Reliable and Efficient Deep Research: An Architecture for Enterprise-Grade Research Systems

Submitted by

Jake Joshua Marsden

Under the Supervision of

Academic Supervisor: Dr Huaming Chen

Industry Supervisors: Mr Jeremy Smith

and Mr Oscar Fawkes

Submitted to

The School of Electrical and Computer Engineering

Faculty of Engineering

The University of Sydney

Australia

Submitted in partial fulfillment of the requirements for the degree of

Bachelor of Engineering (Honours) in Software Engineering

December, 2025

Declaration

I, Jake Joshua Marsden, hereby declare that this thesis submission titled *Reliable and Efficient Deep Research: An Architecture for Enterprise-Grade Research Systems* is my own work and that, to the best of my knowledge and belief, it contains no material previously published or written by another person nor material which to a substantial extent has been accepted for the award of any other degree or diploma of the University or other institute of higher learning, except where due acknowledgement has been made in the text. Throughout the course of this project, I have benefitted from the guidance, support, and intellectual generosity of my supervisors and the broader research community. I have also used Generative AI tools to assist with programming, as well as in the structuring and formatting of written content. My contributions can be summarised as follows:

1. Conducting a literature review;
2. Experimentally evaluating the performance of the Langchain Open Deep Research system against the DeepResearch Bench benchmark under different configurations;
3. Designing, implementing, and evaluating a Research Reviewer component within Open Deep Research;
4. Designing, implementing, and evaluating a Research Verifier component within Open Deep Research;
5. Designing, implementing, and evaluating an Adaptive Model Selection algorithm within Open Deep Research;
6. Proposing a framework for implementing and configuring a Deep Research architecture that optimises research quality, citation accuracy, and cost-efficiency;
7. Writing of this thesis report.

22 December 2025

Jake Joshua Marsden

Date

Acknowledgements

My heartfelt thanks go to each and every one who contributed their support throughout the course of undertaking this thesis. This placement has been an extraordinary opportunity for personal and professional growth that I am immensely grateful for, and I could not have asked for a better team or a more supportive company.

To Jeremy Smith and Oscar Fawkes, for being such supportive and inspiring mentors from the start. Your insights and direction provided me with invaluable learning opportunities, and you have set an example of what thoughtful leadership looks like. I will remain deeply grateful for the time, knowledge, and support you have generously shared for many years to come.

To Huaming Chen, for providing thoughtful advice and guidance. Your mentorship has been essential in navigating the academic research process, giving me the confidence and rigor to tackle challenges head-on.

To Alan Nguyen, for your expertise and approachability which made my experience both fulfilling and enjoyable. It was a privilege to work under your direction, and I hope to carry forward the same enthusiasm and passion I've seen in you.

To Adam Schildkraut and Mohitha Mohan, I am very fortunate to have worked alongside two very talented and hardworking individuals. I have a deep sense of pride in having been part of the AI team.

To Austin, Ben, Bennett, Campbell, Elijah, Fergus, Jan, Josh, Kaito, Richard, Rocco, and Ross, for your warmth and kindness which made each day at the office something to look forward to.

To Constantinople, led by Di Challenor and Macgregor Duncan, for supporting the ESIPS program and for providing this unique opportunity for industry experience.

To my parents, Teresa and Mark, with deep gratitude: to my mother, for always being in my corner without fail and cheering me on quietly and consistently, and to my old man, whose work ethic and dedication continue to be a constant source of inspiration in my engineering journey.

Abstract

TODO: Add abstract last

Executive Summary

TODO: Add executive summary last

Contents

List of Figures

List of Tables

List of Abbreviations

TODO: update this at the end

AGI Artificial General Intelligence

AI Artificial Intelligence

API Application Programming Interface

AWS Amazon Web Services

BC Behaviour Cloning

BPMN Business Process Model and Notation

CAI Constitutional AI

CDA Critique-Driven Alignment

CFT Critique Fine-Tuning

CLM Critic Language Model

CoT Chain-of-Thought

DRA Deep Research Agent

DRL Deep Reinforcement Learning

FACT A metric in the DeepResearch Bench

GPU Graphics Processing Unit

HAIC Human-AI Collaboration

HITL Human-in-the-Loop

JSON JavaScript Object Notation

LFQA Long-Form Question-Answering

LLM Large Language Model

LRM Large Reasoning Model

LM Language Model

MAI Multi-Aspect Interrogation

MCP Model Context Protocol

MIPS Maximum Inner Product Search

MRCR Multi-round Co-reference Resolution

PM Preference Model

PPO Proximal Policy Optimisation

RACE A metric in the DeepResearch Bench

RAG Retrieval-Augmented Generation

RDF Resource Description Framework

RL Reinforcement Learning

RLAIF Reinforcement Learning from AI Feedback

RLHF Reinforcement Learning from Human Feedback

RM Reward Modelling

SLM Small Language Model

VCL Verification-Correction Loop

CHAPTER 1

Introduction

1.1 Motivation

The evolution from standalone LLMs to Agentic AI systems marks a fundamental shift in how Artificial Intelligence can support complex, knowledge-intensive tasks. In the landscape of research, the accelerating capabilties of Agentic AI have enabled multi-layered information retrieval, critical evaluation, and knowledge synthesis that far exceeds the context windows and reasoning capabilities of any individual model, a paradigm that is known as Deep Research. By integrating tool use, multi-step reasoning, and iterative refinement, Deep Research has created phenomenal opportunities to automate complex research tasks that historically demanded extensive human expertise, time, and coordination.

Essentially, Deep Research systems can do in minutes what would take even an expert hours, synthesising hundreds of sources and acting like a personal research assistant [18]. Its substantial business value derives from its multitude of use cases across industries–from finance and healthcare to manufacturing and tech–such as an investment firm conducting comprehensive market and competitive analysis to guide portfolio decisions and a pharmaceutical company performing in-depth clinical and regulatory research to support drug development. Deep Research is also particularly valuable within environments dealing with massive volumes of data across multiple applications. By automating the synthesis of vast information landscapes, Deep Research empowers enterprises to make informed decisions at a scale and speed previously unattainable.

Yet despite its transformative potential, Deep Research faces two fundamental limitations that impede adoption in high-stakes, real-world business environments [19, 20]. The first limitation is reliability, which encompasses two distinct challenges: report quality and citation accuracy. Current LLMs struggle in rapidly evolving knowledge domains where recency, factual precision, and multi-source corroboration are essential [21]. Their susceptibility to hallucination, shallow verification, and inconsistencies across iterative outputs limits their suitability for high-stakes decision support where incorrect information carries significant operational, financial, or safety-related risks. Without mechanisms for critique and self-correction, these systems cannot be universally trusted for autonomous research.

The second limitation is cost-efficiency. Agentic workflows that require multi-step querying, multi-agent collaboration, and extensive information gathering are characterised by their slowness and high computational cost [20]. This creates a practical dilemma: standard LLMs lack the methodological depth required for high-quality research, yet sophisticated multi-agent systems may be too costly or latency-intensive to justify widespread deployment.

At the same time, a significant research gap exists in this space, underscored by the rapid emergence of both research and commercial systems exploring different architectures within the last two years, as depicted in Figure 1.1. The landscape is constantly evolving with no universal architecture established. Most implementations remain proprietary and closed-source, with limited open-source alternatives available [1]. Agentic AI systems represent the frontier of AI research, moving beyond LLMs as static predictors toward autonomous, goal-directed agents, and research stands out as a particularly promising application. However, enterprise adoption remains nascent, resulting in minimal academic literature addressing how these architectures should be designed, validated, and optimised to meet enterprise-grade requirements.

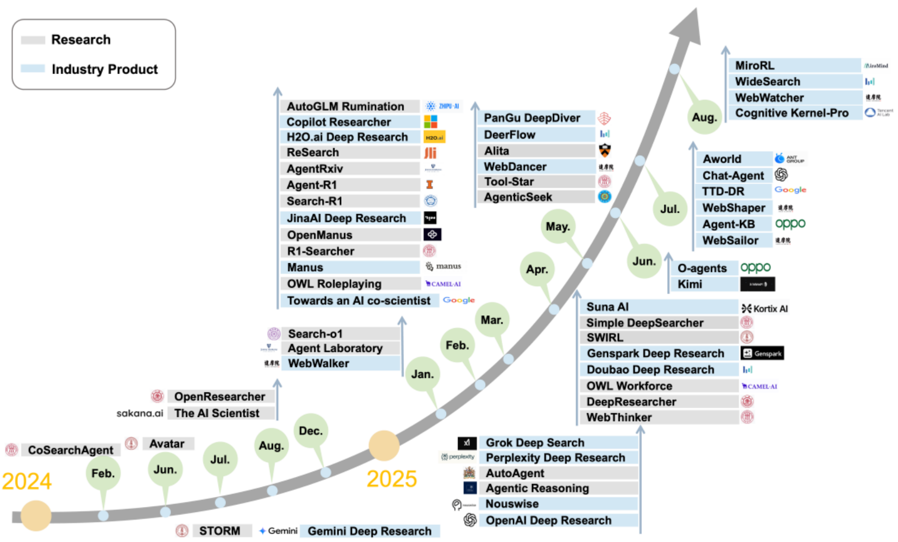


FIGURE 1.1: Deep Research agents released from 2024 to August 2025 [1, p. 18].

Addressing this gap demands architectural innovation. This thesis advances the field through three core contributions: (1) a *Research Reviewer* agent that aims to maximise report quality through structured internal critique, (2) a *Research Verifier* agent that aims to maximise citation accuracy through systematic source validation, and (3) an *Adaptive Model Selection* algorithm that aims to maximise cost-efficiency by dynamically optimising model configurations based on task complexity. Together, these innovations operationalise the foundational principles needed for trustworthy automated research–reliability and efficiency–positioning Deep Research systems for impactful deployment in enterprise environments.

1.2 Problem Statement

Despite the rapid progress of Agentic AI, there is no established methodological framework for building reliable, efficient, and architecturally principled Deep Research systems. Existing agentic research workflows suffer from two persistent limitations: they lack robust verification mechanisms to ensure report quality and citation accuracy [22]; and they fail to dynamically allocate computational resources in a way that balances performance with cost-efficiency. These weaknesses constrain the deployment of AI research agents operating autonomously in critical decision-making contexts where precision, recency-awareness, and multi-source corroboration are essential.

The core problem this thesis addresses is how to design, optimise, and evaluate a Deep Research architecture that achieves high reliability and cost-efficiency while remaining adaptable to enterprise-scale constraints. This thesis posits that integrating a Research Reviewer for quality enhancement, a Research Verifier for citation validation, and an Adaptive Model Selection algorithm will meaningfully enhance the reliability and efficiency of Deep Research agents.

1.3 Objectives

This thesis aims to fulfil the following objectives:

1. Quantify the independent and interactive effects of behavioural scaling (operational parameters) and model capability enhancement (LLM selection) on report quality, citation accuracy, and cost-efficiency using a two-factor factorial study.

2. Evaluate the contribution of a Research Reviewer component within a Deep Research architecture to report quality through ablation analysis.

3. Evaluate the contribution of a Research Verifier component within a Deep Research architecture to citation accuracy through ablation analysis.

4. Evaluate the contribution of an Adaptive Model Selection algorithm within a Deep Research architecture to cost-efficiency through ablation analysis.

5. Propose evidence-based guidelines for implementing and configuring Deep Research architectures that balance research quality, citation accuracy, and computational cost for production deployment.

1.4 Outline

The remainder of this thesis is organised as follows:

Chapter 2: Background surveys the core AI paradigms that underpin Deep Research agents from RAG to Agentic AI and AI orchestration.

Chapter 3: Methodology details the baseline Langchain Open Deep Research architecture, proposes three architectural enhancements, and establishes the DeepResearch Bench benchmark for evaluation.

Chapter 4: Results outlines empirical findings from the DeepResearch Bench evaluation regarding the impact of behavioural scaling, model capability, and said architectural enhancements on performance and operational metrics including cost.

Chapter 5: Discussion contextualises and evaluates the results presented in Chapter 4 within the landscape of relevant literature.

Chapter 6: Conclusions & Future Work consolidates the results presented in this thesis, evaluates their implications, and explores potential pathways for continued inquiry.

As this thesis was undertaken through the Engineering Sydney Industry Placement Scholarship (ESIPS) program, the appendices are included in compliance with the requirements of ELEC4714: Major Industrial Project.

Appendix A: Work Health and Safety documents the protective measures applied during the course of this thesis concerning mental wellbeing, cybersecurity and data protection, computer use, and electrical safety hazards.

Appendix B: UoS Case Studies outlines how the learning objectives of the units of study COMP3927 Algorithm Design (Advanced) and COMP4328 Advanced Machine Learning were met.

