of algorithmic fairness strategies that seek to normalize performance across different groups (Mehrabi et al., 2021). Moreover, maintaining transparency and accountability in model development and application is vital for mitigating bias and increasing trust in LLM technologies (Jo & Gebru, 2019).

In the creation of Llama 2, several steps have been taken to pre-train LLMs responsibly, as exemplified by Touvron et al. (2023). Their pretraining process excluded data from sources known to contain high volumes of personal information and employed standard privacy and legal reviews for each dataset used. This approach aims to minimize ethical risks and reduce the carbon footprint of pretraining by sharing models broadly, thereby mitigating the need for redundant model training efforts (Touvron et al., 2023).

As LLMs increasingly interact with external systems through APIs, they must handle sensitive data securely. This includes personal information, proprietary data, and security credentials, all of which are susceptible to cyber threats such as data breaches and unauthorized access (McGregor et al., 2017; Vaswani et al., 2017). Ensuring robust security measures and adherence to data protection regulations, such as the General Data Protection Regulation (EU, 2016) and the EU AI Act (EU, 2024), is crucial. These measures should encompass privacy-by-design principles and continuous security assessments to address potential vulnerabilities effectively.

The deployment of LLMs can significantly alter employment landscapes and industry practices. While LLMs offer the potential to improve productivity and foster innovation by automating routine tasks, they also pose risks to job security, particularly in sectors heavily reliant on language processing and data management (Autor, 2015; Zinkula & Mok, 2024). The societal impact of these technologies extends to concerns over job displacement in areas such as content creation and legal services, where automation may supplant human roles.

To mitigate these impacts, it is imperative to develop and implement policies that promote workforce adaptation and skill development. Educational and training initiatives are essential to prepare individuals for the evolving job market, equipping them with the necessary skills to work alongside AI technologies (Laker, 2023). Furthermore, ethical guidelines should emphasize the augmentation of human capabilities rather than their replacement, aiming to create a cooperative environment where human expertise and AI efficiency are optimally integrated (Zinkula & Mok, 2024).

The ethical integration of LLMs into societal frameworks requires a careful balance of innovation and caution. As these technologies continue to evolve, the focus must remain on developing ethical practices that address the complex challenges they bring to the forefront. Ensuring that LLMs contribute positively to society involves constant vigilance, proactive regulatory measures, and a commitment to fostering an inclusive, secure, and equitable digital landscape. This approach not only mitigates the risks associated with LLM deployment but also maximizes their potential benefits across various sectors of society.

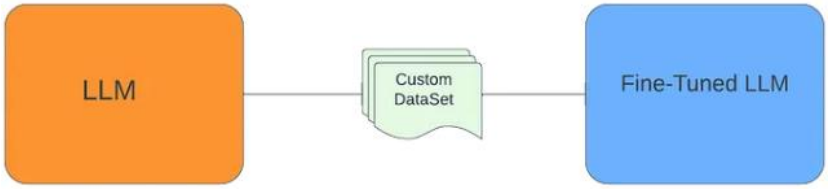
For instance, in healthcare, LLMs can assist in diagnosing diseases and personalizing treatment plans, potentially improving patient outcomes and reducing healthcare costs (Topol, 2019). In education, they can provide personalized learning experiences and support educators in creating more engaging and effective curricula (Holmes et al., 2019).

2.6. Practical Considerations Fine-Tuning Small LLMs

Fine-tuning smaller variants of LLMs, such as Llama 2, presents unique challenges and opportunities. This section draws from a range of scholarly works to outline key practical considerations for fine-tuning these models, addressing common approaches, best practices, technical problems, and areas needing further exploration.

The primary focus in fine-tuning small LLMs is optimizing model performance without the extensive computational resources typically required for larger models. This involves adjusting critical training parameters such as learning rate, batch size, and number of training epochs to suit specific tasks, a process that directly impacts the model's learning efficacy and overall performance (Capelle, 2024a; Capelle, 2024b). Custom datasets play a crucial role in this context, as they allow for targeted training that closely aligns with specific application needs (Das, 2024; Kuo, 2024), as illustrated in Figure 14.

Figure 14 Fine-tuning LLM with Custom DataSet



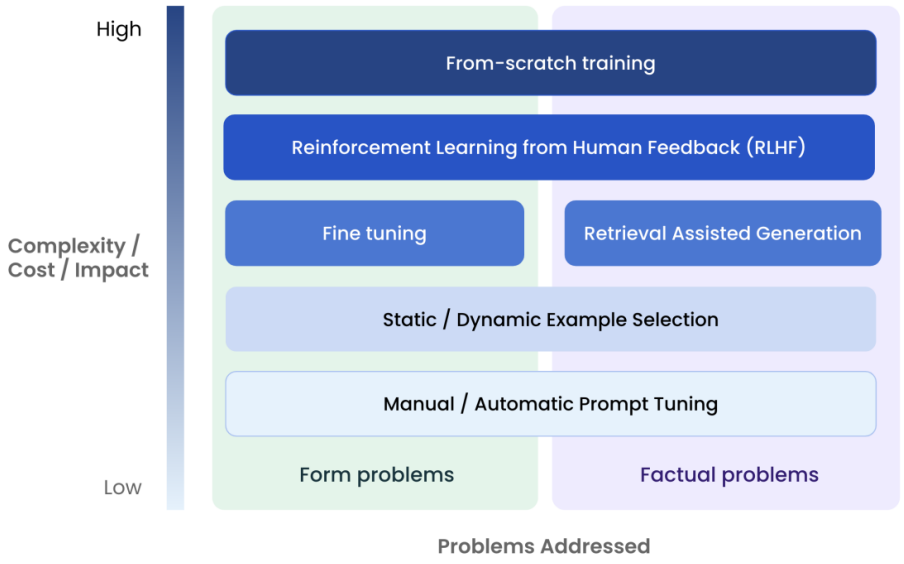
Source: Das, 2024

Integrating function-calling models and APIs, as highlighted in resources like Fireworks.ai (2023), emphasizes the need to improve model performance through external capabilities. This integration involves ensuring that LLMs can effectively interact with these APIs, a complex challenge that requires sophisticated methodologies to maintain seamless interaction and extend the model's application potential (Fireworks.ai, 2023).

One significant technical challenge is the creation and curation of specialized datasets that cater precisely to the model's training requirements. This task is intricate, involving careful selection, preparation, and refinement of data to ensure it effectively trains the model for specific function-calls (Das, 2024; Kuo, 2024).

Moreover, maintaining a balance between performance and computational efficiency is necessary, especially in environments with limited resources. Strategies like gradient accumulation, LoRA, and QLoRA are particularly valuable in this context, as they allow for effective model training and fine-tuning within constrained settings (Raj et al., 2024; John, 2023). Figure 15 one can see how fine-tuning a LLM ranks in terms of complexity and cost. This balance affects the model's deployment potential, making it a crucial consideration for ensuring that LLMs can be effectively utilized in a wide range of settings without necessitating excessive computational resources (John, 2023).

Figure 15 Complexity, Cost, & Impact of Domain-Specific Model Refinement Techniques



Source: Kadous & Hakhamaneshi, 2023

Despite the advancements in fine-tuning techniques, several areas require further research and development. The complexity of integrating external APIs with LLMs calls for more advanced interfacing techniques that can handle diverse and dynamic data inputs seamlessly (Endpoints Team, 2023). Additionally, the process of dataset curation and the strategic selection of training parameters need ongoing refinement to increase the efficacy and efficiency of fine-tuning practices (Hakhamaneshi & Ahmad, 2023).

The practical considerations of fine-tuning LLMs also intersect with broader ethical and societal impacts. As these models become more capable and widespread in their applications, issues such as data privacy, security, and the potential for bias in model outputs become increasingly pertinent (Bender et al., 2021). Addressing these concerns requires not only technical solutions but also robust policy frameworks and ethical guidelines to ensure the responsible use of AI technologies (Touvron et al., 2023).

Fine-tuning small LLMs like Llama 2 involves a complex interplay of technical strategies, dataset management, and integration with external systems. The challenges highlighted in this discussion underscore the need for continued innovation in model training methodologies and the development of more efficient and effective ways to improve model adaptability (Raffel et al., 2020). As the field progresses, it will be essential to keep these considerations in balance with the ethical and societal implications of deploying advanced AI systems, ensuring that they contribute positively to both technological advancement and societal well-being (Raschka, 2023).

2.7. Current Gaps in LLM Fine-Tuning Research

2.7.1. Diversity in Programming Languages and Systems Integration

Current research in the field of LLMs predominantly focuses on widely used programming languages, especially Python. These languages are integral to the development and operation of LLMs due to their extensive libraries, robust community support, and alignment with modern web and application development standards (Vaswani et al., 2017; Devlin et al., 2019). However, this focus overlooks the potential for LLM integration with less common or domain-specific languages, which are crucial in certain sectors such as finance, healthcare, and embedded systems where languages like R, MATLAB, and others play a pivotal role.

The predominance of popular languages in LLM research can be attributed to several factors, including the availability of pre-existing frameworks and datasets, and the commercial focus on applications with a broad market appeal (Touvron et al., 2023). Yet, this focus limits the applicability of research findings to scenarios where these languages are not the norm, potentially alienating sectors that rely on specialized languages for their operations.

The research community has thus far not extensively explored the integration of LLMs with legacy systems and less common programming languages. This gap in research presents several challenges:

- LLMs developed with a focus on popular programming languages may not perform optimally in environments governed by legacy systems or specialized languages. This restricts the deployment of LLM technologies in sectors such as manufacturing, where older languages and systems are prevalent (Greyling, 2023a, Greyling, 2023b).

- The lack of research into the adaptation of LLMs to diverse programming ecosystems can lead to interoperability issues, where systems cannot effectively communicate or integrate, leading to increased costs and reduced efficiency (Hakhamaneshi & Ahmad, 2023).
- By not addressing the needs of industries reliant on specific technologies, the field risks stagnating innovation in areas that could significantly benefit from advanced AI applications, such as automated code generation, bug fixes, and more dynamic data handling in scientific computing or real-time systems (Kaddour, et al., 2023; Raffel et al., 2020).

The opportunities for addressing these gaps are substantial and include:

- Research could focus on developing LLMs that understand and generate code in less common languages, broadening the applicability of these models to various industries. This would involve creating datasets specific to these languages and adjusting model architectures to handle different syntactic and semantic constructs (Kaddour, et al., 2023; Roberts, 2023a; Roberts, 2023b).
- Developing methodologies for integrating LLMs with legacy systems could revive older technologies, making them more efficient and capable of handling modern computational tasks. This would involve research into adapters or middleware that can translate between modern AI outputs and the input requirements of older systems (Capelle, 2024a; Capelle, 2024b).
- Encouraging cross-disciplinary research initiatives that bring together AI researchers with domain experts from fields that rely on specialized languages can foster a deeper understanding of the specific requirements and constraints of these fields, leading to more tailored and effective AI solutions (Capelle, 2024a; Capelle, 2024b).

While the current research landscape for LLMs focuses predominantly on mainstream programming languages, there is a critical need for expansion into less common languages and legacy system integration. Addressing this gap not only broadens the applicability of LLM technologies but also supports essential industries yet currently underserved by advancements in AI. This approach would not only diversify the impact of LLMs but also stimulate innovation in areas that have been slow to adopt modern AI technologies.

2.7.2. Handling Real-Time Data and Dynamic APIs

While APIs have improved the capabilities of LLMs significantly, current methodologies predominantly focus on static API function-calling, where the data requirements and outputs are predefined and unchanged over time. This approach is well-suited for applications where data variability is low and updates are infrequent (Raffel et al., 2020).

Static API function-calling in LLMs, such as the pre-defined API call structures used in models like GPT-3, is designed to handle requests and responses in a predictable format (Brown et al., 2020).

While effective for stable environments, these methods face significant limitations when tasked with handling dynamic or real-time data streams. The primary limitations include:

- Static APIs cannot adjust to changes in data structure or query parameters without manual updates or reconfiguration, limiting their use in environments where data inputs are frequently changing (Greyling, 2023a; Greyling, 2023b).
- As the volume and velocity of data increases, static API function-calling methods struggle to process information efficiently, potentially leading to delays and outdated information being served to the end-users (Hakhamaneshi & Ahmad, 2023).
- Static methods often lack robust mechanisms for dealing with unexpected data types or malformed requests, which are more common in real-time scenarios where data integrity can vary (Zhao et al., 2023; Kaddour et al., 2023).

To overcome these limitations, there is a need to develop methodologies that extend function-calling capabilities to accommodate dynamic and real-time data processing. The future directions in this area involve:

- Developing LLM architectures that can adapt to changing API schemas by incorporating metadata analysis and schema detection techniques. This would allow LLMs to adjust function-calls based on real-time changes in data structure or available API functions (Zhao et al., 2023; Kaddour et al., 2023).
- Implementing event-driven architectures in LLMs, where the model can respond to data events in real-time rather than relying on periodic or batch processing. This could involve the use of webhooks or streaming APIs to facilitate immediate data processing and response generation (Zhao et al., 2023; Kaddour et al., 2023; Touvron et al., 2023).
- Improving LLMs with better error detection and diagnostic tools to manage the inconsistencies and anomalies often present in real-time data streams. This would include developing sophisticated algorithms to identify and correct errors or to provide actionable feedback to users when errors occur (Kaddour et al., 2023; Raffel et al., 2020).

Current methodologies predominantly focus on static function-calling, which, while effective for stable environments, exhibit notable limitations when tasked with dynamic or real-time data streams (Raffel et al., 2020; Brown et al., 2020). Zhao et al. (2023) provide a comprehensive survey that highlights the need for advanced LLM functionalities to manage the increasing volume and velocity of data. Meanwhile, Kaddour et al. (2023) identify broader applications and challenges that include dynamic data handling, reinforcing the necessity for methodologies that can adapt to rapidly changing data environments. This shift promises not only better flexibility and scalability but also positions LLMs to be more responsive in a continuously changing data landscape.

2.7.3. Identifying and Addressing further Research Deficiencies

Recent studies such as those by Kaddour et al. (2023) and Zhao et al. (2023) have extensively documented advancements and applications of LLMs. These studies reveal the increasing complexity and capabilities of LLMs, highlighting their utility in a range of applications from NLP to complex decision-making systems. However, while these research studies provide valuable insights, they also expose significant gaps in the field that need to be addressed.

Kaddour et al. (2023) discuss the challenges LLMs face when applied outside standard datasets or typical application scenarios. There is a noted gap in the adaptability of LLMs to generalize well across diverse real-world domains, which often have unique, unstructured data not represented in the training datasets.

Zhao et al. (2023) touch upon the ethical implications and bias in LLM outputs. Current research often underestimates the reproduction of inherent biases in the training data, leading to ethical concerns in practical applications. There's a need for more rigorous methodologies to detect, mitigate, and manage these biases effectively.

Both sources highlight scalability, but there's a lack of detailed methodologies for efficiently scaling LLMs in production environments. As LLMs grow in parameter size, the computational demand also increases, raising concerns about the practicality of deploying these models in resource-constrained settings.

Kaddour et al. (2023) note a gap in interdisciplinary research involving LLMs. Integrating insights from fields such as cognitive science, psychology, and domain-specific expertise can improve contextual awareness and application-specific performance of LLMs.

To bridge these gaps, future research should focus on developing more robust models that can handle the complexity of real-world data and ethical challenges. This includes creating diverse datasets that better represent global demographics and integrating interdisciplinary knowledge to increase the contextual understanding of LLMs. Additionally, developing new architectures and optimization techniques to improve the efficiency and scalability of LLMs will be crucial for their sustainable growth and application across various industries.

3. Fine-Tuning Experiments with LLMs

3.1. Experimental Design and Setup

The primary objective of these experiments is to assess the effectiveness of fine-tuning an LLM, specifically Llama 2 and Mistral Instruct v0.2, for increased function-calling capabilities. The hypothesis posits that fine-tuning will significantly improve the model's ability to discern user intent and generate accurate, context-appropriate outputs.

For this experiment, the Llama 2 LLM by Meta (Meta, n.D), and Mistral Instruct v0.2 (Jiang et al., 2023), both open-source models known for their robust architecture and broad NLP capabilities, were selected. The 7 billion parameter variants, Llama 2 7B and Mistral Instruct 7B strike a balance between computational efficiency and sophisticated performance, making it suitable for detailed fine-tuning (Touvron et al., 2023; Jiang et al., 2023). The utilization of this specific model iteration allows for a focused investigation into the impacts of fine-tuning on a model of considerable, yet manageable, complexity and size.

In the context of improving LLMs with the ability to interface dynamically with external data sources, the utilization of the `get_current_weather` function, referenced in Code Listings 1, serves as a typical example of extending the functionality of LLMs beyond mere text generation to executing specific, data-driven tasks.

Code Listings 1 get_current_weather Function

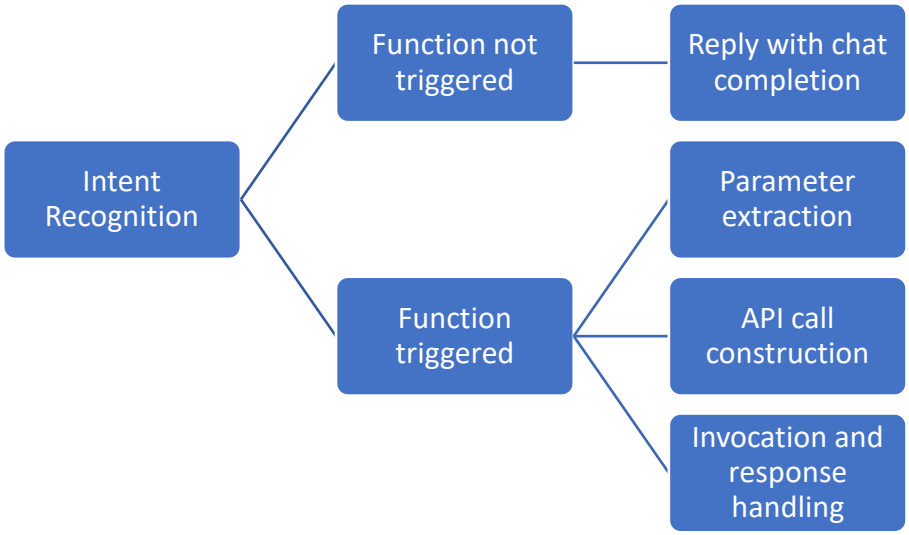
```
{
    "type":"function",
    "function":{
       "name":"get_current_weather",
       "description":"Get the current weather",
       "parameters":{
          "type":"object",
          "properties":{
             "location":{
                "type":"string",
                "description":"The city and state, e.g. San Francisco, CA"
             },
             "format":{
                "type":"string",
                "enum":[
                   "celsius",
                   "fahrenheit"
                ],
                "description":"The temperature unit to use. Infer this from the users location."
             }
          },
          "required":[
             "location",
             "format"
          ]
       }
    }
}
```

Source: Endpoints Team, 2023

The `get_current_weather` function is designed as an API call that enables the LLM to fetch current weather information for a specified location and return the data in a predefined format. The function is structured as follows: it requires a "location" parameter, which is a string denoting the city (e.g., "San Francisco, CA"), and a "format" parameter, which specifies the temperature unit (either "celsius" or "fahrenheit"). The design of this function encapsulates the requirements for invoking the API call, including the necessary parameters and the type of response expected.

When an LLM is presented with a query such as "Is it today sunny in San Francisco?", the model's task is to decipher the intent behind the question, specifically, to retrieve current weather information for San Francisco—and to construct an API call request that aligns with the `get_current_weather` function's structure. This process involves several critical steps as shown in Figure 16:

Figure 16 Decision Flowchart for LLM-Based API Function-Call Generation



Source: own creation, 2024

- The LLM must first understand that the query pertains to obtaining weather information, which involves recognizing keywords and the overall context of the question.
- If the questions involve nothing that intends to get current information about the weather, the function is not triggered, and a reply is generated using chat completion.
- If the model deduced a function-call intent, it then extracts relevant information from the query to populate the API call's parameters. In this instance, it identifies "San Francisco" as the location and infers from the context or default settings that the temperature format requested is "fahrenheit".
- Based on the extracted parameters and the predefined structure of the `get_current_weather` function, the LLM constructs the API call. This call is formatted as a JSON object that specifies the function name (`get_current_weather`) and the necessary parameters (in this case, `"location": "San Francisco, CA"` and `"format": "fahrenheit"`).

- The constructed API call is then invoked, and the LLM processes the returned data to generate a human-readable response that addresses the original query.

In this example, the LLM should send a reply to the user in a valid JSON format that looks as follows: {"type": "function", "function": {"name": "get_current_weather", "parameters": {"location": "San Francisco, CA", "format": "fahrenheit"}}} (Taoofstefan, 2024).

In evaluating the efficacy of the fine-tuning experiment, it is essential to employ evaluation metrics that capture the model's performance across various dimensions of language understanding and task execution. The following are the primary metrics proposed for the assessment:

- Accuracy: The Accuracy metric gauges the LLM's proficiency in discerning user intent, distinguishing whether a user's input requires a chat-completion response or triggers a function-call. This binary classification task is important, as it forms the foundation for appropriate model behavior in response to user queries. The accuracy of intent discrimination is thus a critical measure of the model's understanding of the context and purpose of the user's input (fireworks.ai, 2023; Hakhamaneshi & Ahmad, 2023; Greyling, 2023b).
- Precision: Upon the correct identification of a function-call intent, the Precision metric evaluates the LLM's ability to generate structured output that aligns with the specified API's requirements. This involves assessing whether the model's output correctly incorporates the necessary parameters and adheres to the JSON format prescribed by the `get_current_weather` function. ACSC is indicative of the model's ability to translate user intent into an executable action that interfaces effectively with external systems (Roberts, 2023b; Raffel et al., 2020).
- Text Coherence Evaluation (TCE): When the LLM's intent recognition leads to a chat-completion task, TCE is used to assess the relevance and coherence of the generated response. This metric evaluates whether the model's reply is contextually sound and logical within the framework of the conversation, reflecting the model's capacity for generating sensible and contextually appropriate language (Roberts, 2023b; Liu et al., 2020; fireworks.ai, 2023).

This experimental setup not only tests the practical applications of fine-tuning in improving LLMs' functional capabilities but also contributes to the theoretical understanding of how fine-tuning impacts LLM performance across varied tasks. The insights gained from this research could guide future developments in LLM technologies, particularly in optimizing models for specific functional tasks like API interactions.

The design of this experiment is intended to systematically evaluate how fine-tuning affects the ability of an LLM to perform specialized tasks, specifically through function-calling. By rigorously testing the model across well-defined metrics (accuracy, precision, TCE), this experiment aims to contribute

valuable findings to the field of NLP, offering evidence-based insights into the benefits and limitations of fine-tuning LLMs for greater interactive capabilities.

This research uses one dataset to evaluate the function-calling capabilities of the LLMs, targeting different aspects of model performance. The dataset (Taoofstefan, 2024) is tailored for the specific function-call specified in Code Listings 1. It contains a small, focused collection of entries for detailed analysis. It will be further described in chapter 4.3 Datasets.

3.2. Execution of Experiments

The execution phase of fine-tuning experiments with LLMs, such as Llama 2, requires substantial computational resources due to the intensive nature of these processes. This section details the deployment of cloud-based GPU computing services and introduces QLoRA as the primary methodological approach for fine-tuning.

The research utilizes cloud-based GPU computing services provided by Brev.dev, which specializes in virtual development environments equipped with high-performance computing capabilities (Brev.dev, n.d.). Cloud GPUs offer essential advantages for machine learning tasks, including scalability, cost-efficiency, and the ability to handle parallel computations effectively. These attributes are critical for managing the significant computational demands of fine-tuning LLMs and help in reducing the time required for training neural networks (Touvron et al., 2023; Capelle, 2024a; Nickolls et al., 2008). Utilizing cloud infrastructure allows researchers to access state-of-the-art computing power without the overhead of maintaining physical hardware, thus providing flexibility to scale resources according to the project's needs (Bergstra et al., 2011).

QLoRA appears as a prominent fine-tuning methodology, building on the foundational concepts of LoRA introduced by Hu et al. (2021). This approach has been refined further by Dettmers et al. (2023) to improve its efficiency and application in fine-tuning LLMs. QLoRA involves quantization techniques and low-rank matrix factorization, which streamline the adaptation of pre-trained models to specific tasks without extensive retraining (Dettmers et al., 2023).

Recent empirical studies underscore the effectiveness of QLoRA. Das (2023) discusses its application on custom datasets, demonstrating how QLoRA maintains the integrity of pre-trained models while effectively adapting them to task-specific nuances. Geronimo (2023) corroborates these findings with practical insights into fine-tuning Llama 2 and Mistral models using QLoRA, noting significant performance improvements on specialized tasks. Moreover, John (2023) outlines accessible strategies for implementing QLoRA in open-source LLMs, increasing its utility for a wide range of AI practitioners.

Guidelines provided by Run.ai (2023) and Schmid (2023) offer detailed technical procedures for implementing QLoRA, substantiating its role as a robust fine-tuning method in both academic and

applied contexts. These resources ensure that researchers can apply QLoRA effectively, optimizing LLMs for greater computational efficiency and better task performance.

The Python code used during the fine-tuning process incorporates these best practices and is fundamentally based on the principles of QLoRA (Raschka, 2023). This integration ensures that the fine-tuning not only adheres to current methodological standards but also leverages advanced techniques to maximize the potential of the LLM.

