ESIPS - Draft Report

MOHITHA MOHAN

B.Eng (Hons) (Software)

Academic Supervisor: Dr. Guodong Shi
Industry Supervisor: Oscar Fawkes

A thesis submitted in fulfilment of
the requirements for the degree of
Bachelor of Engineering Honours (Software)

School of Electrical Engineering
Faculty of Engineering
The University of Sydney
Australia

4 August 2025

Contents

Contents ii

Chapter 1 Introduction 1

1.1 Motivation 1

1.2 Problem Statement 4

1.3 Objectives 5

1.4 Contributions 6

1.5 Outline 7

Chapter 2 Literature review 8

2.1 Artificial Intelligence in Banking 8

2.2 History of AI & AI literature 9

2.3 Language Models & Large Language Models (LLMs) 10

2.3.1 History of Language Models 11

2.3.2 State-of-the-Art Models 12

2.3.3 LLMs in Business 13

2.3.4 Augmentation of LLMs 14

2.4 Tool Learning & Tool Usage 15

2.4.1 Task Planning 17

2.4.2 Tool Selection 18

2.4.3 Tool Calling 19

2.4.4 Response Generation 20

2.4.5 Benchmarks 21

2.5 Model Context Protocol (MCP) for Standardisation 22

2.5.1 MCP Within AI Tooling 22

2.5.2 Architecture, Design and Primitives 24

2.5.3 Challenges 25
2.6 Conclusion 26

Chapter 3 Research Design **27**
3.1 Experimentation Set Up 28
3.2 Datasets 28
3.3 Evaluation Methodology & AI System 29
3.4 Metrics 30
3.5 Models 31
3.6 MCP Servers 32

Chapter 4 Experimentation **34**
4.1 Evaluation Framework - 1 34
4.2 Evaluation Framework - 2 36

Chapter 5 Data & Results **38**

Chapter 6 Analysis & Discussion **40**

Chapter 7 Conclusion & Future Work **41**

Chapter 8 Case Study **43**
8.1 COMP4318 - Machine Learning & Data Mining 43
8.2 COMP4328 - Advanced Machine Learning 44

References **47**

CHAPTER 1

Introduction

1.1 Motivation

The first large-scale neural language model was introduced by Google in 2015 [1]. A couple of years later in 2017, the world saw a breakthrough with the self-attention mechanism at the heart of transformers discussed in Vaswani et al. [2]. Now, large language models (LLMs) as we know them, in products like Microsoft Copilot and your everyday ChatGPT, came about from 2019 onwards with models such as OpenAI's GPT series and Google's BERT, which have been stepping stones to the power of LLMs we see these days.

There is no doubt that LLMs are extremely powerful in generating human-like text and solving a wide range of tasks, but they exhibit several limitations, including the inability to access and retrieve up-to-date knowledge after training, a tendency to hallucinate information, and incapability to interact with the real world. They have also shown to be incapable of complex reasoning tasks such as multi-step logical deduction and causal inference.

In response, the augmentation of LLMs—particularly through the integration of external tools—has recently become a prominent area of research and innovation. A tool is defined as any external functional interface that to a system [3], and most of the time such tools are in the form of application programming interfaces, or APIs for short. From 2023 to late 2024, most implementations of tool-learning or tool-augmentation in LLMs involved custom integrations of APIs within the AI system to help train and evaluate them [3]. This required the collection of diverse public APIs for datasets, manual coding to handle input and output to the API calls, and robust error management. These integration challenges make it difficult to scale and

standardise tool learning in research. But in November 2024, Anthropic introduced the model context protocol (MCP) [**IntroducingAnthropicngAnthropic**] an open-standard framework to designed to streamline the interactions between LLMs and external tools or data sources. The protocol adopts a client-server architecture inspired by the language server protocol (LSP) - a widely adopted standard for connecting code editors to language services—but extends this paradigm to AI systems, enabling bidirectional communication with databases, APIs, and other resources. While the long term success of MCP hinges on ecosystem adoption, it's early uptake by major industry players including OpenAI (March 2025) and Google DeepMind (April 2025) shows promising momentum. Some experts compare MCP in AI to significant standards like HTTPS for websites or USB-C for devices. However, for MCP to really succeed, there will need to be broader developer engagement and mitigation of identified security risks of having external integrations

When it comes to evaluating tool-augmented LLMs or AI systems that use external tools prior to the introduction of MCP, one of the earliest benchmarks—APIBank [**Li2023API-Bank:LLMs**]—noted that, despite LLMs demonstrating impressive abilities in conversation and reasoning, their skill in interacting with external tools and servers remains poorly understood. This gap highlights the importance of developing robust evaluation methods to accurately assess and advance tool-augmented LLM capabilities. Considering evaluation consists of 3 main pillars, datasets, metric selection and the evaluation methodology [**Rudd2025ASystems**], APIBank alongwith most other benchmarks, require generation of large training data that includes functioning API calls, and response-answer pairs sometimes even with multi-dialogue sequences for their datasets. Most of these datasets, like in the GAIA benchmark [**Mialon2023GAIA:Assistants**] have unambiguous question-answer pairs with either a phrase or word or number as the answer, even if there are multiple tool calls made. What this means is that the dataset for evaluation is focused on retrieving a single ground truth answer to check the tool-usage capability of the LLM, limiting evaluation of real use cases which may or may not have a single ground truth as the answer. Even in ACEBench [**Chen2025ACEBench:Usage**] where they have split their test samples into normal, special and agent to handle instructions or prompts that might be incomplete or require multi-dialogue sequences, the end answer is always a single truth. Similarly, the metrics for such benchmarks include numerical or term similarity/overlaps, for

example the number of tools calls that were correctly made and whether or not the response has a specific word/answer in it. LLM-as-a-judge is commonly used for this comparision as well.

This brings about the need to introduce more representative datasets and metrics that adequately take into consideration that industry and business use cases more often than not won't amount to a single ground truth answer. This can be addressed by changing either the datasets, metrics or both. In [**Brasoveanu2020RubricRubrics**], the paper attempts to check code answers from university tests by using a exam rubric-style system where each question is marked against a specific rubric. Similarly in the mixture-of-experts model KimiK2 [**TeamKIMIK2**], they've utilised a rubric-style evaluation method in the post-training of KimiK2 to better it's tool calling capability, thus showing promise in the direction of question-specific rubrics in the dataset for more comprehensive evaluation. There is also a huge opportunity to utilise LLM autorater metrics, i.e. LLM-as-a-judge given its widespread adoption in recent research for evaluating outputs that require nuanced or subjective assessment beyond simple correctness.

Evaluation of LLMs that use tools via MCP however is still not very researched, and those that have done so, like MCPBench [**Luo2025EvaluationServers**] and MCPRadar [**Gao2025MCP-RADAR:Models**] still stick to unambiguous ground truths with numerical metrics and purely data-retrieval functionalities; hence allowing a huge area of research. Even [**Chang2023AModels**] states that tool usage evaluation is a significant and important research gap to address.

Despite the introduction of MCP, the evaluation of LLMs that use tools via this protocol remains largely unexplored. Existing studies, such as MCPBench [**Luo2025EvaluationServers**] and MCPRadar [**Gao2025MCP-RADAR:Models**], continue to rely on unambiguous ground truths, numerical metrics, and focus primarily on data-retrieval tasks. This leaves a substantial gap in the research landscape. As noted by [**Chang2023AModels**], the development of robust evaluation methods for tool-augmented LLMs is a significant and pressing area for future work especially with MCP.

1.2 Problem Statement

The evaluation of tool-augmented large language models presents several critical challenges that limit our understanding of their real-world capabilities and hinder their effective deployment in industry applications.

First, existing evaluation benchmarks for tool-augmented LLMs rely heavily on datasets with unambiguous, single ground-truth answers, typically consisting of specific phrases, words, or numbers. While this approach simplifies assessment, it fails to capture the complexity of real-world use cases where problems may have multiple valid solutions or require nuanced, context-dependent responses. This limitation is evident across prominent benchmarks including APIBank, GAIA, and ACEBench, which despite their contributions to the field, do not adequately represent the diverse and often ambiguous nature of practical tool usage scenarios.

Second, current evaluation metrics predominantly focus on numerical similarity measures and binary correctness checks, such as counting correct tool calls or detecting specific terms in responses. While these metrics provide clear, objective measures, they are insufficient for evaluating the quality, appropriateness, and contextual relevance of responses in complex, multi-faceted scenarios that characterize real business and industry applications.

Third, the introduction of the Model Context Protocol (MCP) by Anthropic has created new opportunities for standardized tool integration, yet evaluation methodologies have not evolved to adequately assess LLM performance within this framework. Existing MCP-focused studies like MCPBench and MCPRadar continue to employ traditional evaluation approaches that emphasize data retrieval tasks with clear ground truths, leaving a significant gap in understanding how LLMs perform across the broader spectrum of tool-augmented capabilities that MCP enables.

Additionally, there remains insufficient research into how the availability of multiple tools affects LLM performance. As AI systems are deployed with access to increasingly diverse tool ecosystems, understanding potential performance degradation—including the system's

ability to select appropriate tools and generate semantically accurate responses—becomes crucial for optimizing tool selection and system design.

These limitations collectively hinder the development of robust, industry-ready tool-augmented LLM systems and impede researchers' ability to accurately assess and improve these systems for real-world deployment.

1.3 Objectives

Given the identified research gaps outlined above, this paper aims to develop and evaluate novel assessment methodologies that better represent real industry use-cases through modified datasets and enhanced metrics for measuring tool-usage capabilities in AI systems.

This study will implement and test two distinct evaluation frameworks designed to address current limitations in tool-augmented LLM assessment:

(1) Dual-metric framework: Combining semantic similarity evaluation with LLM-as-a-judge assessment of prompt adherence to determine whether responses adequately address the input question.
(2) Adaptive rubric-based framework: Implementing a question-dependent rubric system that dynamically adjusts evaluation criteria based on the specific requirements and context of each individual question.

Additionally, this paper will investigate the relationship between tool availability and system performance by examining whether an increase in the number of tools provided to the LLM results in performance degradation. Performance degradation is defined as a decline in both the system's ability to select and call appropriate tools and its capacity to generate semantically accurate responses that meet the expected quality standards.

Through these comprehensive evaluation approaches, this research seeks to establish more robust and representative methodologies for assessing tool-augmented LLM capabilities in real-world scenarios.

1.4 Contributions

The contributions of this research to the literature and field of tool-augmented LLM evaluation include:

(1) Novel evaluation methodologies: Development of two innovative evaluation frameworks that move beyond traditional single ground-truth assessments to better represent the complexity and ambiguity inherent in real-world industry applications of tool-augmented LLMs.
(2) Comprehensive model performance analysis: Systematic evaluation and comparison of multiple Claude Sonnet models (3.5, 3.7, and 4.0) across the proposed evaluation frameworks, providing insights into the relative strengths and capabilities of different LLM architectures in tool utilization scenarios.
(3) Tool scalability impact assessment: Empirical demonstration of how increasing the number of available tools affects LLM performance, specifically examining degradation in tool selection accuracy and response quality as the tool ecosystem expands.
(4) Methodological advancement for MCP-based systems:} Introduction of evaluation approaches specifically designed to assess tool-augmented LLMs operating within standardized frameworks like the Model Context Protocol, addressing a significant gap in current evaluation practices.
(5) Introduction of a systematic data generation pipeline to create synthetic datasets tailored for evaluating tool-augmented LLM performance, addressing the scarcity of appropriate evaluation data in this domain.