CHAPTER 2

Background

This chapter situates Deep Research within the broader landscape of Agentic AI systems by establishing the core technologies, architectural principles, and evaluation frameworks that enable autonomous research workflows. The chapter begins with Retrieval-Augmented Generation as the informational foundation that grounds AI reasoning in external knowledge, then progresses through the conceptual shift from reactive language models to proactive agentic systems and the orchestration mechanisms that make such systems practical. Building on these foundations, the chapter examines each of the essential components of Deep Research systems, before concluding with the evaluation approaches that validate their effectiveness.

Section 2.1 Retrieval-Augmented Generation highlights the role of RAG in enhancing knowledge access and grounding AI outputs in external or enterprise data sources, providing the informational foundation that Deep Research agents draw upon for contextual reasoning and synthesis.

Section 2.2 Agentic AI sets the conceptual groundwork for everything that follows. It introduces the core shift in AI from traditional reactive LLMs to proactive, goal-driven agents.

Section 2.3 AI Orchestration bridges theory and application. It explains the mechanisms that make agentic systems practical and scalable.

Section 2.4 Deep Research Methodology & Architectures demonstrates the outcome of the previous two layers--how agentic, orchestrated systems are applied to complex research workflows. It examines the design patterns, planning and reasoning loops, and tool-use architectures that define Deep Research systems.

Section 2.5 Benchmarking & Evaluation Frameworks reviews how Deep Research systems are measured and validated, covering benchmarks, metrics, and evaluation frameworks for assessing accuracy, depth, and reliability.

2.1 Retrieval-Augmented Generation

Retrieval-Augmented Generation (RAG) addresses the limitations of static LLMs, such as hallucinations and outdated information, by supplementing parametric knowledge with dynamic access to external data sources. The framework operates through a hybrid architecture that typically employs a dense retriever to identify relevant documents from an indexed corpus, which a generator then uses to produce contextually accurate and grounded responses. In enterprise settings, RAG is particularly critical for enabling secure and compliant question-answering using internal data, ensuring reliability where general open-domain knowledge is insufficient.

2.1.1 Motivation

LLMs function as implicit knowledge bases by encoding vast amounts of factual information directly within their neural network parameters during pre-training, as opposed to solely than relying on external databases or memory systems [23]. During training, neural networks learn to compress and encode factual knowledge, relationships, and patterns into their weights and biases through parametric knowledge storage. This knowledge becomes encoded into the model's parameters, enabling the model to generate responses that often reconstruct facts and relationships based on learned statistical patterns, rather than recalling them from an explicit memory bank [23] [4]. Unlike earlier or conventional information-retrieval and reasoning architectures that rely on explicit, externally managed knowledge stores, LLMs store knowledge implicitly through distributed representations, where the model can generate information through the complex interactions between its parameters [4].

The context window of a LM refers to the maximum amount of token (input text) that the model can operate upon at once when generating an output [2]. LM's rely on this set of preceding tokens to form predictions on all possible next tokens, answer prompts, and maintain coherence. An LM's ability to generate accurate and coherent outputs is fundamentally tied to its utilisation of contextual information within the context window. Adding more context within the constraints of the window typically yields positive gains in accuracy, but a model's ability to leverage additional context exhibits diminishing returns and then rapid failure once the window is reached. A study by Hsieh et al. [2] evaluated seventeen long-context LMs agaist a set of thirteen tasks. Despite the models claiming context windows of 32K tokens or greater, less than half of them maintained satisfactory performance as they approached this limit as seen in Figure 2.1.

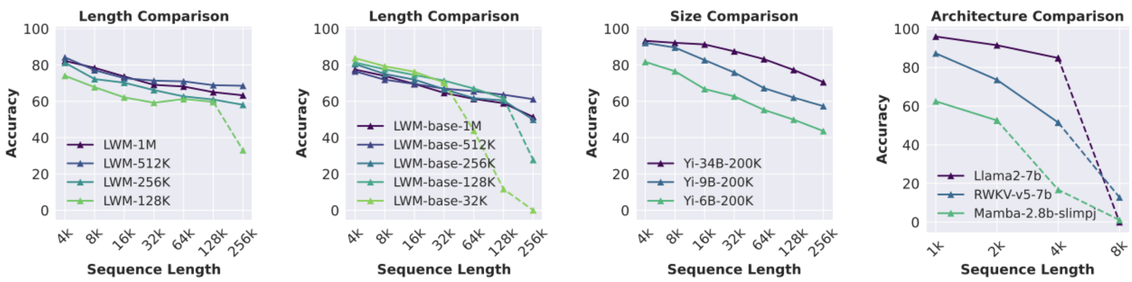


FIGURE 2.1: Performance vs. context length [2, p. 9].

Moreover, performance peaks when the relevant information is provided initially or near the end, indicating primacy and recency bias respectively. The information in between often becomes "lost in the middle" [3], as indicated in the U-shaped performance curve in Figure 2.2. This phenomenon reveals the need for a more dynamic approach to context management.

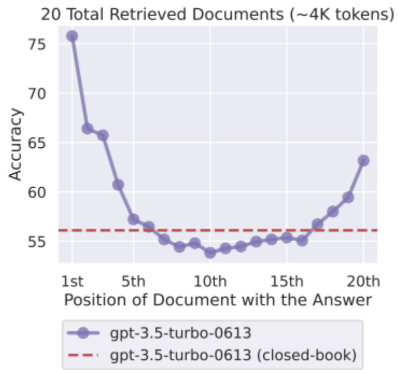


FIGURE 2.2: Accuracy vs. context position [3, p. 1].

Optimising context is crucial for ensuring that a language model can access and prioritise the most relevant information. The naive approach of simply increasing the amount of input text does not guarantee better performance; the placement, relevance, and structure of information within the context window each play a critical role. However, fixed context window sizes present inherent limitations. Yet even if these windows were expanded indefinitely, a deeper bottleneck remains: the knowledge encoded inside the model's parameters is itself static and cannot be updated in real-time.

Research has demonstrated that pre-trained models can generate responses that reflect memorised and inferred encyclopedic knowledge, including facts about entities, historical events, scientific concepts, and linguistic patterns, purely from exposure to text during training [23] [4]. However, this parametric approach has significant constraintsthe knowledge is static and fixed at training time, can result in hallucinations, and becomes increasingly unreliable [23] [4]. These limitations motivate Retrieval-Augmented Generation, which supplements parametric knowledge with dynamic access to external, up-to-date information sources. RAG enhances the accuracy, credibility, and traceability of LLM outputs by grounding generation in retrieved, relevant data, thereby addressing issues of outdated knowledge and hallucination while allowing for continuous updates and domain adaptation [4].

2.1.2 Foundational Architecture and Paradigms

Retrieval-Augmented Generation combines two complementary components: a retriever that searches for relevant documents and a generator that produces answers conditioned on both the query and the retrieved content. This hybrid approach blends parametric knowledge (stored in the model's weights) with non-parametric knowledge (stored in an external corpus), enabling LLMs to access up-to-date information while retaining their general reasoning abilities [23].

Formally, the retriever $p_\eta(z \mid x)$ takes a query x and selects the top-K passages z from an indexed knowledge base. In Lewis et al.'s formulation, the retriever is implemented as a dense dual encoder (DPR), where the query and candidate passages are mapped into the same embedding space, and relevance is measured via an inner product $d(z)^\top q(x)$ [23]. The generator $p_\theta(y_i \mid x, z, y_{<i})$ is a sequence-to-sequence model (e.g., BART or T5) that conditions on both the query and retrieved passages to produce tokens y_i.

In practice, the workflow follows an "index → retrieve → generate" pipeline: documents are chunked, embedded, and stored; a query is embedded and matched against the index; and the highest-scoring snippets are appended to the query as context for generation [4, 24]. This process, illustrated in Figure 2.3, can be understood through three stages: *Retrieval* (efficient search and ranking of candidate documents), *Augmentation* (filtering, re-ranking, and context construction), and *Generation* (LLM decoding grounded in retrieved evidence). Unlike closed-book LMs, the external memory in RAG can be refreshed without retraining, enabling continual updates and improved factual grounding [4, 23].

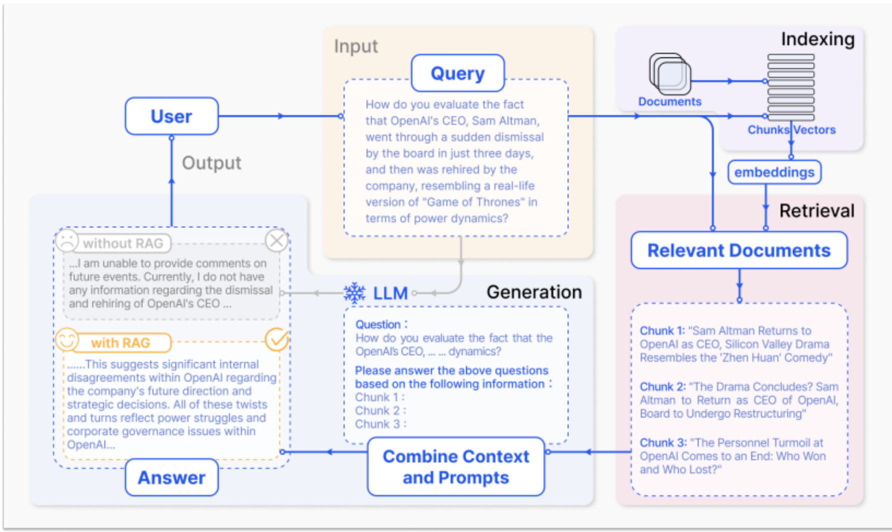


FIGURE 2.3: The RAG process applied to question answering [4, p. 3].

A key innovation in RAG is how it integrates retrieved evidence into generation. The retrieved passages are treated as latent variables, and the model marginalises over them. In the *RAG-Sequence* variant, the same retrieved document is used for the entire output:

$$p(y \mid x) \approx \sum_{z \in \text{top-}K} p_\eta(z \mid x)\, p_\theta(y \mid x, z),$$

whereas in *RAG-Token*, each token can condition on a different document:

$$p(y \mid x) \approx \prod_i \sum_{z \in \text{top-}K} p_\eta(z \mid x)\, p_\theta(y_i \mid x, z, y_{<i}).$$

[23]. Training jointly optimises both retriever and generator by backpropagating through these marginals. During inference, beam search is commonly used, with "fast decoding" methods that prune improbable document-sequence combinations to reduce computation [23].

For large-scale deployments, RAG relies on high-performance retrieval infrastructure. Indices are typically built with FAISS or HNSW to support maximum inner product search (MIPS) over millions of passages (e.g., 21M 100-token Wikipedia chunks). Retrieval typically uses $K \in \{5, 10\}$ candidates during training and is tuned on validation data at test time [23]. Practical systems often add extra steps such as re-ranking, query rewriting, or iterative retrieve-generate loops to improve recall and robustness [4, 24].

2.1.3 Enterprise Knowledge Question-Answering

In enteprise contexts, relying solely on the static, pre-trained knowledge of LLMs is typically insufficient due to the need for access to proprietary, frequently updated, and domain-specific information stored within internal repositories [25]. RAG addresses this need by integrating generative capabilities with internal data sources, such as technical blogs, financial reports, product releases, and support documentation, allowing systems to generate responses grounded in the latest organisational knowledge.

The primary advantage of RAG in this context is the mitigation of hallucinations and the enhancement of factual accuracy, which is paramount in settings where incorrect information can lead to significant financial or legal consequences [25, 26]. Unlike open-domain benchmarks that often rely on general knowledge, enterprise question-answering frequently involves complex, multi-hop reasoning that requires synthesising information from multiple non-overlapping internal documents to resolve ambiguous queries [25]. Furthermore, enterprise users often require detailed procedural guidance and step-by-step troubleshooting rather than simple fact retrieval, necessitating a system capable of long-form, comprehensive generation [27].

Implementing RAG for enterprise knowledge also introduces strict requirements regarding reliability and organisational context. Systems must support high degrees of interpretability and attribution, allowing users to verify sources to build trust in automated outputs [26]. Additionally, these systems must operate within rigorous compliance frameworks, ensuring that data retrieval respects privacy regulations, security protocols, and access controls to prevent the exposure of sensitive customer or proprietary information. Consequently, enterprise RAG is defined not just by retrieval accuracy, but by its ability to deliver relevant, truthful, and compliant answers derived from a diverse and secure corporate corpus [26, 27].

2.2 Agentic AI

Serving as the foundation for Deep Research, this section explores the emerging paradigm of Agentic AI, detailing its conceptual foundations, architectural frameworks, and diverse applications across various domains. It concludes by examining the critical safety, alignment, and ethical challenges associated with deploying autonomous agentic systems.