The execution phase of fine-tuning LLMs with QLoRA is carefully planned to exploit cloud-based GPU resources, ensuring efficient and scalable computational support (Eassa, 2022). The choice of QLoRA as a fine-tuning methodology reflects a strategic decision to improve model adaptability and performance while maintaining computational sustainability (Raschka, 2023). This approach is expected to yield insights into the optimization capabilities of LLMs, particularly in how they can be tailored to meet specific functional requirements through advanced fine-tuning techniques (Hakhamaneshi & Ahmad, 2023).

4. Research Methodology

4.1. Methodological Framework

This research adopts a mixed-methods approach, combining empirical, qualitative, and quantitative techniques to comprehensively evaluate the function-calling capabilities of LLMs. The methodological framework is designed to combine the strengths of each approach to provide a robust analysis of LLM performance in various programming environments and contexts.

Qualitative methods in this study rely on evaluating current research papers, blogs, and website articles to understand the characteristics, properties, and external influences affecting LLMs' function-calling capabilities. This approach includes a detailed analysis of the textual data from these sources, synthesizing insights from the literature and their expert commentary on LLM interactions with APIs (Creswell, 2009). The aim is to uncover nuanced perspectives on the usability and adaptability of LLMs in handling complex function-calls.

Empirical methods involve direct or indirect observation of the phenomenon under study. In the context of this thesis, empirical research is conducted through experiments that assess the precision and efficiency of LLMs in generating API-compliant function-calls (Creswell, 2009). These experiments are designed to simulate real-world scenarios where LLMs interact with different APIs to perform specific tasks, providing concrete data on their performance metrics.

This research uses a combination of empirical experimentation and qualitative analysis of the existing literature to explore the function-calling capabilities of LLMs. By focusing on empirical evidence gathered from experiments and synthesizing findings from scholarly articles, blogs, and expert discussions, the study aims to ascertain the effectiveness of LLMs in specific scenarios. This targeted approach helps understand the practical applications and challenges of deploying LLMs, providing insights into how they perform in real-world settings.

The methodological framework is supported by extensive literature that discusses the theoretical underpinnings of each method and its application in AI research. Critically, while the mixed-methods approach provides a thorough understanding of the subject matter, it also introduces complexities in data integration and interpretation. Balancing the insights from different methods requires careful consideration to ensure that the conclusions drawn are valid and reliable.

4.2. Experimental Design

4.2.1. Selection of Llama 2 for Fine-Tuning

Opting for an open-source model and specifically choosing Llama for fine-tuning experiments offer distinct benefits. The choice between the two is influenced by a range of factors including accessibility, the scope of customization, and alignment with specific project requirements. We will shortly discuss the reasons for using an open-source LLM architecture and Llama 2 7B in particular.

Open-source models provide a foundational platform that allows for transparency, flexibility, and cost savings. This accessibility fosters a diverse ecosystem of contributions, facilitating rapid advancements and the exchange of ideas (IBM Data and AI Team, 2023a). Furthermore, open-source models enable reproducibility and verification of research results, a cornerstone for scientific progress (Pineau et al., 2021). They also offer flexibility for customization, allowing researchers and developers to adapt and refine models to suit specific tasks or domains. The communal nature of open-source projects often means that models are continuously improved through collective efforts, ensuring that they remain at the cutting edge of technology (IBM Data and AI Team, 2023a).

Llama 2, as introduced by Touvron et al. (2023), exemplifies the latest advancements in LLMs, offering several unique benefits for fine-tuning experiments. One of the key strengths of Llama 2 is its performance efficiency across a wide range of parameters, from 7 billion to 70 billion, making it adaptable for both large-scale and resource-constrained environments. This scalability ensures that Llama 2 can be fine-tuned for a variety of applications, from complex reasoning tasks to dialogue systems, without significant loss in performance. The model's architecture, which incorporates innovations such as grouped-query attention, facilitates improved inference scalability and computational efficiency (Touvron et al., 2023).

Moreover, Llama 2's fine-tuning methodologies, including Supervised Fine-Tuning (SFT) and Reinforcement Learning with Human Feedback (RLHF), provide a structured approach to model alignment with human preferences, improving its usability and safety in real-world applications. The comprehensive description of fine-tuning and safety improvement strategies presented by Touvron et al. (2023) offers valuable insights for researchers looking to replicate and build upon their work. By opting for Llama 2, researchers can leverage these documented methodologies to expedite the development process and achieve better outcomes in terms of model performance and safety.

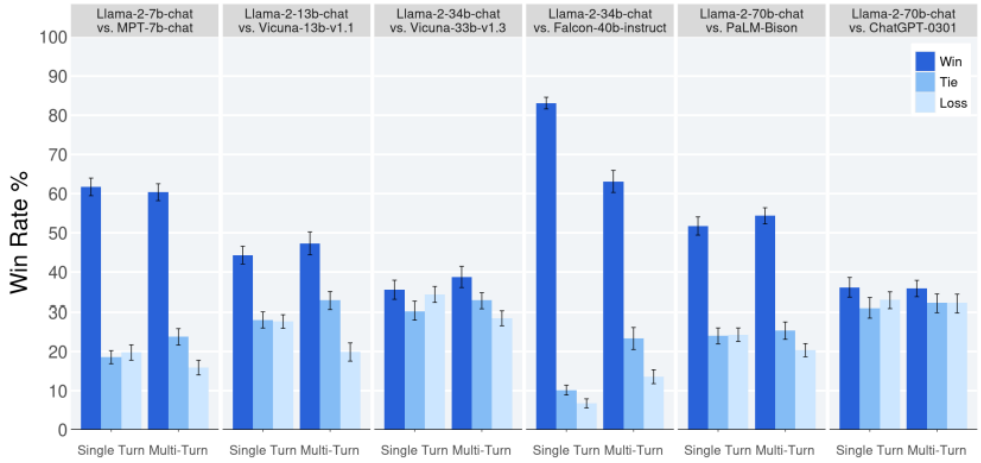
The models introduced by Touvron et al. (2023), particularly the Llama 2-Chat variants, are optimized for dialogue applications and exhibit superior performance on various benchmarks compared to other open-source models, as shown in Table 1, and in Figure 17 we can see the overall performance against open and closed models with Llama 2 7B-chat outperforming its peers significantly The authors provide a comprehensive overview of the methodologies employed for fine-tuning and increasing the safety of Llama 2-Chat, aiming to enable the community to replicate their work and further the responsible development of LLMs.

Table 1 Overall performance compared to open-source base models

Model	Size	Code	Commonsense Reasoning	World Knowledge	Reading Comprehension	Math	MMLU	BBH	AGI Eval
MPT	7B	20.5	57.4	41.0	57.5	4.9	26.8	31.0	23.5
	30B	28.9	64.9	50.0	64.7	9.1	46.9	38.0	33.8
Falcon	7B	5.6	56.1	42.8	36.0	4.6	26.2	28.0	21.2
	40B	15.2	69.2	56.7	65.7	12.6	55.4	37.1	37.0
LLAMA 1	7B	14.1	60.8	46.2	58.5	6.95	35.1	30.3	23.9
	13B	18.9	66.1	52.6	62.3	10.9	46.9	37.0	33.9
	33B	26.0	70.0	58.4	67.6	21.4	57.8	39.8	41.7
	65B	30.7	70.7	60.5	68.6	30.8	63.4	43.5	47.6
LLAMA 2	7B	16.8	63.9	48.9	61.3	14.6	45.3	32.6	29.3
	13B	24.5	66.9	55.4	65.8	28.7	54.8	39.4	39.1
	34B	27.8	69.9	58.7	68.0	24.2	62.6	44.1	43.4
	70B	**37.5**	**71.9**	**63.6**	**69.4**	**35.2**	**68.9**	**51.2**	**54.2**

Source: Touvron et al., 2023, p. 8

Figure 17 Llama 2 Human Evaluation Results

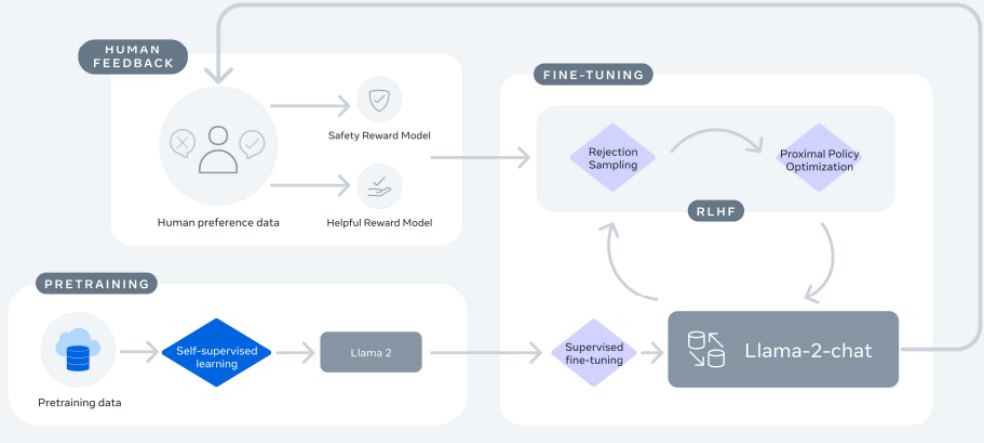


Source: Touvron et al., 2023, p. 19

Touvron et al. (2023) describe the development process involved extensive pretraining on a diverse dataset followed by fine-tuning to align the models with human preferences through techniques such as SFT and RLHF, as showcased in Figure 18. The paper emphasizes the importance of safety in model training, detailing the implementation of safety-specific data annotation and tuning, red-teaming, and iterative evaluations to mitigate potential risks associated with LLM deployment.

According to Touvron et al. (2023), Llama 2's performance improvement is attributed to several factors, including an expanded pretraining corpus, increased context length, and the use of grouped-query attention for improved inference scalability. The fine-tuning phase incorporated novel strategies such as SFT with high-quality, diverse data and RLHF for aligning model outputs with human preferences, thereby improving dialogue consistency and safety. The authors also highlight the significance of iterative reward modeling and the introduction of a novel technique, Ghost Attention, to manage dialogue flow across multiple turns effectively.

Figure 18 Training of Llama 2-Chat



Source: Touvron et al., 2023, p. 5

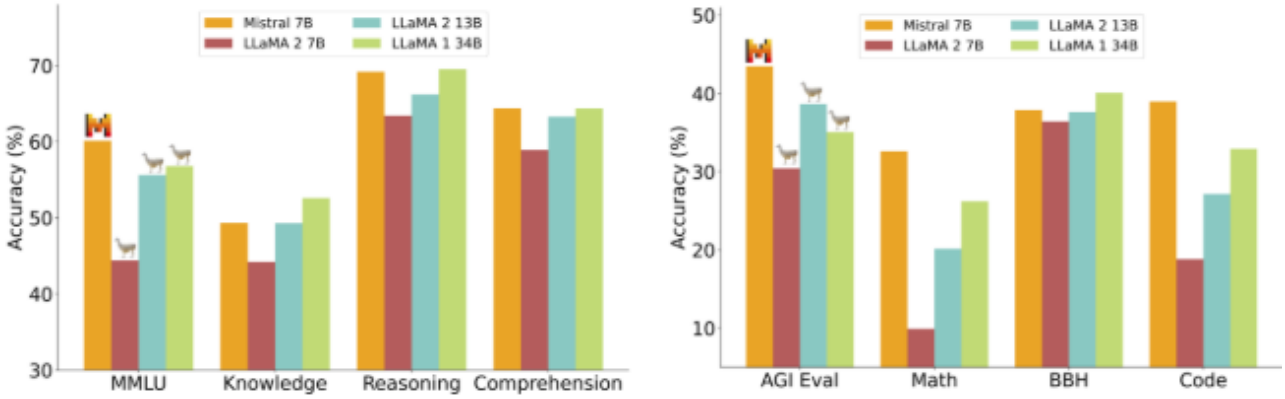
4.2.2. Selection of Mistral for Fine-Tuning

In addition to Llama 2, Mistral Instruct 7B was selected as a second small and open-source LLM for fine-tuning in this research. It has similar advantages compared to Llama 2 (Jiang et al., 2023). For this reason, this chapter will only give a brief overview, focusing on the performance differences of Llama 2 and Mistral 7B.

Mistral 7B's documented performance in various benchmarks further justifies its selection for this research. The model has shown superior performance in tasks requiring structured output and complex reasoning, which are critical areas of focus for this study (Jiang et al., 2023).

It is particularly noteworthy that Mistral 7B not only outperforms Llama 7B in different tasks and tests, as shown in Figure 19, but it also manages to outperform or be at par with the bigger Llama models, such as Llama 13B and 34B (Jiang et al., 2023)

Figure 19 Performance comparison of Mistral 7B and Llama 2 7B, 13B, and 34B



Source: Jiang et al., 2023, p. 4

Leveraging these strengths, Mistral 7B complements Llama 2, enabling a more comprehensive exploration of fine-tuning techniques and their effects on LLM performance.

4.2.3. Python Code and Fine-Tuning Parameters

In this chapter, the essential parts of the Python code used to fine-tune the LLM will be introduced and discussed. The code is based on a Jupyter Notebook from the GitHub repository of Schmid (2023). Some adjustments had to be made, especially for the training dataset and the prompt generation.

The dataset is directly loaded from Hugging Face (Hugging Face, n.D.d) as shown in Code Listings 2. The used dataset can easily be changed to train on different datasets.

Code Listings 2 Load Dataset

```
from datasets import load_dataset

train_dataset = load_dataset('taoofstefan/function_call_weather', split='train')
```

Source: Adapted from: Schmid, 2023

In Code Listings 3, a language model is initialized for fine-tuning using advanced quantization techniques facilitated by the 'BitsAndBytesConfig' from the 'transformers' library. Specifically, the model, identified by "meta-llama/Llama-2-7b-hf", is configured to load with 4-bit quantization. This quantization is further improved by double quantization ('bnb_4bit_use_double_quant=True') and uses a noise-free 4-bit quantization type ('bnb_4bit_quant_type="nf4"'). The computation employs 'torch.bfloat16', optimizing performance on CUDA-enabled devices. The model is then instantiated with these parameters to run on GPU, leveraging PyTorch's capabilities for efficient computation (Hugging Face, n.D.a).

Code Listings 3 Load LLM and Quantization

```
import torch
from transformers import AutoTokenizer, AutoModelForCausalLM, BitsAndBytesConfig

# Hugging Face model id
model_id = "meta-llama/Llama-2-7b-hf"

# BitsAndBytesConfig int-4 config
bnb_config = BitsAndBytesConfig(
    load_in_4bit=True,
    bnb_4bit_use_double_quant=True,
    bnb_4bit_quant_type="nf4",
    bnb_4bit_compute_dtype=torch.bfloat16
)

# Load model and tokenizer
model = AutoModelForCausalLM.from_pretrained(
    model_id,
    quantization_config=bnb_config,
    use_cache=False,
    device_map="auto"
)
```

Source: Schmid, 2023

In Code Listings 4, an 'AutoTokenizer' is instantiated for a pre-trained model using the 'from_pretrained' method from the 'transformers' library. The tokenizer is configured with the

model's identifier and is specifically modified to include both end-of-sequence (EOS) tokens. This setup ensures that the tokenization process accommodates sequences that are delineated, aiding in precise model training and inference tasks (Hugging Face, n.D.b; Hugging Face, n.D.c).

Code Listings 4 AutoTokenizer

```
tokenizer = AutoTokenizer.from_pretrained(model_id)
tokenizer.pad_token = tokenizer.eos_token
tokenizer.padding_side = "right"
```

Source: Schmid, 2023

As a next step, we need to construct a structured prompt intended for a language model-based assistant, as seen in Code Listings 5. It creates a scenario where the assistant is required to respond to a user query, aiming to transform a question into an API call. In this case, the prompt expects the LLM to infer the function-call from the input and output pairs.

The ’sample["Question"]’ and ‘sample["Answer"]’ are incorporated into the full prompt to set the expected interaction pattern. This constructed prompt is then passed to a separate function, tokenize, which tokenizes it, preparing it for further processing by a language model. This approach aids in training and fine-tuning LLMs for specific tasks related to conversational AI and structured data extraction (Wolf et al., 2020).

Code Listings 5 Generate Training Prompt

```
def format_instruction(sample):
          return f"""### Instruction:
Use the Input below to infer using the get_current_weather function-call correctly.

### Question:
{sample['Question']}

### Answer:
{sample['Answer']}
"""
```

Source: Adapted from: Schmid, 2023

Now we need to implement the LoRA configuration for PEFT of a language model. In Code Listings 6 we set LoRA hyperparameters such as ‘r=64’, ‘lora_alpha=16’, and ‘lora_dropout=0.1’, and specify target modules for low-rank adaptation. The ‘get_peft_model’ function applies this configuration to the given model, allowing it to leverage LoRA for efficient training (Mangrulkar et al., 2022). We use the QLoRA adapters in a later step. The LoRA hyperparameters are higher than the ones suggested by Dettmers et al. (2023) in their QLoRA paper, as there is some evidence that higher values can lead to better training results (Raschka, 2023a).

Code Listings 6 LoRA Configuration

```
from peft import LoraConfig, prepare_model_for_kbit_training, get_peft_model

# LoRA config based on QLoRA paper
peft_config = LoraConfig(
        lora_alpha=16,
        lora_dropout=0.1,
        r=64,
        bias="none",
        task_type="CAUSAL_LM",
)

# prepare model for training
model = prepare_model_for_kbit_training(model)
```

Source: Schmid, 2023

In Code Listings 7, we setup the training arguments. It outlines the setup and configuration for fine-tuning a LLaMA 2-based model using the 'transformers.TrainingArguments'. It begins by defining project-specific identifiers and output directories, followed by setting the tokenizer's pad token to the end-of-sequence token for uniformity. The arguments are then configured with the model, training dataset, and various training arguments. These include the output directory, warmup steps, batch size, gradient accumulation, maximum training steps, learning rate, and logging details. Additionally, it specifies checkpointing and evaluation strategies. The data collator is set for language modeling without masked language modeling, and model caching is disabled during training to avoid warnings. (Wolf et al., 2020).

Code Listings 7 Fine-Tuning the Base Model

```
from transformers import TrainingArguments

args = TrainingArguments(
    output_dir="llama-7",
    num_train_epochs=50,
    per_device_train_batch_size=6 if use_flash_attention else 4,
    gradient_accumulation_steps=2,
    gradient_checkpointing=True,
    optim="paged_adamw_32bit",
    logging_steps=10,
    save_strategy="epoch",
    learning_rate=2e-4,
    bf16=True,
    fp16=False,
    tf32=True,
    max_grad_norm=0.3,
    warmup_ratio=0.03,
    lr_scheduler_type="constant",
    disable_tqdm=False,
)
```

Source: Schmid, 2023

Once configured, the training process is initiated with 'trainer.train()', providing a comprehensive strategy for fine-tuning LLMs.

In determining the optimal batch size for training, studies suggest that smaller batch sizes may provide a better generalization error, but larger batches are more computationally efficient and stable

during training. Balancing these factors, a moderate batch size was chosen to ensure both efficiency and model performance (Kandel & Castelli, 2020).

The selection of the learning rate is critical, as it influences the convergence speed and stability of the training process. The learning rate for this project was set following empirical evidence suggesting that lower rates might slow down convergence but lead to more stable and reliable training outcomes (Kandel & Castelli, 2020).

Gradient accumulation steps were introduced to allow the model to effectively train on larger batches if needed without exceeding memory limits, following recommendations by Raschka (2023), who found that this technique helps in managing computational resources more efficiently while maintaining training performance.

These parameter settings are designed to optimize the training process under the constraints and capabilities of the Llama 2 model, as supported by current literature and empirical tests within the domain of deep learning (Capelle, 2024b; Das, 2024; Geronimo, 2023; Hakhamaneshi & Ahmad, 2023).

In Code Listings 8, we use a specialized training framework known as 'SFTTrainer' from the 'trl' library is employed to fine-tune a pre-trained language model (Hugging Face, n.D.). This framework integrates the use of a PEFT configuration, specifically designed to update a small subset of the model's parameters, thereby offering a computationally efficient alternative to full model retraining (Mangrulkar et al., 2022).

The SFTTrainer provides a robust framework for efficiently fine-tuning large-scale models by leveraging advanced techniques in parameter efficiency and data handling, thereby improving both the performance and practicality of model training in resource-constrained environments (Li & Liang, 2021; Niederfahrenhorst et al., 2023).

Code Listings 8 SFTTrainer Implementation

```
from trl import SFTTrainer

max_seq_length = 2048 # max sequence length for model and packing of the dataset

trainer = SFTTrainer(
    model=model,
    train_dataset=dataset,
    peft_config=peft_config,
    max_seq_length=max_seq_length,
    tokenizer=tokenizer,
    packing=True,
    formatting_func=format_instruction,
    args=args,
)
```

Source: Schmid, 2023

Now we can load the fine-tuned model with the QLoRA adapter from the previously initialized checkpoint (Brev.dev, 2024) and export it to Hugging Face to turn the model into a GGUF file format to run it locally if desired (Hugging Face, n.D.e).

To evaluate the base and the fine-tuned models we have a loop, in which we go through all the test questions. For each loop we run inference on the loaded LLM, giving it access to the tool as defined in Code Listings 1 and asking for chat completion, as exemplarily shown in Code Listings 9. In this setup for local inference, the model has to decide on it's on if it makes a function-call or not.

Code Listings 9 Running inference, showcased on Llama 2 base model

```
# Define the message structure for the LLM inference
message = [
    {"role": "system", "content": "You are a helpful assistant. Use functions if appropriate."},
    {"role": "user", "content": question}
]

# Run the LLM inference
result = llm.create_chat_completion(
    messages=message,
    functions=tools,
    tools=tools,
    tool_choice="auto",
    temperature=0.5,
    stop=['<</SYS>>']
)
```

Source: own creation, 2024

The code used to test the base models (Taoofstefan, 2024e) and to fine-tune and test the fine-tuned models (Taoofstefan, 2024f) can be found on GitHub.

4.3. Datasets

For the fine-tuning of LLMs, particularly for increasing function-calling capabilities, the selection and preparation of datasets are critical. This section describes the datasets created for this research to train Llama 2 and Mistral Instruct but also test the model, each chosen to target different aspects of the model's performance.

The dataset (Taoofstefan, 2024) is purpose-built for the specific function highlighted in Code Listings 1. This dataset is relatively small, containing 843 training entries and 100 test entries, designed to fine-tune the model's ability to handle the specific function-call 'get_current_weather'. The controlled size and focus of this dataset allow for a detailed analysis of the model's response accuracy in a constrained domain, facilitating focused improvements.

The specific training and test datasets include diverse prompt types to evaluate various model capabilities:

- General Text-Completion Questions: Test the model's basic language processing and generation capabilities.

- Nonsensical Prompts: Aim to assess the model's ability to handle illogical queries, evaluating its response robustness.
- Past Weather Questions: Focus on the model's capacity to deal with ambiguous historical weather inquiries.
- Future Weather Questions: Similar to past weather questions but oriented towards future conditions, testing the model's ability to predict based on learned data.
- Current Weather Questions: Specifically designed to test the model's API interaction capabilities for real-time data fetching.

Table 2 illustrates samples from each category, showing the diversity of queries and the expected precision in responses, particularly in how the model handles real-time data interaction through function-calls.

Table 2 Sample Data for Fine-Tuning and Testing Model

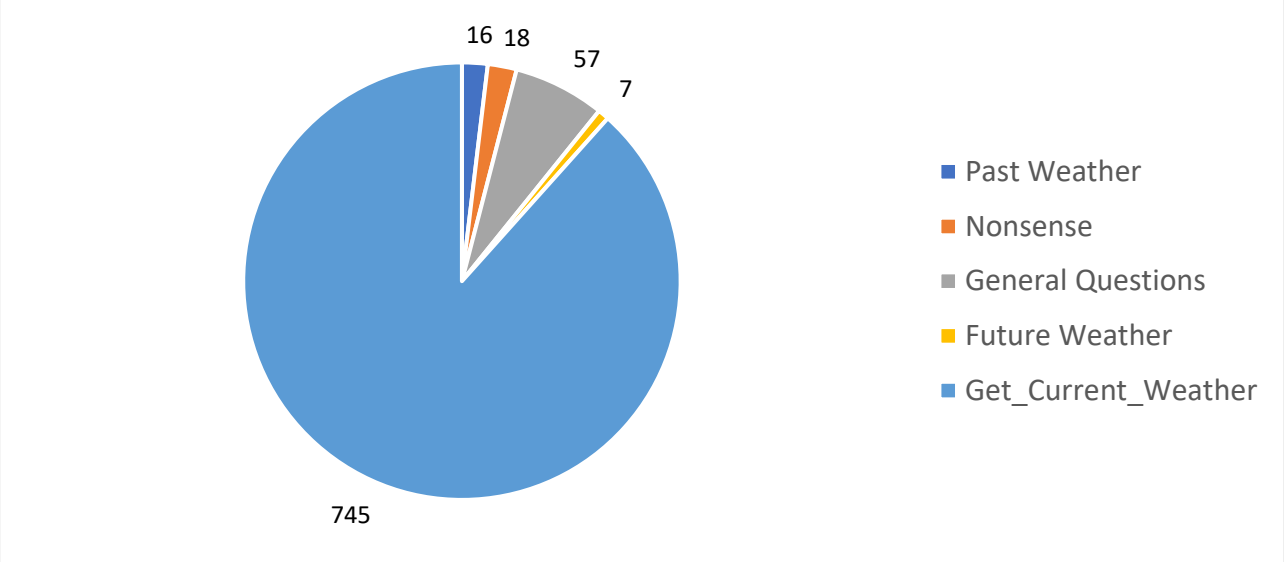
Use Case #	Question	Expected Reply
1	What does HTML stand for?	HTML stands for HyperText Markup Language.
	Who was the first President of the United States?	The first President of the United States was George Washington.
2	Snarfle	The gentle purring sound of a friendly alien pet.
	Glimblox	Tiny, luminescent creatures found in deep magical forests.
3	Was it windy last month in Marrakech?	I currently don't have real-time access to weather data or historical weather records. To find out specific weather conditions, checking a reliable meteorological service is advised.
	Has Addis Ababa been hotter than usual this spring?	Accurate and detailed historical weather data, such as temperature extremes or precipitation levels, can be accessed through specialized weather databases or meteorological agencies.
4	Will Seoul see an increase in air quality issues due to weather?	Weather advisories, including storm warnings or temperature rises, are issued by meteorological authorities. Check their updates regularly.
	Will it rain tomorrow in Istanbul?	Weather advisories, including storm warnings or temperature rises, are issued by meteorological authorities. Check their updates regularly.
5	Could you give me the current weather in Zurich?	{"type": "function", "function": {"name": "get_current_weather", "parameters": {"location": "Zurich, Switzerland", "format": "celsius"}}}

	May I have the weather forecast for Bangkok?	{“type": "function", "function": {"name": "get_current_weather", "parameters: {"location": "Bangkok, Thailand", "format": "celsius"}}}

Source: own creation, 2024

The dataset focuses heavily on the function 'get_current_weather' with a significant number of entries in both training, Figure 20, and testing, Figure 21, components. This distribution reflects a targeted approach, which is consistent with best practices in machine learning where specific tasks are prioritized to improve model performance in particular domains (Brown et al., 2020). The inclusion of diverse categories such as past weather, nonsense, general questions, and future weather in smaller quantities ensures the model's capability to handle a variety of inputs while maintaining a focus on its primary functional task.