These contributions collectively advance the field's capacity to evaluate, understand, and optimize tool-augmented LLM systems for practical applications, while providing researchers and practitioners with more robust methodologies for assessing these increasingly important AI capabilities.

1.5 Outline

This thesis begins by introducing the motivation behind evaluating tool-augmented large language models (LLMs), with a particular focus on the limitations of current benchmarks and the emerging role of the Model Context Protocol (MCP). The introduction also defines the problem space and outlines the research objectives, followed by a summary of the thesis contributions.

The literature review that follows situates this work within the broader context of AI development, tracing the evolution of LLMs and their growing integration with external tools. It then examines existing methods for tool learning and the early adoption of MCP as a standardised interface. Building on this foundation, the research design section details the construction of two novel evaluation frameworks and the datasets used to simulate real-world tool use. This leads into the experimentation and analysis, where each framework is applied to assess LLM performance under varying tool availability. The thesis concludes with a discussion of key findings, observed performance degradation patterns, and directions for future work.

CHAPTER 2

Literature review

2.1 Artificial Intelligence in Banking

In the modern banking landscape, digital innovation has become increasingly important for financial institutions to stay up to date in the customer market and continue to be competitive. Lately, many of these innovations have been driven by the utilisation of artificial intelligence (AI) [4]. AI has fundamentally transformed the banking sector, with financial institutions increasingly adopting AI-based technologies across all banking disciplines, including the front (customer-facing voice-assistants and biometrics), middle (anti-fraud risk monitoring and complex legal workflows) and back offices (credit underwriting with smart contracts) [5]. This has led to the innovative disruption of banking channels, services and solutions.

Much of the existing research on artificial intelligence in banking has focused on its applications in business strategy, operational processes, and customer engagement [5], with automating manual laborious banking functionalities being at the forefront of what could be improved. In the past 2 decades as per KAUR et al. [6] and in relation to the processes theme in research, AI has primarily been used to analyse, approve and oversee customer loans. More recently there has also been ongoing research in human language analysis, image and voice recognition as well as deep learning of human emotions [5] which fits into the customer service research category. Fraud detection is another major domain of interest, which according to Mytnyk et al. [7] AI systems offer banks a much more advanced solution compared to traditional methods.

In addition to understanding its significant benefits, banks must also thoroughly understand the potential challenges associated with AI technology before planning and implementing it [8]. Organisational cultural barriers [9], complicated regulatory environments [5] and detailed privacy laws and regulations [10] are some of the biggest hurdles to the successful integration of AI functionality. It is also essential for organisational leaders to embrace AI, as this encourages senior executives to take responsibility and drive the transformation actively throughout the company [11].

Thus, while the use of AI is crucial for banks to remain competitive, successful integration demands a thorough understanding and acceptance of its role in the industry to ensure effective implementation [8]. In this context, agile fintech startups—unencumbered by the legacy systems and rigid structures that constrain traditional banks—are especially well positioned to lead the adoption of AI within the financial sector. Accordingly, this research is conducted from the unique perspective of operating inside such an agile fintech startup.

2.2 History of AI & AI literature

Artificial intelligence is a field of science concerned with building computers and machines that can reason, learn, and act in such a way that would normally require human intelligence or that involves data whose scale exceeds what humans can analyse [12]. The origins of AI can be traced back to the 1950s, notably with Alan Turing's seminal work, "Computing Machinery and Intelligence," in which he posed the question, "Can machines think?" and introduced the concept of the Turing Test as a measure of machine intelligence [13]. Later in the summer of 1956 at the Dartmouth Conference, John McCarthy and other members of the board officially coined the term "AI", formally establishing AI as a field [14].

The early years of artificial intelligence, from 1956 to 1974, are often referred to as the "golden years" [15] marked by optimism and significant progress in symbolic reasoning and problem-solving. However, limited computational resources and unmet expectations led to the first "AI winter" from 1974 to 1987, when funding and interest in the field sharply declined [15]. A brief revival occurred in the 1980s and 1990s, driven by advances in neural networks,

including the development of convolutional neural networks and the introduction of long short-term memory (LSTM) networks in 1997 [16]. Other notable milestones included IBM's Deep Blue defeating world chess champion Garry Kasparov in 1997 [17], signalling a new era in AI's capabilities. Despite this, AI research faced another downturn in the 2000s, often referred to as the second AI winter [15].

The trajectory of AI shifted dramatically again in the 2010s, with breakthroughs in deep learning and natural language processing. Key events include the triumph of IBM's Watson in 2011 [18] and Google DeepMind's AlphaGo defeating the world Go champion in 2016 [19]. The introduction of the transformer architecture in 2017 [2] further accelerated progress, enabling the development of powerful large language models. Most recently, the launch of ChatGPT in 2022 [20] brought generative AI to widespread public attention, highlighting the rapid development and increasing influence of AI technologies—a topic explored further in the following sections.

2.3 Language Models & Large Language Models (LLMs)

Language modelling (LM) is a fundamental part of natural language processing (NLP). Its main objective is to predict the next word or character in a sequence of text and to achieve this, researchers had to build models that could understand and generate meaningful human language. The core aim is to estimate the probability of different word combinations, which enables these models to generate new text, finish incomplete sentences, and assess how likely certain word sequences are.

Language models can be divided into 4 categories: statistical, machine learning, deep learning, and transformer-based models [21]. The development of large language models specifically was driven by advances in deep learning, the availability of vast datasets, and improvements in computational power. These modern models, built on transformer architectures, are trained on massive amounts of data, which allows them to understand and generate not only natural language but also other types of content, supporting a broad range of applications [21]. Transformers use neural networks with encoder and decoder components, both of which rely

on self-attention mechanisms. This structure enables the model to capture the meaning of text and the relationships between words and phrases—an important step forward compared to earlier approaches [22].

2.3.1 History of Language Models

The earliest language models, developed in the 1950s, were based on hand-crafted rules and linguistic features [23]. In the 1980s, statistical language models emerged, making use of probabilistic techniques to predict the next word in a sequence based on its context [24]. While these models outperformed rule-based systems, they still struggled with deeper understanding of meaning and context [25]. A major advance came in the 2010s with neural language models, which used deep learning methods to learn language patterns from large datasets. The first of these, the recurrent neural network language model (RNNLM), appeared in 2010 and improved the ability to model context and generate more natural text [26]. Soon after, Google's introduction of the Neural Machine Translation system in 2015 marked the first large-scale neural language model [1]. Although in this paragraph we have discussed the evolution of language models from rule-based to neural language models, the article "The Bitter Truth" by Rich Sutton in 2019 talks about how most research in AI is conducted as if the computation available to the agent were constant, but if you take that away, general methods that leverage higher computational power are always more effective. This opposing approach based on scaling computation by search and learning argues that progress in AI is better achieved by developing general-purpose algorithms that can take advantage of ever-increasing computational resources, rather than relying on hand-engineered, domain-specific solutions.

Building on this previous foundation, the most significant leap in LLMs occurred with the introduction of the transformer architecture by Vaswani et al. in 2017 [2]. The self-attention mechanism at the heart of transformers allowed for efficient parallel processing and the management of long-range dependencies, making it possible to train much larger models using multiple GPUs. This breakthrough paved the way for models such as OpenAI's GPT series and Google's BERT, which have set new benchmarks across a variety of NLP tasks [27].

2.3.2 State-of-the-Art Models

The evolution of LLMs from 2019 onwards has seen exponential growth in model size, enhanced multilingual and few-shot capabilities, innovations in training efficiency and a focus on practical, specialised applications [21][28].

The initial stages from 2019-2020 saw the development of models like GPT-3 (2020) from OpenAI [29] and T5 (2019) from Google [30]. GPT-3 had an enormous 175 billion parameters at that time, demonstrating the power of few-shot learning and establishing LLMs as meta-learners (algorithms that focus on learning general strategies, enabling them to adapt efficiently to new tasks by leveraging prior experiences) [29]. Another significant model from this era is mT5 (2020) [31], which demonstrated that large multilingual models can perform on par with their single-language counterparts.

In 2021 many of the models such as ERNIE 3.0 [32], CPM-2 [33] and Gopher [34] emphasised few-shot learning, modular architectures, and enhanced representation capabilities. Importantly, the strong multilingual performance of mT5 paved the way for later models such as ERNIE 3.0 and HyperCLOVA [35], which built on these advances to further improve multilingual capabilities and task-specific fine-tuning. 2022 brought about models that showcased increased parameter sizes and innovations in efficiency, like ERNIE 3.0 Titan [36] and GLaM [37]. GLaM also introduced the use of mixture-of-experts (MoE) within transformer layers, illustrating how models can retain high capability while reducing computational demands. DeepMind's Chinchilla highlighted the need to proportionally scale model size and training data for optimal performance [38].

In 2023, the landscape of large language models expanded to include diverse options such as UL2 [39], GPT-4 [40], Claude Instant [41], and GPT-4 Turbo [40], each offering advances in mode switching, multimodal abilities, and cost efficiency. GPT-4, notable for its 1.76 trillion parameters, demonstrated improved reliability and creativity, while models like Claude Instant and GPT-4 Turbo focused on speed and affordability. The emergence of specialised models, such as StarCoder [42] for programming and Mistral Tiny [43] for mobile devices, reflected a trend towards tailored AI solutions.

In 2024, models such as Gemini 1.5 [44], WebGPT [45], and LLaMA-2-Chat [46] continued to drive innovation in instruction tuning and safety. WebGPT is designed to provide factually accurate responses with supporting references [45], while LLaMA-2-Chat uses reinforcement learning from human feedback to improve safety and resist jailbreak attempts. The adoption of multilingual training, as seen in models like mT0 and BLOOMZ, remains a key trend shaping the next generation of large language models [47].

This progression of LLMs from 2019-2024 has been characterised by increases in model size, improvements in multilingual capabilities, enhanced few-shot learning and innovations in training methodologies. 2025 is expected to usher in further advancements, with a strong focus on agentic AI and multi-agent systems, enabling LLMs to not only process information but also autonomously coordinate, plan, and execute complex tasks across diverse domains [48]. We are likely to see greater integration of LLMs with external tools and APIs through standardised protocols like MCP, supporting more robust, context-aware, and interactive applications. Additionally, research is moving towards improving efficiency and scalability, with innovations such as mixture-of-experts architectures and memory-augmented models, which aim to deliver higher performance with reduced computational costs [21]. Overall, the next phase of LLM evolution will likely be defined by greater autonomy, collaboration, and adaptability, positioning these models as central components in the future of AI-driven applications.