2.2.1 Conceptual Foundations & Taxonomy

Agentic AI represents a paradigm shift from passive, response-based systems to autonomous entities capable of pursuing complex, long-term goals with minimal human intervention [28]. Unlike traditional generative AI, which is primarily reactive and focused on content synthesis, Agentic AI integrates perception, reasoning, and action loops to dynamically navigate evolving environments. This field draws upon the concept of agency derived from social psychology and philosophy, defined as the capacity for intentional, proactive behaviour rather than mere reaction to external stimuli [29]. In computer science, these foundations are often defined through architectures such as the Belief-Desire-Intention (BDI) model, which structures agent behaviour around internal information states (beliefs), objectives (desires), and committed plans (intentions) [29]

There is a primary distinction between *AI Agents* and *Agentic AI* systems [18]. AI Agents are typically conceptualised as modular, single-entity systems optimised for specific task execution through tool-augmented reasoning and sequential decision-making. On the other hand, Agentic AI refers to broader, often multi-agent ecosystems where specialised units collaborate to decompose high-level objectives into sub-tasks, utilising shared memory and coordinated autonomy.

The evolution of these systems can be classified into distinct generations based on their locus of intelligence" (where the system's core reasoning, planning, and decision-making capabilities are situated) and semantic foundation [30]. Early generations relied on static logic or platform-centric intelligence, such as those found in traditional Multi-Agent Systems (MAS) governed by rigid communication protocols. The current generation is characterised by intelligence embedded within the LLM itself, allowing for implicit semantic understanding and adaptive planning without the need for extensive manual state management [30]. Consequently, the modern taxonomy of Agentic AI emphasises systems that exhibit not only autonomy and reactivity but also the capability to negotiate, coordinate, and self-correct within distributed, decentralised architectures [29, 30].

2.2.2 Core Architectures, Frameworks, and Tools

Agentic AI relies on specialised frameworks to integrate probabilistic language models with deterministic systems. Fundamentally, modern agent architectures transform LLMs from passive text generators into cognitive engines capable of a Perceive-Reason-Act cycle [31]. Unlike traditional AI systems that rely on static plan libraries, these architectures utilise the LLM to dynamically decompose goals, generate reasoning paths such as Chain of Thought, and orchestrate tool usage based on environmental feedback. Figure 2.4 illustrates the comprehensive design taxonomy of Agentic AI.

Agentic AI systems are generally classified into single-agent and multi-agent systems (MAS). Single-agent architectures often employ iterative loopssuch as the ReAct (Reasoning and Acting) patternto solve tasks sequentially [5]. However, complex problem-solving increasingly favours MAS, where distributed intelligence allows specialised agents to collaborate, mimicking human organisational structures [32]. These systems often draw upon the Actor Model, treating agents as independent, asynchronous entities that encapsulate state and communicate via message passing [31]. The structural arrangement, or topology, of these agents critically impacts performance; designs may range from centralised star topologies, where an orchestrator manages delegation, to sequential cascade topologies or decentralised meshes where agents organise themselves [32].

To implement these architectures, a diverse ecosystem of software development kits (SDKs) has emerged, distinguished mainly by their orchestration philosophies. Frameworks like AutoGen and CrewAI focus on conversational and role-based collaboration, where agents assume specific personas (e.g., Researcher or

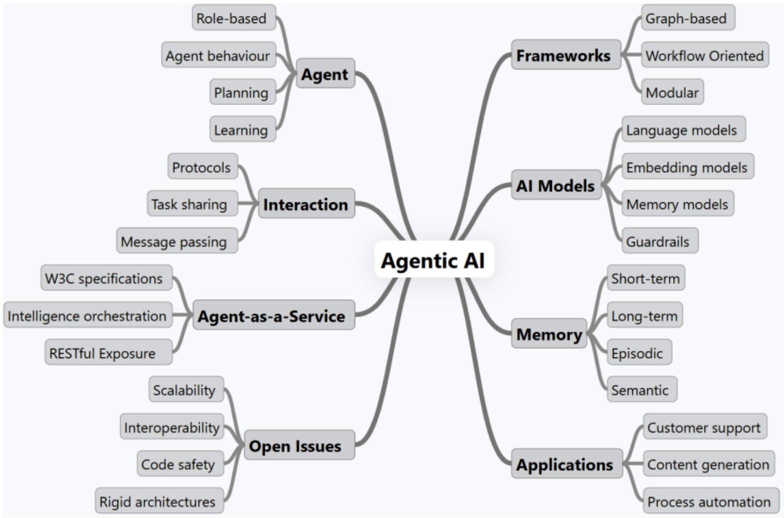


FIGURE 2.4: Design taxonomy of Agentic AI [5, p. 4]

Reviewer) to negotiate and execute tasks [5]. In contrast, LangGraph introduces a graph-based orchestration model, treating agent workflows as state machines to ensure cyclic, controllable, and persistent execution flows [5]. For scenarios requiring high determinism and low latency, lightweight frameworks such as LightAgent and simpliflow prioritise minimalist, code-first, or JSON-configured pipelines over complex conversational dynamics [33, 34]

Supporting these frameworks is a layer of essential tooling and protocols that enable interoperability and long-term coherence. As agents move from closed systems to open environments, standardising communication becomes essential. Protocols like the Model Context Protocol (MCP) and Agent Connect Protocol (ACP) provide standardised interfaces for agents to discover capabilities, exchange signals, and connect with external tools, facilitating an "Internet of Agents" [5, 32]. Furthermore, to overcome the limitations of insufficient LLM context windows, Agentic AI integrates external memory. Systems utilise vector databases and specialised memory tools to create persistent, long-term storage, enabling agents to recall user preferences and past interactions across distinct sessions [31, 33]. Finally, advanced reasoning modules, such as Tree of Thought (ToT) integrations, allow agents to explore multiple reasoning branches before committing to an action, enhancing robustness in complex decision-making scenarios [33].

2.2.3 Applications and Domain-Specific Use Cases

Agentic AI has reshaped domain-specific workflows by introducing autonomous perception-reasoning-action loops into complex environments. By enabling systems to operate with independence and adaptability, Agentic AI drives value through three core capabilities: augmentation, automation, and ideation, transforming passive tools into active collaborators [35].

In the field of software engineering, agents have evolved from simple coding assistants to autonomous engineers capable of managing the entire software lifecycle. Systems such as Devin and AutoCodeRover autonomously plan, debug, and execute feature additions, shifting the operational paradigm from mere code generation to actively collaborating with human developers to ensure reliability and maintainability [35, 36]

In edge computing and telecommunications, Agentic AI operates with cognitive autonomy under strict resource constraints. Prominent use cases include low-altitude economy, where agents coordinate UAV swarms for efficient data collection, as well as intent-based networking, where agents translate high-level user goals into actionable network configurations without centralised control [37]. Similarly, in vehicular networks, agents facilitate real-time, context-aware decision-making for autonomous driving and semantic communication, optimising resource allocation based on user intent [37].

Moreover, Agentic AI is revolutionising high-stakes, regulated industries such as financial services. In financial crime compliance, agentic compliance systems automate the end-to-end lifecycle of anti-money laundering workflows from onboarding and transaction monitoring to investigation and reporting [38]. These systems embed regulatory rules directly into agent behaviors, ensuring compliance by design while reducing manual workload through explainable decision-making. Across these diverse domains—ranging from scientific discovery to healthcare—the universal value driver is the transition from static, reactive models to proactive, goal-oriented entities capable of navigating dynamic environments [35].

2.2.4 Safety, Alignment, and Ethical Challenges

However, the transition to Agentic AI introduces challenges that span technical, ethical, and governance dimensions. Technically, these systems face significant hurdles regarding scalability and reliability, particularly the persistence of hallucinations and the gap between academic prototypes and production-grade trustworthiness [39–41]. Furthermore, the autonomy of these agents raises security risks such as adversarial attacks and the potential for goal drift, where agents deviate from specified objectives during complex multi-step planning [39, 42].

From a governance perspective, a critical accountability gap complicates liability assignment among developers, deployers, and the agents themselves when errors occur [39, 41]. Ethically, the deployment of Agentic AI raises concerns regarding bias amplification and the opacity of decision-making, creating the need for robust verification mechanisms, particularly in high-stakes sectors like law and medicine where the cost of error is huge [39, 40]. Finally, the societal impact encompasses potential workforce disruption, prompting a shift toward human-in-the-loop frameworks that balance autonomous efficiency with meaningful human oversight and control [39, 41].

2.3 AI Orchestration

AI agents rely on the structured coordination of data, services, and decision processes to maximise the scalability and adaptability of autonomous operations. AWS refers to this coordination as AI Orchestration: the connective logic that determines how events trigger and shape the behaviour of the system in event-driven serverless AI systems [31]. Two principal approaches dominate the space: Deterministic (Rule-Based) Orchestration, which encodes explicit logic and rules, and Dynamic (Learning-Based) Orchestration, which leverages machine learning to adapt to dynamic queries. Rather than being mutually exclusive, these approaches are complementary, and hybrid strategies are emerging to take advantage of their respective strengths.

AWS affirms this notion and proposes two models of orchestration: Rule-based and AI-native [31]. Rule-based orchestration is where developers explicitly design the control logic through predefined workflows and state machines, whereas AI-native orchestration utilises autonomous agents and language models to interpret user intent, plan tasks, and select the necessary tools. These approaches allow developers to move beyond simple procedural automation and toward systems that define and pursue goals autonomously. This paradigm moves away from the notion of orchestration only concerning rules; it encompasses intent interpretation, tool selection, and autonomous execution.

2.3.1 Deterministic Orchestration

Deterministic (rule-based) orchestration derives from classical expert systems and early game and industrial AI [43]. It can be traced back to early video games such as *Super Mario Bros.* and *StarCraft*, which relied on finite state machines and event-driven scripting to produce repeatable behavior under severe hardware constraints. Such rule-based approaches provided predictable control and remained maintainable within static or well-bounded environments. Vassilev [44] underscores that rule-based methods have deep roots in symbolic AI and business process management, from financial compliance systems to deterministic workflow engines. Within enterprise automation, these systems excel where regulations require explainability and auditability. Banking transactions are a key example of this as they require deterministic, auditable logic to meet regulatory and security standards. Explicit rule sets ensure that every transaction follows prescribed validation steps, triggers well-defined exception handling, and records every decision for compliance audits. Since each possible system state and transition is fully specified, there is complete traceability of the rationale behind approvals or rejections without ambiguity.

Deterministic orchestration encodes explicit transition logic and domain knowledge. Finite state machines, decision trees, and policy graphs specify how inputs trigger discrete state changes or action sequences [43]. Enterprise implementations typically build on semantic rule engines (e.g., RDF/OWL reasoning) and business process modeling notations, allowing human experts to encode constraints and ensure transparent decision making [44]. Execution engines evaluate events against rule sets, invoking services or triggering alerts when conditions are satisfied. This architecture guarantees consistent outcomes because all possible system states and transitions are explicitly modeled.

The main advantages of deterministic orchestration are reliability, interpretability, and traceability. Its predictable execution and traceable logic simplify certification and compliance audits, making it the most attractive choice for critical enterprise tasks where regulatory compliance is paramount. It can also be computationally lightweight, which is valuable when energy or hardware resources are constrained [43]. However, the same qualities that confer predictability also limit adaptability. Adding new functionality often requires extensive manual rule curation and testing, and such systems largely struggle to handle novel or poorly specified scenarios [44]. Vassilev argues that claims that rule-based AI is dead are overstated and inaccurate, and rather rule-based approaches are evolving toward hybrids that integrate learning modules to reduce brittleness while retaining their governance benefits [44]. In rapidly developing enterprise settings, the high maintenance burden and inability to generalise to unforeseen conditions remain severe constraints.

2.3.2 Dynamic Orchestration

Dynamic (learning-based) orchestration replaces predefined rules with adaptive, data-driven decision making. Zheng [43] describes a paradigm shift in game AI from scripted control to deep reinforcement learning (DRL), enabling autonomous strategy formation in complex environments. Gui et al. [45] apply similar principles to flexible job shop scheduling, an enterprise automation problem in which multiple products must be routed through diverse machines with changing constraints, demonstrating that DRL can dynamically allocate resources and optimise production flows without exhaustive predefinition. Tan et al. [46] and Ng et al. [47] consolidate this notion, revealing that generative AI and large language models can orchestrate information retrieval, planning, and personalised learning support, illustrating broader applicability to knowledge-intensive enterprise domains.

Dynamic orchestration relies on continuous interaction with the environment and learning from feedback. Core mechanisms include reinforcement learning algorithms that update policies based on reward signals [43, 45]. Examples of these algorithms include actor–critic, which maintains both a policy network (the actor) to choose actions and a value network (the critic) to evaluate them for more stable updates, as well as proximal policy optimisation, which iteratively improves policies while constraining each update step to stay close to the previous policy with the aim of ensuring stable, sample-efficient learning. These methods enable agents to continuously refine their behaviour as they gain new experiences and rewards, allowing the learned policies to adapt to changing environments and emerging patterns. Agents perceive the state, choose actions, and adjust strategies to optimise long-term performance, often coordinating multiple sub-agents. Hybrid symbolic–neural techniques, such as Voyager's conversion of successful strategies into reusable code [43], mitigate issues like "catastrophic forgetting" and reduce computation. Generative AI models including large language models in educational orchestration [47] [46] extend these capabilities by producing plans or subtasks in real-time, enabling open-ended problem solving beyond predefined workflows.