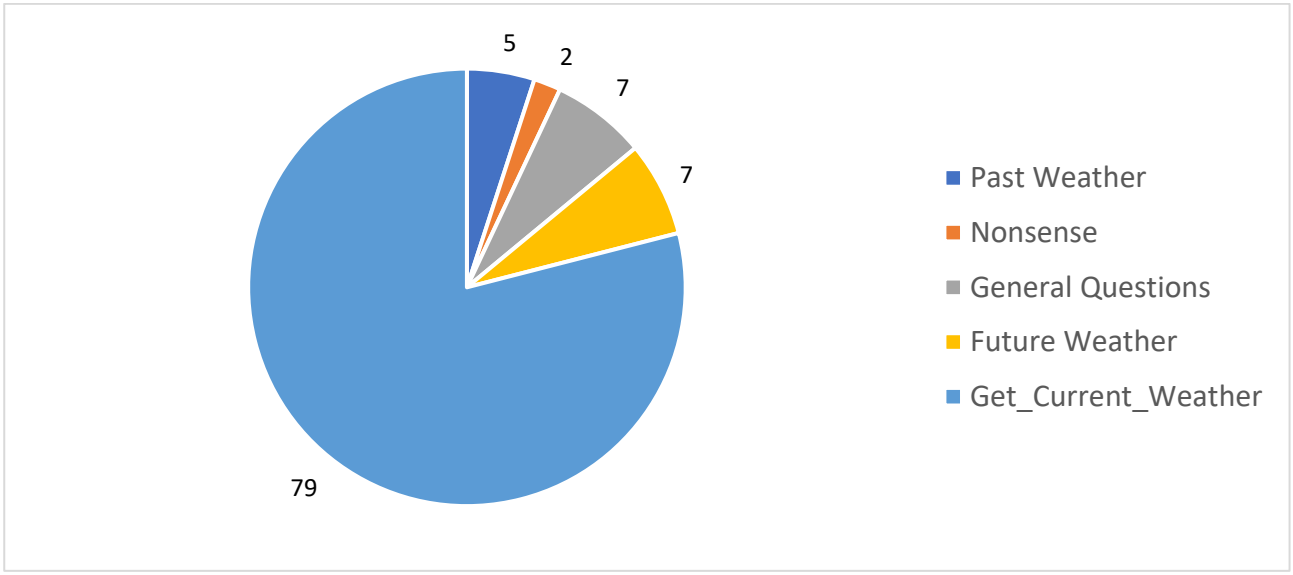
Figure 20 Training Data Distribution



Source: own creation, 2024

The concentration on 'get_current_weather' with 745 instances in training and 79 in testing aligns with empirical research strategies that recommend focused training on key functionalities to improve precision and reduce error rates in primary task domains (Hakhamaneshi & Ahmad, 2023). Such a distribution not only supports specialized learning but also facilitates the testing of generalizability and adaptability across different types of queries, which is critical for deploying robust, real-world applications (Kaddour et al., 2023).

Figure 21 Test Data Distribution



Source: own creation, 2024

This methodological approach, supported by literature, leverages the strengths of mixed data types to improve both specific task performance and overall model resilience, making it a strategic choice for developing high-performance LLMs capable of sophisticated function-calling in dynamic environments.

4.4. Fine Tuning Techniques

The development of fine-tuning methodologies for LLMs such as GPT-3, GPT-4, and Llama 2, has been marked by significant innovations aimed at improving model adaptability and performance across diverse tasks. Drawing from recent advancements and insights from seminal works, this section explores the comprehensive strategies employed in the fine-tuning process, emphasizing the most important techniques.

The essence of fine-tuning LLMs lies in the customization of the training process to suit the unique demands of the task. This involves the strategic adjustment of several key parameters within the training loop, including the learning rate, batch size, and the number of epochs (Niederfahrenhorst et al., 2023). The learning rate dictates the pace at which the model updates its weights in response to the error it observes, making it a critical factor in the model's ability to learn effectively. The batch size affects the model's generalization capabilities and its computational demands, while the choice of the number of epochs influences the depth of training and the potential for overfitting. By meticulously calibrating these parameters, researchers can significantly improve the model's performance, ensuring that it is optimally tuned to the specificities of the desired application (Li & Liang, 2021).

Intending to maximize the efficacy and efficiency of the fine-tuning process, the adoption of PEFT techniques has emerged as a cornerstone strategy (Houlsby et al., 2019). PEFT-based

methodologies, such as those proposed by Li & Liang (2021), concentrate on the judicious adaptation of models to new tasks with minimal adjustments to the pre-existing model parameters. This approach enables the preservation of the original model's strengths while incorporating new capabilities tailored to the task at hand. By doing so, PEFT not only circumvents the need for extensive retraining, which can be resource-intensive and time-consuming but also mitigates the risk of catastrophic forgetting, where a model loses its ability to perform on previous tasks following adaptation to new ones (Kwak and Kim, 2024). Consequently, PEFT represents a sophisticated balance between efficiency and efficacy in the model optimization process, making it a highly regarded strategy in the fine-tuning of LLMs.

The integration of PEFT methodologies, specifically as illustrated through the deployment of LoRA and QLoRA configurations in Code Listings 3 and Code Listings 6, represents a strategic implementation to optimize the fine-tuning process of LLMs. These techniques, grounded in the foundational works by Houlsby et al. (2019) and further elaborated by Li & Liang (2021), focus on minimal yet effective modifications to the model's parameters. This approach significantly improves the adaptability of LLMs to new tasks without extensive retraining, conserving resources, and retaining the pre-trained strengths of the model (Kwak & Kim, 2024). By employing PEFT, this research not only aligns with best practices in computational efficiency but also strategically mitigates the risk of catastrophic forgetting, a critical consideration in maintaining the long-term utility and effectiveness of LLMs in dynamic applications. This methodological choice underscores a commitment to leveraging advanced, resource-conscious strategies in the evolution of LLM capabilities, ensuring that the models remain robust and versatile across varied tasks.

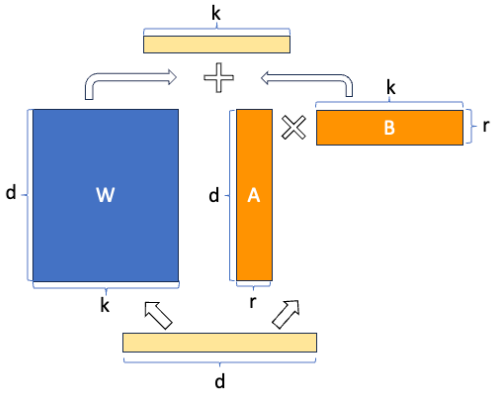
Gradient accumulation became an essential technique in the training of LLMs, particularly when dealing with the constraints of available hardware. This method involves accumulating the gradients over multiple forward passes before conducting a single update to the model's weights (Raj et al., 2024). This approach effectively simulates larger batch sizes than those that the hardware could normally handle (Raschka, 2023), thereby enabling the training of sophisticated models on limited computational resources (Y. Wang et al, 2024; Zhang et al., 2024). The principal advantage of gradient accumulation lies in its capacity to improve training stability and model performance without necessitating the procurement of more powerful hardware. By allowing for the utilization of larger batch sizes, it contributes to more stable gradient estimates and improved generalization of the model, making it a crucial strategy in the optimization of LLMs (Raschka, 2023).

Among the advanced fine-tuning techniques, LoRA stands out for its innovative approach to model optimization. LoRA introduces low-rank matrices into the model weights, enabling the model to undergo selective updates that specifically target the improvement of its capabilities concerning new tasks (Hu et al., 2021). This method retains the original structure and knowledge base of the model while seamlessly integrating new information. One of the key benefits of LoRA is its ability to maintain the efficiency of the model by limiting the number of parameters that need to be updated during the

fine-tuning process, thereby preserving computational resources, and reducing the risk of overfitting (Niederfahrenhorst et al., 2023), showcased in Figure 10. The selective update mechanism of LoRA ensures that the core competencies of the model are preserved, making it an effective strategy for incrementally building upon the model's existing strengths (Hu et al., 2021).

Utilizing the methodology proposed by Hu et al. (2021), LoRA considerably streamlines the computational complexity of matrix operations within the fine-tuning paradigm of deep learning models. By employing low-rank matrices denoted as A and B, as opposed to directly modifying the comprehensive weight matrix *W* inherent in pre-trained models, the fine-tuning process becomes computationally efficient (refer to Figure 22 for a visual depiction). The modification sequence, as explained by Jawade (2023), involves the exclusive updating of matrices A and B, thus significantly reducing the number of parameters requiring optimization during the fine-tuning phase. The refinement of outputs is accomplished through the computation of the matrix product $A \times B$, which is subsequently superimposed upon the pre-existing outputs produced by W. This resulting product effectively acts as an adaptation layer to the original weight matrix, thereby encapsulating the updates within a lower-dimensional space. The designation 'lower-dimensional space' is attributed to the dimension *k*, which is the rank of the low-rank matrices A and B introduced during the LoRA fine-tuning process. Although the dimensions d and r, corresponding to the original weight matrix W, retain their substantial magnitudes reflective of the extensive parameter space of pre-trained models, the dimension k is intentionally set to a smaller value. Consequently, the modifications encapsulated within the matrices A and B reside within a subspace of reduced rank, which is computationally less intensive to optimize, hence yielding a more efficient fine-tuning regimen while leveraging the existing structure and knowledge embedded in W (Jawade, 2023).

Figure 22 LoRA Update Mechanism for Efficient Fine-Tuning of Pre-Trained Models



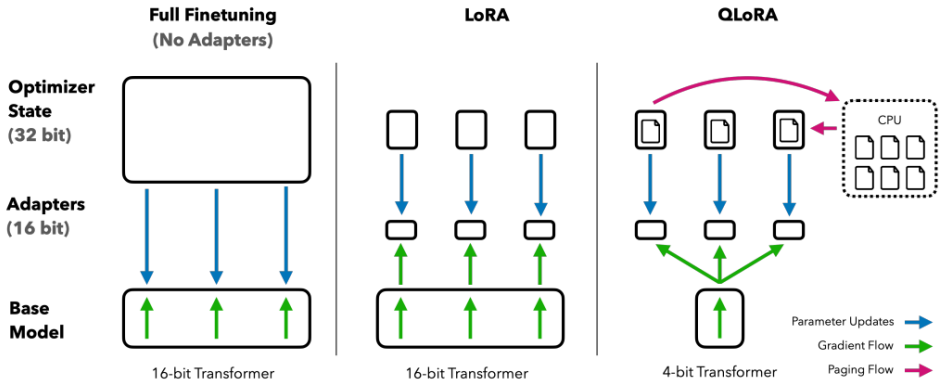
Source: Kuo, 2023

QLoRA stands as a significant improvement over LoRA. QLoRA integrates quantization techniques, substantially reducing the model's computational demand and memory requirements (Dettmers et al., 2023). This makes QLoRA particularly suitable for environments where computational resources

are limited. By quantizing model parameters, QLoRA allows for efficient memory usage while maintaining robust performance, aligning to deploy advanced models in resource-constrained settings (Dettmers et al., 2023). The superiority of QLoRA lies in its ability to preserve the high performance of LLMs, proving essential for the fine-tuning experiment where maintaining the balance between efficiency and effectiveness is crucial (Raschka, 2023a). This approach not only supports the scalability of LLM deployments but also improve their accessibility, making it a strategic choice for fine-tuning in this research.

The integration of quantization with low-rank adaptation offers a balanced approach to model optimization, ensuring that performance is maintained or even improved while achieving greater efficiency. By reducing the computational overhead, QLoRA facilitates the broader application of LLMs across various domains, democratizing access to state-of-the-art NLP capabilities (Dettmers et al., 2023). In Figure 23 different fine-tuning techniques are depicted in a simplified manner.

Figure 23 Full Finetuning, LoRA and QLoRA



Source: Dettmers et al., 2023, p. 3

Adapter modules represent a further advancement in the fine-tuning of LLMs, offering a path toward efficient adaptation without the need for extensive retraining or modification of the model's core parameters. These modules are small, task-specific neural networks that are inserted into the pre-existing architecture of pre-trained models (Houlsby et al., 2019), see Figure 12. The primary allure of adapter modules lies in their ability to facilitate the learning of new tasks or adapting to new domains by only adjusting these inserted components, rather than the entire model. This method significantly reduces the computational burden and resource requirements traditionally associated with fine-tuning, making it an attractive option for extending the versatility of LLMs (Hu et al., 2023; Houlsby et al., 2019). One of the key advantages of using adapter modules is their modularity, which allows for rapid experimentation and deployment across various tasks with minimal overhead. Additionally, because the original model parameters remain untouched, this approach greatly diminishes the risk of catastrophic forgetting, where a model loses its ability to perform previously learned tasks after being fine-tuned on new data. By preserving the integrity of the base model while

enabling task-specific adaptations, adapter modules offer a balanced approach to leveraging the capabilities of LLMs across a wider array of applications (Houlsby et al., 2019).

In the context of this research, while adapter modules represent a potent fine-tuning methodology, enabling specific, task-oriented adaptations without extensive modification to a model's core architecture (Houlsby et al., 2019), the decision to work with QLoRA was driven by significant computational efficiency considerations. The project's resource constraints, particularly the need to manage cloud-GPU costs effectively, made QLoRA a more suitable choice. QLoRA optimizes the low-rank matrices integral to the model's structure through quantization, reducing both the memory requirements and computational load significantly (Dettmers et al., 2023). This approach not only ensures that the fine-tuning process is less burdensome on the limited resources but also retains the model's performance efficacy, making it an ideal strategy in scenarios where economic and computational efficiency are paramount. Thus, despite the versatility offered by adapter modules, the specific needs and constraints of this project guided the selection towards QLoRA, aligning with our goal to achieve optimal results within a resource-constrained setting.

Complementing the use of adapter modules, prompt tuning emerges as a nuanced strategy for refining the interaction between LLMs and specific tasks or queries. Prompt tuning involves the careful engineering of prompts, which are inputs designed to guide the model's generation process toward desired outcomes. By crafting prompts that effectively communicate the task at hand or the context within which a response is sought, researchers can significantly improve a model's ability to interpret user queries accurately and generate responses that are not only relevant but also contextually appropriate (Y. Wang, et al., 2024; Lyu, et al., 2024). The art and science of prompt tuning rest on a deep understanding of the model's underlying mechanisms and how it processes inputs. This includes insights into the types of prompts that elicit the most accurate and useful responses from the model. Prompt tuning can range from simple adjustments, such as the rephrasing of questions or the inclusion of specific keywords, to more complex strategies like the use of exemplars or the structuring of prompts to mimic the format of the desired output. When applied judiciously, prompt tuning can dramatically improve the efficiency and effectiveness of LLMs in task performance, bridging the gap between generic pre-training and specialized application needs (Yang et al., 2024; Li and Liang, 2021).

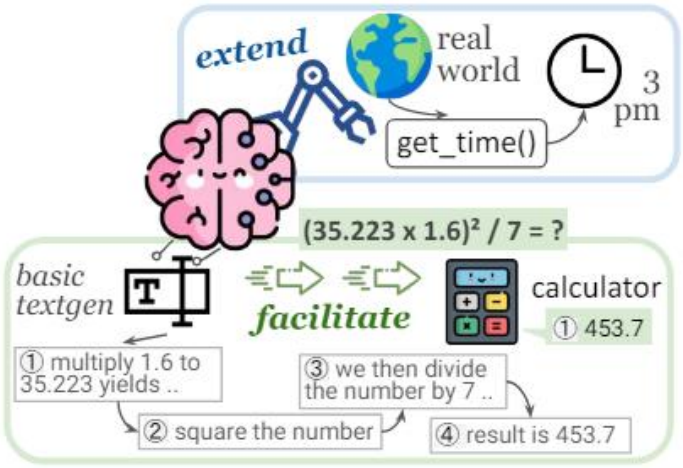
The integration of adapter modules and prompt tuning as suggested by Hu et al. (2023) represents a synergistic approach to fine-tuning that harnesses the strengths of both strategies. Adapter modules provide a structural basis for task-specific adaptation, while prompt tuning refines the model's interpretative and generative capabilities within the context of those tasks. Together, they enable a more nuanced and effective customization of LLMs, facilitating their application across a diverse range of domains and challenges. This combined approach underscores the evolving landscape of LLM fine-tuning, highlighting the innovative methodologies that continue to push the boundaries of what these powerful models can achieve (Hu et al., 2023).

While this research does not employ the combination of adapter modules and prompt tuning as suggested by Hu et al. (2023), it acknowledges the important role of prompt design in fine-tuning LLMs, particularly in maintaining alignment with intended functions and behaviors. Recent studies, such as the findings from Qi et al. (2023), underscore the significance of prompt structures in mitigating risks associated with misalignment during the fine-tuning process (Qi et al., 2023).

The dataset from Taoofstefan (2024), specifically designed for the 'get_current_weather' function, illustrates this strategy. By carefully creating prompts that guide the LLM towards precise function-calling, the dataset ideally allows the fine-tuning process to maximize the LLM's performance in generating accurate and contextually appropriate responses. This approach is reflected in the structured prompts within the dataset, which are formulated to elicit specific responses, thereby harnessing the full potential of the LLM in practical applications (see Table 2). The careful design of these prompts aligns with current research advocating for the strategic use of prompt engineering to maintain the integrity and functionality of LLMs post-fine-tuning (Lyu et al., 2024).

Tool augmentation significantly increases the functional scope of LLMs by integrating external tools or knowledge sources, typically through APIs. This strategic incorporation enables LLMs to interact with and retrieve data from external sources in real-time, thus allowing them to respond to queries with up-to-date information or execute tasks that depend on current data (Wang et al., 2024). The utility of this approach is particularly evident in domains where the relevancy of information changes rapidly, such as in weather forecasting, financial market analysis, or any area where dynamic data is crucial for accurate responses or analyses (Li, et al., 2023). For example, an LLM equipped with tool augmentation capabilities could access the latest weather data from meteorological APIs to provide real-time weather updates or query financial databases to offer the most current stock market trends and predictions. This capability represents a significant evolution from the traditional use of LLMs, which relied predominantly on static datasets compiled during the pre-training phase. By leveraging live data, LLMs can provide more accurate and contextually relevant outputs, increasing their utility and effectiveness across a broad spectrum of applications (Wang et al., 2024). Figure 24 gives an illustration of how tools can potentially extend different language model tasks.

Figure 24 Illustration of tools extending and facilitating LM task-solving



Source: Wang et al., 2024, p. 1

Incorporating the function-call feature within the tool augmentation strategy exemplifies a practical application of extending LLM capabilities through external APIs. Specifically, the fine-tuning process using the 'get_current_weather' function aligns with the methodology of tool augmentation. This function enables the LLM to fetch and integrate real-time weather data directly from external sources, thereby improving the model's ability to provide timely and contextually relevant responses. This application not only demonstrates the utility of LLMs in accessing dynamic external data but also underscores the model's adaptability to specific operational needs, which is critical in domains where up-to-date information is essential for accuracy and relevance (Wang et al., 2024; Li et al., 2023). This strategic improvement through tool augmentation facilitates a broader functional scope, enabling the LLM to perform complex tasks that rely on current data, thus significantly increasing its practical utility across various applications.

In combination with tool augmentation, the application of advanced learning strategies such as zero-shot (Li et al., 2024), one-shot, and few-shot (Brown et al., 2020) learning further amplifies the adaptability and efficiency of LLMs. These strategies enable models to understand and perform tasks even with minimal or no task-specific training data. Zero-shot learning allows LLMs to generalize to tasks without any specific examples or training on those tasks. The model uses its pre-existing knowledge and understanding to infer the correct output for a completely new task (Li et al., 2024). One-shot learning involves providing the model with a single example or instruction from which it learns and generalizes the task, adapting its responses based on this limited input. Few-shot learning extends this concept by supplying a small number of examples or instructions, which significantly improve the model's ability to accurately perform the task (Brown et al., 2020).

In comparing these fine-tuning methods, it is evident that each offers distinct advantages and limitations. PEFT and LoRA are notable for their computational efficiency and ability to preserve the model's integrity, while QLoRA extends these benefits to scenarios with stringent resource

constraints. Adapter modules provide a highly flexible fine-tuning approach, suitable for a wide range of applications. In contrast, prompt tuning stands out for its simplicity and effectiveness in improving model performance without extensive retraining.

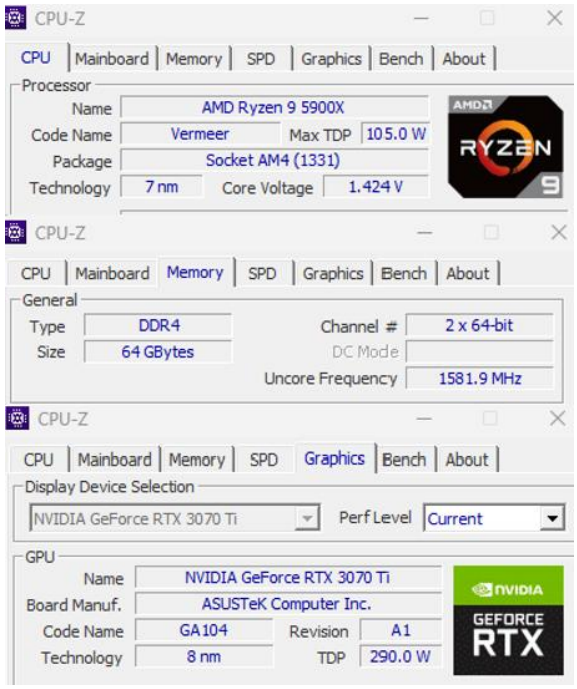
4.5. Computational Resources and Tools

This section provides an overview of the computational resources and tools utilized for the fine-tuning and inference phases of the LLM training. To meet the intensive computational demands of training LLMs, this research leverages cloud-based GPUs provided by Brev.dev. The service offers a scalable, pay-per-minute GPU computing solution, which not only ensures cost-efficiency but also provides the necessary computational power required for extensive model training (Brev.dev, 2024).

Brev.dev has kindly supported the creation of this thesis by sponsoring the fine-tuning process on their platform.

For the initial testing of the base models and the OpenAI GPT, local computing resources are employed to manage the computational load effectively. The setup includes an AMD Ryzen 9 5900X processor and an NVIDIA GeForce RTX 3070 TI equipped with CUDA technology. This configuration supports faster inference processes due to its robust processing capabilities and substantial memory of 64GB DDR4 RAM (see Figure 25). The use of this hardware locally facilitates quick and efficient handling of LLM operations post-training, catering to the need for rapid response and data processing in real-time applications. For the sake of efficiency, the fine-tuned models will be directly tested via the already deployed GPU-computing solution.

Figure 25 Computing resources for local testing



Source: own creation, 2024

The selection of cloud-based GPUs for training and high-performance local hardware for inference strategically addresses the dual requirements of cost-efficiency and computational adequacy. This approach not only optimizes the financial investment in computational resources but also ensures that the infrastructure is well-suited to support the advanced computational needs of deploying LLMs in practical scenarios (Eassa, 2022).

The methodology for this research incorporates Python and Jupyter Notebook, predominantly due to their widespread acceptance and utility in scientific computing and data analysis within academic and development circles (Capelle, 2024a; Das, 2024; Geronimo, 2023). This preference is supported by many academic papers and blog posts that highlight Python's extensive library ecosystem (Nagpal & Gabrani, 2019) and Jupyter's interactive computing environment, making them indispensable tools for AI research (Raffel et al., 2020; Brown et al., 2020).

For local testing and running inference, the research employs “llama-cpp-python” (Llama-cpp-python, n. D.), a Python wrapper for the “llama.cpp” library (Llama.cpp, n. D.), which is known for its performance and ease of use in Windows environments. This choice is strategic, leveraging llama-cpp-python's compatibility and efficiency to ensure reliable local execution of the LLMs, thereby facilitating rapid iteration and testing without the need for extensive computational infrastructure.