2.3.3 LLMs in Business

While LLMs show heavy promises on various general tasks, it still remains to prove its effectiveness within business functions and the industry. Haque et al. [49] specifically brings forth the reality of bringing LLMs into real-world applications and how ready the business world is for this major implementation.

Currently, LLMs have been used in various phases of business operations including planning, implementation and decision-making. Some major areas LLMs have been known to do well

in are chatbots in customer service, translations [50] and within text and data analysis [51] to extract insights and process customer comments.

Several limitations of ChatGPT and large language models (LLMs) have been identified in recent literature. Zhou et al. [20] highlights issues such as reliability, logical reasoning, knowledge acquisition, and robustness. Azaria [52] further emphasises the tendency of LLMs to hallucinate and their inability to provide up-to-date information. Kocoń et al. [53] even finds that ChatGPT consistently lags behind specialised state-of-the-art models designed for specific tasks, underperforming by as much as 25%. Notably, the performance gap widens as task complexity increases. It also becomes increasingly clear that the more context the LLM has with regard to the business, the more value it outputs, reiterating the fact that contextual, customised knowledge is essential for optimal performance in domain specific tasks. However, most LLMs have context limitations, restricting the amount of relevant information the model can consider at once.

This suggests that the practical deployment of LLMs in specialised sectors such as banking remains challenging. Key obstacles include their tendency to hallucinate and their limitations in handling up-to-date or time-sensitive knowledge. Equipping LLMs with up-to-date and context-specific understanding in these sectors could help address the challenges outlined above. The following section outlines several effective strategies for tailoring LLMs to incorporate this crucial business-specific context.

2.3.4 Augmentation of LLMs

Mialon et al. [54] discusses augmented language models, which represent an evolution beyond traditional language models by equipping them with either (or both) enhanced reasoning capabilities or the ability to use external tools.

Teaching an LLM to inculcate enhanced reasoning can be done by prompting, using either recursive, few-shot or zero-shot techniques, or by manually fine-tuning the language models. Prompting is considered an excessively time and resource intensive task and is limited by LLM context windows. Fine-tuning on the other hand has proven to achieve remarkable

improvements and show that small scale instruction fine-tuned models can perform better than non-finetuned large scale models, especially in complex tasks where instruction following is important.

When it comes to expanding LLMs to utilise tools, it usually involves either calling another model to implement the task, or information retrieval itself from a database. Retrieval of this information is where retrieval-augmentation and querying search engines come into play, while if the LLM uses another model to act out the task, that results in what is known as an agentic AI system. Sapkota et al. [48], distinguishes agentic AI from conventional AI applications by highlighting the architectural leap from isolated, tool-augmented agents to orchestrated, multi-agent frameworks that can collaborate, communicate, and dynamically allocate responsibilities within a broader system. Similarly, multi-agent systems provide the foundational architecture for agentic AI, enabling distributed problem-solving through the interaction of autonomous agents [55].

A few ways of *how* to teach LMs with such capabilities as mentioned above, include approaches such as [54]:

(1) Supervision with fine-tuning or human written demonstrations
(2) Reinforcement learning with human-feedback or hardcoded reward functions.

So, although LLMs represent a remarkable advancement beyond prior language models, it is evident significant challenges and improvements remain, augmenting LLMs stands out as a promising direction for addressing these limitations.

2.4 Tool Learning & Tool Usage

As discussed in our above analysis, large language models (LLMs) demonstrate significant limitations due to their finite context windows, temporal knowledge constraints, and their propensity to hallucinate. Further, one of the biggest challenges with LLMs is the fact that they can only take input and output a result, they are unable to perform an action on results and improve their responses based on feedback from doing these actions. This lack of interactivity

limits their ability to carry out tasks, adapt to dynamic environments, or iteratively refine their responses based on the success or failure of previous actions. Tool learning represents a sophisticated augmentation methodology designed to address these fundamental challenges, or at minimum, substantially mitigate their effects on the model performance and reliability.

A tool, as defined by Qu et al. [3] is a functional interface to a computer program that runs externally to the LLM, and the LLM generates function calls and input arguments to utilise this tool. In turn, tool learning is the process by which an AI system or LLM learn to effectively use external tools to accomplish tasks beyond their native capabilities. It involves recognising when a tool would be helpful, and interpreting the results to incorporate them into a useful response. When it comes to utilising LLMs in a specific business domain, especially in context and data sensitive fields like banking, there is even more reliance on LLMs and AI systems to not hallucinate and dynamically be able to update and retrieve data. Expertise enhancement is also a valuable improvement that could enable these systems to provide domain-specific insights while maintaining regulatory compliance and auditability.

With this focus on these three challenges; expertise enhancement, dynamic knowledge acquisition and reducing hallucinations, let us understand how tool learning is implemented currently in literature.

Qu et al. [3] discuss the four main stages of tool learning within LLMs; task planning, tool selection, tool calling and response generation, with each one serving an important step in the entire input text to output response pipeline of LLMs interacting with users and tools. Task planning and tool selection detail how an LLM receives user input, breaks it down into further atomic tasks and maps each task to a specific tool that can be used to action them. Tool calling embodies the actual calling of the API or tool, requiring the LLM to have an understanding of the API schema, its input parameters, and being able to extract relevant information from the broken down tasks and other input provided to pass on to the API. The LLM then requests data from these tool servers. This stage is especially important with multi-modal LLMs and tool servers. Now coming to the last step, response generation involves the LLM receiving data from the tools and using this response to concoct a coherent, comprehensible reply to

the user, with the relevant information. This may take place as direct data insertion into the LLMs initial response, or it may iterate over its initial reply and update it as required.

The following subsections detail each phase, highlighting a few relevant examples and the different approaches that can be taken at each stage.

2.4.1 Task Planning

In real world scenarios, user queries are often complex and not singular in action. Hence, it is essential for the LLM to first engage in task planning to correctly interpret user intent and actions. This stage involves the breakdown of complex user queries into smaller, atomic tasks that can be directly mapped to specific action tasks. This can be done either by:

(1) Utilising the innate abilities of LLMs and enable effective planning through zero-shot or few-shot prompting (without explicit fine-tuning); or by,
(2) Manually fine-tuning the LLMs weights to assist the model in the planning stage (with explicit fine-tuning).

Without Fine-tuning

The majority of papers not using fine-tuning utilise systematic prompting techniques to help the LLM to break down each query task, sometimes iteratively like in Huang et al. [56] where they refine sub-queries till initial query is satisfied, or using templated prompts like in Liu et al. [57] and Shen et al. [58]. Some research diverges from the norm of prompting techniques to venture into graph-based territory, with sub-tasks being viewed as nodes in a graph and the edges representing dependencies [59].

Many of these papers also iterate and improve on the tool descriptions to help the LLM learn without actual fine-tuning, for example in Huang et al. [56] where in their "Edit & Ground" paradigm they rewrite tool descriptions if the tool is continuously failing for queries, and update them on new queries that use the tool.

With Fine-tuning

General areas of research for using fine-tuning in task planning is utilising human feedback (RLHF) on top of a foundation model to help critique a model into more accurate task planning [60] and training models on large function-calling examples [61]. TPTU-v2 also assembles domain-specific API user queries and trains LLMs on this [62], while UMi uses a two-phase regime with broad fine-tuning first to generate a "planner model" and then further planner-focused training to sharpen its task planning capability [63].

2.4.2 Tool Selection

After the task planning stage, it is necessary for the LLM to select the appropriate tools to utilise for each decomposed atomic task. The tool selection process involves choosing tools through either a retriever or directly allowing the LLM to pick from a provided list of tools. If there are too many tools, often a tool retriever is useful to identify the top-K relevant tools to offer to the LLMs; this is a process known as retriever-based tool selection. Although, if the quantity of tools is minimal and there is enough of a context window for the LLM, then allowing the LLM to pick a specific tool itself when given all tool options within its context is also another appropriate option, i.e. LLM-based selection.

LLM-Based Selection

Within LLM-based selection, there have been examples of both tuning-free and tuning based. Some early examples of tuning-free methods include Chain-of-Thought prompting (“let’s think step by step”) to externalise reasoning [64], ReACT’s interleaving of textual reasoning and tool actions with feedback loops from tool outputs [65], and depth-first search strategies (DFSDT) to reduce error propagation in multi-step tool chains [66]. ToolNet represents tools as nodes in a directed graph to be traversed [67], and retrieval accelerators like AnyTool and GeckOpt apply hierarchical indexing or intent gating to prune large tool sets before selection [68][69].

Tuning-based methods focus on fine-tuning open-source LLMS directly on curated tool-use datasets, embedding explicit knowledge of tool semantics and invocation patterns into model

parameters. Toolbench [70] demonstrates that fine-tuning with demonstrations and system prompts bolsters tool proficiency in LLMS.

Retriever-Based Selection

Retriever-based task selection involves the usage of some type of RAG or retrieval system to first retrieve top-k relevant tools, out of which the LLM then selects the most appropriate one. This retrieval can be achieved by term-based methods (sparse retrieval), which represent tools and queries as high-dimensional sparse vectors based on terms, and employ exact term matching to achieve alignment and ultimate selection of tools with respect to the query. Conversely, it can also be done with semantic-based methods (dense-retrieval), which utilise neural networks to understand the semantic relationship between queries and tool descriptions (cosine similarity).

Gorilla [71] employs BM25 and GPT-Index to construct a term-based retriever for implementing tool retrieval, while CRAFT [72] instructs LLMs to generate a fictitious tool description based on the given query and employs this fabricated tool to conduct a search in semantic-based methods.

The sheer volume of tools that can be used coupled with the limited context lengths of LLMs, most of the time it is impractical to use purely LLM-based methods, and LLMs with retrieval systems consistently perform better.

2.4.3 Tool Calling

In the 3rd stage, LLMs are required to extract required parameters from user query in accordance to the specifications outlined in the tool schema and description. It is important that the LLMs adhere strictly to the output format to prevent the generation of superfluous sentences sent to the tool server. The process can be carried out in either tuning-free or tuning-based methods, with tuning-free methods being more flexible and robust, since they don not require the construction of a specific dataset.

Without Fine-Tuning

Tuning-free methods primarly leverage the few-shot example to provide demonstrations for parameter extraction [57][73], while other methods include reverse chain [74] which utilises reverse thinking by first selecting a final tool for a task and then having the LLMs populate the necessary parameters. Rewriting and improving tool descriptions is also prevalent, with Easytool [75] prompting chatgpt to rewrite tools to be more concise, and Xu et al. [76] compressing tool documentation into summary sequences while preserving key information.

With Fine-Tuning

Tuning-based approaches refine an LLM's ability to invoke tools by updating its weights on curated tool-usage data. For instance, GPT4Tools [77] applies LoRA fine-tuning with a ChatGPT-generated instruction dataset, while Toolkengpt uses special "toolken" tokens to trigger and parameterise API calls seamlessly [78]. Models like Themis and STE embed tool reasoning into the generation process—dynamically deciding which tools to call and leveraging trial-and-error, imagination, and memory to improve reliability [79][80]. Crucially, these methods also incorporate error-handling loops that adjust inputs based on call failures, enhancing overall robustness.