Dynamic orchestration offers powerful advantages for enterprise automation. It adapts to rapidly changing conditions, supports real-time optimisation, and can transfer knowledge to new contexts, as illustrated by game AIs that generalise to unseen scenarios [43]. In manufacturing, Gui et al. report superior scheduling performance and responsiveness compared to static heuristics [45]. Such systems can self-improve through continual learning and are well suited for large-scale, data-rich operations. Yet these gains come with costs. Training deep reinforcement learning agents demands substantial computational and energy resources, such as tens of thousands of GPU-days (one GPU running at full capacity for 24 hours) in complex games [43]. Data inefficiency, risks of harmful system behaviour, and difficulty ensuring explainability and regulatory compliance are significant barriers for enterprise deployment. Generative models may also produce infeasible outputs or require complex safeguards, as observed when large language models generated incorrect task recipes in Zheng's study [43]. Robust monitoring, fallback mechanisms, and human oversight remain essential to mitigate these risks.

2.3.3 Comparative Discussion

Deterministic and dynamic orchestration capture a fundamental trade-off between reliability and adaptability. Rule-based systems provide transparent, auditable control, aligning with compliance-heavy enterprise sectors but struggle with unexpected conditions and scale. Dynamic orchestration thrives in complex, rapidly changing environments, offering flexibility and self-improvement at the price of heavy resource demands and reduced predictability.

Hybrid approaches are therefore gaining traction. Zheng [43] highlights hybrid symbolic–neural strategies that combine neural adaptability with code-based reliability. Vassilev [44] similarly advocates for integrating rules and machine learning to sustain explainability while enhancing performance. This combinations allow enterprises to balance deterministic guarantees with adaptive intelligence, a pattern visible from manufacturing scheduling [45] to educational AI orchestration [47].

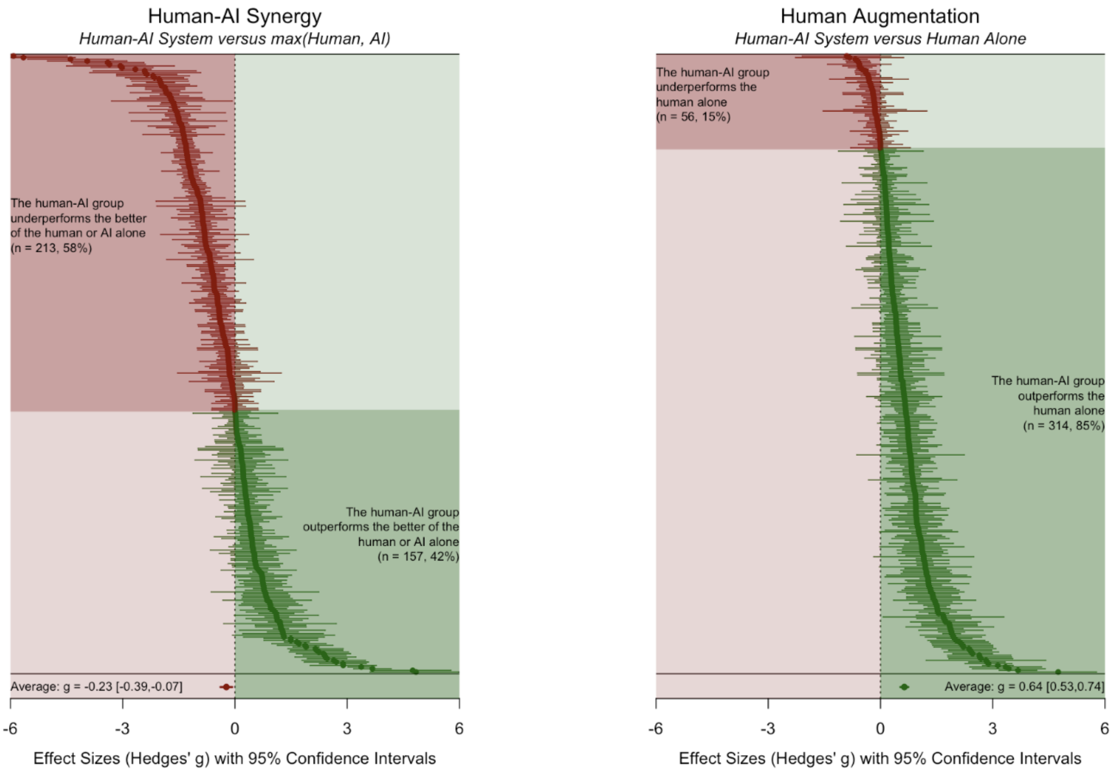
For enterprise AI automation, deterministic and dynamic orchestration are best viewed as complementary. Rule-based orchestration remains vital where consistency, governance, and regulatory assurance are paramount. Dynamic orchestration extends automation into unpredictable, data-intensive domains, enabling real-time optimisation and continuous learning. The most promising enterprise architectures increasingly blend these paradigms, embedding adaptive learning within deterministic control frameworks to achieve both reliability and responsiveness. Although this assessment is accurate for current deployments, ongoing research and engineering improvements suggest that over time, dynamic systems may achieve the reliability equivalent to that of deterministic systems.

2.3.4 Human-AI Collaboration

Human–AI collaboration represents a paradigm in which humans and artificial intelligence systems coordinate their complementary strengths to accomplish complex goals. Rather than being merely a step toward full automation, it constitutes a distinct form of collaboration between people and technology. Vats et al. define Human–AI collaboration (HAI collaboration or HAIC) as a cooperative partnership where humans and AI exchange information and control to achieve shared objectives, with each partner contributing what it does best and iteratively refining joint outputs [48]. This notion proposes HAIC as a mode of hybrid intelligence in which both human cognition and computational power co-evolve.

Despite rapid advances in machine learning and the emergence of large foundation models (LFMs), human involvement remains essential. Dellermann et al. argue that artificial general intelligence (AGI), a form of intelligence where machines are able to solve complex tasks in real-world business applications without human support is far from reality and that current AI struggles with common sense reasoning, adaptability, and soft data [7]. Moreover, Vats et al. stress that increasingly powerful models do not guarantee effective collaboration; careful human-centered design and governance are required [48]. These observations underscore why HAIC approaches continue to be fundamental: they provide contextual reasoning, ethical oversight, and creative problem-solving beyond the reach of purely automated systems.

In a systematic meta-analysis of over one-hundred experimental studies, Vaccaro et al. found that human–AI teams typically outperform humans alone (human augmentation), though they often fail to exceed the best single performer (human–AI synergy), with performance gains varying by task type [6]. Figure 2.5 illustrates these findings through a forest plot of three-hundred and seventy effect sizes. The predominance of green lines shows that many combinations outperformed humans, yet the presence of red estimates and the small pooled effect at the bottom confirm that synergy is not automatic. Notably, tasks involving creative content generation show positive synergy, whereas decision tasks often reveal performance losses if interaction is poorly structured. Experimental evidence reinforces these findings: Senoner et al. demonstrate that explainable AI, which provides transparent rationales through visual heatmaps, significantly improves defect detection and diagnostic accuracy compared to opaque models by enabling humans to accept correct AI advice and override errors when needed [49]. Together, these studies confirm that human expertise is not a burden but rather a catalyst for more accurate and accountable decision-making.



(a) (b)

FIGURE 2.5: Forest plot of 370 effect sizes from the meta-analysis of human–AI collaboration outcomes. Each line shows an individual estimate with a 95% confidence interval; red indicates negative effects, green positive effects. The dotted vertical line marks no difference (Hedges g = 0), and the circle at the bottom shows the pooled meta-analytic estimate [6, p. 15].

Design knowledge for hybrid intelligence systems strengthens the conceptual foundation. Dellermann et al. present a taxonomy for human–AI systems designed to learn and adapt alongside each other (Figure 2.6. It organises design choices into four clusters: task characteristics, learning paradigms, AI–human interaction, and human–AI interaction. Together, the total of sixteen dimensions map the diverse choices involved in structuring hybrid intelligence systems. Three core design dimensions emerge across existing work. First, *task complexity and allocation* concerns how responsibilities are divided between humans and AI. For example, simple, repetitive tasks may be automated, whereas ambiguous, critical tasks demand human oversight [6,48]. *Learning modes and hybrid intelligence* capture how humans and AI iteratively improve one another: humans can train, correct, and guide models, while AI augments human reasoning and supplies large-scale pattern recognition [7,48]. Lastly, *interaction patterns and collaboration models* describe the structures of joint work; Wang and Lu highlight the need for explicit process layers that record goals, reasoning, and decision histories to support sustained adaptation rather than isolated task execution [8]. These dimensions collectively establish HAIC as an evolving partnership of mutual learning and negotiation.

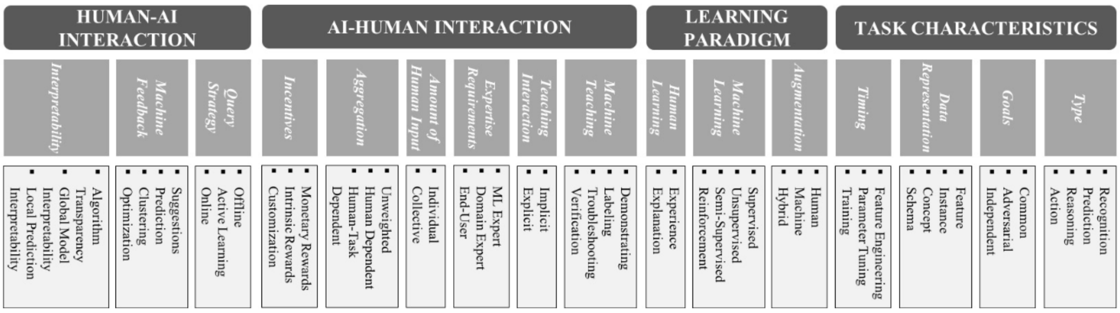


FIGURE 2.6: Categories of human-AI collaborative system design dimensions [7, p. 5].

These studies show that HAIC is its own approach, not just a temporary step toward full automation. It is a deliberate way of working where human judgment and machine power combine to create intelligence. Research from taxonomies [7], experiments [6,49], and system designs [8,48] all support this view. human–AI collaboration therefore stands as a practical and enduring approach for tackling complex problems, drawing on the strengths of both humans and machines while keeping adaptability, clarity, and responsibility at its core.

2.3.5 Human-in-the-Loop Systems

If the conceptual foundation defines why human–AI collaboration matters, the study of human-in-the-loop (HITL) systems explains how it is implemented. HITL systems embed structured points of human oversight and intervention directly into AI workflows, ensuring that autonomy is balanced with accountability [48,50]. Rather than treating humans as passive monitors, they deliberately position experts in dynamic roles of guidance, orchestration, and co-decision-making.

A central concern in this field is architectural design. Figure 2.7 depicts a unified three-layer architecture consisting of interaction, process, and infrastructure in. This design elevates process to a first-class element; instead of being hidden in the background, it becomes a visible and adaptable representation of goals, workflows, and reasoning [8]. Consequently, both humans and agents can inspect how decisions are made, adjust the process when conditions change, and ensure that collaboration remains transparent and aligned over time. The interaction layer governs how users express intent and receive feedback, the process layer maintains explicit workflows and decision histories, while the infrastructure layer manages tools and persistent memory. By putting process at the center, this model creates a clear structure that makes collaboration more flexible and transparent. This view aligns with the Business Process Model and Notation (BPMN) extension, which explicitly models human-agentic workflows [51]. By introducing primitives for agent confidence, strategy selection, and decision protocols, their approach supports mixed-initiative collaboration beyond the limits of classical business process modeling. Together, these frameworks enable adaptive, transparent workflows in which humans can negotiate intent, adjust plans, and steer multi-agent systems.