This combination of advanced computational tools and platforms is selected to optimize the research process, ensuring that the computational demands of training and testing LLMs are met efficiently and effectively.

4.6. Data Analysis Methods

This section will further outline the chosen evaluation metrics introduced in chapter 3.1 and explain their alignment with the overarching research goals:

- Accuracy: Measures the LLM’s ability to correctly identify user intentions, which is essential for determining whether a query should trigger a function-call or a textual response. This metric directly correlates to the research objective of increasing user-interaction effectiveness through precise function-call activation (Hakhamaneshi & Ahmad, 2023; fireworks.ai, 2023).
- Precision: Assesses the precision with which the LLM constructs function-calls, ensuring they meet the specified requirements. This is crucial for integrating LLM outputs with real-world applications, thereby supporting the objective of practical deployment of LLMs in API-driven environments (Roberts, 2023b; Raffel et al., 2020).
- TCE: Evaluates the coherence and context-appropriateness of the LLM’s textual responses, reinforcing the objective of developing LLMs capable of maintaining engaging and relevant conversations (Roberts, 2023b; Liu et al., 2020).

The selected metrics not only facilitate a comprehensive evaluation of the LLM's function-calling performance but also enable a nuanced interpretation of how well the model meets the practical requirements stipulated by the research objectives. By measuring accuracy, precision, and TCE, this research can:

- Quantitatively assess the precision and reliability of the LLM in real-world settings.
- Identify areas where the LLM excels or requires further optimization, guiding subsequent refinements in model training and setup.
- Provide empirical evidence to support the efficacy of the chosen fine-tuning approaches discussed in earlier sections, thereby validating the methodological framework of the study.

For the evaluation of the models in chapter 5.2 Presentation and Interpretation of Experimental Findings, the metrics will be evaluated as follows:

- Accuracy: The proportion of correct decisions made by the model in determining whether a function-call should occur, or a chat completion is required. It is a critical measure of the model's decision-making capability.
- Precision: Specifically evaluated when the model decides to perform a function-call, this metric measures the correctness of the structured output. It is an important indicator of the model's ability to generate accurate and usable structured data.
- TCE: Assesses the helpfulness and meaningfulness of the model's reply when a chat completion is required. This metric evaluates the quality and coherence of the text generated by the model.

This comprehensive analysis approach ensures that the results are not only statistically valid but also meaningfully aligned with the practical demands and expectations for modern LLMs in varied application contexts.

4.7. Ethical Considerations

This research includes several ethical considerations concerning the use of artificial intelligence and data handling. Ethical engagement with AI involves addressing both the potential benefits and risks associated with deploying models like Llama 2, which has been openly released to foster responsible AI innovation and ensure broad access to AI advancements (Touvron et al., 2024).

The use of LLMs, particularly in sensitive applications, can lead to challenges such as non-factual generations, unintended biases, and the propagation of harmful content due to training on diverse internet-sourced datasets (Touvron et al., 2024). These issues are exacerbated when models like Llama 2, which were initially trained predominantly on English data, are applied in multilingual contexts. This limitation can lead to suboptimal performance and inadvertent cultural insensitivity in non-English languages, posing significant ethical concerns (Touvron et al., 2024).

To mitigate these risks, several strategies have been adopted:

- Continual Fine-Tuning: Efforts to update and refine AI models continually are crucial. While Llama 2 was fine-tuned to reduce harmful outputs, continuous updates are necessary to address emergent issues effectively (Touvron et al., 2024).
- Diverse Data Set: The dataset (Taoofstefan, 2024) was specifically designed to include diverse geographical contexts, aiming to reduce regional biases and improve the model's global applicability. By incorporating data that represents various global regions, this approach strives to improve the fairness and inclusivity of the AI application.
- Ethical Use Guidelines: Following ethical guidelines and promoting responsible AI usage are imperative. The release of Llama 2 with an open license is intended to democratize AI technology. However, it is accompanied by a Responsible Use Guide to encourage ethical application and mitigate misuse (Touvron et al., 2020).

Adopting an open science approach (UNESCO, 2023), the research leverages the collective expertise of the global AI community to increase model safety and effectiveness. By sharing findings and methodologies openly, the research not only advances scientific knowledge but also contributes to the ethical development of AI technologies. This collaborative approach is essential in addressing complex ethical issues associated with AI, ensuring that advancements are both innovative and responsibly implemented (Spirling, 2023).

These ethical considerations are integral to the research methodology, ensuring that the study not only advances technical understanding but also contributes positively to the broader societal impacts of AI technology.

4.8. Methodological Critique

This section provides a critical evaluation of the methodologies and technologies applied throughout the research, with an emphasis on identifying potential limitations and biases inherent in the tools and methods used.

The fine-tuning techniques, primarily PEFT, LoRA and QLoRA, while advanced, present limitations that may impact the generalizability and robustness of the findings. One significant concern is that these methods might not fully capture the complex, real-world scenarios in which LLMs are deployed, potentially leading to overfitting the specific tasks or datasets used (Houlsby et al., 2019; Li & Liang, 2021). Furthermore, the reliance on cloud-based computational resources, while increasing processing power, introduces potential biases related to the specific architectures and optimization strategies inherent in these platforms (Wang et al., 2024).

The methodologies employed, though rigorous, are subject to limitations that may affect the validity and reliability of the research outcomes. For instance, the use of LLMs trained primarily on English data may not adequately represent linguistic diversity, impacting the applicability of the findings

across different linguistic contexts (Touvron et al., 2020). Additionally, adaptation methods like QLoRA, while efficient, may introduce subtle alterations in model behavior that are not immediately apparent, posing challenges in fully understanding the impact of quantization on the nuanced capabilities of the models (Dettmers et al., 2023).

These methodological constraints necessitate cautious interpretation of the results, particularly in terms of their applicability to broader AI applications. Researchers must consider these limitations when extrapolating the findings to different contexts or datasets. The ongoing development and refinement of fine-tuning techniques and computational frameworks are crucial to improving the robustness, reliability, and fairness of LLM applications in diverse real-world scenarios.

This critique underscores the necessity for continuous methodological improvement and critical assessment of AI technologies to ensure that they meet the evolving standards of research rigor and ethical AI deployment.

5. Research Findings

5.1. Presentation and Interpretation of Literature Findings

5.1.1. Presentation of Literature Findings

Fine-tuning significantly increases the performance of LLMs on specific tasks such as function-calling and factual accuracy. For instance, the Berkeley Function-calling Leaderboard (BFCL) demonstrates substantial performance gains for models like GPT-4 and Mistral-7B following fine-tuning. Figure 9 from the BFCL clearly shows that GPT-4, after fine-tuning, leads in function-calling accuracy with a performance score of 93%, while fine-tuned models like Mistral-7B and Llama-2-70B achieve scores of 81.5% and 81.0%, respectively. This underscores the effectiveness of fine-tuning in improving model precision and reliability (Yan, et al., 2024).

While fine-tuning improves task accuracy, it often comes with higher computational costs during both training and inference phases. As highlighted by Z. Wang et al. (2024) in their paper, models such as ToolFormer show a significant increase in performance (e.g., +30.4 points on math tasks) but also require substantial computational resources (e.g., 3,864.2 million tokens for training). This trade-off is critical for practitioners to consider, balancing between the benefits of improved task performance and the resource demands (Z. Wang, et al., 2024). A detailed overview can be found in Figure 26.

Figure 26 Performance gain and learning costs for different methods, tools, and datasets

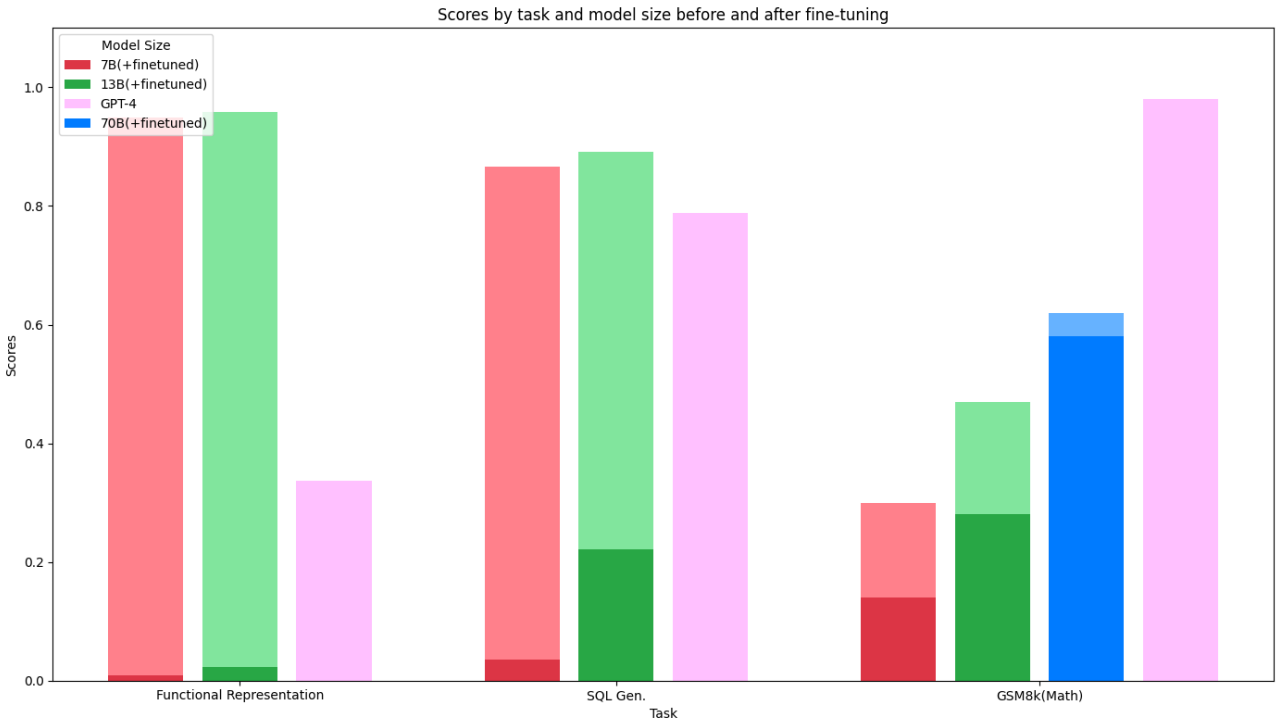
Type	Method	Task	Δ Perf.	# Params (B)	# Tokens (M) train	# Tokens (M) test
tool use	ToolFormer	cloze	+ 14.7	6.7	642.1	269.0
		math	+ 30.4	6.7	3864.2	421.0
		QA	+ 5.8	6.7	1101.2	189.0
		multilingual	- 0.2	6.7	606.0	274.0
		temporal	+ 13.0	6.7	508.8	202.0
	API-Bank	API	+ 24.4	7	**190414.6**	0.0
	ToolAlpaca	API	+ 45.2	7	**241889.3**	0.0
	Chameleon	science	+ 2.6	-	0.0	88.3
		table	+ 1.9	-	0.0	325.9
tool making	LATM	BigBench	+ 29.1	-	28.5	4720.0
	CREATOR	math	+ 4.5	-	0.0	5113.6
		table	+ 0.0	-	0.0	**6827.6**
	CRAFT	math	+ 13.2	-	4126.6	4098.5
		table	+ 17.2	-	2750.6	5018.2
	TroVE	math	+ 21.0	-	0.0	1825.2
		table	+ 12.0	-	0.0	1358.8

Source: Z. Wang, et al., 2024, p. 10

Hakhamaneshi & Ahmad (2023) go on to show that the scalability of fine-tuning approaches varies with the size of the models. Fine-tuning smaller models, such as Llama-2-7B and Llama-2-13B, results in significant performance improvements while being more cost-effective compared to larger models like GPT-4. Hakhamaneshi & Ahmad (2023) demonstrate that fine-tuning smaller models

can yield performance that surpasses even larger models on specific tasks. For instance, as highlighted in Figure 27, Llama-2-13B's accuracy improved from 58% to 98% on functional representations and from 42% to 89% on SQL generation after fine-tuning (Hakhamaneshi & Ahmad, 2023). This evidence supports the idea that fine-tuning is an efficient and scalable approach, particularly for improving performance in targeted application domains.

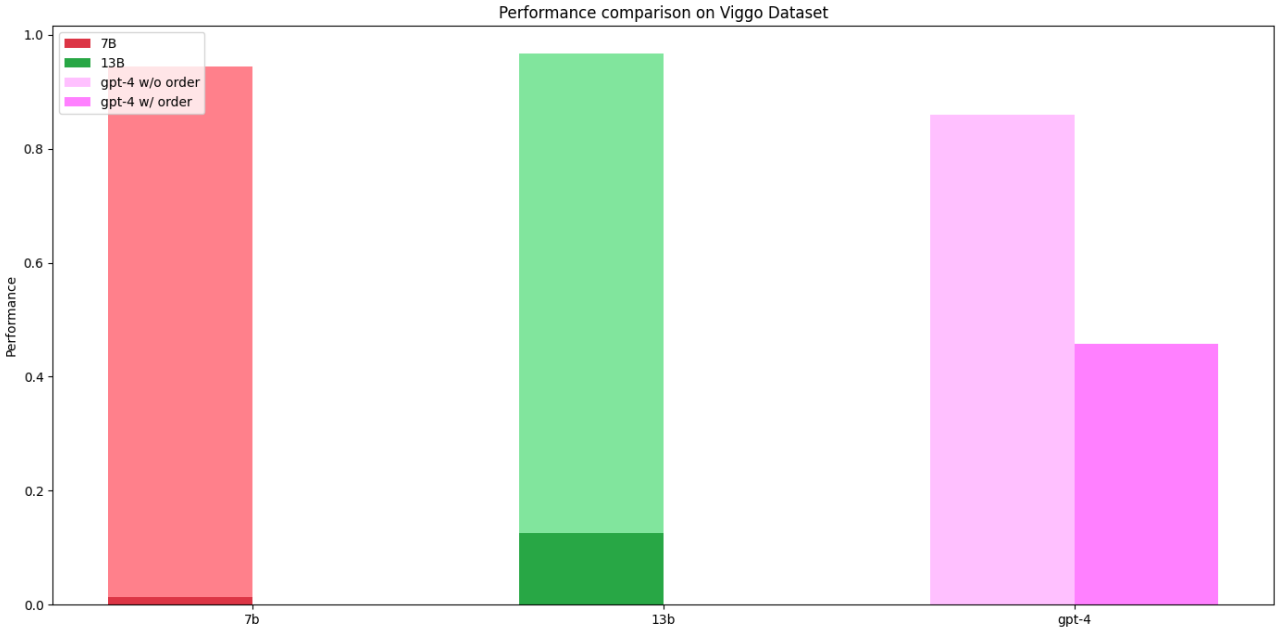
Figure 27 Performance Improvements of Llama-2 Models on SQL Generation, Functional Representation, and Math Tasks



Hakhamaneshi & Ahmad, 2023

Fine-tuning increases the adaptability of LLMs to specific datasets and tasks, as evidenced by performance comparisons across different model configurations. Data from Roberts (2023b) and Hakhamaneshi & Ahmad (2023) show that fine-tuned models not only improve in accuracy but also adapt better to the quirks and specific requirements of various datasets. For instance, fine-tuned models exhibited a 90% success rate in structured tasks such as those evaluated in the ViGGO dataset, maintaining high accuracy even under constraints like attribute precedence (Hakhamaneshi & Ahmad, 2023). Hakhamaneshi & Ahmad (2023) further show in their work, that when considering attribute precedence, GPT-4's accuracy declines substantially, whereas the fine-tuned models consistently maintain high accuracy, demonstrating their robustness under this evaluation constraint (see Figure 28). The ViGGO dataset underscores the advantages of fine-tuning, particularly for tasks requiring structured output. Fine-tuning ensures that models can not only determine the necessity of arguments but also adhere to their correct order, surpassing the capabilities of simple regex or JSON format matching (Hakhamaneshi & Ahmad, 2023).

Figure 28 Performance Improvements on the ViGGO dataset

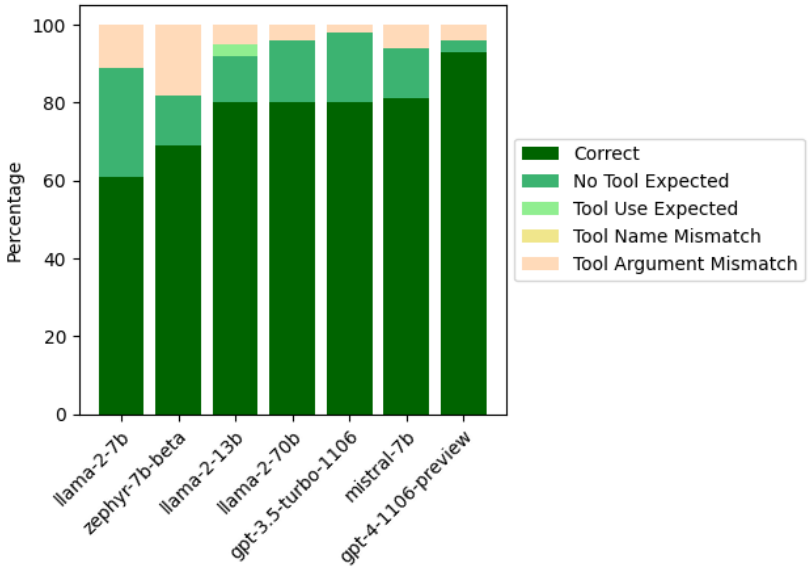


Hakhamaneshi & Ahmad, 2023

The integration of structured output like JSON, and function-calling capabilities into LLMs significantly improves their utility, particularly in structured tasks (Endpoints Team, 2023). Having a particular JSON mode makes sure that LLM outputs follow proper JSON formats that are designed to meet certain schema needs. Function-calling empowers LLMs to use APIs effectively, selecting the right function and arguments for a given task. The implementation of JSON mode and function-calling capabilities in models like Mistral-7B highlights their potential in enabling function-calling on open models (Endpoints Team, 2023).

The Endpoints Team (2023) reveals in their evaluation of function-calling across various models significant insights. Figure 29 presents the success and error rate breakdown for models like Llama-2-7B, Zephyr-7B-beta, Llama-2-13B, Llama-2-70B, GPT-3.5-turbo-1106, Mistral-7B, and GPT-4-1106-preview. This evaluation shows that Llama-2 13B, 70B, and Mistral-7B models perform comparably to GPT-3.5-turbo in single-turn function-calling without specialized training. Notably, Mistral-7B demonstrates better accuracy in distinguishing scenarios where a function-call isn't necessary. It then chooses to use a normal response instead (Endpoints Team, 2023).

Figure 29 Succes and Error Rate Breakdown of Function-Calling



Source: Endpoints Team, 2023

In their tests Fireworks.ai (2023) introduced a function-calling model based on a fine-tuned CodeLlama-34B, showcasing its potential in handling complex tasks involving external API calls. The performance evaluation across different models highlights the effectiveness of the fine-tuned CodeLlama-34B model, particularly in comparison to other models like GPT-4. As shown in the data, the Fireworks fine-tuned CodeLlama-34B model outperforms the standard CodeLlama-34B-Instruct in both single-turn and multi-turn interactions (see Table 3). Specifically, the fine-tuned model achieves a 0.58 accuracy in single-turn scenarios and a 0.90 accuracy in multi-turn scenarios, compared to 0.44 and 0.45, respectively, for the standard model. This indicates the fine-tuned model's superior capability in maintaining context and accuracy across multiple turns.

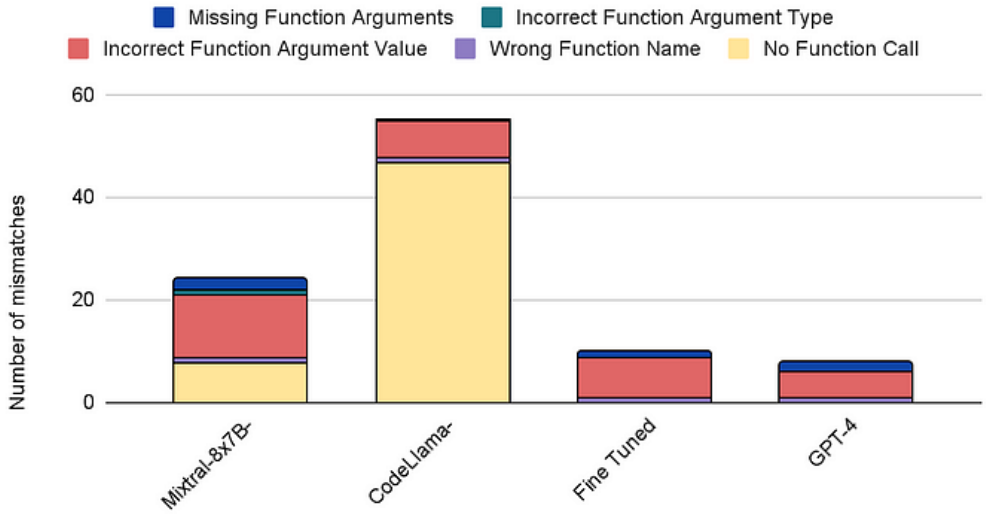
Table 3 Breakdown across 2 datasets with accuracy

Dataset Name	Number of Functions	Fireworks Fine Tuned CodeLlama-34B	CodeLlama-34B -Instruct	Mixtral-8x7B -Instruct	GPT-4
Single Turn	13	0.58	0.44	0.43	0.50
Multi Turn	1 to 5	0.9	0.45	0.76	0.92

Source: Fireworks.ai, 2023

They further report the mismatch breakdown for multi-turn scenarios (Figure 30) which reveals that the fine-tuned CodeLlama-34B model exhibits fewer errors compared to other models, particularly in categories such as missing function arguments and incorrect function argument types. The fine-tuned model shows a balanced performance with fewer total mismatches, indicating its robustness in managing complex interactions and maintaining accurate function-calls.

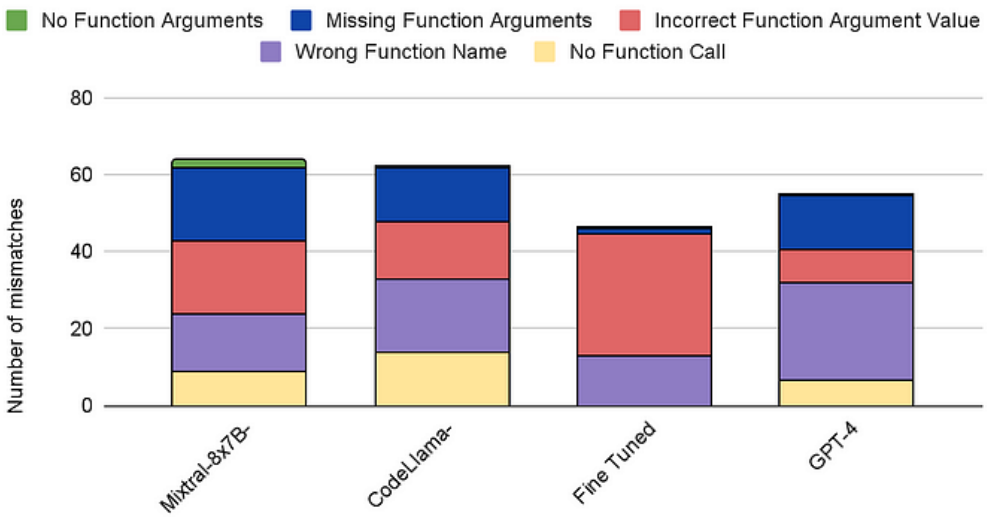
Figure 30 Mismatch Breakdown – Multi-Turn



Source: Fireworks.ai, 2023

Similarly, in single-turn scenarios (Figure 31), the fine-tuned CodeLlama-34B model demonstrates a lower rate of errors related to missing function arguments and incorrect argument values. This performance underscores the model's effectiveness in accurately interpreting and executing function-calls even in straightforward, single-turn interactions (Fireworks.ai, 2023).

Figure 31 Mismatch Breakdown – Single Turn



Source: Fireworks.ai, 2023

Further analysis (Fireworks.ai, 2023) of the mean scores and win rates shows that the fine-tuned CodeLlama-34B model achieves a mean score of 7.20 with a standard deviation of 2.7, and a win rate of 48%, compared to GPT-4's mean score of 6.3 with a standard deviation of 2.9, and a win rate of 40%. This performance metric highlights the fine-tuned model's ability to consistently perform well in function-calling tasks, often outperforming GPT-4 in accuracy and reliability (Fireworks.ai, 2023).

Overall, the findings from Fireworks.ai (2023) emphasize the importance and effectiveness of fine-tuning LLMs for function-calling tasks. The fine-tuned CodeLlama-34B model's superior performance in handling multi-turn interactions and maintaining high accuracy in function-calls positions it as a powerful tool for applications requiring dynamic and real-time data integration.

Raschka (2023) shows that fine-tuning LLMs introduces significant latency and cost considerations, impacting their practical deployment. For example, QLoRA provides substantial memory savings, reducing memory usage by up to 6 GB. However, as seen in Table 4, this comes with a trade-off of approximately 30% slower training times due to the added quantization and dequantization steps (Raschka, 2023). This trade-off between memory efficiency and increased processing time underscores the need to balance computational efficiency with performance gains in real-world applications.