Although tool-calling is a key part of tool learning, the introduction of the Model Context Protocol (MCP)—which will be discussed later—could help standardise the process in which tools are called and given to a model, effectively making tool calling much more portable across systems.

2.4.4 Response Generation

Outputs of tools can be very unpredictable in different use cases, so it is important to have a layer between the response from tools, and the direct response sent to the user. Research has been done to utilise either direct insertion methods for response generation or information integration methods, which involves the LLM refining the response before sending it back to the user.

Methods in early-work involved direct insertion where the output of the tools are inserted into the generated response without a layer of evaluation [81][82] but this may result in under performing conversational user experience. However, information integration methods where there is a layer of evaluation between tool response and sending a response to the user, fare a lot better [73]. Some studies even suggest that refining the response generated by LLMs using the tool feedback is more effective than generating the response after invoking the tool [83][84].

2.4.5 Benchmarks

Evaluation of LLMs in tool learning is just as important as the action of utilising tool calls itself. No feature is meaningful if it cannot be reliably measured. Robust evaluation ensures that improvements in tool learning translate to real-world performance. Different methods and benchmarks exist for each stage of the tool learning process, from task planning to response generation.

Tools like Metatool [85] focus on task planning and tool selection stages, assessing whether LLMs accurately recognise the necessity of utilising tools and selecting the most suitable one to fulfill the user requirement. While APIBench [71] and ToolBench1 [70] are more involved with the tool calling and selection stages, addressing the ability of an LLM to choose the right tool and provide the necessary parameters according to the tool schema/descriptions. Some studies like Tooleyes [86], ToolBench2 [66], RestBench [73] and TaskBench [87] encompass three, or all four stages of the tool learning process.

The majority of current research does rely on synthetic queries generated by LLMs, rather than real user data, but studies GTA [88], ShortcutsBench [89] and AppWorld [90] overcome these limitations by gathering real-world tools and creating queries that reflect genuine complex user requirements.

Many other studies are more specific in what they assess, for example RoTBench [91] and Toolsword [92] emphasise the robustness and safety issues in tool learning while SoayBench [93] specialises in assessment of academic information seeking. ToolQA [94] even goes

ahead an tries to enhance the question answering capability of an LLM through the use of external tools.

As discussed, there are many different approaches to assessing tool learning in large language models, specific to each stage. The choice of evaluation strategy is dependent on what the user plans to optimise on, and is essential in measuring and understanding progress. Rigorous and targeted assessments also ensure that advances in tool learning are meaningful and transferable beyond controlled research settings.

2.5 Model Context Protocol (MCP) for Standardisation

Following the development in enhancement of reasoning and tool usage in large language models (LLMs), many of these systems are still trapped behind information silos and legacy code. And each new data source requires its own custom integration, slowing down both adoption and research. This problem led Anthropic to introduce the Model Context Protocol (MCP) in November 2024 as a solution [41]. MCP is a universal, open standard that allows AI systems and LLMs to connect to tools and data sources replacing the fragmented, custom-built API integrations that currently hinder scalability. Even top-performing models are limited by their static training data making it hard for them to provide relevant, up-to-date information. Tool learning *with* MCP offers a potential way to overcome these limitations.

2.5.1 MCP Within AI Tooling

Before the introduction of MCP, AI applications relied on methods such as manual API wiring, plugin-based interfaces and agent frameworks to interact with external tools. These approaches require integrating each external service with a specific API, leading to increased complexity, increased manual work and limited scalability [95].

Traditional implementations involved establishing manual API connections for each tool or service that an AI application interacted with. This further required custom authentication,

data transformation and error handling for every integration, making it extremely difficult to scale or modify with increase in the number of APIs due to its tight coupling. MCP eliminates this complexity by providing a unified interface, allowing AI systems to connect with multiple tools dynamically, without needing customer API wiring for each tool call.

OpenAI ChatGPT Plugins [96] are a good example of standardised plugin interfaces that aimed to reduce the complexity of manual wiring. It allowed AI models to connect to external tools through standardised API schemes such as OpenAPI, and had an entire OpenAI Plugin ecosystem, before it was discontinued in 2024 and replaced with custom GPTs [97]. Although these plugin-type interactions were an improvement, they were still one-directional and could not maintain state or coordinate multiple steps in a task. Other LLM app stores like ByteDance Coze [98] and Tencent Yuanqi [99] also emerged providing plugins, and while these platforms expanded available tool options, they created isolated ecosystems where plugins are platform-specific, limiting cross-platform compatibility and requiring duplicate maintenance efforts. MCP solves this by being open-source and platform-agnostic, enabling AI applications to engage in two-way interactions with external tools.

The third method with AI agent tool integrations, tool orchestration frameworks like Langchain [100] that have emerged recently, provide a structured way for models to invoke external tools through predefined interfaces, improving automation and adaptability. However, integrating and maintaining these tools remained largely manual, requiring custom implementations and increasing complexity as the number of tools grew. MCP simplifies this process by offering a standardised protocol that enables AI agents to seamlessly invoke, interact with, and chain multiple tools through a unified interface. Even with Retrieval-Augmented Generation (RAG), where vector-based searches are used to retrieve relevant knowledge from databases, this approach was still only passive retrieval of information, and does not allow models to perform active operations such as modifying data or triggering workflows [101]. MCP extends beyond passive information retrieval and enables AI models to actively interact with external data sources and tools, facilitating both retrieval and action in a unified workflow.

Therefore the introduction of MCP within tool learning, where all the above tool invocation methods (manual API wiring, plugin interfaces and agent frameworks) are used, has the

potential to redefine AI-to-tool interactions and enable more autonomous and intelligent agent workflows. However, the impact of MCP will depend on its uptake within the industry. Widespread adoption is necessary for MCP to become a standard, as its benefits can only be fully realised if developers and organisations actively use it in real-world applications.

2.5.2 Architecture, Design and Primitives

Anthropic was inspired by established standardisation successes like web APIs and the language server protocol (LSP) [102], and aims to establish a unified framework for tool calling, that significantly simplifies interactions between AI systems and external resources. The way it has been designed to do so is around a structured client-server model which includes a host, client and server to enable communication [103].

The host refers to the LLM application, such as an IDE or a platform like Anthropic's Claude Desktop, which supports MCP communication. The host typically includes a client component that holds all the software to manage the one-to-one connection with a server. In contrast, servers are the only part of the system with direct access to external resources. They supply the client with essential data and tools by making API calls to these resources. The server is often also run on the client side in cases with desktop hosts and clients, but it may also run remotely if needed in deployed systems. This client-server structure encourages modularity, scalability, and smooth interoperability. As a result, AI systems can securely and efficiently integrate with external resources while maintaining strong security and operational efficiency.

Further, MCP uses a layered approach to manage communication between these components [104]. The protocol layer standardises how messages are framed, handled, and delivered, supporting asynchronous, non-blocking exchanges and robust error management. This structure makes integration easier and improves system reliability. The transport layer handles how messages are sent, with options like Stdio for local use and HTTP with SSE for remote or streaming scenarios. The choice of transport depends on factors such as security and scalability.

MCP also defines four main message types: requests, results, errors, and notifications, which together support clear, efficient, and reliable interactions between clients and servers.

Coming into how MCP interacts, it defines three main primitives: prompts, resources, and tools. Prompts are structured instructions that guide LLM behaviour, ensuring consistent and efficient outputs. They are user-controlled and can be customised for different contexts, with their discovery and use managed through standardised APIs. Resources refer to data provided by the server, such as text, images, or other files, which enrich LLM interactions. These are uniquely identified and accessed via standard methods, with advanced servers supporting real-time updates to keep context current. Tools are server-side functions that LLMs can invoke to perform actions or access external systems. These tools are documented and securely managed, with strict controls in place to ensure safe operation. Together, these primitives enable flexible, reliable, and secure integration between LLMs and external resources.

This shift marks a move away from explicit, structured tool learning—where models are directly trained on how to use tools—towards more implicit and contextual tool learning. Within the standardised MCP framework, models learn to use tools by exploring and responding to demonstrations, rather than relying on manual API wiring [95].

2.5.3 Challenges

Further research by Hou et al. [95] identifies critical security challenges in the Model Context Protocol (MCP) ecosystem, primarily the lack of centralised security oversight and standardised authentication mechanisms. With no unified platform to enforce security standards or manage packages, MCP implementations suffer from inconsistent configurations and vulnerable installations. Additional concerns include insufficient monitoring capabilities, challenges in maintaining workflow consistency across tools, and isolation issues in multi-tenant environments. As MCP adoption expands in enterprise and IoT settings, these weaknesses create significant risks that require urgent attention from researchers and industry stakeholders.

2.6 Conclusion

This literature review provides a comprehensive overview of large language models (LLMs), tool learning, and the Model Context Protocol (MCP), along with the challenges these technologies face when applied in business contexts. Additionally, it explores the historical development of AI and LLMs, examines various LLM augmentation techniques, and outlines the key steps involved in implementing tool learning.

Large language models have evolved significantly, progressing from basic language models to augmented models and then to the use of tool learning to enhance reasoning and interaction with external resources. This enhancement of LLMs to better suit the context and environment it is applied in, was found vital for its effective implementation within domain-specific fields and one prevalent method to do so is with the augmentation technique of tool learning. Tool learning is multi-faceted and can be done in many ways, but the introduction of the open Model Context Protocol (MCP) in November of 2024 is playing a pivotal role in standardising how AI systems interact with external tools and resources, particularly in this context of tool learning. . By providing a unified framework for communication between models and external APIs, MCP has made it significantly easier to integrate, manage, and scale tool use across different AI applications—much like how the USB-C port streamlined device connectivity. This development opens up new research opportunities, particularly in evaluating the effectiveness of tool selection using MCP. This paper aims to investigate whether MCP improves tool selection, specifically when combined with retrieval-augmented generation (RAG) and LLM-based methods, to deliver tailored, information sensitive and domain-specific results within banking. We also aim to assess whether this approach can be generalised to many use cases and multiple tools within an agile fintech startup, and to identify what factors are necessary for the broader success of tool learning with MCP.

CHAPTER 3

Research Design

This research originated from a practical need within a company environment. Our team developed an internal chatbot platform, designed to mirror the functionality of tools like Claude Desktop or ChatGPT, but with the critical distinction of leveraging our own secure database and proprietary codebase. There were two primary motivations for this initiative:

(1) **Confidentiality and Data Security:** To enable employees to utilise an LLM for tasks involving confidential personal information (PI) data. By ensuring that all data remained within our internal infrastructure, we were able to meet strict compliance requirements that prohibit the use of external providers like ChatGPT for sensitive information.

(2) **Comprehensive Knowledge Integration:** To serve as a unified knowledge base for employees, seamlessly connecting to all core internal systems. This integration aimed to facilitate efficient information retrieval and streamline workflows across the organisation.