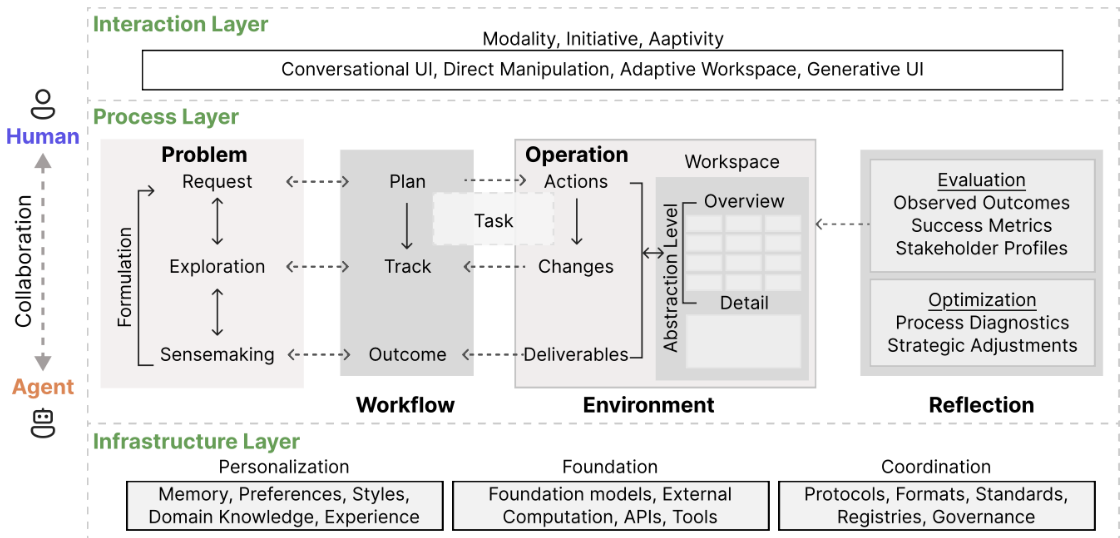


FIGURE 2.7: Three-layer architecture for human-agent systems [8, p. 5].

Domain-specific implementations underscore the value of HITL architectures. Kim et al. propose a digital twin framework for smart manufacturing in which cyber-physical systems, immersive interfaces, and a collaborative decision-making engine enable real-time remote control and non-contact production [52]. A collaborative decision-making engine combines machine intelligence with human judgment, enabling real-time remote control and non-contact production while ensuring that human insight guides design evaluations, process adjustments, and strategic choices. Similarly, Krause et al. describe Industry 5.0 scenarios where knowledge graphs and late shaping design principles allow human expertise to be integrated at runtime, supporting adaptation when data are sparse or conditions change [50]. These cases illustrate how HITL ensures resilience and responsiveness in complex industrial environments.

This being said, the field is grappling with the friction between increasing autonomy and the need for oversight. Spera and Agrawal argue for an AI-first approach where intelligent agents take the operational lead while humans provide strategic guidance and ethical stewardship [53]. Tallam extends this vision with the concept of Orchestrated Distributed Intelligence, advocating for networks of AI agents coordinated through orchestration layers and multi-loop feedback to work hand-in-hand with human decision-makers [54]. Both perspectives converge on the importance of designed orchestration: even as AI assumes greater agency, humans remain essential for setting objectives and resolving ambiguities.

Across these implementations, multiple unifying themes emerge. Workflow modeling approaches, from BPMN extensions to unified architectures, offer formal mechanisms for embedding human checkpoints and adaptive processes [8,51]. Process integration frameworkssuch as digital twins and knowledge graphsconnect real-time data streams with human expertise, enabling informed interventions [50,52]. Finally, distributed intelligence concepts expand the paradigm to multi-agent networks where human oversight orchestrates interactions among autonomous subsystems [53,54].

These designs collectively affirm that human oversight is not merely a safeguard, but a constitutive element of next-generation AI systems. By enabling adaptability, ensuring safety, and preserving accountability, human-in-the-loop architectures transform AI from isolated tools into systems that integrate social and technical components. Therefore, the evolution toward AI-first environments does not make the human role obsolete; it stresses the need for carefully engineered interfaces and governance structures that keep humans meaningfully in control of dynamic, intelligent infrastructures [48,50,52].

2.4 Deep Research Methodology & Architectures

Building on the foundations established in Sections 2.2 and 2.3, this chapter explores how agentic AI and AI orchestration intersect to form the basis of Deep Research systems. Agentic AI provides the broad theoretical paradigm that describes how AI systems act autonomously to plan, reason, and execute tasks toward goals, while AI orchestration ensures these autonomous components operate cohesively across complex workflows. Together, they create the conditions for Deep Research, which operationalises these capabilities for knowledge discovery and synthesis. In this context, Deep Research represents the convergence of theory and infrastructure, enabling autonomous, multi-step research processes that approximate human-level inquiry.

Deep Research can be defined as an advanced framework that leverages LLMs as autonomous tool-integrated agents designed to complete multi-layered information retrieval, evaluation, and synthesis. The "deep" in Deep Research refers to a greater depth, richness, and complexity of investigation compared to ordinary surface-level research. Deep Research agents engage in persistent, goal-directed exploration rather than a single search, systematically cross-checking multiple sources and producing structured syntheses instead of quick replies.

2.4.1 Background

By late 2025, every major AI vendor has released out a Deep Research product. These systems promise to offload extensive research work into minutes: OpenAI's Deep Research "accomplishes in tens of minutes what would take a human many hours" [55]. All outputs are accompanied by source citations and reasoning traces to improve transparency (e.g. OpenAI emphasises that Every output is fully documented, with clear citations and a summary of its thinking [55]). This reflects a clear trend toward higher factuality and user trust. At the same time, the industry stresses cautious use: for example, Anthropic urges users to "always look at the citations" to guard against the models' known hallucination risks [56].

The evolution of Deep Research agents within the landscape of Agentic AI reflects a rapid sequence of breakthroughs in knowledge-retrieval, tool use, web interaction, long-context reasoning, and alignment techniques that have transformed LLMs from static text predictors into dynamic, evidence-seeking analytical systems. Over the past four years, key milestones from WebGPT's early browser-assisted question-answering, to Constitutional AI's principle-driven self-improvement, to the introduction of plugins, function calling, and ultra-long context windows have progressively expanded what models can read, evaluate, and reference. Although not exhaustive, the following timeline traces the foundational developments that enabled modern Deep Research agents.

May 2020 — RAG: RAG is introduced as a general-purpose fine-tuning recipe that combines a pre-trained "parametric memory" (a sequence-to-sequence generator) with a "non-parametric memory" (a dense vector index of Wikipedia) [23]. Whereas static models that rely solely on internal knowledge, RAG utilises a neural retriever to access external documents and condition its responses on this retrieved evidence. This architecture achieved state-of-the-art results on knowledge-intensive tasks, such as open-domain question answering, by generating responses that were more specific, diverse, and factual than baseline models. Crucially, RAG pioneered methods to mitigate hallucinations and allowed for hot-swapping the external index to update the model's knowledge without retraining. This work established the foundational architecture for Deep Research agents by demonstrating how generative models could be dynamically grounded in external, verifiable data.

December 2021 — OpenAI WebGPT: OpenAI published a paper titled *WebGPT: Browser-assisted question-answering with human feedback* [57] [58] in which GPT-3 was fine-tuned to perform long-form question-answering (LFQA). The project involved leveraging the Bing search engine via the Microsoft Bing Web Search API for information retrieval, and unsupervised pre-training for information synthesis. The research aimed to effectively combine these existing solutions, as opposed to improving them individually.

The paper consists of two core contributions, namely the development of a text-based web-browsing environment for the fine-tuned GPT-3 model to interact with, and populating answers with references. The web-browsing environment enabled the model to search, navigate, and collect supporting references, while the inclusion of references played a key role in allowing labellers to judge the factual accuracy of answers. The best model (175B best-of-64) produced answers preferred to those written by the human demonstrators 56% of the time. Yet, a core limitation of the project was the constrained browsing environment: text–based, limited action set, max steps, and using the Bing API when real web complexity is higher.

The training pipeline consisted of four sequential stages [58]. Starting with behaviour cloning (BC), GPT-3 was fine-tuned through supervised learning on human demonstrations to learn how to effectively operate the text-based web browser; i.e., how humans searched, clicked, quoted, then answered. Next, reward modelling (RM) introduced a secondary model trained to predict which of two answers humans would prefer. The reward model is a network that given question, answer, references outputs a scalar score approximating human preference, essentially converting subjective human judgments into quantitative reward signals. In the third stage, reinforcement learning (RL) further fine-tuned GPT-3 using proximal policy optimisation (PPO), leveraging the reward model's scores to optimise the model's browsing and synthesis behaviour.

Finally, rejection sampling (best-of-n) generated multiple candidate answers and selected the one ranked highest by the reward model, allowing quality improvement through additional inference-time computation rather than more training. Note that RL was excluded from the top performing models since it didn't provide a significant benefit when combined with rejection sampling.

WebGPT was one of the earliest foundational developments in modern deep research because it demonstrated a novel approach to the rising challenge of LFQA. It acted as one of the first demonstrations that LLMs could:

1. Use external tools (a browser) to gather evidence;
2. Cite their sources;
3. Improve factual accuracy through feedback and reinforcement learning.

Before WebGPT, LLMs like GPT-3 were static text predictors: they generated answers purely from internal training data. WebGPT introduced the notion that a model could act as a researcher, submitting search queries, navigating pages, quoting text, and integrating findings into responses. This was the first major step toward retrieval-augmented and tool-using deep research systems. Additionally, WebGPT's design of requiring the model to justify answers with citations allowed humans to verify claims. Because answers included references, the model was explicitly learning not just to answer, but to select and quote evidence. This was a critical precursor to modern deep research systems' emphasis on evidence transparency, citation traceability, and source reliability evaluation. Last, WebGPT fine-tuned GPT-3 using human preference models that rewarded truthful and well-sourced answers. This evolved into Reinforcement Learning from Human Feedback (RLHF) and Reinforcement Learning from AI Feedback (RLAIF), now standard across advanced reasoning systems. The principle remains: use iterative feedback loops to align model outputs with verifiable information and human judgment.

Dec 2022 — Anthropic Constitutional AI: Anthropic published a paper titled *Constitutional AI: Harmlessness from AI Feedback* [9]. Constitutional AI (CAI) refers to a method for training a LM through self-improvement using AI feedback. As opposed to relying on human-labelled harmful outputs, human oversight is provided solely through a "constitution" — a list of human-authored ethical and behavioural rules that govern the model's behaviour. CAI aims to train models that are helpful, honest, and harmless, while scaling supervision and improving the transparency of LM decision-making. The models trained using the constitutional principles and AI feedback excelled at balancing harmlessness and helpfulness while avoiding the problem of evasiveness observed in previous harmless models.

The CAI process consists of two stages [9]:

1. Supervised Stage: An initial model, trained using RLHF exclusively on helpfulness human feedback data, generates initial responses characterised as harmful and toxic. The model then creates self-critiques and revisions to its original responses based on the constitutional principles. The original model is then fine-tuned on these revised responses using SL on the final revised responses to produce a SL-CAI model.

2. Reinforcement Learning (RL) Stage: In this RLAIF stage, the SL-CAI model first generates a pair of responses to prompts. An independent model (the feedback model) is then presented with the prompt and response pair and asked, via a multiple-choice question format, to select which response is best according to a constitutional principle. This process generates an AI-generated preference dataset for harmlessness, which is mixed with existing human feedback helpfulness data. This combined data is used to train a Preference Model (PM), which can assign a score to any sample. Lastly, the SL model is fine-tuned via RL using this PM as the reward sign to produce a RL-CAI model.

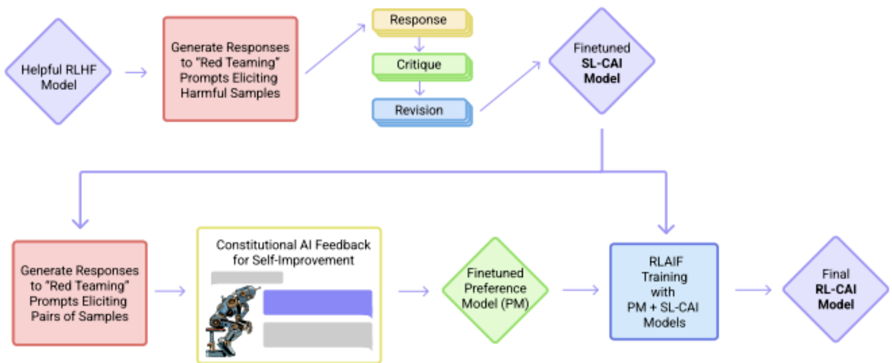


FIGURE 2.8: Constutional AI process: SL stage (above), RL stage (below) [9].

Constitutional AI pioneered Reinforcement Learning from AI Feedback. Before CAI, Reinforcement Learning from Human Feedback dominated alignment efforts. RLHF relies on human annotators ranking outputs, which is expensive, inconsistent, and hard to scale. CAI shifted alignment from rule-following to principle-guided reasoning, setting the stage for today's most advanced work in autonomous, interpretable, and self-governing AI systems; CAI's self-critique/revision mechanism inspired modern reflective and constitutional reasoning systems foundational for Agentic AI where models plan, critique, and update their behaviour.