Table 4 Trade-offs between memory usage and training time when using QLoRA

Model	Training Time	Memory Used
Default LoRA (with bfloat-16)	6685.75s	21.33 GB
QLoRA via –-quantize “bnb.nf4”	10059.53s	14.18 GB
QLoRA via –quantize “bnb.fp4”	9334.45s	14.19 GB

Source: Adapted from: Raschka, 2023

The findings of Raschka (2023) on QLoRA, as illustrated in Table 4, highlight its impact on training time, memory usage, and model performance. Implementing QLoRA results in notable memory savings, making it feasible to run large models like Llama-2-7B on smaller GPUs. Figure 32 shows that quantizing the model has a minor performance decline in benchmarks such as MMLU Global Facts but an improvement in arithmetic benchmarks. This trade-off is acceptable, considering the substantial memory savings that offset the increased training time by around 30%. Therefore, QLoRA provides a balanced approach, maintaining overall model performance while reducing computational resource requirements (Raschka, 2023).

Figure 32 Impact of QLoRA on Model Performance

	TruthfulQA MC1	TruthfulQA MC2	Arithmetic 2ds	Arithmetic 4ds	BLiMP Causative	MMLU Global Facts
Llama 2 7B base	0.2534	0.3967	0.508	0.637	0.787	0.32
LoRA default 1	0.2876	0.4211	0.3555	0.0035	0.75	0.27
LoRA default 2	0.284	0.4217	0.369	0.004	0.747	0.26
LoRA default 3	0.2815	0.4206	0.372	0.004	0.747	0.27
QLoRA (nf4)	0.2803	0.4139	0.4225	0.006	0.783	0.23
QLoRA (fp4)	0.2742	0.4047	0.5295	0.016	0.744	0.23

Source: Raschka, 2023

Raffel et al. (2020) highlight the balance between task-specific performance gains and potential decreases in generalization due to fine-tuning in their exploration of the T5 model. They found that while fine-tuning improves task accuracy, it may lead to overfitting, limiting performance on unseen tasks. Similarly, it was demonstrated that fine-tuning the BERT model increases specific tasks like biomedical text mining (Lee et al., 2020) or improves performances for scientific tasks but these improvements can vary significantly over different domains (Beltagy et al., 2019). Brown et al. (2020) show that while GPT-3 generalizes well in few-shot settings, extensive fine-tuning can cause overfitting, leading to exceptional performance on specific tasks but less effectiveness on others. This underscores the importance of balancing specialization and generalization to maintain broad applicability and avoid overfitting (Raffel et al., 2020; Lee et al., 2020; 2019; Beltagy et al., 2019; Brown et al., 2020).

Fine-tuning LLMs raises significant ethical and bias concerns, particularly when models are trained on niche or skewed datasets. These issues are amplified due to the models' sensitivity to the specific data they are fine-tuned on. For instance, the Llama 2 paper by Touvron et al. (2023) identifies that fine-tuning predominantly on English-language data can lead to performance fragility in other languages, thus perpetuating linguistic biases (Touvron et al., 2023).

Ethical considerations also extend to the potential misuse of fine-tuned models. As highlighted in the Llama 2 paper (Touvron et al., 2023), there is a risk of these models being employed for malicious purposes, such as generating misinformation or aiding in cybercrime. This underscores the need for robust safeguards and continuous monitoring to mitigate such risks (Touvron et al., 2023).

The performance evaluation of Llama 2 models, as shared by Touvron et al. (2023) reveals key insights into their effectiveness and limitations. Larger Llama 2 variants, like the 70B model, show significant improvements in factual accuracy with a TruthfulQA score of 50.18 (see Figure 33), the highest among evaluated models. However, ToxiGen scores, which measure the generation of toxic content, present mixed results. The 7B model shows better performance in mitigating toxic content (ToxiGen score of 21.25), while larger models like the 13B and 70B have higher scores, indicating a trade-off between model size and ethical output. Fine-tuning improves factual accuracy but necessitates continuous efforts to address biases, especially in larger models. These findings highlight the balance needed between performance and ethical considerations for practical deployments of Llama 2 models (Touvron et al., 2023).

Figure 33 Evaluation of pre-trained LLMs on automatic safety benchmarks

		TruthfulQA ↑	ToxiGen ↓
MPT	7B	29.13	22.32
	30B	35.25	22.61
Falcon	7B	25.95	**14.53**
	40B	40.39	23.44
Llama 1	7B	27.42	23.00
	13B	41.74	23.08
	33B	44.19	22.57
	65B	48.71	21.77
Llama 2	7B	33.29	21.25
	13B	41.86	26.10
	34B	43.45	21.19
	70B	**50.18**	24.60

Source: Touvron et al., 2023, p. 23

Bender et al. (2021) highlight the risks of amplifying biases inherent in training data when deploying LLMs. They argue that these models can inadvertently reinforce and propagate harmful biases present in publicly available datasets. Such biases can have serious ethical repercussions, particularly when LLMs are used in sensitive applications like healthcare, law, and finance. The authors emphasize the need for careful consideration and mitigation strategies to prevent biased and unethical outcomes from LLM deployments.

A detailed examination of bias in fine-tuned models is illustrated by the performance of the TruthfulQA and MMLU Global Facts benchmarks (see Figure 32). The QLoRA fine-tuning approach, despite providing memory savings and reducing training costs, shows a slight decline in performance on the MMLU Global Facts benchmark, indicating a potential trade-off between efficiency and maintaining unbiased, accurate outputs (Raschka, 2023).

5.1.2. Interpretation of Literature Findings

The findings from the literature review and results presented in chapter 5.1.1 underscore the critical role of fine-tuning in increasing the performance of LLMs across various tasks, particularly function-calling and factual accuracy. This section interprets these findings by linking them to the theoretical foundations and empirical evidence discussed earlier in the thesis.

Fine-tuning has been demonstrated to significantly improve the performance of models such as GPT-4, Mistral-7B, and Llama-2-70B in function-calling tasks, as evidenced by their high scores on the BFCL (Yan et al., 2024). The superior performance of fine-tuned models highlights the effectiveness of this approach in refining model precision and reliability. This aligns with the theoretical premise that fine-tuning allows models to adapt better to specific tasks by adjusting parameters to optimize performance (Li & Liang, 2021).

However, the trade-off between improved task accuracy and increased computational costs, as highlighted by Z. Wang et al. (2024), necessitates a careful consideration of resource allocation. The

substantial computational demands associated with fine-tuning underscore the importance of balancing performance gains with resource efficiency. This finding supports the notion that while fine-tuning can significantly improve performance, it also requires substantial investment in computational resources, which must be justified by the performance improvements achieved (Z. Wang et al., 2024).

The scalability of fine-tuning approaches, as discussed by Hakhamaneshi & Ahmad (2023), further illustrates the nuanced benefits of this technique. Fine-tuning smaller models, such as Llama-2-7B and Llama-2-13B, results in considerable performance improvements while being more cost-effective compared to larger models like GPT-4. This evidence suggests that fine-tuning can be an efficient and scalable approach, particularly for specific, targeted applications. The improved performance of smaller models on tasks like SQL generation and functional representations, as shown in Figure 27, demonstrates that fine-tuning can enable smaller models to outperform larger, less specialized models in specific domains (Hakhamaneshi & Ahmad, 2023).

The results from Fireworks.ai (2023) further reinforce the benefits of fine-tuning. The fine-tuned CodeLlama-34B model exhibited superior accuracy in both single-turn and multi-turn function-calling tasks compared to its non-fine-tuned counterpart and even GPT-4 in certain scenarios. This performance is attributed to the model's improved ability to maintain context and execute function-calls accurately, highlighting the importance of fine-tuning in improving model reliability and effectiveness in complex interactions (Fireworks.ai, 2023).

Moreover, the success and error rate breakdowns presented by the Endpoints Team (2023) and Fireworks.ai (2023) reveal that fine-tuned models like Mistral-7B and CodeLlama-34B handle function-calling tasks with greater precision and fewer errors compared to non-fine-tuned models. These findings emphasize the robustness of fine-tuned models in managing complex function-calls, a critical capability for applications requiring dynamic and real-time data integration.

Despite these advantages, fine-tuning also introduces challenges such as increased latency and higher training costs, as discussed by Raschka (2023). The implementation of techniques like QLoRA, which provides memory savings at the expense of slower training times, highlights the need to balance computational efficiency with performance gains. This trade-off is crucial for practical deployments, where resource constraints and performance requirements must be carefully managed (Raschka, 2023).

Furthermore, fine-tuning raises ethical and bias concerns, particularly when models are trained on niche or skewed datasets. The findings from Touvron et al. (2023) and Bender et al. (2021) underscore the importance of addressing these issues to prevent the propagation of harmful biases and ensure ethical deployment of LLMs. The performance evaluation of Llama 2 models, for instance, reveals a trade-off between improved factual accuracy and the potential for generating toxic content,

highlighting the need for continuous efforts to mitigate biases and ensure responsible AI usage (Touvron et al., 2023; Bender et al., 2021).

5.2. Presentation and Interpretation of Experimental Findings

5.2.1. Presentation of Experimental Findings

In this chapter, we will present the impact fine-tuning has on a base model. The base model, Llama 2 7B, will be compared to Mistral Instruct 7B v0.2 (Jiang et al., 2023) and three different OpenAI GPT models:

- gpt-3.5-turbo-0125
- gpt-4-turbo-2024-04-09
- gpt-4o-2024-05-13

For better readability of the figures, the OpenAI GPT model names will be shortened to:

- gpt-3.5-turbo
- gpt-4-turbo
- gpt-4o

At the moment of the research, those models were the most current versions that OpenAI had accessible (OpenAI, n. D.a). OpenAI has been picked as a baseline as their models consistently rank very high in online rankings (Morrison, 2024). Mistral is an LLM of similar size then Llama 2 but seems to have overall better performance than the base Lllama 2 7B (Mistral AI Team, 2023)

The general expectation is that the OpenAI GPT models will outperform the base models. The Mistral 7B model should also outperform the Llama 2 7B model. The data gathered for the baseline model performance and evaluation (Taoofstefan, 2024a), as well as the code used to produce the figures in this chapter (Taoofstefan, 2024d) can be found on GitHub.

In the first step, we will evaluate the results before the models have been fine-tuned. The baseline results for the evaluation metrics are presented in Table 5. This evaluation includes three different performance metrics: accuracy, precision, and TCE. The models assessed in this baseline evaluation are Llama 2, Mistral Instruct, gpt-3.5-turbo-0125, gpt-4-turbo-2024-04-09, and gpt-4o-2024-05-13.

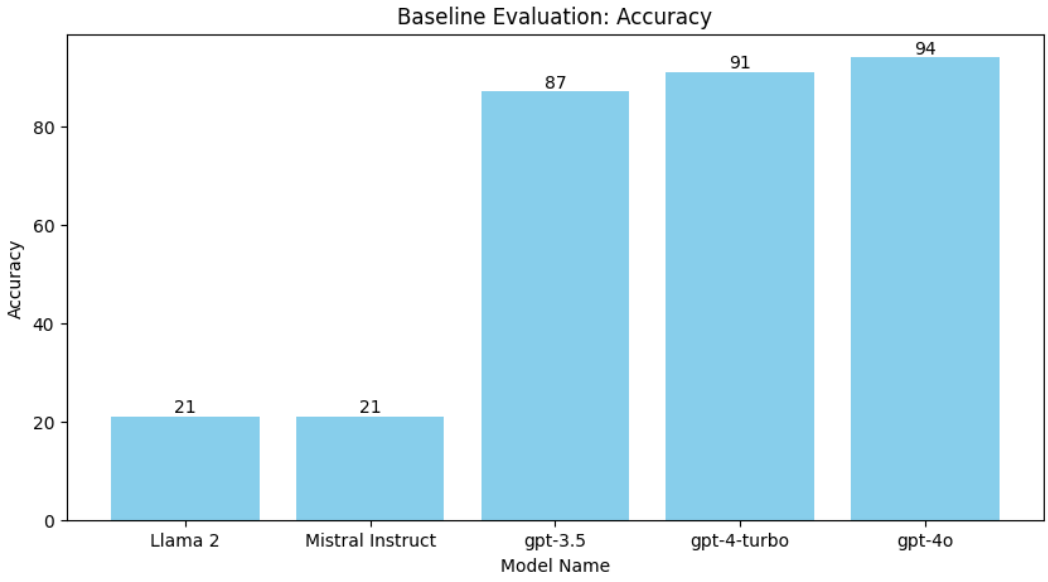
Table 5 Baseline Model Evaluation

Model Name	Accuracy	Precision	TCE
Llama 2	21	0	4
Mistral Instruct	21	0	21
gpt-3.5-turbo-0125	87	5	7
gpt-4-turbo-2024-04-09	91	26	15
gpt-4o-2024-05-13	94	31	15

Source: own creation, 2024

The Accuracy metric measures the model's ability to correctly decide whether a function-call should occur or if a chat completion is required. Higher values indicate better performance in making correct decisions. The results are presented in Figure 34.

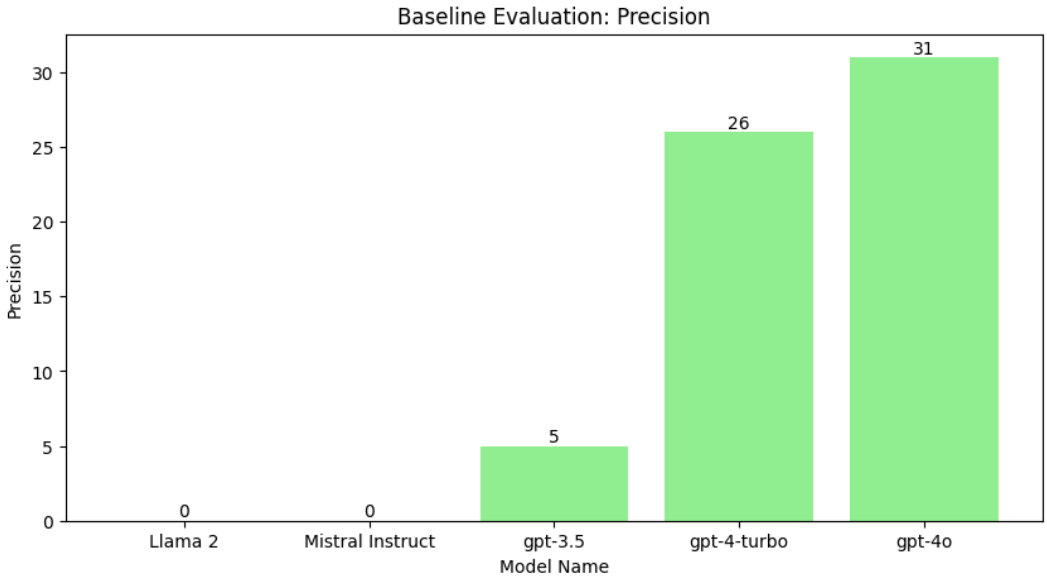
Figure 34 Baseline Accuracy Evaluation



Source: own creation, 2024

The Precision metric evaluates the correctness of the structured output when the model decides to make a function-call. Higher values indicate a higher rate of correct structured outputs. The results are presented in Figure 35.

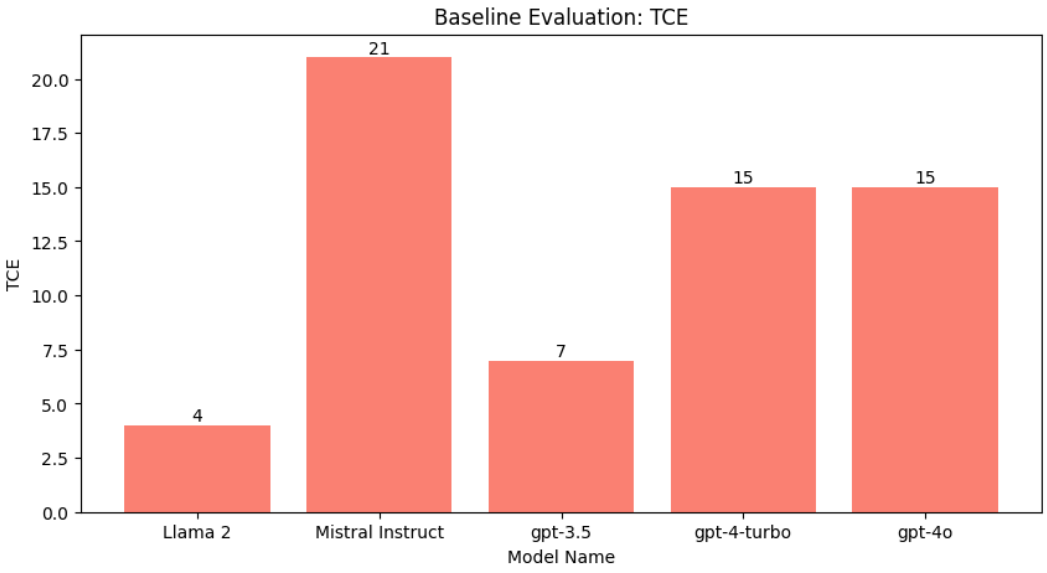
Figure 35 Baseline Precision Evaluation



Source: own creation, 2024

The TCE metric assesses the helpfulness and meaningfulness of the model's reply when a chat completion is required. Higher values indicate better performance in generating coherent and useful text responses. The results are presented in Figure 36.

Figure 36 Baseline TCE Evaluation



Source: own creation, 2024

The data in Table 5 indicates varying levels of performance across the different models for each of the evaluation metrics. The GPT series models, particularly the gpt-4 series, show higher scores across all metrics compared to Llama 2 and Mistral Instruct, with the sole exception of Mistral Instruct in the TCE metric.

Additionally, Llama 2 and Mistral Instruct were not able to perform any function-calls pre-fine-tuning, which is reflected in their Precision scores of 0. The Accuracy scores of 21 for both models are attributed entirely to chat completions.

All GPT models demonstrated the ability to perform function-calls, as indicated by their non-zero Precision scores. The gpt-4 series models, in particular, exhibited higher Precision and TCE scores.

Next, we are presenting the results from the fine-tuned models. The data gathered for the fine-tuned model performance and evaluation can be found on GitHub (Taoofstefan, 2024b).

The post-fine-tuning evaluation metrics for the language models are presented in Table 6. This evaluation assesses the impact of fine-tuning on the models' performance across three key metrics: Accuracy, Precision, and TCE. To better differentiate the base and fine-tuned models, both fine-tuned models will be listed with the suffix FT. The models evaluated in this post-fine-tuning phase include Llama 2 FT, Mistral Instruct FT, gpt-3.5-turbo-0125, gpt-4-turbo-2024-04-09, and gpt-4o-2024-05-13.

Llama 2 FT (Taoofstefan, 2024g; Taoofstefan, 2024h) and Mistral Instruct FT (Taoofstefan, 2024i; Taoofstefan, 2024j) have been shared and stored on Hugging Face.

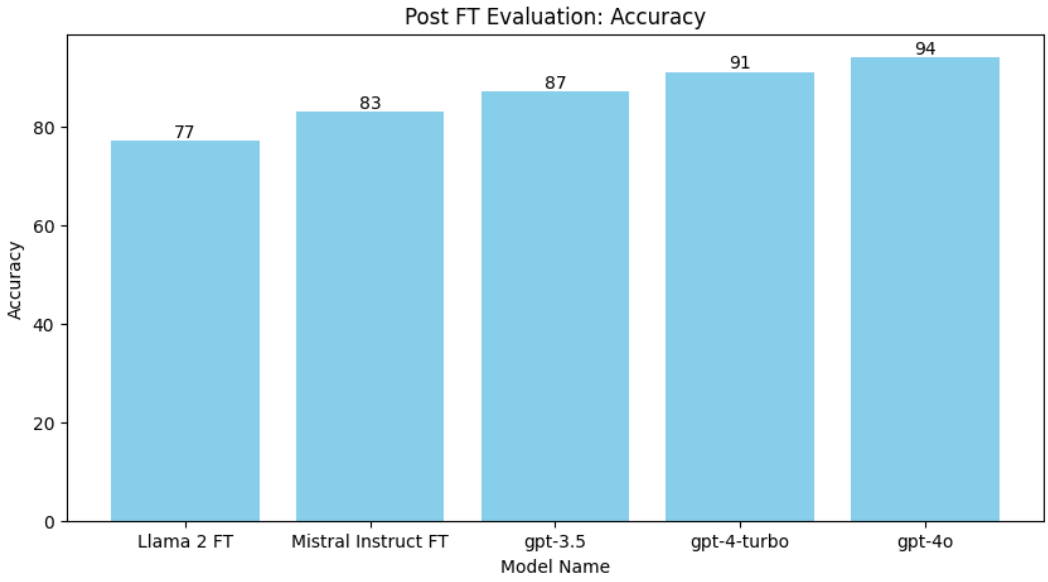
Table 6 Fine-Tuned Model Evaluation

Model Name	Accuracy	Precision	TCE
Llama 2 FT	77	35	9
Mistral Instruct FT	83	50	4
gpt-3.5-turbo-0125	87	5	7
gpt-4-turbo-2024-04-09	91	26	15
gpt-4o-2024-05-13	94	31	15

own creation, 2024

The Accuracy metric measures the model's ability to correctly decide whether a function-call should occur or if a chat completion is required. The results are presented in Figure 37. Higher values indicate better performance in making correct decisions.

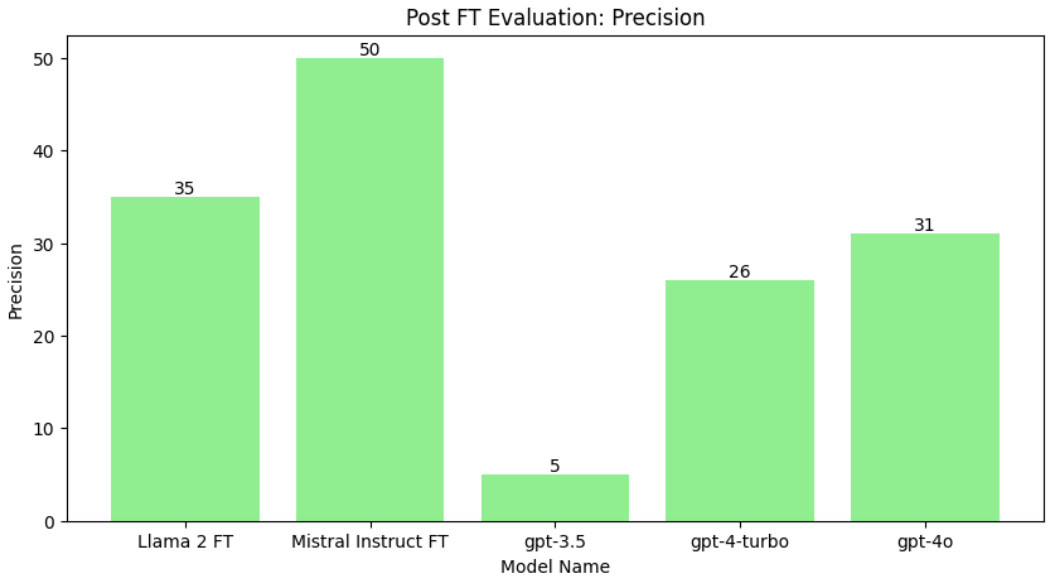
Figure 37 Fine-Tuning Accuracy Evaluation



Source: own creation, 2024

The Precision metric evaluates the correctness of the structured output when the model decides to make a function-call. The results are presented in Figure 38. Higher values indicate a higher rate of correct structured outputs.

Figure 38 Fine-Tuning Precision Evaluation

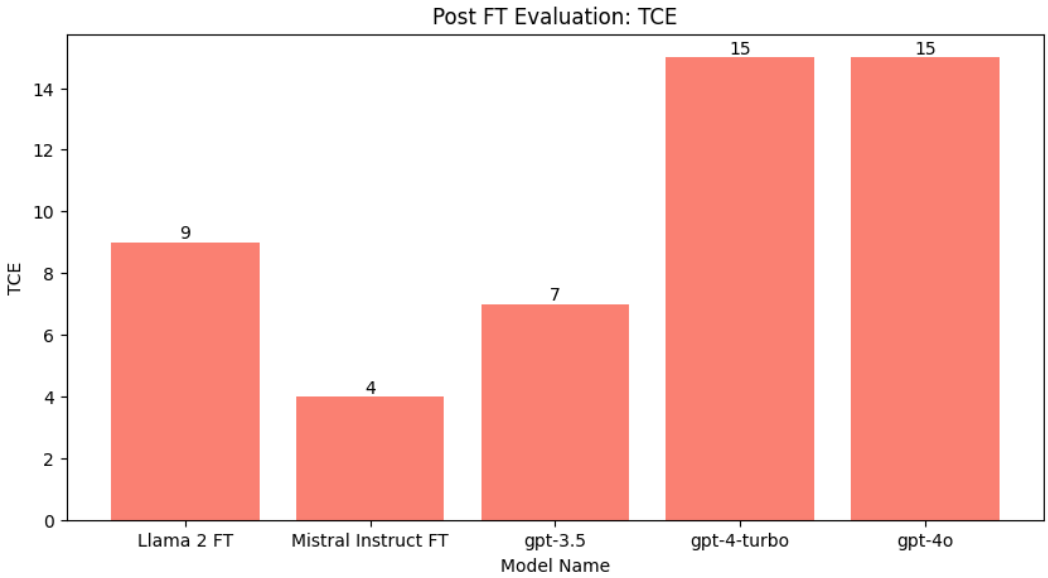


Source: own creation, 2024

The TCE metric assesses the helpfulness and meaningfulness of the model's reply when a chat completion is required. The results are presented in Figure 39. Higher values indicate better performance in generating coherent and useful text responses.