Achieving these objectives, especially the latter, required robust integration between the chatbot and various internal services through function and tool-calling. We specifically chose to implement this integration using the Model Context Protocol (MCP), which was introduced in November 2025. The initial deployment connected the chatbot to Slack, enabling bi-directional communication, and was subsequently expanded to include MCP servers for tools such as Jira, Confluence, and Gitlab.

The internal AI system quickly gained popularity among employees, becoming an indispensable part of daily operations. However, when it came to evaluating the system's effectiveness,

we encountered a significant gap in the literature: few existing offline evaluation frameworks were capable of handling the diversity of real user inputs or provided meaningful metrics for such dynamic environments. This challenge inspired the development and testing of novel, question-dependent, rubric-based evaluation methods tailored to real-world use cases.

3.1 Experimentation Set Up

Given that robust evaluation frameworks are typically structured around three core pillars—datasets, metrics, and evaluation methodology [**Rudd2025ASystems**]—this section will systematically address each component as it relates to the two evaluation frameworks developed in this research. Following this, we will provide an in-depth discussion of the MCP tool servers and the large language models (LLMs) employed within the system.

Both evaluation frameworks are specifically designed to assess data-retrieval functionality via tool servers, focusing on single-turn user input dialogues within the AI system. This targeted approach ensures that our findings are directly applicable to real-world, production-like scenarios.

To rigorously investigate performance degradation as the number of available tools increases, we leverage both evaluation frameworks alongside their corresponding datasets and metrics. This dual-framework analysis enables a comprehensive and nuanced understanding of system behaviour under varying operational conditions.

3.2 Datasets

Due to the absence of suitable benchmark datasets in existing literature that reflect the specific requirements of our tool server—particularly one with Slack-based data retrieval functionality—we constructed a bespoke dataset tailored to this use case. This dataset was generated using a large language model (LLM), with the intent to simulate realistic user interactions that would be encountered in a production environment.

To ensure the authenticity of the generated inputs, we provided the LLM with representative examples from real-world usage of our internal chatbot. These examples were carefully curated to reflect actual production data, and the LLM was instructed to generate inputs that mimic genuine user behavior and language patterns.

Additionally, the LLM was supplied with detailed information about the available tools on the tool server, so it could accurately model how users would invoke these tools in practice. The generated dataset follows a structured `jsonl` format, with each entry including the following attributes:

- `unique_id`: A unique identifier for each interaction.
- `prompt`: The simulated user input.
- `answer`: The expected system response.
- `tools_required`: A list of tools invoked or necessary to generate the response.

For the second framework evaluated in this work, an additional field—`rubrics`—is included for each data point. This field captures the specific criteria on which the AI-generated response is intended to be evaluated, enabling more nuanced and structured analysis of performance beyond simple correctness.

3.3 Evaluation Methodology & AI System

Both evaluation frameworks employed in this work utilize the LLM-as-a-judge method to assess the quality and correctness of responses generated by the AI system. This evaluation strategy leverages the judgment capabilities of a large language model to automate the scoring and feedback process, offering a scalable alternative to manual evaluation.

The evaluation pipeline is built upon and extends the foundational codebase introduced by [**Luo2025EvaluationServers**], which provides a minimal yet fully functional model context protocol (MCP) host. This existing infrastructure includes core capabilities such as sending prompts to the LLM of choice, handling tool server interactions, and managing the necessary pre-processing and post-processing logic required to interface between components.

Our methodology follows a two-stage process:

(1) **System Response Generation**: The pipeline begins by issuing the user's prompt to the LLM-based AI system. If the task requires access to external tools, the system is responsible for orchestrating tool server calls, retrieving responses, and integrating this information before formulating the final answer. This step simulates how an AI assistant would perform in a real-world environment that involves both reasoning and tool use.
(2) **Response Evaluation**: Once a response has been generated, it is evaluated by a separate instance of an LLM, which receives both the prompt and the AI system's output, along with task-specific evaluation criteria embedded in a structured evaluation prompt. The LLM is then tasked with scoring the response based on defined rubrics, such as factual accuracy, relevance, completeness, and adherence to formatting constraints.

This pipeline allows us to leverage the LLM's reasoning abilities not only for answering prompts but also for evaluating its own or another model's outputs in a systematic manner. To ensure consistency and reduce variance in judgment quality, the same LLM—Claude Sonnet 3.5—is used across all evaluation runs.

3.4 Metrics

Both evaluation frameworks use two types of metrics, combining one objective numeric measure with a few subjective LLM-based metrics to comprehensively evaluate AI system performance.

The numeric metric provides a straightforward assessment of tool usage accuracy. It operates as a simple conditional check: if all tools called by the AI system match the expected tools listed in the dataset's `tools_required` attribute, the response is marked as successful. This binary evaluation ensures the system correctly identifies and utilizes the appropriate computational tools.

The second type of LLM-based metrics differ between the two frameworks, each designed for specific evaluation needs.

First Evaluation Framework

This framework employs two LLM-based metrics alongside the numeric tool check. The prompt adherence score evaluates whether the response adequately addresses the input prompt, using only the prompt and response as inputs and scoring out of 5 points. The content accuracy score compares the generated response against an expected reference response, semantically evaluating alignment and correctness, also scored out of 5 points.

The framework combines these scores additively—responses scoring above 6 out of 10 total points are considered successful. However, final success requires both the numeric tool check and the LLM-based evaluation to pass; only when both conditions are met does the framework return a positive result.

Second Evaluation Framework

This framework takes a more flexible approach, relying entirely on custom criteria specified in each prompt's "rubrics" attribute. The evaluation adapts to different requirements through a templating system with placeholders, allowing the assessment to focus on various aspects such as technical accuracy, creativity, or domain-specific considerations depending on the rubric specifications.

Both frameworks utilize structured evaluation prompts that guide the LLM judges, ensuring consistent assessment while maintaining flexibility for different evaluation scenarios. This dual-framework approach provides both standardized evaluation criteria and adaptable, context-specific assessment capabilities suitable for diverse AI testing requirements.

3.5 Models

The primary models utilized in this study are from the Claude Sonnet series, specifically sonnet-3.5, sonnet-3.7, and sonnet-4. These models were selected to provide a comprehensive

evaluation across different capabilities within the same model family, allowing for meaningful performance comparisons while maintaining consistency in underlying architecture and training methodologies. Claude Sonnet-3.5 was additionally designated as the evaluator model within the LLM-as-a-judge pipeline, responsible for assessing all generated responses across both evaluation frameworks. This choice was maintained consistently throughout the entire evaluation process to ensure standardized assessment criteria and eliminate potential variability that could arise from using different evaluator models, thereby ensuring that performance differences observed between the tested models reflect genuine capability variations rather than inconsistencies in the evaluation process.

3.6 MCP Servers

With the internal AI chatbot at work, one of the first tool server integrations added was the slack MCP server found here https://github.com/modelcontextprotocol/servers-archived/tree/main/src/slack (which has now since been archived), due to slack being a widely used work service which could provide valuable functionality to an AI system. The tool server was further improved to better filter tool response outputs, since there was excessive metadata being returned from the API calls, which resulted in the LLM not being able to handle all the tokens passed to it. This slack tool server is also the main server of interest in our research, it specifically being picked because of it's industry value and widespread use, and also specifically because it is a good database for data-retrieval functionality which is primarily what we'll be focusing on. The dataset generation pipeline also utilises this server's tools and information to generate the synthetic data.

Other MCP servers to bulk up the tools passed to the LLM include:

- Atlassian MCP server (Jira and confluence products' functionality) - api key
- Brave search MCP Server - api key auth
- Gitlab MCP server - api key auth
- Memory MCP server - no auth
- Deepwiki remote SSE MCP server - no auth

- Excel remote SSE MCP server - no auth
- Playwright MCP Server - no auth
- Apify remote SSE Server - API key
- Semgrep remote SSE MCP server - no auth
- Git remote SSE MCP Server - no auth
- Neon remote SSE MCP Server - api key auth

These were picked primarily because most were easily available online as remote servers with minimal authorisation and required less integration via code. An additional two mock MCP servers for spotify and notion were also created, to help focus on addition applications that have data-retrieval type functionality and to bulk up the number of tools. All of these increased number of tools to 181, which will help in evaluation of performance degradation with increase in tools provided to the LLM.

CHAPTER 4

Experimentation

Experimentation started off at work by creating unit test-like use cases for the slack tool server by looking through the tools provided by it. This helped initial integration into the chatbot at work and brought to our attention the shortcomings of the tool server like the excess metadata in tool call responses. This was fixed by making a custom slack MCP server using the code from the out-of-box server as a base. Evaluation was done on the 14 use cases created initially, and evaluation framework 1, which is described as below. Due to the use cases being quite direct and simple, the LLMs in the internal AI chatbot had no problem answering the questions with the mutated/updated slack MCP server.

Later, when reusing this MCP server for this thesis, all the write functionality of the server, for example responding to a message thread on slack or adding an emoji to message, was removed, keeping only data-retrieval functionality. The dataset used also had to be updated to provide a much more expansive list of use cases, which has been done according to the data generation pipeline described in the experimentation set up. A deeper look into the experiments done are below.

4.1 Evaluation Framework - 1

As already discussed, the implementation of the 3 pillars of this framework are as follows:

(1) Dataset: The base dataset that includes attributes of unique_id, prompt, answer and tools_required is used. This was synthetically generated as per our data generation pipeline discussed above, and includes 50 test cases to evaluate.

(2) Metrics: Includes the one objective numeric metric of tool calls and two subjective, LLM-as-a-judge style metrics. To consolidate the two types, if both the numeric and subjective metrics return a success=true value for the response, then the response for that data point is said to have passed or been a success, and contributes to the overall success percentage that is calculated at the end. All 3 are listed below:
   (a) tools_required metric: Operates as a simple conditional check where if all the tools called by the LLM are within the tools_required attribute in the dataset, the response is said to have passed this measurement.
   (b) Prompt adherence metric: An LLM-as-a-judge metric which takes the input prompt and response and evaluates whether the response adequately answered the prompt, and then scores it out of 5. Scoring it out of 5 was chosen because 3 is too less and 10 was too much.
   (c) Content accuracy metric: The second LLM-as-a-judge metric which takes response and expected response and compares the two semantically, i.e. whether the answers' inherent meaning is the same, and also scores it out of 5. The score of this metric is clubbed together with the prompt adherence metric to give a total score out of 10. If the total score is above 6, then the success of the LLM-as-a-judge metrics is said to be true.
(3) Evaluation methodology: At a high-level this consists of 2 steps for each datapoint in the dataset, first the generation of the response and second the evaluation of this response. The DSPy framework for building and evaluating modular AI systems is utilised, and helps handle both of these steps. The AI system in use is extended upon the open source AI system and host that [**Luo2025EvaluationServers**] have initially created.

created, with additional benchmarks to handle the evaluation frameworks within this thesis.