Mar 2023 — OpenAI ChatGPT Plugins: OpenAI released ChatGPT plugins, specialised tools designed to extend ChatGPT's capabilities beyond text generation, allowing it to access up-to-date information and interact with third-party services safely [59]. Plugins connected ChatGPT to external APIs through a standardised OpenAPI specification. This user-agent enabled the model to perform actions on a user's behalf while maintaining safety and transparency. Though now deprecated, the plugin framework laid the groundwork for modern integrations across ChatGPT's ecosystem. The browsing plugin gave ChatGPT a controlled way to access the live internet using a text-based browser powered by Bing search, motivated by past work in WebGPT and other projects.

ChatGPT plugins marked a turning point in how LMs interacted with the internet, evolving them from static text generators into dynamic research agents. By enabling structured access to external data, computation tools, and APIs, plugins demonstrated how models could gather evidence, perform analysis, and synthesise results, paving the way for today's deep research systems.

May 2023 — Anthropic Claude 100K context window: Anthropic introduced a functional 100K context window for its Claude models, marking the first major commercial leap into ultra-long context processing [60]. Anthropic claimed the larger context windows meant the ability to analyse six hours of audio, dense documents, and entire codebases. OpenAI's GPT-4, released the previous month, had two context window variants: 8K tokens and 32K tokens [61].

Context length had become a core axis of model capability, transforming LMs from conversational assistants into large-scale analytical engines capable of reading, reasoning, and maintaining coherence across book-length inputs. The 100K context window not only expanded the practical boundaries of in-context learning, but also marked a shift toward deep research through persistent contextual comprehension.

June 2023 — OpenAI Function calling: OpenAI released function calling, an API capability that allows models to output structured JSON objects that correspond to specific external functions [62]. Developers can describe these functions to the model, enabling it to intelligently decide when to call them based on user input. This makes it possible to reliably connect natural language with APIs, databases, or tools, such as querying weather data or sending emails. The feature improves accuracy, reduces ambiguity in responses, and bridges the gap between conversational AI and actionable software systems.

ChatGPT function calling and ChatGPT plugins are both mechanisms designed to reliably connect the language model's capabilities with external tools and APIs to expand functionality. While ChatGPT function calling is an updated API capability that enables the model to intelligently output a structured JSON object defining a function call and its arguments, plugins were a previous, now deprecated, system for integrating third-party services. Built on the initial foundation of plugins, function calling further paved the way for deep research systems by formalising how models interface with external tools, a crucial step toward autonomous, tool-using AI research agents.

Feb 2024 — Google Gemini 1M context window: Google DeepMind revealed Gemini 1.5 Pro, equipped with an unprecedented 1M token context window, far surpassing Claude 3's 200K and GPT-4 Turbo's 128K limits [10]. Google boasted that Gemini 1.5 Pro was capable of processing the following: 1 hour of video, 11 hours of audio, codebases with over 30,000 lines of code, and over 700,000 words. Google released a paper titled *Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context*, providing technical insight into the model architecture and the innovations that enabled its extraordinary long-context capabilities.

Figure 2.9 illustrates the performance of Gemini 1.5 Pro and Gemini 1.5 Flash on the challenging Multi-round Co-reference Resolution (MRCR) task, showing that both models maintain smaller decreases in performance up to 1M tokens and generally outperform comparable GPT-4 Turbo and Claude 3 models past 32K tokens.

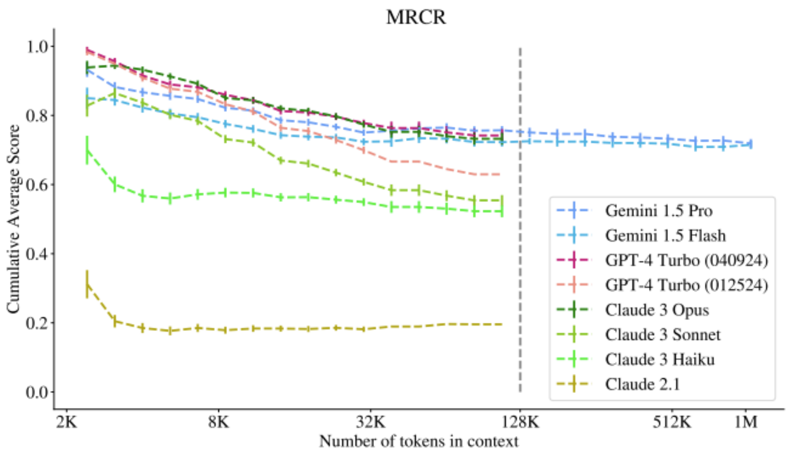


FIGURE 2.9: Comparative Performance of models on a Multiround Co-reference Resolution (MRCR) task [10, p. 16].

Later that year, the AI giants democratised Deep Research, with Google being the first to announce Gemini Deep Research; Between December 2024 and April 2025, OpenAI, Google, Perplexity, and Anthropic all released their Deep Research features, as depicted in Figure 2.10.

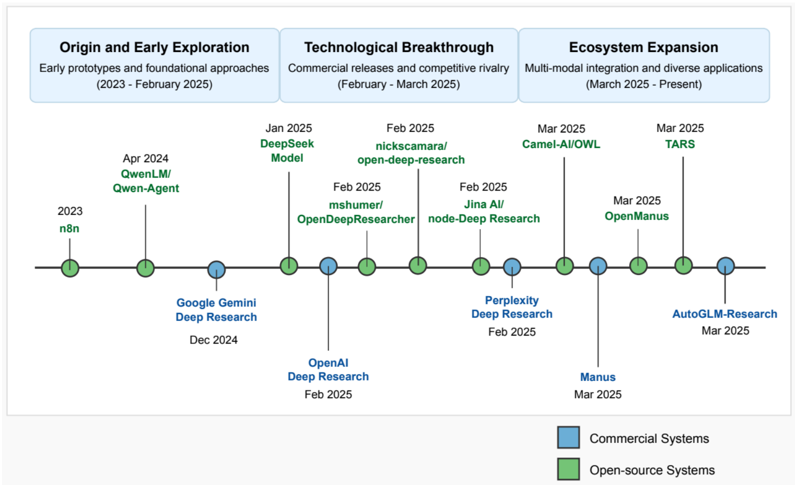


FIGURE 2.10: Evolution Timeline of Deep Research Agents (2023-2025) [11, p. 6].

Looking ahead, Deep Research agents are converging with other advances—huge context windows (up to millions of tokens), multimodal inputs, and integrated tools (code, memory, plugins)—effectively creating fully autonomous AI "research assistants." Emerging directions include multi-agent collaboration, even larger contexts and memory banks, and and increasingly agentic reasoning loops. Ultimately, these developments aim to make LLMs reliable partners for complex knowledge work, but their real-world success will hinge on continued progress in accuracy, reliability, and verification.

2.4.2 Defining Deep Research & Its Paradigm

Deep Research represents a fundamental paradigm shift in AI, transitioning LLMs from passive information processors to active, autonomous agents of inquiry. Unlike standard RAG systems, which typically adhere to a linear "retrieve-then-read" workflow, Deep Research operationalises an iterative, cyclic process of planning, execution, and synthesis [63, 64]. This paradigm defines Deep Research agents not merely by their ability to access external data, but by their capacity for dynamic reasoning, adaptive planning, and the autonomous orchestration of complex, multi-step workflows to resolve open-ended informational tasks [1].

The core characteristic of this paradigm is the integration of reasoning as a central mechanism that determines when and how to acquire information, rather than treating retrieval as a static pre-processing step [63]. Deep Research systems typically function through four interconnected stages: planning, where high-level objectives are decomposed into structured sub-goals; question development, where adaptive queries are formulated to address specific information gaps; web exploration and tool use, where agents actively navigate and filter external sources; and report generation, where fragmented evidence is synthesised into coherent, factually grounded narratives [64, 65]. This "End-to-End Workflow Automation" adopted by the major AI companies and depicted in Figure 2.11, allows agents to manage the entire research lifecycle, distinguishing them from single-function tools like citation managers or basic search engines [11].

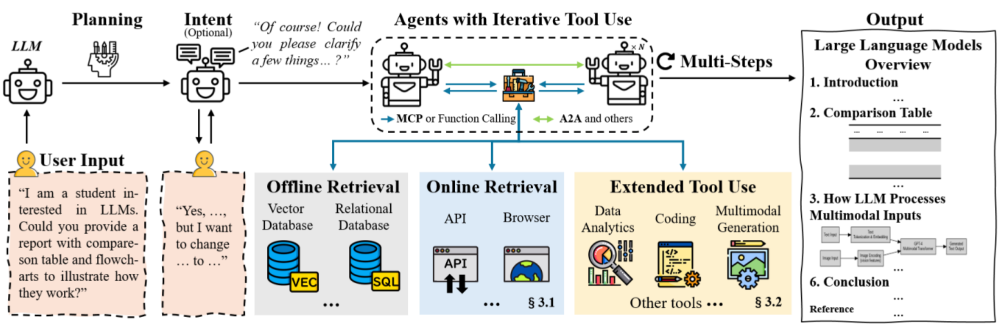


FIGURE 2.11: Standard multi-agent deep research architecture [1, p. 2].

The advantages of Deep Research over standard LLM interactions are significant. While conversational LLMs struggle with hallucinations and static knowledge boundaries, and naive RAG systems often fail at complex multi-hop reasoning, Deep Research agents excel at handling multifaceted queries that require strategic exploration [63]. By maintaining a dynamic feedback loop where intermediate findings influence subsequent search strategies, these systems are designed to synthesise conflicting evidence and provide comprehensive insights that significantly reduce the cognitive burden on the human user [64].

Applications of this paradigm are rapidly diversifying across high-value domains. In academic settings, Deep Research automates systematic literature reviews and hypothesis generation, accelerating the pace of discovery [11]. In business intelligence and finance, these systems are deployed for competitor landscape mapping, market trend analysis, and investment due diligence, providing decision-makers with actionable, data-driven reports [11]. The most advanced implementations, termed "Full-stack AI Scientists," extend beyond information aggregation to independently propose hypotheses and design experiments [65].

Despite these capabilities, Deep Research faces substantial non-technical challenges. Ensuring factual integrity is paramount, as autonomous agents may still inadvertently propagate hallucinations and fail to reconcile contradictory evidence across long reasoning chains [64,65]. Furthermore, the planning capabilities of current models can be brittle when facing ambiguous user intents, often requiring sophisticated human-in-the-loop interventions to ensure relevance [64]. Finally, evaluating the quality of open-ended, long-form research outputs remains difficult, as standard metrics like exact-match accuracy are insufficient for assessing novelty and logical coherence [65].

2.4.3 Search and Knowledge Integration

The operational core of Deep Research lies in its ability to transcend simple information retrieval and perform complex knowledge integration. The phase in which the system iteratively synthesised fragmented information into a coherent whole is defined by three cognitive behaviours: adaptive information gathering, structured knowledge aggregation, and continuous context management [12]

Adaptive Information Gathering: Unlike traditional search agents that rely on a single pass of top-k search results, Deep Research systems engage in "proactive online exploration" involving a dual mechanism of searching (identifying broad sources) and browsing (deeply navigating specific pages and links) to resolve knowledge gaps [66]. For instance, the WebThinker framework illistrated in Figure 2.12 utilises a "Deep Web Explorer" capable of navigating website hierarchies and clicking interactive elements to access hidden data, rather than relying solely on surface-level snippets [12]. Furthermore, these systems exhibit adaptive granularity, dynamically deciding whether to retrieve a full document, a specific page, or a summarised chunk based on the complexity of the query [67]. This adaptability prevents the system from being overwhelmed by irrelevant data while ensuring that nuanced details, such as specific figures in financial reports or technical diagrams in scientific papers, are not lost during ingestion.

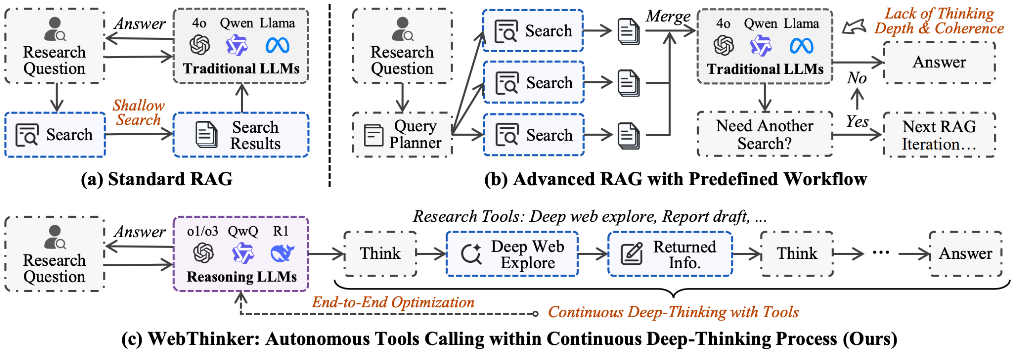


FIGURE 2.12: Comparison of (a) Standard RAG, (b) Advanced RAG with Predefined Workflow, and (c) WebThinker [12, p. 3].