In the post-fine-tuning evaluation, the Llama 2 FT model demonstrated substantial improvements in its performance metrics. The accuracy of Llama 2 FT improved significantly, achieving a score of 77, which indicates a marked improvement in its decision-making capability following fine-tuning. This improvement highlights the model's increased proficiency in correctly determining whether a function-call should occur, or a chat completion is required. Additionally, the precision score of Llama 2 FT increased to 35, showcasing the model's newfound ability to produce correct structured outputs. This ability was not observed during the baseline evaluation. This significant increase in precision reflects the effectiveness of the fine-tuning process in increasing the model's structured output generation capabilities. Furthermore, Llama 2 FT achieved a TCE score of 9, indicating moderate improvements in the coherence and usefulness of its text completions. This score reflects the model's improved ability to generate coherent and meaningful replies, thereby improving the overall quality of its chat completions.

Figure 39 Fine-Tuning TCE Evaluation



Source: own creation, 2024

In the post-fine-tuning evaluation, the Mistral Instruct FT model exhibited significant improvements in its performance metrics. The accuracy of Mistral Instruct FT achieved a score of 83, showcasing a marked improvement in the model's ability to make correct decisions. This substantial increase underscores the higher decision-making capability of the model following the fine-tuning process. The precision score of Mistral Instruct FT reached 50, signifying a strong improvement in the model's ability to produce correct structured outputs. This notable improvement indicates the model's increased proficiency in generating accurate and usable structured data, a critical aspect of its performance. However, the TCE score for Mistral Instruct FT was 4, indicating a reduction compared to its baseline performance. This decrease suggests a potential trade-off in the coherence and usefulness of text completions for other gains achieved through fine-tuning, highlighting the complexities and nuanced impacts of the fine-tuning process on different performance metrics.

The post-fine-tuning evaluation presents a clear and detailed comparison of the performance metrics for the models. The data indicates significant improvements for Llama 2 FT and Mistral Instruct FT across most metrics, particularly in precision, where they previously scored zero. Notably, both Llama 2 FT and Mistral Instruct FT showcased better performance in precision than even the best GPT model, highlighting the remarkable improvements achieved through fine-tuning in their ability to produce correct structured outputs. This underscores the effectiveness of the fine-tuning process in significantly boosting the precision of these models, surpassing the established benchmarks set by the GPT models.

The post-fine-tuning evaluation reveals substantial improvements in the performance metrics of Llama 2 FT and Mistral Instruct FT, particularly in terms of accuracy and precision. The following analysis investigates these improvements and highlights detailed observations based on statistical measures (see Table 7). The code for the calculation can be found on GitHub (Taoofstefan, 2024).

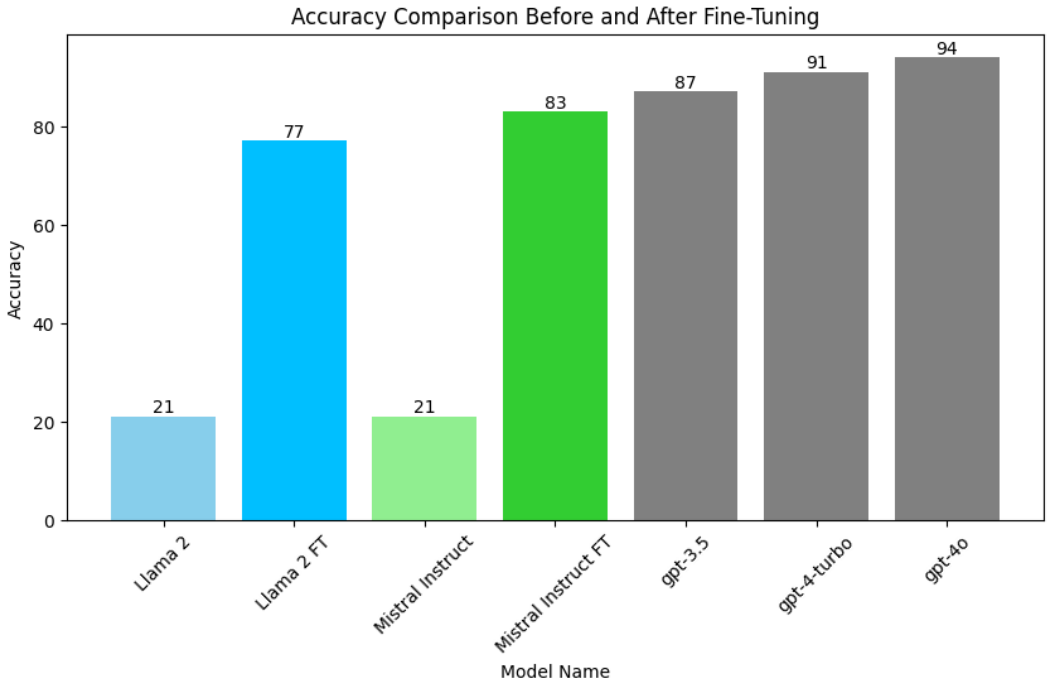
Table 7 Comparison of Statistical Metrics and Changes

Metric	Baseline Mean	Post FT Mean	Mean Change	Baseline Median	Post FT Median	Baseline Std Dev	Post FT Std Dev
Accuracy	62.8	86.4	+23.6	87	87	33.71	6.48
Precision	12.4	29.4	+17.0	5	31	14.66	16.47
TCE	12.4	10.0	-2.4	15	9	6.51	4.96

Source: own creation, 2024

The mean accuracy across all models improved significantly from 62.8 (baseline) to 86.4 (post-fine-tuning), indicating a marked improvement in decision-making capabilities post-fine-tuning (see Figure 40). This substantial increase underscores the effectiveness of the fine-tuning process in boosting the models' ability to correctly decide whether a function-call should occur, or a chat completion is required. The standard deviation of accuracy scores decreased from 33.71 (baseline) to 6.48 (post-fine-tuning). This reduction indicates a more consistent performance across models following fine-tuning, with less variability in their ability to make correct decisions.

Figure 40 Accuracy Comparison

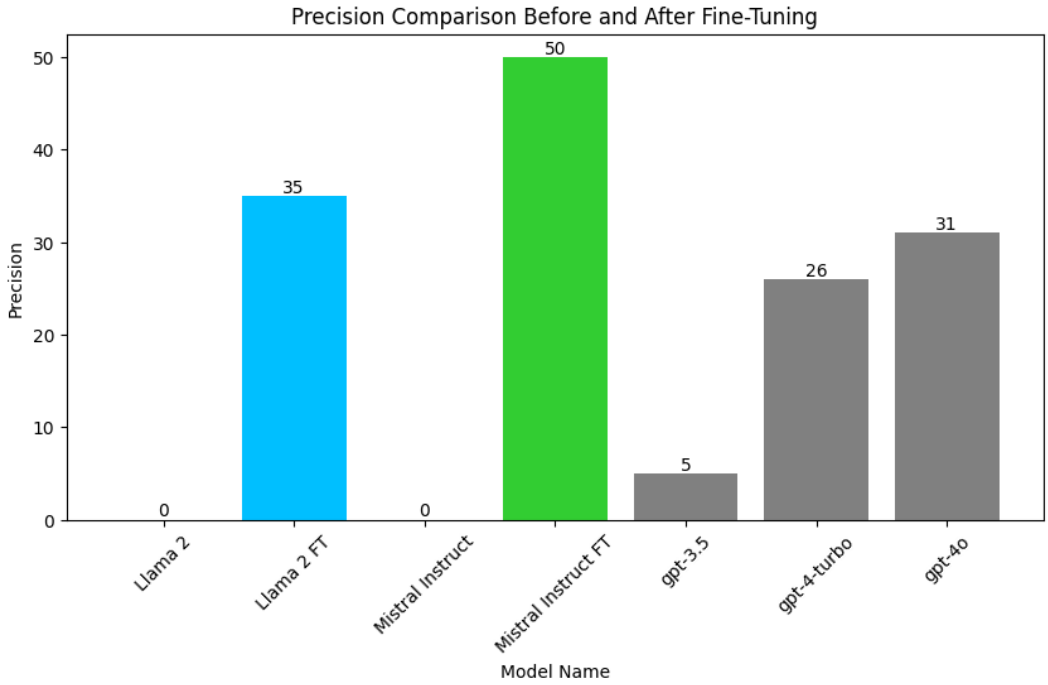


Source: own creation, 2024

The mean precision improved from 12.4 (baseline) to 29.4 (post-fine-tuning), reflecting a notable improvement in the models' ability to produce correct structured outputs (see Figure 41). The precision scores for Llama 2 FT and Mistral Instruct FT increased dramatically, showcasing their newfound ability to generate accurate structured data. This improvement is significant as it highlights the success of the fine-tuning process in addressing previously observed deficiencies in precision. The significant increase in the mean precision and the high standard deviation (16.47) post-fine-tuning highlight substantial improvements in the models Llama 2 FT and Mistral Instruct FT. These models demonstrated remarkable gains in their ability to produce correct structured outputs, indicating the success of the fine-tuning process in increasing this metric.

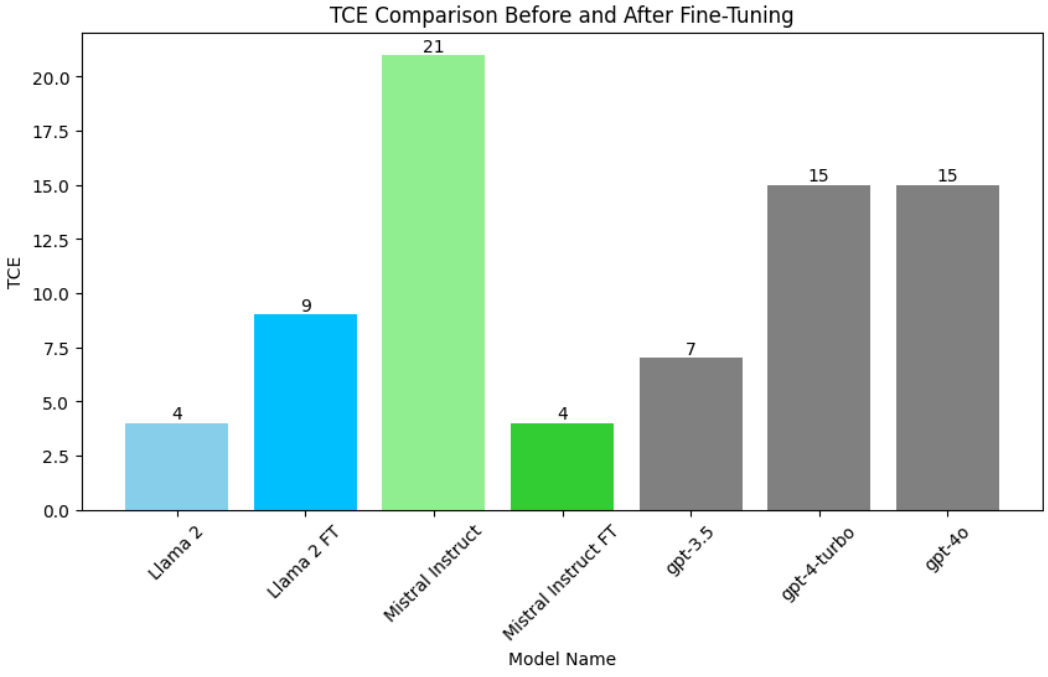
The mean TCE showed a slight decrease from 12.4 (baseline) to 10 (post-fine-tuning). This change is primarily influenced by the reduction in TCE for Mistral Instruct FT (see Figure 42), suggesting a trade-off in text coherence. The decrease indicates that while some models improved in other metrics, maintaining coherence in text generation remains a challenge for certain models post-fine-tuning. The decrease in mean TCE and the accompanying standard deviation (4.96) post-fine-tuning indicate a more varied performance in text coherence. This variability suggests that while some models managed to maintain or improve their text coherence, others experienced a reduction, highlighting the complexity of balancing improvements across multiple performance metrics.

Figure 41 Precision Comparison



Source: own creation, 2024

Figure 42 TCE Comparison



Source: own creation, 2024

The post-fine-tuning evaluation presents a clear and detailed comparison of the performance metrics for the models. The data indicates significant improvements for Llama 2 FT and Mistral Instruct FT across most metrics, particularly in precision, where they previously scored zero. Notably, both Llama 2 FT and Mistral Instruct FT showcased better performance in precision than even the best

GPT model, highlighting the remarkable improvements achieved through fine-tuning in their ability to produce correct structured outputs. This analysis underscores the effectiveness of the fine-tuning process and its impact on model performance, providing a robust foundation for further exploration and application of these models.

5.2.2. Interpretation of Experimental Findings

The fine-tuning experiment demonstrated significant improvements in the performance metrics of Llama 2 FT and Mistral Instruct FT. Notably, both models showed substantial gains in accuracy and precision, surpassing even the best-performing GPT models in precision. However, the trade-off in TCE for Mistral Instruct FT suggests that while fine-tuning can increase structured output capabilities, it may come at the cost of text coherence.

Comparing the fine-tuned models with their baseline counterparts, we observe that fine-tuning significantly developed the decision-making capabilities (Accuracy) and structured output generation (Precision). Llama 2 FT and Mistral Instruct FT, which initially had zero precision scores, demonstrated remarkable improvements, indicating the effectiveness of the fine-tuning process.

The mean accuracy across all models improved significantly from 62.8 (baseline) to 86.4 (post-fine-tuning), indicating a marked improvement in decision-making capabilities post-fine-tuning (see Figure 40). This substantial increase underscores the effectiveness of the fine-tuning process in boosting the models' ability to correctly decide whether a function-call should occur, or a chat completion is required. The standard deviation of accuracy scores decreased from 33.71 (baseline) to 6.48 (post-fine-tuning). This reduction indicates a more consistent performance across models following fine-tuning, with less variability in their ability to make correct decisions.

The mean precision improved from 12.4 (baseline) to 29.4 (post-fine-tuning), reflecting a notable improvement in the models' ability to produce correct structured outputs (see Figure 41). Particularly, the precision scores for Llama 2 FT and Mistral Instruct FT increased dramatically, showcasing their newfound ability to generate accurate structured data. This improvement is significant as it highlights the success of the fine-tuning process in addressing previously observed deficiencies in precision. The significant increase in the mean precision and the high standard deviation (16.47) post-fine-tuning highlight substantial improvements in the models Llama 2 FT and Mistral Instruct FT. These models demonstrated remarkable gains in their ability to produce correct structured outputs, indicating the success of the fine-tuning process in increasing this metric.

The mean TCE showed a slight decrease from 12.4 (baseline) to 10 (post-fine-tuning). This change is primarily influenced by the reduction in TCE for Mistral Instruct FT (see Figure 42), suggesting a trade-off in text coherence. The decrease indicates that while some models improved in other metrics, maintaining coherence in text generation remains a challenge for certain models post-fine-tuning. The decrease in mean TCE and the accompanying standard deviation (4.96) post-fine-tuning indicate a more varied performance in text coherence. This variability suggests that while some

models managed to maintain or improve their text coherence, others experienced a reduction, highlighting the complexity of balancing improvements across multiple performance metrics.

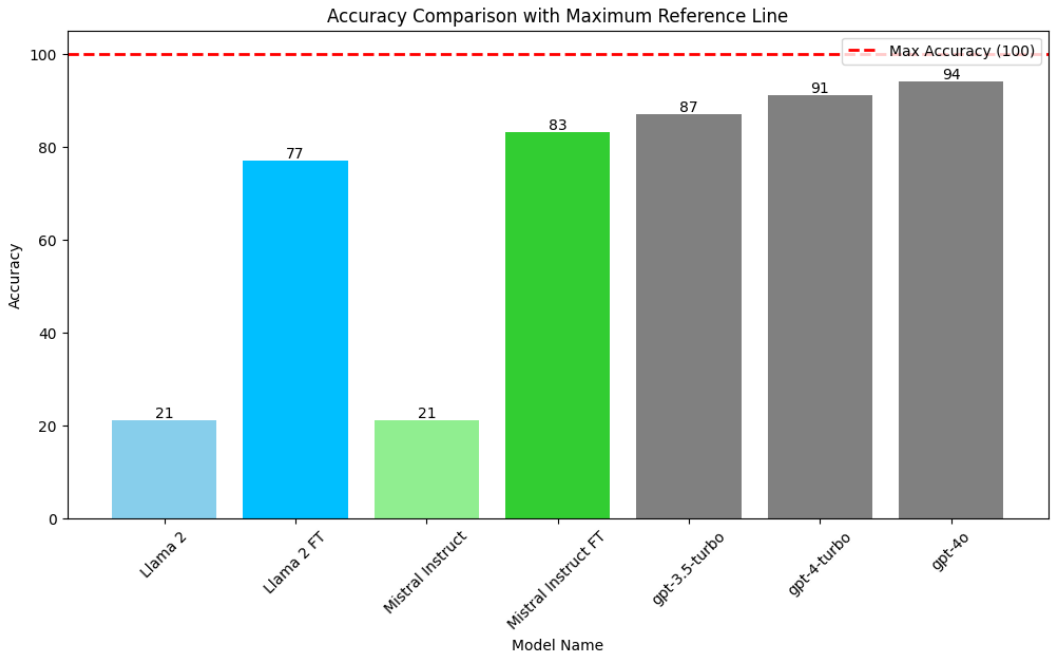
For a comprehensive understanding, it's essential to contextualize the performance improvements against the maximally achievable points. The maximum achievable values are:

- Accuracy: 100
- Precision: 79
- TCE: 21

The figures include a reference line indicating the maximum achievable points. These reference lines provide a clear visual benchmark, highlighting how close the models are to the optimal performance. The code used to produce the figures in this chapter (Taoofstefan, 2024d) can be found on GitHub.

While Llama 2 FT and Mistral Instruct FT have shown significant improvements, they still fall short of the maximum possible accuracy (see Figure 43). This suggests room for further improvements, possibly through more extensive fine-tuning or diverse training datasets.

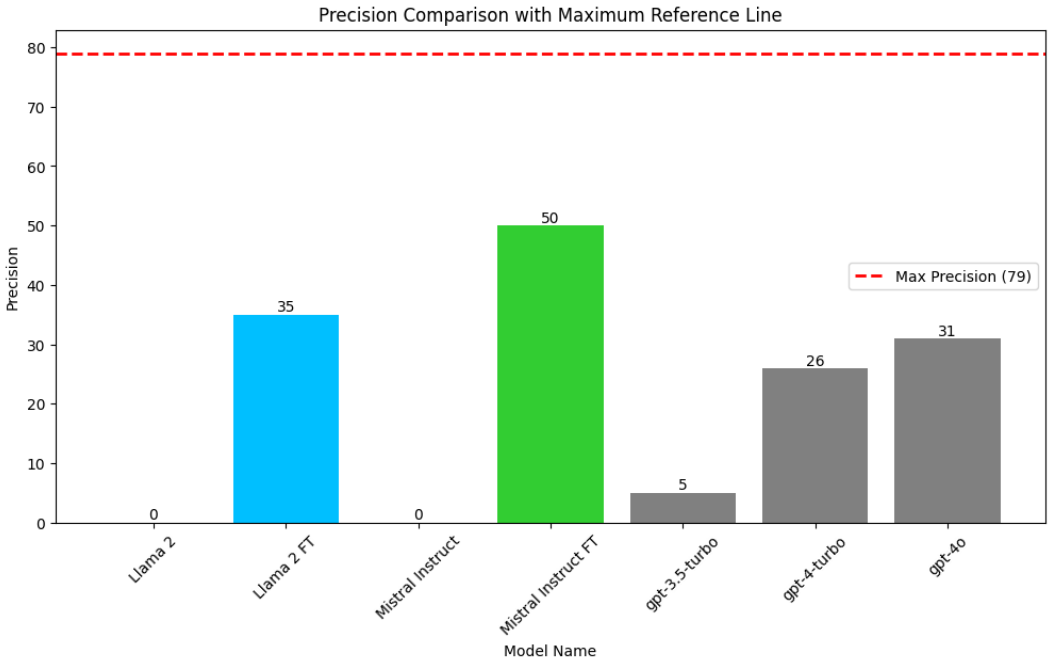
Figure 43 Accuracy Comparison with Maximum Reference Line



Source: own creation, 2024

Despite the remarkable improvements in precision for both Llama 2 FT and Mistral Instruct FT, there is a substantial gap in the maximum achievable precision (see Figure 44). This highlights the need for continued efforts to refine these models to achieve higher precision, which is crucial for applications requiring highly accurate structured outputs.

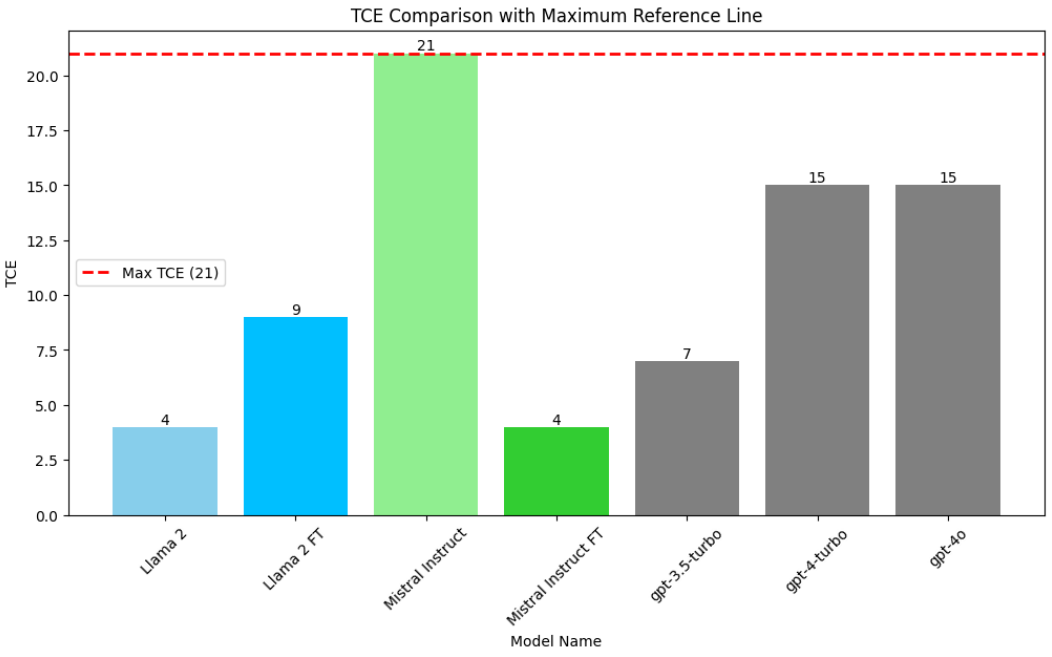
Figure 44 Precision Comparison with Maximum Reference Line



Source: own creation, 2024

The lower TCE score for fine-tuned Mistral Instruct, as shown in Figure 45, indicates a trade-off that may result from overfitting to structured output tasks. Addressing this will be critical for maintaining coherence in text generation.

Figure 45 TCE Comparison with Maximum Reference Line



Source: own creation, 2024

The TCE performance of the baseline Mistral Instruct was notably high due to its excellent chat replies. However, during fine-tuning, the model may have suffered from overfitting, often responding

with structured outputs even in chat completion scenarios where it shouldn't have. Notably, Mistral Instruct FT hallucinated new function-calls (for future weather, {'type': 'function', 'function': {'name': 'get_tomorrow_weather', 'parameters': {'location': 'Tehran, Iran', 'format': 'celsius'}}} or past weather, {'type': 'function', 'function': {'name': 'get_historical_weather', 'parameters': {'location': 'Brisbane, Australia', 'format': 'celsius', 'start_time': 'beginning_of_last_winter', 'end_time': 'end_of_last_winter'}}}). This overfitting led to a significant drop in TCE scores post-fine-tuning. It's important to note that the quality of the replies did not degrade; rather, the intent detection suffered. This suggests that while the model became highly proficient in structured outputs, it struggled with discerning when such outputs were appropriate, impacting its overall coherence in chat completions. This finding underscores the need for a balanced approach in fine-tuning, where improving one capability does not detrimentally affect another.

Llama 2 FT could have performed better in accuracy, particularly in scenarios requiring a nuanced understanding of function-call intents. For instance, the model had difficulty recognizing function-call intents in questions asking about current weather in a nuanced manner (e.g., "Are there any storm warnings for Adelaide today?"). This limitation suggests that Llama 2 FT might benefit from longer training periods or exposure to more varied and nuanced training cases. Addressing these areas could significantly improve its performance, bringing it closer to the capabilities of Mistral Instruct FT and narrowing the gap with the GPT models.

The improved accuracy and precision of Llama 2 FT make it a strong candidate for tasks requiring high decision-making accuracy and structured data generation. Applications such as interactive AI systems and complex data extraction can benefit from these improvements, even with moderate improvements in text coherence.

The increased accuracy and precision of Mistral Instruct FT position it well for applications that prioritize structured data accuracy. However, the reduced TCE suggests caution in applications heavily reliant on coherent text generation. Future fine-tuning strategies should aim to balance these aspects to optimize overall performance.

The fact that Llama 2 FT and Mistral Instruct FT outperform the GPT models in precision is significant. It highlights the potential of fine-tuning to increase specific performance metrics beyond the capabilities of pre-trained models. This finding suggests that with targeted fine-tuning, even smaller or less advanced models can achieve superior performance in critical areas.

The fine-tuning experiment provides valuable insights into the improvements and trade-offs involved in improving language model performance. By understanding these dynamics, researchers and practitioners can better design and deploy language models tailored to specific application needs, balancing accuracy, precision, and text coherence for optimal performance.

Next, we aim to statistically validate the results from the experiment. The calculations used for the validation process below can be found in GitHub (Taoofstefan, 2024c).

The initial assumption is to calculate the paired t-test, which is conventionally used to determine whether there is a statistically significant difference between the means of two related groups. It is particularly appropriate when the differences between paired observations are normally distributed (JMP Statistical Discovery, n. D.).