This framework is utilised as the base framework for evaluation of LLM tool usage performance degradation with increase in tools, and is a good starting point for better representing a

wide range of use cases within the industry due to it being intuitive and highly general. Although some prompts still may not have a single expected response and a semantic similarity score still might not be the best way to measure it's correctness.

Sonnet 3.5, 3.7 and 4 were all run against this framework with the dataset containing 50 test cases twice. Once only with the slack server tools, which totals to 5, and the second time with 181 tools across various MCP servers (as mentioned in the previous chapter).

4.2 Evaluation Framework - 2

For the second evaluation framework, the evaluation methodology pillar is implemented in the same way as the first framework, but the dataset and metrics are slightly different:

(1) Dataset: All the base attributes of unique_id, prompt, answer and tools_required are still present in the dataset test cases, but there's an additional attribute called "rubrics" that is prompt-dependant and has been included to further generalise evaluation methodology to fit any type of input. Although that does mean there is a lot of initial overhead in creating the dataset and choosing which metrics are going to be used to evaluate which questions, including setting the evaluation scale and weight for each metric. This may ultimately give us less return on investment if the evaluation doesn't perform as expected.

(2) Metrics: The numeric metric of tools_required is still used, but for the subjective, LLM-based metrics, the rubric in the dataset for each question is used and the evaluation prompt (<reference appendix>) templatises it to pass to the LLM. All the weights from each criterion must add up to 1, so that the scores can then be multiplied into the weights and summed to get the final total score. Similar to framework 1, if both the tools_required metric and the total sum of the LLM-based metrics are both successful, then the entire response is returned as a successful response.

This framework is more for the contribution of having a generalised, relevant evaluation method for industry standards, and was not focused on for the preliminary experiments. But

```
{
  "unique_id": 1,
  "Prompt": "Can you list all Slack channels related to our ongoing banking software project?",
  "Answer": "",
  "Tool_Calls": ["slack_list_channels"],
  "Rubrics": [
    {
      "criterion": "Tool Selection Accuracy",
      "description": "Must select slack_list_channels exclusively",
      "weight": 0.4,
      "evaluation_scale": [
        {"score": 0, "condition": "Wrong tool selected"},
        {"score": 1, "condition": "Correct tool but extra unnecessary tools"},
        {"score": 2, "condition": "Perfect tool match"}
      ]
    },
    {
      "criterion": "Parameter Precision",
      "description": "Should implicitly filter for banking-related channels",
      "weight": 0.3,
      "evaluation_scale": [
        {"score": 0, "condition": "No parameter handling"},
        {"score": 1, "condition": "Partial parameter capture"},
        {"score": 2, "condition": "Full context captured"}
      ]
    }
```


FIGURE 4.1: Enter Caption

plans for further experimentation can include running all 3 claude sonnet models on this framework, utilising tool sets of both 5 tools and 181 tools.

CHAPTER 5

Data & Results

Preliminary results for the contribution regarding the evaluation of tool usage performance degradation in LLMs with increase in tools, shows that on changing from 5 tools in the slack server to 181 across multiple tool servers decreases the successfulness of responses from 72.5% to 60% with sonnet 3.5. And similarly the decrease in percentage result for sonnet-4 was from 75% to 62.5%, although oddly the result

A table containing the results out of 40 test cases from the dataset is as seen below, with an graph showing the same results in a visual format as well.

Evaluation	Model	Dataset	# of Tools	Result
Eval framework 1	claude-3-5-sonnet-20241022	slack_50_genprompt2	5	29/40 (72.5%)
Eval framework 1	claude-3-5-sonnet-20241022	slack_50_genprompt2	181	24/40 (60%)
Eval framework 1	claude-3-7-sonnet-20250219	slack_50_genprompt2	5	24/40 (60%)
Eval framework 1	claude-3-7-sonnet-20250219	slack_50_genprompt2	181	23/40 (57.5%)
Eval framework 1	claude-sonnet-4-20250514	slack_50_genprompt2	5	30/40 (75%)
Eval framework 1	claude-sonnet-4-20250514	slack_50_genprompt2	181	25/40 (62.5%)

TABLE 5.1: Claude Models' Performance Across Tool Availability Levels

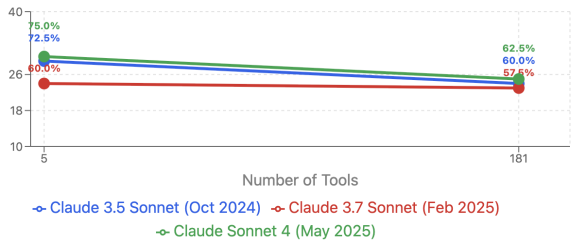

FIGURE 5.1: Claude Model Performance by no. of tools

Claude sonnet 4 clearly shows the best performance with 5 tools, scoring a 30/50 (75%) and sonnet 3.7 shows the lowest with only 24/50 (60%). The score rank with 181 tools is the same, with sonnet 4 scoring the highest, then sonnet 3.5 and sonnet 3.7. All models show a decreased performance when using 181 tools, although sonnet 3.7 is almost negligible with getting only one extra test case wrong from the experiment run with 5 tools, that is it score a 23/50 for 181 tools and a 24/50 with 5 tools.

Considering this is just preliminary data, further results are expected when the experiments are run for different levels of number of tools used, for example at 25 tools, 50 tools and 100 tools. Experiments related to context length of inputs and outputs will also be run along with increasing number of similar tools instead of just random ones.

CHAPTER 6

Analysis & Discussion

From looking at the preliminary results in the previous chapter, there is a clear decrease in performance in the first evaluation framework when the LLMs are run with 5 tools compared to when they are run with 181 tools. This further validates the ongoing discussion that tool usage with LLMs at a large scale haven't been researched much and simply pumping the AI systems with tools via system prompts may not be the most efficient way to go.

It was also seen in the data that the prompts the LLMs got wrong were those whose "tools_required" in the dataset, had similar other tools or tool descriptions within the tool set of 181, therefore showing that the LLM got confused which tool was to be utilised. This directs further experimentation to focus on more similar data-retrieval tools, and making the questions in the dataset less descriptive of the actual application or tool to use.

Hence, future plans for experimentation in this thesis includes:

- Having more levels of "number of tools" to experiment with, like starting with 5, then 25, 50 and 100, and seeing where performance really drops off.
- Having more similar data-retrieval functionality tools
- Updating dataset to be less descriptive of application or tool to be used.
- Checking if increase in context is what is causing performance degradation or if it's purely similarity of tool names and descriptions, by checking total input and output tokens for the experiments.

CHAPTER 7

Conclusion & Future Work

The conclusions drawn from the results, analysis, and discussion are consistent with the broader context of this research. This project has demonstrated significant value by addressing key gaps in the evaluation of large language models (LLMs) for tool use, particularly in settings that mirror real-world industry applications. By developing a reusable data generation pipeline and robust evaluation frameworks, this work enables future research to build on methodologies that are both practical and representative of production environments.

While the findings provide valuable insights, several limitations were encountered, such as the availability of comprehensive datasets and the evolving nature of LLM capabilities. These limitations highlight the need for continued research into dataset creation and evaluation metrics that better capture the complexities of tool-augmented AI systems. Future work should focus on expanding the diversity of datasets, refining evaluation criteria, and exploring additional real-world scenarios to further validate and improve the frameworks introduced here.

Moreover, this research contributes to the relatively unexplored area of performance degradation in LLMs as the number of available tools increases. Understanding how and why this degradation occurs is crucial, especially as the field moves toward more complex, multi-tool environments. As LLMs become increasingly integrated into systems that require interaction with dynamic external resources, effective tool management and evaluation will be paramount.

In summary, this project establishes a strong foundation for future investigations into LLM tool use, offering practical resources and new directions for both academic and industry

research. The methodologies and insights provided here are expected to inform ongoing efforts to make LLMs more reliable, adaptable, and effective in real-world applications.

CHAPTER 8

Case Study

My case studies and how I've achieved each one of the learning outcomes from both are described in detail below.

8.1 COMP4318 - Machine Learning & Data Mining

- L01: Understanding the basic principles, strengths, weaknesses and applicability of machine learning algorithms for solving classification, regression, clustering and reinforcement learning tasks.
  - Specifically through my literature review, I analysed how different ML approaches evolved in language models, from statistical methods to transformers. I identified key strengths like GPT-3's few-shot learning and weaknesses like hallucination, showing how RLHF helps with classification tasks in instruction following.
- LO2: have obtained practical experience in designing, implementing and evaluating machine learning algorithms
  - I designed and implemented two evaluation frameworks for tool-augmented LLMs, building on existing code and extending it with custom evaluation pipelines. I also created a data generation system and modified MCP servers to work with my evaluation setup, while building an AI system from scratch at work as well.
- LO3: have gained practical experience in using machine learning software and libraries

  - I worked hands-on with DSPy framework, MCP servers, Claude models, and various APIs including Slack, Jira, and GitLab integrations. I built evaluation pipelines that could handle responses across 181 different tools in my experiments.
- LO4: present and interpret data and information in verbal and written form.
  - I communicated my research findings clearly in my thesis, explaining complex concepts like performance degradation when tool counts increased from 5 to 181. I structured my results and analysis in an accessible way that shows how tool similarity affects LLM performance.

8.2 COMP4328 - Advanced Machine Learning

- LO1. Present the design and evaluation of a machine learning algorithm, describing the design processes and evaluation.
  - I presented detailed designs for my two evaluation frameworks, explaining how I combined objective tool-calling metrics with subjective LLM-based scoring. I tested these across different Claude models with varying numbers of tools to evaluate performance systematically.
- LO2. Understand the variance and bias trade-off in machine learning algorithms.
  - I recognized how existing benchmarks had high bias toward single answers, limiting real-world applicability. My frameworks balanced this by using semantic similarity (reducing bias) with structured scoring (controlling variance) to better capture diverse scenarios.
- LO3. Understand and analyse some machine learning algorithms and have some knowledge to further improve them.
  - I analysed transformer-based LLMs and identified limitations like context constraints and hallucination. I proposed improvements through better evaluation methods and suggested future work on hierarchical tool organization to address performance degradation issues.

- LO4. Understand and analyse some machine learning problems and have some knowledge to adapt the existing machine learning models to different purposes.
  - I identified the gap in evaluating tool-augmented LLMs for industry use and adapted existing evaluation methods to handle multiple valid responses instead of single ground truths. I repurposed Claude models for meta-evaluation tasks in my LLM-as-a-judge pipeline.
- LO5. Implement machine learning algorithms from peer-reviewed papers.
  - I implemented evaluation methods based on recent research, building on Luo et al. (2025) for MCP evaluation and adapting LLM-as-a-judge approaches from current literature. I also applied tool learning concepts from Qu et al. (2025) in my implementation.
- LO6. Understand the nature of the statistical foundations of designing or adapting learning algorithms.
  - I designed statistically sound evaluation frameworks with appropriate sample sizes (50 test cases), considered measurement validity in my scoring systems, and analyzed the statistical implications of performance changes across different tool quantities.
- LO7. At the completion of this unit, you should be able to demonstrate knowledge of the introduced machine learning models and the relative strengths and weaknesses of each and their most appropriate uses.
  - I analyzed various model types throughout my work - transformers (good attention but limited context), LLMs (versatile but prone to hallucination), and tool-augmented systems (dynamic knowledge but complex integration). I showed appropriate applications for each in different scenarios.
- LO8. At the completion of this unit, you should be able to demonstrate knowledge of methods to analyse machine learning algorithms, such as hypothesis complexities and generalisation bounds.
  - I analyzed how increasing tool numbers affected model complexity and performance, establishing baseline metrics and studying degradation patterns. My

comparison across Claude versions helped understand performance bounds and how tool similarity creates decision complexity that affects generalization.