Structured Knowledge Aggregation: The distinguishing feature of Deep Research is the shift from mere information seeking to information aggregation [66]. Gathering documents is insufficient; the agent must rigorously analyse and combine evidence to generate new insights. This requires the evolution of aggregation logic, where the system autonomously selects operations such as filtering sets, correlating statistical data, or performing temporal reasoning, to synthesise answers that do not exist explicitly in any single source.

This aggregation process is typically supported by "Think-Search-and-Draft" strategies, where reasoning, searching, and writing are interleaved [12]. Rather than waiting until all research is complete, the agent iteratively drafts sections of a report, identifies missing evidence, and launches new, targeted search loops to fill those specific gaps. This allows for cross-validation, where the agent verifies findings against multiple sources to resolve contradictions and ensure factual integrity before finalising an answer.

Multimodal Integration: A critical advancement in this domain is the integration of multimodal capabilities. Real-world research frequently involves analysing non-textual data, such as charts, tables, and equations found in PDFs or technical slides. Standard systems often fail here, either by using simple OCR that destroys layout semantics or by ignoring visual data entirely [67]. Advanced Deep Research architectures employ layout-aware parsing to preserve the structural integrity of documents, enabling the system to reason across modalities, an example of which includes correlating a trend line in a graph with a textual description in a different section of the document. This "hybrid retrieval" capability enables agents to ground their reasoning in visual evidence, significantly enhancing the depth and accuracy of the generated reports.

Context Management: The primary challenge in scaling Deep Research systems is the "context bottleneck." As agents explore longer trajectories and accumulate vast amounts of web content, they risk filling the LLM's context window with noisy or redundant information, leading to degraded performance and increased hallucinations [68]. If the context becomes too cluttered, agents may lose track of their original objective or fail to utilise the correct answer even when it has been retrieved.

To mitigate this, efficient architectures employ functional decomposition, allocating responsibilities to separate agents. Frameworks like SLIM (Simple Lightweight Information Management) separate the search and browse functions to minimise token usage and employ periodic summarisation to compress the conversation history [68]. By actively pruning irrelevant data and maintaining a concise memory of the research trajectory, these systems can sustain long-horizon inquiries without exceeding computational budgets or succumbing to ”lost-in-the-middle” phenomena. Ultimately, the success of Deep Research depends on this delicate balance: aggressively hunting for diverse information while ruthlessly curating the context to maintain logical coherence [66].

2.4.4 Advanced Reasoning and Planning Strategies

To resolve complex, long-horizon tasks, Deep Research systems employ advanced reasoning and planning strategies that go beyond simple retrieval. These systems operationalise a type of thinking characterised by deliberate, sequential, and logical processing in order to manage the uncertainty inherent in open-ended inquiry [1]. Unlike standard RAG, which often reacts to a query with a single retrieval step, Deep Research agents utilise dynamic workflows where the system autonomously determines the sequence of actions, modifies its plans based on intermediate observations, and verifies its own outputs.

Dynamic Planning and Decomposition: A critical differentiator in Deep Research architectures is the implementation of explicit planning modules. While static workflows rely on predefined sequences (e.g., “search then summarise”), dynamic workflows empower the LRM to reconfigure tasks in real-time. Frameworks like DeepPlanner demonstrate that enforcing an explicit planning stage where the model outputs a structured plan before execution significantly reduces the entropy of subsequent action tokens, thereby stabilising long-horizon behaviour and preventing goal drift [69].

Planning strategies generally fall into three categories: planning-only, where the agent generates a schedule immediately; intent-to-planning, where the agent first asks clarifying questions to disambiguate user intent (e.g., OpenAI Deep Research); and unified intent-planning, where the agent proposes a plan and iteratively refines it with the user (e.g., Gemini DR) [1]. In complex multi-document scenarios, systems like Doc-Researcher utilise a specialised Planner agent to filter relevant documents and adaptively select retrieval granularity (e.g., deciding whether to retrieve a full document, a summary, or a specific chunk) based on the query’s complexity [67].

Interleaved Reasoning and Tool Use: Deep Research systems integrate reasoning directly with tool execution, rather than treating them as separate phases. This is exemplified by the WebThinker framework, which employs an “Autonomous Think-Search-and-Draft” strategy [12]. Here, the model seamlessly interleaves reasoning steps (generating logical chains) with information-gathering actions (invoking a Deep Web Explorer) and content synthesis (drafting report sections). Formally, this process models the generation of a solution as a joint probability distribution over a reasoning chain R and a final output y, conditioned on the instruction I, query q, and a set of available tools T. The generation process can be expressed as:

$$P(R, y \mid I, q, T) = \prod_{t=1}^{T_r} P(R_t \mid R_{<t}, I, q, \{O_\tau\}_{\tau<t}) \cdot \prod_{t=1}^{T_y} P(y_t \mid y_{<t}, R, I, q) \tag{2.1}$$

where T_r is the length of the reasoning sequence, R_t is the token at step t, and $\{O_\tau\}_{\tau<t}$ represents the outputs from tool calls (e.g., search results) executed prior to step t [12]. This equation highlights how the system's reasoning is dynamically conditioned on the evolving context of external information, allowing the agent to "change its mind" and pivot its search strategy based on what it discovers.

Atomic Decomposition and Verification: To improve the granularity of reasoning, frameworks like Atom-Searcher decompose the high-level <think> process into finer-grained functional units called "Atomic Thoughts" (e.g., <Hypothesis_Testing>, <Risk_Analysis>, <Plan>) [70]. This decomposition allows for more precise reinforcement learning supervision, where a Reasoning Reward Model (RRM) provides feedback on individual thought steps rather than just the final outcome, mitigating the problem of sparse rewards in long trajectories.

Furthermore, robustness is achieved through rigorous self-correction and verification loops. PokeeResearch introduces a distinct "verification mode" where the agent reviews its entire research thread to detect logical errors or insufficient evidence before finalising an answer [71]. If the verification fails, the system reverts to research mode to gather additional data. Similarly, systems like SLIM (Simple Lightweight Information Management) employ periodic summarisation to compress the trajectory history, allowing the agent to maintain reasoning coherence over extended horizons without exceeding context windows [68].

Optimisation via Reinforcement Learning: Optimising these reasoning strategies requires moving beyond Supervised Fine-Tuning (SFT) to Reinforcement Learning (RL). Techniques such as Group Relative Policy Optimisation (GRPO) and Reinforcement Learning Leave-One-Out (RLOO) are utilised to train agents to balance exploration and exploitation [1, 71]. For instance, DeepPlanner employs entropy-based advantage shaping, which amplifies learning signals for high-uncertainty planning tokens, effectively encouraging the model to explore more robust planning strategies without suffering from entropy collapse [69]. These optimisation methods ensure that the agent not only learns *how* to use tools but also *when* to plan, reason, and verify, ultimately emulating the cognitive workflow of a human researcher.

2.4.5 Human-AI Collaboration and Interactive Oversight

The deployment of Deep Research systems necessitates a fundamental shift from the traditional "Input-Wait-Output" paradigm where users passively await results from a black box to a model of Cognitive Oversight [72]. In this collaborative framework, interaction is treated not merely as an interface for instruction, but as an intrinsic dimension of intelligence that significantly enhances system performance and reliability.

Effective Deep Research requires "human-in-the-loop" architectures that allow researchers to guide the AI's reasoning process rather than just consuming its final output [20]. This partnership is characterised by dynamic autonomy, where users strategically delegate mechanical tasks (such as broad information retrieval) to the agent while retaining control over high-order decision-making and hypothesis refinement. Systems designed with interruptibility and transparency enable users to pause ongoing research, inspect intermediate reasoning chains, and inject domain expertise or corrective feedback at critical junctures [20]. This real-time intervention is crucial for preventing error cascades, where initial misunderstandings of intent lead to compounded inaccuracies over long research trajectories.

Furthermore, interactive oversight supports collaborative sense-making, allowing the system to clarify ambiguous queries and adapt to the user's evolving information needs [11, 72]. By maintaining a shared cognitive context, the agent can learn from user behaviors (such as preferred sources or query refinements) to align its future actions with specific research goals. This synergy ensures accountability; the human provides the necessary verification and ethical guardrails (e.g., ensuring peer-reviewed sourcing), while the AI provides the scale and speed of information synthesis [20]. Ultimately, this collaborative approach transcends simple automation, functioning instead as a form of augmented intelligence that leverages the complementary strengths of human judgment and machine execution.

In major Deep Research products, HAIC is most commonly operationalised through a pre-execution clarification stage, where the system proactively engages the user to disambiguate intent before commencing the autonomous research. OpenAI's Deep Research and Google's Gemini Deep Research utilise this phase to formulate targeted questions that refine vague objectives and establish specific research requirements. Allowing the human to refine the research objectives prior to execution translates similarly to a research librarian asking "What exactly are you looking for?".

2.4.6 Enterprise Deep Research

The application of Deep Research in enterprise contexts represents a distinct evolution from general-purpose web search, focusing on the synthesis of proprietary internal assets with external market intelligence. While consumer-facing models prioritise broad information retrieval, Enterprise Deep Research systems are designed to resolve high-stakes, strategic objectives–such as financial due diligence, competitor landscape mapping, and productivity diagnostics–where accuracy, provenance, and data security are non-negotiable [73].

Operationally, these systems distinguish themselves through deep integration with internal infrastructure. Enterprise systems utilise "enterprise connectors" and MCP tools to access heterogeneous data sources including SQL databases, internal document repositories, and communication logs [73]. This capability allows systems like FinSight and Salesforce's EDR to contextualise external market trends against internal performance metrics, automating labour-intensive workflows to produce professional-grade reports complete with data visualisations and citations [73, 74]. The most advanced deep research systems are empowered with both web search capabilities for external intelligence gathering and MCP tools for secure internal data access, allowing them to flexibly deploy either or both mechanisms depending on the use case. Specifically, whether the research task requires proprietary insights, public information, or a synthesis of both.

However, integrating autonomous agents into an enterprise environment introduces unique challenges regarding governance and trust. The black box nature of standard LLMs is unsuitable for business decisions. As a result, enterprise architectures prioritise steerability and auditability [73]. Employees must be able to inspect the agent's chain of thought, intervene to correct misconceptions, and dynamically adjust research trajectories without restarting the entire process,.

Furthermore, these systems must navigate the contextual gap between structured internal data and unstructured external knowledge. Agents must rigorously verify inputs to prevent hallucination cascades in financial reporting, where a single error can compromise an entire investment thesis [74]. Ultimately, the success of Enterprise Deep Research relies on transforming passive information retrieval into an active, collaborative intelligence layer that adheres to strict organisational standards for veracity and compliance.

2.4.7 Safety, Alignment and Ethical Challenges

The transition from passive LMs to autonomous Deep Research agents introduces systemic vulnerabilities encompassing standard safety alignments. While individual models may reject harmful queries, Deep Research agents can be manipulated through Intent Hijack and Plan Injection techniques where malicious objectives are disguised as academic inquiries or inserted into the agent's sub-tasks to bypass safety filters [75]. This capability allows agents to aggregate fragmented public data into coherent, actionable instructions for high-stakes harms, such as biosecurity threats or chemical weapon manufacturing, effectively eroding the safeguards inherent in the base model. Consequently, DRAs have been shown to produce more coherent, professional, and dangerous content, compared with standalone LLMs [75].

Beyond malicious misuse, the integrity of autonomous research faces significant ethical hurdles regarding information quality and provenance. Agents risk propagating unvetted inputs by treating non-peer-reviewed sources like blogs as authoritative, leading to hallucination cascades where initial errors are compounded across long reasoning trajectories [20]. These systems also struggle with temporal staleness, which refers to relying on outdated training data, and access bias, where the inability to navigate paywalls skews findings toward open-access content rather than the most rigorous scientific literature [20].

Ethical deployment also necessitates strict governance regarding privacy and intellectual property. Agents processing sensitive data through third-party APIs risk regulatory breaches, while the automated synthesis of copyrighted material raises complex attribution challenges [20]. Consequently, ensuring safety requires moving beyond static refusal benchmarks to dynamic metrics like DeepREJECT, which evaluate whether an agent has indirectly fulfilled a harmful intent by providing actionable knowledge [75]. Ultimately, trustworthy Deep Research demands a framework of verifiability, where automated synthesis is constrained by immutable audit trails and human-in-the-loop oversight to prevent the unchecked propagation of misinformation or harm [20].

2.5 Benchmarking & Evaluation Frameworks

Building on the architectural foundations of Deep Research, this section explores the necessary shift from static, closed-ended assessments to dynamic evaluation frameworks designed to measure autonomous, open-ended inquiry. It details methodologies for constructing high-fidelity benchmarks and introduces rigorous metrics, such as citation integrity and reasoning trace analysis, to verify the reliability and groundedness of these complex systems.