First, we check if the data has a normal distribution. In the analysis, the Shapiro-Wilk test (van den Berg, n.D.) was performed to assess the normality of the differences in performance metrics (Accuracy, Precision, TCE) between the baseline and post-fine-tuning evaluations. The results are presented in Table 8 **Shapiro-Wilk Test**.The p-values for all metrics are less than 0.05, indicating that the differences are not normally distributed. Consequently, the assumptions for the paired t-test are not met, making it unsuitable for our analysis.

Given the non-normality of the differences, the Wilcoxon signed-rank test is a more appropriate choice (Wilcoxon signed-rank test, n.D.). This non-parametric test does not assume normal distribution and is suitable for ordinal or continuous data. It examines the two related groups' median differences and is especially helpful when working with small sample sizes or data that has no normal distribution (Wilcoxon signed-rank test, n.D.). The results of the Wilcoxon signed-rank test for our performance metrics are shown in Table 9.

Table 8 Shapiro-Wilk Test

Metric	Statistic	P-Value
Accuracy	0,7138	0,0133
Precision	0,7619	0,0382
TCE	0,7497	0,0296

Source: own creation, 2024

Table 9 Wilcoxon Test

Metric	Statistic	P-Value
Accuracy	0,0000	0,1797
Precision	0,0000	0,1797
TCE	1,0000	0,6547

Source: own creation, 2024

The p-values for Accuracy, Precision, and TCE are all greater than 0.05, indicating that the changes are not statistically significant at the 5% level. However, due to the small sample size, these results should be interpreted with caution.

To address the limitations of the small sample size and provide a more robust estimation of p-values, we applied the bootstrap resampling technique. This method involves repeatedly resampling the data with replacement to estimate the distribution of the test statistics (Yen, 2019). The bootstrap p-values for the performance metrics are presented in Table 10.

Table 10 Bootstrap p-values

Metric	Bootstrap p-value
Accuracy	0.1944
Precision	0.1068
TCE	0.4945

Source: own creation, 2024

Table 11 Effect Sizes

Metric	Effect Size
Accuracy	0.729
Precision	0.712
TCE	-0.284

Source: own creation, 2024

The bootstrap p-values confirm the findings from the Wilcoxon signed-rank test:

- Accuracy: The p-value of 0.1944 indicates that the improvement in accuracy is not statistically significant.
- Precision: The p-value of 0.1068 suggests that the improvement in precision is approaching significance and might be considered marginally significant.
- TCE: The p-value of 0.4945 indicates that the decline in TCE is not statistically significant.

To complement the significance testing, we calculated the effect sizes for each metric using Cohen's d for paired samples. The effect sizes provide a measure of the magnitude of the differences, which is crucial for understanding the practical significance of the findings (Data Nova, n. D.). The results are shared in Table 11.

The effect sizes for Accuracy (0.729) and Precision (0.712) indicate medium to large effects, suggesting substantial improvements due to fine-tuning. The small negative effect size for TCE (-0.284) indicates a slight decline in text coherence, aligning with the observation of potential overfitting.

Despite the lack of statistical significance, the effect sizes indicate meaningful improvements in Accuracy and Precision. The medium to large effect sizes suggest that the fine-tuning process has led to substantial improvements in these metrics, which are practically significant for real-world applications.

The bootstrap p-value for Precision (0.1068) indicates that the improvement is approaching significance and may be considered marginally significant. This suggests that with a larger sample size, the improvement in precision could become statistically significant, highlighting the importance of fine-tuning in improving structured output generation.

The small negative effect size and non-significant p-value for TCE indicate a slight decline in text coherence. This suggests a trade-off where the focus on improving structured outputs might have impacted the model's ability to maintain coherence in text generation. This finding underscores the need for balanced fine-tuning to optimize multiple performance metrics simultaneously.

While the statistical tests did not show significant changes at the conventional 5% level, the medium to large effect sizes for Accuracy and Precision suggest meaningful improvements. These effect sizes indicate substantial practical significance, underscoring the value of fine-tuning in improving model performance. This approach aligns with widely accepted scientific practices, where effect sizes are used to highlight the practical relevance of findings, even when statistical significance is not achieved.

To further validate the results, we can calculate Cohen's d with a pooled standard deviation (Data Nova, n. D.). The results are shared in Table 12.

Table 12 Cohen's d with pooled standard deviation

Metric	Mean Change	Pooled Std Dev	Cohen's d	Interpretation
Accuracy	23.6	24.26	0.97	Large effect
Precision	17.0	15.59	1.09	Large effect
TCE	-2.4	5.79	-0.41	Small to medium effect

Source: own creation, 2024

The statistical analysis of the experimental data reveals that the fine-tuning process has had a substantial impact on the performance metrics of the models. Specifically, the calculation of Cohen's d for Accuracy yields a value of 0.97, which indicates a large effect size. This suggests a significant improvement in the models' accuracy post-fine-tuning. Similarly, the Cohen's d value for Precision is 1.09, also signifying a large effect size and indicating a marked improvement in the models' ability to produce correct structured outputs. These results demonstrate that fine-tuning has effectively improved both Accuracy and Precision to a considerable extent.

In contrast, the TCE metric exhibits a Cohen's d of -0.41, indicating a small to medium effect size. This suggests a slight decline in the coherence and usefulness of text completions following the fine-tuning process. While this trade-off indicates a reduction in TCE, the overall improvements in Accuracy and Precision are significant. The large effect sizes for these metrics underscore the practical significance of fine-tuning, affirming that it has led to meaningful and beneficial improvements in the models' performance, despite the minor decrease in TCE. These findings highlight the effectiveness of fine-tuning in improving critical aspects of model performance.

6. Conclusion

6.1. Summary of Key Findings

6.1.1. Synthesis of Research Contributions

This thesis has made several significant contributions to the field of artificial intelligence, specifically in the domain of LLMs and their fine-tuning for increased function-calling capabilities. The research presented herein addresses critical gaps in the understanding and practical implementation of LLM fine-tuning, providing both theoretical insights and empirical evidence.

This study has demonstrated the effectiveness of fine-tuning LLMs, specifically Llama 2 7B and Mistral Instruct 7B, using QLoRA. The results indicate significant improvements in the accuracy and precision of function-calling capabilities post-fine-tuning. Llama 2 7B, which initially exhibited no function-calling capability, achieved an accuracy of 77% and a precision score of 35 points post-fine-tuning. Similarly, Mistral Instruct 7B saw its accuracy improve to 83% and its precision to 50 points.

The research has provided detailed insights into how fine-tuning impacts the function-calling performance of LLMs. The comparative analysis between fine-tuned and baseline models, including OpenAI's GPT series, highlights the potential of fine-tuned smaller models to outperform even larger, more advanced models in specific metrics such as precision. This underscores the importance of targeted fine-tuning in achieving high-functioning specialized models

The practical implications of these findings are significant for AI developers and practitioners. The research suggests that with careful fine-tuning, even smaller models like Llama 2 7B and Mistral Instruct 7B can be optimized for specific tasks, providing a cost-effective alternative to larger, more resource-intensive models. This has direct applications in fields such as interactive AI systems and complex data extraction, where function-calling accuracy is paramount (Endpoints Team, 2023; Fireworks.ai, 2023).

The thesis has contributed methodologically by validating the use of QLoRA as an effective fine-tuning technique for LLMs. The detailed empirical analysis and statistical evaluation provide a robust framework for future research, offering a replicable methodology for optimizing LLMs for specialized tasks (Dettmers et al., 2023).

In addressing the ethical implications of LLM deployment, the research has underscored the importance of ongoing fine-tuning and monitoring to mitigate biases and ensure the responsible use of AI technologies. The study advocates for the continuous refinement of models to address emergent ethical issues, particularly in multilingual and cross-domain applications (Bender et al., 2021; Touvron et al., 2023).

6.1.2. Assessment of Research Questions

The research conducted in this thesis aimed to address several key questions regarding the fine-tuning of LLMs and their function-calling capabilities. These questions and their respective assessments are as follows:

1. **What are the principal outcomes and investigative approaches in the current scholarly landscape concerning LLMs' capabilities in API-compliant argument generation?**

The literature review conducted in this thesis reveals that fine-tuning LLMs significantly enhances their ability to generate API-compliant arguments. Investigative approaches commonly involve empirical experiments and evaluations using benchmarks like the BFCL. Studies indicate that models such as GPT-4, Mistral-7B, and Llama-2-70B achieve high performance in function-calling tasks post-fine-tuning, demonstrating improved precision and reliability (Yan et al., 2024; Hakhamaneshi & Ahmad, 2023).

2. **What are the prevalent research gaps and challenges identified in the literature regarding function-calling by LLMs?**

The primary challenges identified include the semantic interpretation of user requests, context management, and interoperability with diverse systems and APIs. Additionally, there is a notable gap in research focusing on less common programming languages and legacy systems. Ethical considerations and the potential for bias propagation also remain critical concerns that necessitate further investigation and mitigation strategies (Endpoints Team, 2023; Touvron et al., 2023; Bender et al., 2021).

3. **How has the evolution of LLMs influenced their ability to generate API-compliant function-calls in recent years?**

The evolution of LLMs has significantly enhanced their function-calling capabilities. Improvements in model architectures, such as those seen in GPT-4 and Mistral-7B, have led to better accuracy and precision in generating API-compliant function-calls. Fine-tuning has played a crucial role in this evolution, allowing models to adapt to specific tasks and improve their performance in generating accurate and reliable function-calls (Li & Liang, 2021; Fireworks.ai, 2023).

4. **How can fine-tuning a compact LLM model improve its accuracy and adherence to API standards in function-calling tasks?**

Fine-tuning a compact LLM model can significantly enhance its accuracy and adherence to API standards. This research demonstrated that fine-tuning Llama 2 7B and Mistral Instruct 7B improved their accuracy from 21% to 77% and 83%, respectively, and increased their precision from 0% to 35% and 50%, respectively. These improvements highlight the effectiveness of fine-tuning in optimizing LLMs for specific tasks, thereby enhancing their compliance with API standards (Hakhamaneshi & Ahmad, 2023; Z. Wang et al., 2024).

5. **What fine-tuning strategies are most effective in improving the precision of LLMs for API-compliant function-calls?**

The study identified QLoRA as a particularly effective fine-tuning strategy for improving the precision of LLMs in API-compliant function-calls. This approach not only enhances precision but also optimizes computational resource usage, making it a practical choice for fine-tuning compact models. The success of QLoRA in this research underscores its potential for broader application in enhancing LLM performance (Dettmers et al., 2023; Raschka, 2023).

6. **What is the impact of model fine-tuning on the performance of LLMs in various function-calling scenarios, and how does it compare to baseline models?**

Fine-tuning significantly enhances the performance of LLMs in function-calling scenarios compared to baseline models. The research demonstrated that fine-tuned models like Llama 2 7B and Mistral Instruct 7B not only achieved higher accuracy and precision but also outperformed larger pre-trained models such as GPT-4 in specific tasks. However, trade-offs were noted in text coherence (TCE), particularly for Mistral Instruct 7B, indicating the need for balanced fine-tuning approaches to maintain overall model performance (Fireworks.ai, 2023; Greyling, 2023a).

By addressing these research questions, this thesis provides a comprehensive understanding of the current capabilities, challenges, and advancements in the function-calling abilities of LLMs, particularly through the lens of fine-tuning and its impact on model performance.

6.1.3. Reflection on Objectives Met

Objective 1: Perform an Exhaustive Review of Existing Academic and Industry Research Related to Function-Calling Capabilities of LLMs.

The first objective was comprehensively addressed through an in-depth review of academic and industry research on LLM function-calling capabilities. This review encompassed scholarly articles, technical reports, and case studies, revealing that fine-tuning significantly improves the accuracy and reliability of different models (Yan et al., 2024).

Fine-tuning smaller models such as Llama 2 7B and Llama 2 13B demonstrated substantial performance improvements while being more cost-effective than larger models like GPT-4 (Hakhamaneshi & Ahmad, 2023). This indicates that fine-tuning is efficient and scalable, particularly for resource-constrained applications.

However, the review also identified significant trade-offs between improved task accuracy and increased computational costs (Z. Wang et al., 2024). Fine-tuning requires substantial computational resources, necessitating a balance between performance gains and resource efficiency.

Additionally, the literature highlighted the need to address ethical and bias concerns. Fine-tuning on niche or skewed datasets can propagate harmful biases, requiring continuous efforts to ensure

responsible AI usage and mitigate the risk of generating toxic content (Touvron et al., 2023; Bender et al., 2021).

Objective 2: Refine Function-Calling Accuracy in LLMs Through Empirical Research and Fine-Tuning.

The second objective was to empirically refine the function-calling accuracy of a compact LLM through fine-tuning. This objective was successfully met, as demonstrated by the substantial improvements in accuracy and precision of the fine-tuned Llama 2 7B and Mistral Instruct 7B models. The fine-tuning process, informed by the outcomes of the literature review, involved the application of best practices and addressing known limitations identified in the review. The results showed a significant improvement in the models' accuracy and precision. Particularly with Llama 2 7B's precision scores increasing from 0 to 35 and Mistral Instruct 7B's precision rising from 0 to 50, fine-tuning showcased that it could beat OpenAI's best models. These improvements underscore the effectiveness of the fine-tuning methodologies employed and their potential for advancing the function-calling capabilities of LLMs to meet real-world standards and expectations.

In conclusion, the thesis has successfully met its stated objectives. The exhaustive literature review provided a comprehensive understanding of the current state of function-calling capabilities in LLMs, identifying key research gaps and best practices. The empirical research demonstrated that targeted fine-tuning could significantly increase the precision and accuracy of compact LLMs, making them viable for generating API-compliant outputs. These findings contribute valuable insights to the field and offer practical implications for developers and practitioners aiming to optimize LLMs for specific tasks.

6.2. Theoretical and Practical Implications

The research has practical implications for developers and practitioners. The improvements in accuracy and precision through fine-tuning suggest that targeted fine-tuning can greatly increase LLM functionality, making them more suitable for high-precision applications requiring API compliance.

For developers, the success of fine-tuning smaller models like Llama 2 7B and Mistral Instruct 7B provides a practical solution for resource constraints, enabling these models to achieve performance levels comparable to larger models. This makes advanced LLM capabilities more accessible without extensive computational resources.

Practitioners can benefit from balancing performance gains with computational efficiency, optimizing fine-tuning processes to achieve desired outcomes while managing costs (Z. Wang et al., 2024). This balance is crucial for deploying LLMs in industries with budget and resource limitations.

Moreover, addressing ethical concerns is vital. Continuous monitoring and refinement of models are necessary to prevent the introduction or amplification of biases, ensuring adherence to ethical

standards (Touvron et al., 2023; Bender et al., 2021). This is particularly important in sensitive domains like healthcare, finance, and legal services, where biased outputs can have severe consequences.

6.3. Future Research Directions

This study has highlighted several pathways for future research that can further improve the understanding and application of large LLMs in function-calling tasks. First, expanding the range of models and fine-tuning techniques investigated can provide a more comprehensive understanding of the efficacy and limitations of various approaches. Future studies should explore additional models beyond Llama 2 and Mistral, including newer and more diverse architectures, to generalize findings across a broader spectrum of LLMs (Hakhamaneshi & Ahmad, 2023; Touvron et al., 2023).

Second, integrating more sophisticated techniques to improve text coherence (TCE) alongside function-calling accuracy is crucial. While QLoRA demonstrated significant improvements, exploring hybrid approaches that combine multiple fine-tuning strategies could address the trade-offs observed between structured output precision and text coherence (Raschka, 2023).

Third, future research should go further into addressing ethical and bias concerns in fine-tuning practices. Developing and testing frameworks for continuous bias detection and mitigation will be vital in ensuring the ethical deployment of LLMs (Bender et al., 2021). This includes the creation of more balanced and diverse training datasets to reduce the risk of bias propagation in specialized models.

Finally, exploring real-world applications of fine-tuned models in dynamic environments can provide valuable insights into their practical utility and performance. Longitudinal studies that assess the long-term effectiveness and adaptability of fine-tuned models in various industry settings, such as healthcare and finance, will help in understanding their practical implications and scalability (Z. Wang et al., 2024).

6.4. Limitations of the Study

This study acknowledges several limitations that have impacted the scope and generalizability of its findings. First, the research was constrained by time and computational resources, limiting the investigation to only two smaller LLMs, Llama 2 and Mistral Instruct 7B. While these models provided valuable insights, the findings may not fully represent the performance characteristics of larger or different LLM architectures.

Second, the focus on QLoRA as the primary fine-tuning technique, though effective, narrowed the exploration of alternative methods that might offer different or complementary benefits. Techniques such as adapter tuning and prompt engineering were not extensively examined, which may have provided a more holistic view of fine-tuning strategies (Dettmers et al., 2023; Li & Liang, 2021).

Third, the evaluation metrics used, while comprehensive, highlighted trade-offs between precision and text coherence without fully exploring methods to balance these aspects. Future studies should incorporate additional metrics and more nuanced evaluation frameworks to better capture the multi-dimensional performance of fine-tuned models.

Finally, the ethical and bias considerations, although addressed, were based on existing literature and did not include extensive empirical testing of bias mitigation strategies. This limitation suggests the need for more targeted research to develop robust frameworks for identifying and mitigating biases in fine-tuned LLMs (Touvron et al., 2023; Bender et al., 2021).

In conclusion, while this study provides important contributions to the understanding of fine-tuning LLMs for function-calling tasks, it also highlights the need for ongoing research to address its limitations and expand its findings. Future work should build on these foundations to further develop the capabilities and ethical deployment of LLMs in various real-world applications.

.

VI. References

Ahmed, T., Bird, C., Devanbu, P., Chakraborty, S. (2024, February 23): Studying LLM Performance on Closed- and Open-source Data, arXiv, https://arxiv.org/abs/2402.15100

Autor, D. H. (2015): Why Are There Still So Many Jobs? The History and Future of Workplace Automation, Journal of Economic Perspectives, Vol. 29, No. 3, pp. 3-30, https://www.aeaweb.org/articles?id=10.1257/jep.29.3.3

Beltagy, I., Lo, K., Cohan, A. (2019): SciBERT: A pretrained language model for scientific text, arXiv, https://arxiv.org/abs/1903.10676

Bender, E. M., Gebru, T., McMillan-Major, A., Shmitchell, S. (2021): On the Dangers of Stochastic Parrots: Can Language Models Be Too Big?, Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency, pp. 610-623, https://dl.acm.org/doi/10.1145/3442188.3445922

Bergstra, J., Bardenet, R., Bengio, Y., Kégl, B. (2011): Algorithms for hyper-parameter optimization, Advances in Neural Information Processing Systems, pp. 2546–2554, https://api.semanticscholar.org/CorpusID:11688126

Brev.dev (2024): Fine-tuning Llama 2 7B using QLoRA, GitHub, https://github.com/brevdev/notebooks/blob/main/llama2-finetune.ipynb

Brev.dev (2024a): Export your Fine-Tuned Model to GGUF to Run Locally, GitHub, https://github.com/brevdev/notebooks/blob/main/gguf-export.ipynb

Brev.dev (n.D): Getting started | Brev docs, brev.dev, https://brev.dev/

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., Amodei, D. (2020, May 28): Language Models are Few-Shot Learners, arXiv, https://arxiv.org/abs/2005.14165

Capelle, T., (2024a, February 20): How to Fine-Tune an LLM Part 2: Instruction Tuning Llama 2, Weights & Biases, https://wandb.ai/capecape/alpaca_ft/reports/How-to-Fine-Tune-an-LLM-Part-2-Instruction-Tuning-Llama-2--Vmlldzo1NjY0MjE1

Capelle, T., (2024b, January 15): How to Fine-Tune an LLM Part 1: Preparing a Dataset for Instruction Tuning, Weights & Biases, https://wandb.ai/capecape/alpaca_ft/reports/How-to-implement-fine-tuning-of-an-LLM-Part-1-Dataset-for-Instruction-Tuning--Vmlldzo1NTcxNzE2

Chen, M., Tworek, J., Jun, H., Yuan, Q., Pinto, H. P. de O., Kaplan, J., Edwards, H., Burda, Y., Joseph, N., Brockman, G., Ray, A., Puri, R., Krueger, G., Petrov, M., Khlaaf, H., Sastry, G., Mishkin, P., Chan, B., Gray, S., Ryder, N., Pavlov, M., Power, A., Kaiser, L., Bavarian, M., Winter, C., Tillet, P., Such, F. P., Cummings, D., Plappert, M., Chantzis, F., Barnes, E., Herbert-Voss, A., Guss, W. H., Nichol, A., Paino, A., Tezak, N., Tang, J., Babuschkin, I., Balaji, S., Jain, S., Saunders, W., Hesse, C., Carr, A. N., Leike, J., Achiam, J., Misra, V., Morikawa, E., Radford, A., Knight, M., Brundage, M., Murati, M., Mayer, K., Welinder, P., McGrew, B., Amodei, D., McCandlish, S., Sutskever, I., Zaremba, W. (2021, July 7): Evaluating Large Language Models Trained on Code, arXiv, https://arxiv.org/abs/2107.03374

Choi, S., Gazeley, W., Wong, S. H., Li, T. (2023, November 10): Conversational Financial Information Retrieval Model (ConFIRM), arXiv, https://arxiv.org/abs/2310.13001

Conneau, A., Khandelwal, K., Goyal, N., Chaudhary, V., Wenzek, G., Guzmán, F., Grave, E., Ott, M., Zettlemoyer, L., Stoyanov, V. (2019, November 5): Unsupervised Cross-lingual Representation Learning at Scale, arXiv, https://arxiv.org/abs/1911.02116

Creswell, J. W. (2009): Research design: Qualitative, quantitative, and mixed methods approaches, 3rd edition, Sage publications, https://www.ucg.ac.me/skladiste/blog_609332/objava_105202/fajlovi/Creswell.pdf

Das, S. (2024, January 25): Fine Tune Large Language Model (LLM) on a Custom Dataset with QLoRA, Medium, https://dassum.medium.com/fine-tune-large-language-model-llm-on-a-custom-dataset-with-qlora-fb60abdeba07

Data Nova (n.D.): T-TEST ESSENTIALS: DEFINITION, FORMULA AND CALCULATION , Data Nova, https://www.datanovia.com/en/lessons/t-test-effect-size-using-cohens-d-measure/

Dettmers, T., Pagnoni, A., Holtzman, A., Zettlemoyer, L. (2023, May 23): QLoRA: Efficient Finetuning of Quantized LLMs, arXiv, https://arxiv.org/abs/2305.14314

Devlin, J., Chang, M.-W., Lee, K., Toutanova, K. (2019, May 24): BERT: Pre-training of deep bidirectional transformers for language understanding, arXiv, https://arxiv.org/abs/1810.04805

Dey, V. (2023, June 12): Salesforce announces AI Cloud to empower enterprises with trusted generative AI, VentureBeat, https://venturebeat.com/ai/salesforce-announces-ai-cloud-to-empower-enterprises-with-trusted-generative-ai/

Doerrfeld, B. (2024, February 29): LLM Security Hinges on API Security, Nordic APIs, https://nordicapis.com/llm-security-hinges-on-api-security/

Eassa, A. (2022, March 1): Saving Time and Money in the Cloud with the Latest NVIDIA-Powered Instances, NVIDIA, https://developer.nvidia.com/blog/saving-time-and-money-in-the-cloud-with-the-latest-nvidia-powered-instances/

Endpoints Team (2023, December 12): Anyscale Endpoints: JSON Mode and Function-calling Features, Anyscale Inc., https://www.anyscale.com/blog/anyscale-endpoints-json-mode-and-function-calling-features

EU (2016): General Data Protection Regulation, Europa.eu, https://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CELEX:32016R0679

EU (2024, March 13): Artificial Intelligence Act: MEPs adopt landmark law, Europa.eu, https://www.europarl.europa.eu/news/en/press-room/20240308IPR19015/artificial-intelligence-act-meps-adopt-landmark-law

Fernandez, T., Ackerson, D. (2023, August 3): Function-calling: Integrate Your GPT Chatbot With Anything, semaphore, https://semaphoreci.com/blog/function-calling

Fireworks.ai (2023, December 20): Fireworks Raises the Quality Bar with Function-calling Model and API Release, fireworks.ai, https://blog.fireworks.ai/fireworks-raises-the-quality-bar-with-function-calling-model-and-api-release-e7f49d1e98e9

Geronimo (2023, November 5): Finetuning Lllama 2 and Mistral, Medium, https://medium.com/@geronimo7/finetuning-llama2-mistral-945f9c200611

GlaiveAI (n.D.): glaive-function-calling, Hugging Face, https://huggingface.co/datasets/glaiveai/glaive-function-calling

GlaiveAI (n.D.a): glaive-function-calling-v2, Hugging Face, https://huggingface.co/datasets/glaiveai/glaive-function-calling-v2

Greyling, C. (2023a, June 13): OpenAI Function-calling, Medium, https://cobusgreyling.medium.com/openai-function-calling-98fbf9539d2a

Greyling, C. (2023b, June 16): Practical Examples of OpenAI Function-calling, Medium, https://cobusgreyling.medium.com/practical-examples-of-openai-function-calling-a6419dc38775

Gupta, M. (2023, August 25): Thoughts from LLama-2 paper, Medium, https://medium.com/@manavg/thoughts-from-llama-2-paper-b8013bab3a8

Hakhamaneshi, K., Ahmad, R. (2023, August 11): Fine-Tuning Llama-2: A Comprehensive Case Study for Tailoring Models to Unique Applications, anyscale, https://www.anyscale.com/blog/fine-tuning-llama-2-a-comprehensive-case-study-for-tailoring-models-to-unique-applications