References

[1] Yonghui Wu et al. 'Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation'. In: (Sept. 2016). URL: http://arxiv.org/abs/1609.08144.

[2] Ashish Vaswani et al. 'Attention Is All You Need'. In: (June 2017). URL: http://arxiv.org/abs/1706.03762.

[3] Changle Qu et al. *Tool learning with large language models: a survey*. Aug. 2025. DOI: 10.1007/s11704-024-40678-2.

[4] 'ARTIFICIAL INTELLIGENCE (AI) - THE TECHNOLOGY THAT SHAPES THE WORLD'. In: ().

[5] Omar H. Fares, Irfan Butt and Seung Hwan Mark Lee. 'Utilization of artificial intelligence in the banking sector: a systematic literature review'. In: *Journal of Financial Services Marketing* 28.4 (Dec. 2023), pp. 835–852. ISSN: 14791846. DOI: 10.1057/s41264-022-00176-7.

[6] NAVLEEN KAUR et al. 'BANKING 4.0: "THE INFLUENCE OF ARTIFICIAL INTELLIGENCE ON THE BANKING INDUSTRY & HOW AI IS CHANGING THE FACE OF MODERN DAY BANKS"'. In: *INTERNATIONAL JOURNAL OF MANAGEMENT* 11.6 (June 2020). ISSN: 0976-6502. DOI: 10.34218/ijm.11.6.2020.049.

[7] Bohdan Mytnyk et al. 'Application of Artificial Intelligence for Fraudulent Banking Operations Recognition'. In: *Big Data and Cognitive Computing* 7.2 (June 2023). ISSN: 25042289. DOI: 10.3390/bdcc7020093.

[8] John Smit. 'A Literature Review on the Impact of Artificial Intelligence on the Future of Banking and How to Achieve a Smooth Transition'. In: *Open Journal*

of Business and Management 12.01 (2024), pp. 509–520. ISSN: 2329-3284. DOI: 10.4236/ojbm.2024.121031.

[9] *Building the AI-Powered Organization*. URL: https://hbr.org/2019/07/building-the-ai-powered-organization?ref=chrislubasch.com.

[10] Binny Naik et al. 'The impacts of artificial intelligence techniques in augmentation of cybersecurity: a comprehensive review'. In: *Complex and Intelligent Systems* 8.2 (Apr. 2022), pp. 1763–1780. ISSN: 21986053. DOI: 10.1007/s40747-021-00494-8.

[11] Sulaiman Abdallah Alsheibani et al. *Association for Information Systems Association for Information Systems00 AM Winning AI Strategy: Six-Steps to Create Value from Artificial Winning AI Strategy: Six-Steps to Create Value from Artificial Intelligence Intelligence*. Tech. rep. URL: https://aisel.aisnet.org/amcis2020.

[12] *What Is Artificial Intelligence (AI)? | Google Cloud*. URL: https://cloud.google.com/learn/what-is-artificial-intelligence.

[13] A M Turing. *Computing Machinery and Intelligence*. Tech. rep. 236. 1950, pp. 433–460. URL: http://www.jstor.orgStableURL:http://www.jstor.org/stable/2251299Accessed:25/08/200818:56.

[14] *Dartmouth Summer Research Project: The Birth of Artificial Intelligence - History of Data Science*. URL: https://www.historyofdatascience.com/dartmouth-summer-research-project-the-birth-of-artificial-intelligence/.

[15] *AI Winter: The Highs and Lows of Artificial Intelligence - History of Data Science*. URL: https://www.historyofdatascience.com/ai-winter-the-highs-and-lows-of-artificial-intelligence/.

[16] Sepp Hochreiter and J ¨ Urgen Schmidhuber. *Long Short-Term Memory*. Tech. rep.

[17] Murray Campbell, A Joseph Hoane and Feng-Hsiung Hsu. *Deep Blue*. Tech. rep. 2002, pp. 57–83.

[18] D. A. Ferrucci. 'Introduction to "This is Watson"'. In: *IBM Journal of Research and Development* 56.3-4 (May 2012). ISSN: 00188646. DOI: 10.1147/JRD.2012.2184356.

[19] David Silver et al. 'Mastering the game of Go with deep neural networks and tree search'. In: *Nature* 529.7587 (Jan. 2016), pp. 484–489. ISSN: 14764687. DOI: 10.1038/nature16961.

[20] Jie Zhou et al. *ChatGPT: potential, prospects, and limitations*. Jan. 2024. DOI: 10.1631/FITEE.2300089.

[21] Muhammad Usman Hadi et al. *Large Language Models: A Comprehensive Survey of its Applications, Challenges, Limitations, and Future Prospects*. Sept. 2024. DOI: 10.36227/techrxiv.23589741.v7. URL: https://www.techrxiv.org/users/618307/articles/682263-large-language-models-a-comprehensive-survey-of-its-applications-challenges-limitations-and-future-prospects?commit=b47368cc9cd41e01c01a4695bd

[22] *What is LLM? - Large Language Models Explained - AWS*. URL: https://aws.amazon.com/what-is/large-language-model/.

[23] Peter F Brown et al. *A STATISTICAL APPROACH TO MACHINE TRANSLATION*. Tech. rep.

[24] Thomas Hofmann. *Unsupervised Learning by Probabilistic Latent Semantic Analysis*. Tech. rep. 2001, pp. 177–196.

[25] Ronald Rosenfeld. *Two Decades of Statistical Language Modeling: Where Do We Go from Here?* Tech. rep. 2000.

[26] Tomaš Mikolov et al. 'Recurrent neural network based language model'. In: *Proceedings of the 11th Annual Conference of the International Speech Communication Association, INTERSPEECH 2010*. International Speech Communication Association, 2010, pp. 1045–1048. DOI: 10.21437/interspeech.2010-343.

[27] Jacob Devlin et al. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. Tech. rep., pp. 4171–4186. URL: https://github.com/tensorflow/tensor2tensor.

[28] Jaime Sevilla et al. 'Compute Trends Across Three Eras of Machine Learning'. In: (Feb. 2022). DOI: 10.1109/IJCNN55064.2022.9891914. URL: http://arxiv.org/abs/2202.05924%20http://dx.doi.org/10.1109/IJCNN55064.2022.9891914.

[29] Tom B. Brown et al. ‘Language Models are Few-Shot Learners’. In: (May 2020). URL: http://arxiv.org/abs/2005.14165.

[30] Colin Raffel et al. ‘Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer’. In: (Oct. 2019). URL: http://arxiv.org/abs/1910.10683.

[31] Linting Xue et al. ‘mT5: A massively multilingual pre-trained text-to-text transformer’. In: (Oct. 2020). URL: http://arxiv.org/abs/2010.11934.

[32] Yu Sun et al. ‘ERNIE 3.0: Large-scale Knowledge Enhanced Pre-training for Language Understanding and Generation’. In: (July 2021). URL: http://arxiv.org/abs/2107.02137.

[33] Zhengyan Zhang et al. ‘CPM-2: Large-scale Cost-effective Pre-trained Language Models’. In: (June 2021). URL: http://arxiv.org/abs/2106.10715.

[34] Jack W. Rae et al. ‘Scaling Language Models: Methods, Analysis & Insights from Training Gopher’. In: (Dec. 2021). URL: http://arxiv.org/abs/2112.11446.

[35] Kang Min Yoo et al. ‘HyperCLOVA X Technical Report’. In: (Apr. 2024). URL: http://arxiv.org/abs/2404.01954.

[36] Shuohuan Wang et al. ‘ERNIE 3.0 Titan: Exploring Larger-scale Knowledge Enhanced Pre-training for Language Understanding and Generation’. In: (Dec. 2021). URL: http://arxiv.org/abs/2112.12731.

[37] Nan Du et al. ‘GLaM: Efficient Scaling of Language Models with Mixture-of-Experts’. In: (Dec. 2021). URL: http://arxiv.org/abs/2112.06905.

[38] Jordan Hoffmann et al. ‘Training Compute-Optimal Large Language Models’. In: (Mar. 2022). URL: http://arxiv.org/abs/2203.15556.

[39] Yi Tay et al. ‘UL2: Unifying Language Learning Paradigms’. In: (May 2022). URL: http://arxiv.org/abs/2205.05131.

[40] OpenAI et al. ‘GPT-4 Technical Report’. In: (Mar. 2023). URL: http://arxiv.org/abs/2303.08774.

[41] *Introducing Claude \ Anthropic*. URL: https://www.anthropic.com/news/introducing-claude.

[42] Raymond Li et al. ‘StarCoder: may the source be with you!’ In: (May 2023). URL: http://arxiv.org/abs/2305.06161.

[43] *Mistral Small 3.1 | Mistral AI*. URL: https://mistral.ai/news/mistral-small-3-1.

[44] Gemini Team et al. ‘Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context’. In: (Mar. 2024). URL: http://arxiv.org/abs/2403.05530.

[45] Reiichiro Nakano et al. ‘WebGPT: Browser-assisted question-answering with human feedback’. In: (Dec. 2021). URL: http://arxiv.org/abs/2112.09332.

[46] Hugo Touvron et al. ‘Llama 2: Open Foundation and Fine-Tuned Chat Models’. In: (July 2023). URL: http://arxiv.org/abs/2307.09288.

[47] Niklas Muennighoff et al. ‘Crosslingual Generalization through Multitask Finetuning’. In: (Nov. 2022). URL: http://arxiv.org/abs/2211.01786.

[48] Ranjan Sapkota, Konstantinos I. Roumeliotis and Manoj Karkee. ‘AI Agents vs. Agentic AI: A Conceptual Taxonomy, Applications and Challenges’. In: (May 2025). URL: http://arxiv.org/abs/2505.10468.

[49] Sakib Haque et al. ‘A Reality check of the benefits of LLM in business’. In: *IEEE International Conference on Program Comprehension*. Vol. 2022-March. IEEE Computer Society, 2022, pp. 36–47. ISBN: 9781450392983. DOI: 10.1145/nnnnnnn.nnnnnnn.

[50] Rohan Anil et al. ‘PaLM 2 Technical Report’. In: (May 2023). URL: http://arxiv.org/abs/2305.10403.

[51] Fan Huang, Haewoon Kwak and Jisun An. ‘Is ChatGPT better than Human Annotators? Potential and Limitations of ChatGPT in Explaining Implicit Hate Speech’. In: *ACM Web Conference 2023 - Companion of the World Wide Web Conference, WWW 2023*. Association for Computing Machinery, Inc, Apr. 2023, pp. 294–297. ISBN: 9781450394161. DOI: 10.1145/3543873.3587368.