Hochreiter, S., Schmidhuber, J. (1997): Long short-term memory, Neural Computation, Vol. 9, pp. 1735-1780, https://www.researchgate.net/publication/13853244_Long_Short-term_Memory

Holmes, W., Bialik, M., & Fadel, C. (2019): Artificial Intelligence in Education: Promises and Implications for Teaching and Learning, Center for Curriculum Redesign, https://www.researchgate.net/publication/332180327_Artificial_Intelligence_in_Education_Promise_and_Implications_for_Teaching_and_Learning

Houlsby, N., Giurgiu, A., Jastrzebski, S., Morrone, B., de Laroussilhe, Q., Gesmundo, A., Attariyan, M., Gelly, S. (2019, February 2): Parameter-Efficient Transfer Learning for NLP, arXiv, https://arxiv.org/pdf/1902.00751.pdf

Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., Chen, W. (2021, June 17): LoRA: Low-Rank Adaptation of Large Language Models, arXiv, https://arxiv.org/abs/2106.09685

Hu, Z., Wang, L., Lan, Y., Xu, W., Lim, E., Bing, L., Xu, X., Poria, S., Lee, R. (2023, April 4): LLM-Adapters: An Adapter Family for Parameter-Efficient Fine-Tuning of Large Language Models, arXiv, https://arxiv.org/abs/2304.01933

Huang, Q., Wan, Z., Xing, Z., Wang, C., Chen, J., Xu, X., Lu, Q. (2023, September 28): Let's Chat to Find the APIs: Connecting Human, LLM and Knowledge Graph through AI Chain, arXiv, https://arxiv.org/abs/2309.16134

Hugging Face (n.D.): Supervised Fine-tuning Trainer, Hugging Face, https://huggingface.co/docs/trl/en/sft_trainer

Hugging Face (n.D.a): Quantization documentation, Hugging Face, https://huggingface.co/docs/transformers/en/main_classes/quantization

Hugging Face (n.D.b): AutoTokenizer documentation, Hugging Face, https://huggingface.co/docs/transformers/v4.39.3/en/autoclass_tutorial#autotokenizer

Hugging Face (n.D.c): Llama2 documentation, Hugging Face, https://huggingface.co/docs/transformers/model_doc/llama2

Hugging Face (n.D.d): Loading a Dataset, Hugging Face, https://huggingface.co/docs/datasets/v1.11.0/loading_datasets.html

Hugging Face (n.D.e): GGUG-MY-REPO, Hugging Face, https://huggingface.co/spaces/ggml-org/gguf-my-repo

IBM Data and AI Team (2023a, September 27): Open source large language models: Benefits, risks and types, IBM, https://www.ibm.com/blog/open-source-large-language-models-benefits-risks-and-types/

IBM Data and AI Team (2023b, October 16): Shedding light on AI bias with real world examples, IBM, https://www.ibm.com/blog/shedding-light-on-ai-bias-with-real-world-examples/

Jawade, B. (2023, December 22): Understanding LoRA — Low Rank Adaptation For Finetuning Large Models, Towards Data Science, https://towardsdatascience.com/understanding-lora-low-rank-adaptation-for-finetuning-large-models-936bce1a07c6

Jentzsch, S., Turan, C. (2023, June 27): Gender Bias in BERT -- Measuring and Analysing Biases through Sentiment Rating in a Realistic Downstream Classification Task, arXiv, https://arxiv.org/abs/2306.15298

Jiang, A. Q., Sablayrolles, A., Mensch, A., Bamford, C., Chaplot, D. S., de las Casas, D., Bressand, F., Lengyel, G., Lample, G., Saulnier, L., Lavaud, L. R., Lachaux, M.-A., Stock, P., Le Scao, T., Lavril, T., Wang, T., Lacroix, T., El Sayed, W. (2023, October 10): Mistral 7B, arXiv, https://arxiv.org/abs/2310.06825

JMP Statistical Discovery (n. D.): The Paired t-Test, JMP Statistical Discovery, https://www.jmp.com/en_be/statistics-knowledge-portal/t-test/paired-t-test.html

Jo, E. S., Gebru, T. (2019, December 22): Lessons from Archives: Strategies for Collecting Sociocultural Data in Machine Learning, arXiv, https://arxiv.org/abs/1912.10389

John, R. (2023, September 4): 2 easy ways for fine-tuning LLAMA-v2 and other Open source LLMs, Medium, https://trojrobert.medium.com/4-easier-ways-for-fine-tuning-llama-2-and-other-open-source-llms-eb3218657f6e

Kaddour, J., Harris, J., Mozes, M., Bradley, H., Raileanu, R., McHardy, R. (2023): Challenges and Applications of Large Language Models, arXiv, https://arxiv.org/abs/2307.10169

Kadous, W., Hakhamaneshi, K. (2023, July 5): Fine tuning is for form, not facts, anyscale, https://www.anyscale.com/blog/fine-tuning-is-for-form-not-facts

Kandel, I., Castelli, M. (2020): The effect of batch size on the generalizability of the convolutional neural networks on a histopathology dataset, ICT Express, Vol. 6, Issue 4, pp. 312-315, https://doi.org/10.1016/j.icte.2020.04.010

Kim, S., Moon, S., Tabrizi, R., Lee, N., Mahoney, M. W., Keutzer, K., Gholami, A. (2023, December 7): An LLM Compiler for Parallel Function-calling, arXiv, https://arxiv.org/abs/2312.04511

Kim, S., Moon, S., Tabrizi, R., Lee, N., Mahoney, M., Keutzer, K., Gholami, A. (2023, December 7): An LLM Compiler for Parallel Function-calling, arXiv, https://arxiv.org/abs/2312.04511

Kuo, C. (2023, June 19): Fine-tuning a GPT — LoRA, Medium, https://dataman-ai.medium.com/fine-tune-a-gpt-lora-e9b72ad4ad3

Kwak, N., Kim, T. (2024, January 29): X-PEFT: eXtremely Parameter-Efficient Fine-Tuning for Extreme Multi-Profile Scenarios, arXiv, https://arxiv.org/abs/2401.16137

Kwon, Y., Wu, E., Wu, K., Zou, J. (2023, October 2): DataInf: Efficiently Estimating Data Influence in LoRA-tuned LLMs and Diffusion Models, arXiv, https://arxiv.org/abs/2310.00902

Laker, B. (2023, November 21): Adapting To AI: Interesting Insights From LinkedIn On The Job Market, Forbes, https://www.forbes.com/sites/benjaminlaker/2023/11/21/adapting-to-ai-interesting-insights-from-linkedin-on-the-job-market/

Lee, J., Yoon, W., Kim, S., Kim, D., Kim, S., So, C. H., Kang, J. (2020): BioBERT: a pre-trained biomedical language representation model for biomedical text mining, Bioinformatics, Vol. 36, Issue 4, pp. 1234-1240, https://academic.oup.com/bioinformatics/article/36/4/1234/5566506

Lewis, M., Liu, Y., Goyal, N., Ghazvininejad, M., Mohamed, A., Levy, O., Stoyanov, V., Zettlemoyer, L. (2019, October 29): BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension, arXiv, https://arxiv.org/abs/1910.13461

Li, M., Zhao, Y., Yu, B., Song, F., Li, H., Yu, H., Li, Z., Huang, F., Li, Y. (2023, April 14): API-Bank: A Comprehensive Benchmark for Tool-Augmented LLMs, arXiv, https://arxiv.org/abs/2304.08244

Li, X. L., Liang, P. (2021, January 1): Prefix-Tuning: Optimizing Continuous Prompts for Generation, arXiv, https://arxiv.org/abs/2101.00190

Li, Z., Chen, Z. Z., Ross, M., Huber, P., Moon, S., Lin, Z., Dong, X. L., Sagar, A., Yan, X., Crook, P. A. (2024, February 16): Large Language Models as Zero-shot Dialogue State Tracker through Function-calling, arXiv, https://arxiv.org/abs/2402.10466

Lin, X., Wang, W., Li, Y., Yang, S., Feng, F., Wei, Y., Chua, T.-S. (2024, January 30): Data-efficient Fine-tuning for LLM-based Recommendation, arXiv, https://arxiv.org/abs/2401.17197

Liu, S., Zeng, S., Li, S. (2020): Evaluating Text Coherence at Sentence and Paragraph Levels, arXiv, https://arxiv.org/abs/2006.03221

Llama API (n. D.): Function-calling, Llama Api, https://docs.llama-api.com/essentials/function

Llama.cpp (n. D.): llama.cpp, GitHub, https://github.com/ggerganov/llama.cpp

Llama-cpp-python (n. D.): llama-cpp-python, GitHub, https://github.com/abetlen/llama-cpp-python

Longpre, S., Hou, L., Vu, T., Webson, A., Chung, H. W., Tay, Y., Zhou, D., Le, Q. V., Zoph, B., Wei, J., Roberts, A. (2023, January 31): The Flan Collection: Designing Data and Methods for Effective Instruction Tuning, arXiv, https://arxiv.org/abs/2301.13688

Lyu, K., Zhao, H., Gu, X., Yu, D., Goyal, A., Arora, S. (2024, February 28): Keeping LLMs Aligned After Fine-tuning: The Crucial Role of Prompt Templates, arXiv, https://arxiv.org/abs/2402.18540

Mangrulkar, S., Gugger, S., Debut, L., Belkada, Y., Paul, S., Bossan, B. (2022): PEFT: State-of-the-art Parameter-Efficient Fine-Tuning methods, GitHub, https://github.com/huggingface/peft

McGregor, S., Charters, P., Holliday, J., Roesner, F. (2017): Investigating the Computer Security Practices and Needs of Journalists, USENIX Security Symposium, https://www.usenix.org/conference/usenixsecurity15/technical-sessions/presentation/mcgregor

Meta (n.D.): Llama 2: open source, free for research and commercial use, Meta, https://llama.meta.com/llama2

Mistral AI Team (2023, September 27): Mistral 7B - The best 7B model to date, Apache 2.0, Mistral AI, https://mistral.ai/news/announcing-mistral-7b/

Morrison, R. (2024, March 27): Claude takes the top spot in AI chatbot ranking — finally knocking GPT-4 down to second place, Tom's Guide, https://www.tomsguide.com/ai/claude-takes-the-top-spot-in-ai-chatbot-ranking-finally-knocking-gpt-4-down-to-second-place

Nagpal, A., Gabrani, G. (2019): Python for Data Analytics, Scientific and Technical Applications, 2019 Amity International Conference on Artificial Intelligence, pp. 140-145, DOI: 10.1109/AICAI.2019.8701341

Nickolls, J., Buck, I., Garland, M., Skadron, K. (2008): Scalable parallel programming with CUDA, IEEE Hot Chips Symposium, Vol. 6, Issue 2, pp. 40–53. https://api.semanticscholar.org/CorpusID:7917593

Niederfahrenhorst, A., Hakhamaneshi, K., Ahmad, R. (2023, September 6): Fine-Tuning LLMs: LoRA or Full-Parameter? An in-depth Analysis with Llama 2, anyscale, https://www.anyscale.com/blog/fine-tuning-llms-lora-or-full-parameter-an-in-depth-analysis-with-llama-2

OpenAI (2023): GPT-4 Technical Report, arXiv, https://arxiv.org/html/2303.08774v4

OpenAI (n. D.): Fine-tuning, OpenAI, https://platform.openai.com/docs/guides/fine-tuning

OpenAI (n. D.a): Models, OpenAI, https://platform.openai.com/docs/models

Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C. L., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Ray, A., Schulman, J., Hilton, J., Kelton, F., Miller, L., Simens, M., Askell, A., Welinder, P., Christiano, P., Leike, J., Lowe, R. (2022, March 4): Training language models to follow instructions with human feedback, arXiv, https://arxiv.org/abs/2203.02155

Pineau, J., Vincent-Lamarre, P., Sinha, K., Larivière, V., Beygelzimer, A., d'Alché-Buc, F., Fox, E., Larochelle, H. (2021): Improving reproducibility in machine learning research, Journal of Machine Learning Research, Vol. 22, Issue 1, pp. 1-20, https://arxiv.org/abs/2003.12206

Qi, X., Zeng, Y., Xie, T., Chen, P.-Y., Jia, R., Mittal, P., Henderson, P. (2023, October 5): Fine-tuning Aligned Language Models Compromises Safety, Even When Users Do Not Intend To!, arXiv, https://arxiv.org/abs/2310.03693

Radford, A., Narasimhan, K., Salimans, T., Sutskever, I. (2018, June 11). Improving language understanding by generative pre-training, OpenAI, https://cdn.openai.com/research-covers/language-unsupervised/language_understanding_paper.pdf

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., Liu, P. J. (2020): Exploring the limits of transfer learning with a unified text-to-text transformer, Journal of Machine Learning Research, Vol. 21, No. 140, pp. 1-67, https://jmlr.org/papers/volume21/20-074/20-074.pdf

Raj J., M., VM, K., Warrier, H., Gupta, Y. (2024, March 23): Fine Tuning LLM for Enterprise: Practical Guidelines and Recommendations, arXiv, https://arxiv.org/abs/2404.10779

Raman, S. (2023, December 21): The Rise of AI-Powered Applications: Large Language Models in Modern Business, IEEE Computer Society, https://www.computer.org/publications/tech-news/trends/large-language-models-in-modern-business

Raschka, S. (2023, March 28): Finetuning LLMs on a Single GPU Using Gradient Accumulation, Lightning.AI, https://lightning.ai/blog/gradient-accumulation/

Raschka, S. (2023a, October 12): Finetuning LLMs with LoRA and QLoRA: Insights from Hundreds of Experiments, Lightning.AI, https://lightning.ai/pages/community/lora-insights/

Roberts, A. (2023a, October 10): Prepare for Extraction, arize, https://arize.com/blog-course/structured-data-extraction-openai-function-calling/

Roberts, A. (2023b, December 7): Calling All Functions, arize, https://arize.com/blog/calling-all-functions-benchmarking-openai-function-calling-and-explanations/

Ruder, S., Peters, M. E., Swayamdipta, S., Wolf, T. (2019): Transfer Learning in Natural Language Processing, Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Tutorials, pp. 15–18, Minneapolis, Minnesota. Association for Computational Linguistics, https://aclanthology.org/N19-5004/#:~:text=This%20approach%20requires%20a%20large,model%20with%20better%20generalization%20properties.

Run.AI (n. D.): LLaMA 2 Fine Tuning, Run.Ai, https://www.run.ai/guides/generative-ai/llama-2-fine-tuning

Schmid, P. (2023, July 26): Extended Guide: Instruction-tune Llama 2, philschmid, https://www.philschmid.de/instruction-tune-llama-2

Shah, C., White, R., Andersen, R., Buscher, G., Counts, S., Das, S., Montazer, A., Manivannan, S., Neville, J., Ni, X., Rangan, N., Safavi, T., Suri, S., Wan, M., Wang, L., Yang, L. (2023, September 14): Using Large Language Models to Generate, Validate, and Apply User Intent Taxonomies, arXiv, https://arxiv.org/abs/2309.13063

Shickel, B., Tighe, P., Bihorac, A., Rashidi, P. (2017, June 17): Deep EHR: A Survey of Recent Advances in Deep Learning Techniques for Electronic Health Record (EHR) Analysis, arXiv, https://arxiv.org/abs/1706.03446

Spirling, A. (2023, April 18): Why open-source generative AI models are an ethical way forward for science, Nature, https://www.nature.com/articles/d41586-023-01295-4

Srinivasan, V. K., Dong, Z., Zhu, B., Yu, B., Mosk-Aoyama, D., Keutzer, K., Jiao, J., Zhang, J. (2023, November 7): A Commercially-Permissive Language Model for Function-calling, NeurIPS 2023 Foundation Models for Decision Making Workshop, https://openreview.net/forum?id=5lcPe6DqfI

Taoofstefan (2024): function_call_weather, Hugging Face, https://huggingface.co/datasets/taoofstefan/function_call_weather

Taoofstefan (2024a): base evaluation, GitHub, https://github.com/taoofstefan/Fine-Tuning-Experiment/tree/main/results/base%20evaluation

Taoofstefan (2024b): fine tuned evaluation, GitHub, https://github.com/taoofstefan/Fine-Tuning-Experiment/tree/main/results/fine%20tuned%20evaluation

Taoofstefan (2024c): Statistical Evaluation, GitHub, https://github.com/taoofstefan/Fine-Tuning-Experiment/blob/main/Code/Evaluation/Statistical%20Evaluation.ipynb

Taoofstefan (2024d): Visuals, GitHub, https://github.com/taoofstefan/Fine-Tuning-Experiment/blob/main/Code/Evaluation/Visuals.ipynb

Taoofstefan (2024e): base model testing, GitHub, https://github.com/taoofstefan/Fine-Tuning-Experiment/tree/main/Code/base%20model%20testing

Taoofstefan (2024f): Fine tuning and testing, GitHub, https://github.com/taoofstefan/Fine-Tuning-Experiment/tree/main/Code/Fine%20tuning%20and%20testing

Taoofstefan (2024g): taoofstefan/llama2-7B-MT-FT-llama2-S-v0.3, Hugging Face, https://huggingface.co/taoofstefan/llama2-7B-MT-FT-llama2-S-v0.3

Taoofstefan (2024h): taoofstefan/llama2-7B-MT-FT-llama2-S-v0.3-Q4_K_M-GGUF, Hugging Face, https://huggingface.co/taoofstefan/llama2-7B-MT-FT-llama2-S-v0.3-Q4_K_M-GGUF

Taoofstefan (2024i): taoofstefan/mistral-7B-MT-FT-Mistral-S, Hugging Face, https://huggingface.co/taoofstefan/mistral-7B-MT-FT-Mistral-S

Taoofstefan (2024j): https://huggingface.co/taoofstefan/mistral-7B-MT-FT-Mistral-S-Q4_K_M-GGUF, Hugging Face, https://huggingface.co/taoofstefan/mistral-7B-MT-FT-Mistral-S-Q4_K_M-GGUF

Topol, E. J. (2019, January 7): High-performance medicine: the convergence of human and artificial intelligence, Nature Medicine, Vol. 25, pp. 44-56, https://www.nature.com/articles/s41591-018-0300-7

Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., Bashlykov, N., Batra, S., Bhargava, P., Bhosale, S., Bikel, D., Blecher, L., Ferrer, C., Chen, M., Cucurull, G., Esiobu, D., Fernandes, J., Fu, J., Fu, W., Fuller, B., Gao, C., Goswami, V., Goyal, N., Hartshorn, A., Hosseini, S., Hou, R., Inan, H., Kardas, M., Kerkez, V., Khabsa, M., Kloumann, I., Korenev, A., Koura, P., Lachaux, M., Lavril, T., Lee, J., Liskovich, D., Lu, Y., Mao, Y., Martinet, X., Mihaylov, T., Mishra, P., Molybog, I., Nie, Y., Poulton, A., Reizenstein, J., Rungta, R., Saladi, K., Schelten, A., Silva, R., Smith, E., Subramanian, R., Tan, X., Tang, B., Taylor, R., Williams, A., Kuan, J., Xu, P., Yan, Z., Zarov, I., Zhang, Y., Fan, A., Kambadur, M., Narang, S., Rodriguez, A., Stojnic, R., Edunov, S., Scialom, T. (2023, July 18): Llama 2: Open Foundation and Fine-Tuned Chat Models, arXiv, https://arxiv.org/abs/2307.09288

UNESCO (2023, September 21): UNESCO Recommendation on Open Science, UNESCO, https://www.unesco.org/en/open-science/about

Utemuratov, P. (2024, January 26): Enhancing Open-Source LLMs with Function-calling Feature, deepinfra, https://deepinfra.com/blog/function-calling-feature

Vaid, A., Lampert, J., Lee, J., Sawant, A., Apakama, D., Sakhuja, A., Soroush, A., Lee, D., Landi, I., Bussola, N., Nabeel, I., Freeman, R., Kovatch, P., Carr, B., Glicksberg, B., Argulian, E., Lerakis, S., Kraft, M., Charney, A., Nadkarni, G. (2024, January 5): Generative Large Language Models are autonomous practitioners of evidence-based medicine, arXiv, https://arxiv.org/abs/2401.02851

van den Berg, R. G. (n.D.): SPSS Shapiro-Wilk Test – Quick Tutorial with Example, SPSS TUTORIALS, https://www.spss-tutorials.com/spss-shapiro-wilk-test-for-normality/

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., Polosukhin, I. (2017, June 12). Attention is all you need, arXiv, https://arxiv.org/abs/1706.03762

Wang, Y., Si, S., Li, D., Lukasik, M., Yu, F., Hsieh, C., Dhillon, I., Kumar, S. (2024, January 16): Two-stage LLM Fine-tuning with Less Specialization and More Generalization, arXiv, https://arxiv.org/abs/2211.00635

Wang, Z. Z., Cheng, Z., Zhu, H., Fried, D., Neubig, G. (2024): What Are Tools Anyway? A Survey from the Language Model Perspective, GitHub, https://zorazrw.github.io/files/WhatAreToolsAnyway.pdf

Weis, T. (2023, October 12): From Chatbots to Agents: Augmenting LLMs With Tools, dataiku, https://blog.dataiku.com/from-chatbots-to-agents-augmenting-llms-with-tools

Wilcoxon signed-rank test, n.D.): Wilcoxon signed-rank test, Wikipedia, https://en.wikipedia.org/wiki/Wilcoxon_signed-rank_test

Williams, N., Ivanov, S., Buhalis, D. (2023, March 10): Algorithmic Ghost in the Research Shell: Large Language Models and Academic Knowledge Creation in Management Research, arXiv, https://arxiv.org/abs/2303.07304

Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., Davison, J., Shleifer, S., von Platen, P., Ma, C., Jernite, Y., Plu, J., Xu, C., Le Scao, T., Gugger, S., Drame, M., Lhoest, Q., Rush, A. M. 2019, October 9): HuggingFace's Transformers: State-of-the-art Natural Language Processing, arXiv, https://arxiv.org/abs/1910.03771

Wu, T., Luo, L., Li, Y.-F., Pan, S., Vu, T.-T., Haffari, G. (2024, February 2): Continual Learning for Large Language Models: A Survey, arXiv, https://arxiv.org/abs/2402.01364

Yan, F., Mao, H., Ji, C. C.-J., Zhang, T., Patil, S. G., Stoica, I., Gonzalez, J. E. (2024): Berkeley Function-calling Leaderboard, berkeley.edu, https://gorilla.cs.berkeley.edu/blogs/8_berkeley_function_calling_leaderboard.html

Yang, K., Liu, J., Wu, J., Yang, C., Fung, Y. R., Li, S., Huang, Z., Cao, X., Wang, X., Wang, Y., Ji, H., Zhai, C. (2024, January 1): If LLM Is the Wizard, Then Code Is the Wand: A Survey on How Code Empowers Large Language Models to Serve as Intelligent Agents, arXiv, https://arxiv.org/abs/2401.00812

Yen, L. (2019, January 26): An Introduction to the Bootstrap Method, Towards Data Science, https://towardsdatascience.com/an-introduction-to-the-bootstrap-method-58bcb51b4d60

Yuan, S., Song, K., Chen, J., Tan, X., Shen, Y., Kan, R., Li, D., Yang, D. (2024, January 11): EASYTOOL: Enhancing LLM-based Agents with Concise Tool Instruction, arXiv, https://arxiv.org/abs/2401.06201

Zhang, B., Liu, Z., Cherry, C., Firat, O. (2024, February 27): When Scaling Meets LLM Finetuning: The Effect of Data, Model and Finetuning Method, arXiv, https://arxiv.org/abs/2402.17193

Zhang, Y., Li, P., Hong, J., Li, J., Zhang, Y., Zheng, W., Chen, P.-Y., Lee, J. D., Yin, W., Hong, M., Wang, Z., Liu, S., Chen, T. (2024, February 18): Revisiting Zeroth-Order Optimization for Memory-Efficient LLM Fine-Tuning: A Benchmark, arXiv, https://arxiv.org/abs/2402.11592

Zhang, Z., Zheng, C., Tang, D., Sun, K., Ma, Y., Bu, Y., Zhou, X., Zhao, L. (2023, October 7): Balancing Specialized and General Skills in LLMs: The Impact of Modern Tuning and Data Strategy, arXiv, https://arxiv.org/abs/2310.04945

Zhao, W. X., Zhou, K., Li, J., Tang, T., Wang, X., Hou, Y., Min, Y., Zhang, B., Zhang, J., Dong, Z., Du, Y., Yang, C., Chen, Y., Chen, Z., Jiang, J., Ren, R., Li, Y., Tang, X., Liu, Z., Liu, P., Nie, J.-Y., Wen, J.-R. (2023): A Survey of Large Language Models, arXiv, https://arxiv.org/abs/2303.18223

Zhao, Y., Varma, R., Huang, C.-C., Li, S., Xu, M., Desmaison, A. (2022, March 14): Introducing PyTorch Fully Sharded Data Parallel (FSDP) API, Pytorch, https://pytorch.org/blog/introducing-pytorch-fully-sharded-data-parallel-api/

Zhuang, Y., Yu, Y., Wang, K., Sun, H., Zhang, C. (2023): ToolQA: A Dataset for LLM Question Answering with External Tools, arXiv, https://arxiv.org/abs/2306.13304

Zinkula, J., Mok, A. (2024, March 6): ChatGPT may be coming for our jobs. Here are the 10 roles that AI is most likely to replace., Business Insider, https://www.businessinsider.com/chatgpt-jobs-at-risk-replacement-artificial-intelligence-ai-labor-trends-2023-02