[52] Amos Azaria. *ChatGPT Usage and Limitations*. Tech. rep. URL: https://hal.science/hal-03913837v1.

[53] Jan Kocoń et al. 'ChatGPT: Jack of all trades, master of none'. In: (Feb. 2023). DOI: 10.1016/j.inffus.2023.101861. URL: http://arxiv.org/abs/2302.10724%20http://dx.doi.org/10.1016/j.inffus.2023.101861.

[54] Grégoire Mialon et al. 'Augmented Language Models: a Survey'. In: (Feb. 2023). URL: http://arxiv.org/abs/2302.07842.

[55] Anand Ramachandran. *A Survey of Agentic AI, Multi-Agent Systems, and Multimodal Frameworks Architectures, Applications, and Future Directions*. 2024. URL: https://www.researchgate.net/publication/387577302_A_Survey_of_Agentic_AI_Multi-Agent_Systems_and_Multimodal_Frameworks_Architectures_Applications_and_Future_Directions.

[56] Tenghao Huang, Dongwon Jung and Muhao Chen. 'Planning and Editing What You Retrieve for Enhanced Tool Learning'. In: (Mar. 2024). URL: http://arxiv.org/abs/2404.00450.

[57] Zhaoyang Liu et al. 'ControlLLM: Augment Language Models with Tools by Searching on Graphs'. In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. Vol. 15070 LNCS. Springer Science and Business Media Deutschland GmbH, 2025, pp. 89–105. ISBN: 9783031732539. DOI: 10.1007/978-3-031-73254-6{_}6.

[58] Yongliang Shen et al. *HuggingGPT: Solving AI Tasks with ChatGPT and its Friends in Hugging Face*. Tech. rep. URL: https://github.com/microsoft/JARVIS.

[59] Xixi Wu et al. 'Can Graph Learning Improve Planning in LLM-based Agents?' In: (May 2024). URL: http://arxiv.org/abs/2405.19119.

[60] Yaobo Liang et al. 'TaskMatrix.AI: Completing Tasks by Connecting Foundation Models with Millions of APIs'. In: *Intelligent Computing* 3 (Jan. 2024). ISSN: 27715892. DOI: 10.34133/icomputing.0063.

[61] Zuxin Liu et al. *APIGen: Automated PIpeline for Generating Verifiable and Diverse Function-Calling Datasets*. Tech. rep. URL: https://huggingface.co/datasets/Salesforce/xlam-function-calling-60k.

[62] Yilun Kong et al. 'TPTU-v2: Boosting Task Planning and Tool Usage of Large Language Model-based Agents in Real-world Systems'. In: (Nov. 2023). URL: http://arxiv.org/abs/2311.11315.

[63] Weizhou Shen et al. 'Small LLMs Are Weak Tool Learners: A Multi-LLM Agent'. In: (Jan. 2024). URL: http://arxiv.org/abs/2401.07324.

[64] Jason Wei et al. *Chain-of-Thought Prompting Elicits Reasoning in Large Language Models Chain-of-Thought Prompting*. Tech. rep.

[65] Shunyu Yao et al. 'ReAct: Synergizing Reasoning and Acting in Language Models'. In: (Oct. 2022). URL: http://arxiv.org/abs/2210.03629.

[66] Yujia Qin et al. 'ToolLLM: Facilitating Large Language Models to Master 16000+ Real-world APIs'. In: (July 2023). URL: http://arxiv.org/abs/2307.16789.

[67] Xukun Liu et al. 'ToolNet: Connecting Large Language Models with Massive Tools via Tool Graph'. In: (Feb. 2024). URL: http://arxiv.org/abs/2403.00839.

[68] Yu Du, Fangyun Wei and Hongyang Zhang. 'AnyTool: Self-Reflective, Hierarchical Agents for Large-Scale API Calls'. In: (Feb. 2024). URL: http://arxiv.org/abs/2402.04253.

[69] Michael Fore, Simranjit Singh and Dimitrios Stamoulis. 'GeckOpt: LLM System Efficiency via Intent-Based Tool Selection'. In: *Proceedings of the ACM Great Lakes Symposium on VLSI, GLSVLSI*. Association for Computing Machinery, June 2024, pp. 353–354. ISBN: 9798400706059. DOI: 10.1145/3649476.3658784.

[70] Qiantong Xu et al. 'On the Tool Manipulation Capability of Open-source Large Language Models'. In: (May 2023). URL: http://arxiv.org/abs/2305.16504.

[71] Shishir G. Patil et al. 'Gorilla: Large Language Model Connected with Massive APIs'. In: (May 2023). URL: http://arxiv.org/abs/2305.15334.

[72] Lifan Yuan et al. 'CRAFT: Customizing LLMs by Creating and Retrieving from Specialized Toolsets'. In: (Sept. 2023). URL: http://arxiv.org/abs/2309.17428.

[73] Yifan Song et al. 'RestGPT: Connecting Large Language Models with Real-World RESTful APIs'. In: (June 2023). URL: http://arxiv.org/abs/2306.06624.

[74] Yinger Zhang et al. 'Reverse Chain: A Generic-Rule for LLMs to Master Multi-API Planning'. In: (Oct. 2023). URL: http://arxiv.org/abs/2310.04474.

[75] Siyu Yuan et al. 'EASYTOOL: Enhancing LLM-based Agents with Concise Tool Instruction'. In: (Jan. 2024). URL: http://arxiv.org/abs/2401.06201.

[76] Yang Xu et al. 'Concise and Precise Context Compression for Tool-Using Language Models'. In: (July 2024). URL: http://arxiv.org/abs/2407.02043.

[77] Rui Yang et al. *GPT4Tools: Teaching Large Language Model to Use Tools via Self-instruction*. Tech. rep. URL: https://github.com/AILab-CVC/GPT4Tools..

[78] Shibo Hao et al. *ToolkenGPT: Augmenting Frozen Language Models with Massive Tools via Tool Embeddings*. Tech. rep. URL: https://github.com/Ber666/ToolkenGPT.

[79] Lei Li et al. 'Tool-Augmented Reward Modeling'. In: (Oct. 2023). URL: http://arxiv.org/abs/2310.01045.

[80] Boshi Wang et al. 'LLMs in the Imaginarium: Tool Learning through Simulated Trial and Error'. In: (Mar. 2024). URL: http://arxiv.org/abs/2403.04746.

[81] Timo Schick Jane Dwivedi-Yu Roberto Dessì and Roberta Raileanu Maria Lomeli Eric Hambro Luke Zettlemoyer Nicola Cancedda Thomas Scialom FAIR. *Toolformer: Language Models Can Teach Themselves to Use Tools*. Tech. rep.

[82] Zhiruo Wang et al. 'What Are Tools Anyway? A Survey from the Language Model Perspective'. In: (Mar. 2024). URL: http://arxiv.org/abs/2403.15452.

[83] Zhibin Gou et al. 'CRITIC: Large Language Models Can Self-Correct with Tool-Interactive Critiquing'. In: (May 2023). URL: http://arxiv.org/abs/2305.11738.

[84] Deepak Nathani et al. 'MAF: Multi-Aspect Feedback for Improving Reasoning in Large Language Models'. In: (Oct. 2023). URL: http://arxiv.org/abs/2310.12426.

[85] Yue Huang et al. 'MetaTool Benchmark for Large Language Models: Deciding Whether to Use Tools and Which to Use'. In: (Oct. 2023). URL: http://arxiv.org/abs/2310.03128.

[86] Junjie Ye et al. 'ToolEyes: Fine-Grained Evaluation for Tool Learning Capabilities of Large Language Models in Real-world Scenarios'. In: (Jan. 2024). URL: http://arxiv.org/abs/2401.00741.

[87] Yongliang Shen et al. *TaskBench: Benchmarking Large Language Models for Task Automation*. Tech. rep. URL: https://github.com/microsoft/JARVIS.

[88] Jize Wang et al. *GTA: A Benchmark for General Tool Agents*. Tech. rep. URL: https://github.com/open-compass/GTA..

[89] Haiyang Shen et al. 'ShortcutsBench: A Large-Scale Real-world Benchmark for API-based Agents'. In: (June 2024). URL: http://arxiv.org/abs/2407.00132.

[90] Harsh Trivedi et al. 'AppWorld: A Controllable World of Apps and People for Benchmarking Interactive Coding Agents'. In: (July 2024). URL: http://arxiv.org/abs/2407.18901.

[91] Junjie Ye et al. 'RoTBench: A Multi-Level Benchmark for Evaluating the Robustness of Large Language Models in Tool Learning'. In: (Jan. 2024). URL: http://arxiv.org/abs/2401.08326.

[92] Junjie Ye et al. 'ToolSword: Unveiling Safety Issues of Large Language Models in Tool Learning Across Three Stages'. In: (Feb. 2024). URL: http://arxiv.org/abs/2402.10753.

[93] Yuanchun Wang et al. 'A Solution-based LLM API-using Methodology for Academic Information Seeking'. In: (May 2024). URL: http://arxiv.org/abs/2405.15165.

[94] Yuchen Zhuang et al. *ToolQA: A Dataset for LLM Question Answering with External Tools*. Tech. rep. URL: https://github.com/night-chen/ToolQA.

[95] Xinyi Hou et al. 'Model Context Protocol (MCP): Landscape, Security Threats, and Future Research Directions'. In: (Mar. 2025). URL: http://arxiv.org/abs/2503.23278.

[96] *ChatGPT plugins | OpenAI*. URL: https://openai.com/index/chatgpt-plugins/.

[97] *Introducing GPTs | OpenAI*. URL: https://openai.com/index/introducing-gpts/.

[98] *Coze: Next-Gen AI App Developing Platform*. URL: https://www.coze.com/.

[99] *Tencent Yuanqi- is an intelligent agent open platform launched by Tencent's Mixed Yuan Large Model team, supporting developers in quickly building high-quality intelligent agents and publishing them to platforms like QQ and WeChat*. URL: https://www.aibase.com/tool/30426.

[100] *Introduction | LangChain*. URL: https://python.langchain.com/docs/introduction/.

[101] Wenqi Fan et al. 'A Survey on RAG Meeting LLMs: Towards Retrieval-Augmented Large Language Models'. In: (May 2024). URL: http://arxiv.org/abs/2405.06211.

[102] Jonas Kjær Rask et al. 'The specification language server protocol: A proposal for standardised LSP extensions'. In: *Electronic Proceedings in Theoretical Computer Science, EPTCS*. Vol. 338. Open Publishing Association, Aug. 2021, pp. 3–18. DOI: 10.4204/EPTCS.338.3.

[103] *Core architecture - Model Context Protocol*. URL: https://modelcontextprotocol.io/docs/concepts/architecture.

[104] *Transports - Model Context Protocol*. URL: https://modelcontextprotocol.io/docs/concepts/transports.