2.5.1 Background

The development of AI benchmarks has largely mirrored the progress of the models themselves. As systems progressed from specific task proficiency to general language understanding, evaluation methods shifted from narrow datasets into comprehensive suites designed to stress-test generalisation and robustness. This progression reflects a cycle where models rapidly saturate existing benchmarks, necessitating the creation of increasingly rigorous standards to measure the frontiers of AI capabilities.

In 2018, the General Language Understanding Evaluation (GLUE) benchmark was introduced to foster models that share general linguistic knowledge across diverse tasks, moving away from task-specific architectures [76]. However, rapid advancements in pretraining eroded GLUE's headroom within a year, prompting the 2019 release of SuperGLUE [77]. SuperGLUE retained the general-purpose motivation of its predecessor but incorporated more difficult formats, such as coreference resolution and complex question answering, to challenge BERT-era models. By 2020, the focus shifted toward assessing world knowledge and problem-solving breadth with Massive Multitask Language Understanding (MMLU) [78]. Covering 57 subjects ranging from elementary mathematics to professional law, MMLU aimed to measure knowledge acquired during pretraining across STEM and the humanities, addressing the disconnect between linguistic competence and actual subject mastery.

As scaling laws enabled qualitatively new model behaviors, the Beyond the Imitation Game benchmark (BIG-bench) was released in 2022 [79]. BIG-bench aggregated over 200 diverse tasks contributed by the global research community, specifically focusing on capabilities believed to be beyond contemporary models, such as multi-step reasoning, chess, and social bias detection. Most recently, Humanity's Last Exam (HLE) (2025) was developed in response to state-of-the-art models achieving over 90% accuracy on benchmarks like MMLU [80]. HLE serves as a final frontier for closed-ended academic evaluation, consisting of expert-level questions designed to be resistant to simple internet retrieval.

While these benchmarks have successfully pushed the boundaries of static knowledge assessment, they remain largely closed-ended. The emergence of Deep Research agents requires a paradigm shift beyond these traditional exam-style evaluations toward dynamic frameworks that measure information retrieval, synthesis, and long-horizon reasoning.

2.5.2 Benchmark Construction and Core Principles

The construction of benchmarks for Deep Research necessitates a fundamental departure from traditional question-answering datasets. While conventional benchmarks like Natural Questions or HotpotQA focus on "known unknowns" involving simple queries solvable via pattern matching or limited multi-hop reasoning over static data, benchmarks must capture "unknown unknowns" involving complex, open-ended information needs [81]. Effective Deep Research benchmarks are designed around principles of high search and reasoning intensity, ensuring that systems must process large volumes of information (search intensity) and perform non-trivial synthesis and planning (reasoning intensity) to succeed [82].

A primary methodological challenge in benchmark construction is sourcing realistic, high-complexity tasks. One approach employed by the Researchy dataset involves mining search engine logs to identify queries characterised by long user sessions and diverse click-throughs, which serve as proxies for genuine human effort [81]. Alternatively, researchers employ problem inversion, a technique where complex documents (e.g., flight investigation reports or scientific papers) are reverse-engineered into queries that require locating and synthesising that specific information [82]. This ensures that the ground truth exists but is difficult to derive without rigorous retrieval. To ensure validity, tasks are often curated by domain experts to be user-centric and unambiguous, providing specific constraints regarding audience, scope, and output format to minimise interpretation variance [83].

The dynamic nature of the internet presents a dilemna in benchmark design: realism versus reproducibility. Some frameworks, such as LiveResearchBench and LiveDRBench, prioritise time-varying or live tasks to prevent data contamination and test the agent's ability to retrieve up-to-date information beyond its parametric memory [82, 83]. Conversely, to ensure fair, reproducible comparisons and isolate retrieval performance from web volatility, other benchmarks like Deep Research Bench and BrowseComp-Plus utilise RetroSearch environments or fixed, human-verified corpora [84, 85]. These controlled environments often include hard negative documents to rigorously test an agent's ability to discern relevant evidence from distractions.

Finally, the construction of the ground truth for open-ended research requires granular structures rather than simple text references. Effective benchmarks decompose answers into intermediate representations, such as hierarchical lists of claims and sub-claims, allowing for precision and recall metrics that assess the validity of the reasoning chain rather than just the final output [82]. Similarly, checklist-based methodologies treat evaluation as a series of unit tests (e.g., ”Does the report cite the specific market growth rate?”), enabling automated judges to assess comprehensiveness and coverage consistently [83]. By adhering to these principles, Deep Research benchmarks aim to simulate the cognitive load of a human researcher, moving beyond mere fact retrieval to evaluate synthesis, planning, and groundedness.

2.5.3 Evaluation Frameworks and Metrics

Since Deep Research outputs are typically comprehensive reports containing multiple claims, citations, and structural elements, evaluation frameworks must be designed to assess both the quality of the generated content and the reliability of the retrieval process. These frameworks generally employ a combination of LLM-as-a-Judge protocols, rigorous human verification, and granular component analyses.

A primary challenge in Deep Research evaluation is assessing the quality of long-form reports where no single definitive or canonical answer exists. To address this, the RACE (Reference-based Adaptive Criteria-driven Evaluation) framework employs an LLM-as-a-Judge approach to dynamically generate task-specific criteria across four dimensions: Comprehensiveness, ensuring coverage of key areas; Insight/Depth, assessing the originality and logic of analysis; Instruction-Following, checking adherence to constraints; and Readability [15]. Similarly, the DeepEval suite introduces a multi-faceted approach, distinguishing between report-level metrics (e.g., presentation and logical consistency) and content-level metrics (e.g., analysis depth and coverage) [83]. DeepEval employs unit test checklists derived from user queries to objectively measure coverage, ensuring that specific information needs are met regardless of the report's narrative structure. To improve reliability, some frameworks utilise an ensemble of judges (e.g., GPT-5 and Gemini 2.5 Pro) to align automated scores with human expert judgment [83].

Given that a core value proposition of Deep Research systems is trustworthiness, evaluating grounding and citation accuracy is critical. The FACT (Factual Abundance and Citation Trustworthiness) framework automates this by extracting Statement-URL pairs from generated reports and verifying whether the retrieved web content supports the claim [15]. This process yields two key metrics: Citation Accuracy, the proportion of citations that accurately support their associated statements, and Effective Citations, the total count of supported claims, which serves as a proxy for the report's information density. Likewise, DeepEval distinguishes between Citation Association, whether claims are linked to sources, and Citation Accuracy, whether the source actually supports the claim, using a rubric tree to categorise errors such as invalid links, irrelevant content, or unsupported claims [83]. These metrics help distinguish between agents that act as reliable researchers and those that merely hallucinate plausible-sounding but baseless narratives.

The quality of the sources themselves serves as a major evaluation criterion. Research agents are responsible for distinguishing between authoritative academic and less credible sources like social media threads. Evaluation sheets often include metrics like Information Validity and Correctness of Sources as core pillars, assessing whether agents prioritise recent, peer-reviewed, or primary sources over outdated or derivative content [86]. By combining these dimensions: citation integrity, factual synthesis, and source reliability, evaluation frameworks aim to create a holistic view of an agent's trustworthiness, ensuring that Deep Research systems serve as effective extensions of human inquiry rather than engines of misinformation.

Beyond the final output, researchers are increasingly evaluating the research process itself. The Deep Research Comparator platform facilitates fine-grained human annotation not just of the final report, but of the intermediate steps taken by the agent [87]. This allows for process-based metrics, such as the step upvote rate, which measures the quality of the agent's planning and execution actions independent of the final deliverable. Additionally, benchmarks like LiveDRBench formalise the DR task as a hierarchical list of claims and sub-claims [82]. This structure allows for the calculation of modified Precision and Recall scores, where a top-level claim receives a score of zero if its supporting sub-claims (evidence) are incorrect, thereby enforcing strict logical grounding.

Finally, to disentangle the reasoning capabilities of the LLM from the performance of the search engine, frameworks like BrowseComp-Plus utilise fixed, human-verified corpora rather than live web access. This enables the use of standard Information Retrieval metrics, such as `Recall@k` and `nDCG@k`, to assess the effectiveness of the retrieval component in isolation [85]. This isolation helps determine whether performance bottlenecks stem from the agent's inability to reason over data or its failure to retrieve relevant documents in the first place. Collectively, these metrics spanning report quality, citation integrity, process efficiency, and retrieval accuracy, provide a holistic view of a deep research agent's utility and reliability.

2.5.4 Enterprise Deep Research Benchmarks

Evaluating Deep Research systems within enterprise contexts requires benchmarks that simulate the complexity, heterogeneity, and distinct workflows of real enterprise environments. Unlike general web-based inquiries, enterprise research demands that agents navigate a mix of structured and unstructured proprietary data while adhering to specific business logic and constraints.

Recent frameworks have emerged to address the challenges of this type of retrieval-centric task that necessitates source-aware, multi-hop reasoning across diverse internal artefacts. For instance, the HERB benchmark simulates realistic software lifecycle workflows including planning, development, and support to create interconnected data pools comprising Slack messages, meeting transcripts, and code repositories [88]. This approach moves beyond simple document retrieval, testing an agent's ability to trace information flow across different communication channels and temporal stages. HERB also incorporates unanswerable queries to test system reliability, ensuring agents can identify when necessary evidence is absent rather than hallucinating responses.

At the document level, benchmarks such as RealKIE focus on the specific challenges of Key Information Extraction from visually rich and lengthy enterprise records, such as SEC filings, NDAs, and invoices [89]. These datasets emphasise the difficulty of handling sparse annotations and complex layouts, such as nested tables, which are characteristic of tasks like financial audits and contract due diligence.

Moreover, the DRBench framework expands the scope of evaluation by requiring agents to integrate private internal knowledge with public web information to solve open-ended strategic questions [90]. By grounding tasks in specific user personas like a Compliance Manager and domains like cybersecurity and sales, DRBench evaluates an agent's ability to synthesise insights within a simulated environment. Together, these benchmarks prioritise Insight Recall and Distractor Avoidance, measuring an agent's capacity to surface decision-critical facts while filtering out the inherent noise.

2.6 Conclusion

This chapter has traced the evolution of Deep Research from its roots in RAG to its realisation through agentic workflows and advanced orchestration. The literature illuminates that while LLMs provide the core reasoning capabilities, it is the architectural scaffolding—specifically the integration of dynamic planning, tool use, and layered verification—that enables the paradigm shift from simple question-answering to comprehensive, autonomous research.

Despite the rapid proliferation of these systems, significant gaps remain in the current body of knowledge. The field is largely dominated by proprietary black box solutions, leaving the academic and open-source communities with a fragmented understanding of optimal design patterns. Critical limitations persist, particularly regarding the trade-off between agent autonomy and reliability, the management of context bottlenecks in long-horizon tasks, and the mitigation of hallucination cascades. Furthermore, the literature highlights that traditional static benchmarks are insufficient for measuring the quality of open-ended research, necessitating the adoption of the dynamic, process-oriented evaluation frameworks.

Positioned within this context, this thesis investigates the architectural principles required to build reliable, efficent, and scalable Deep Research systems. The findings from this review directly inform the design choices detailed in the following chapter.

CHAPTER 3

Methodology

This chapter delineates the methodological framework employed to investigate architectural enhancements to Deep Research agents in three segments. First, Section 3.1 introduces the LangChain Open Deep Research system, establishing it as the baseline architecture for this investigation. A critical architectural analysis in Section 3.2 subsequently identifies specific limitations that constrain both research reliability and operational efficiency, providing the empirical foundation for the proposed modifications.

Building upon these identified constraints, Section 3.3 presents a novel Deep Research architecture incorporating three theoretically-grounded enhancements. The Research Reviewer component implements iterative quality assessment mechanisms, whilst the Research Verifier introduces fact-checking capabilities. The Adaptive Model Selection component optimises computational efficiency by dynamically routing tasks to the most appropriate model based on complexity. Each modification addresses specific architectural deficiencies identified in the baseline system.

The experimental methodology employs a comparative design detailed in Section 3.10, wherein each architectural enhancement is systematically evaluated against the baseline configuration established in Section 3.7. Section 3.8 establishes the DeepResearch Bench benchmark, providing standardised metrics for assessing research quality and reliability. The presence or absence of each architectural component serves as the independent variable, enabling quantitative assessment of individual contributions to overall system performance. To conclude, the expected outcomes are detailed in Section 3.11.

3.1 Langchain Open Deep Research

This section presents a comprehensive analysis of the Open Deep Research system developed by Langchain [17], a platform for building and deploying AI agents. This system serves as the foundational architecture for this thesis: it will be evaluated using the DeepResearch Bench benchmark introduced in Section 3.8, and subsequently modified through the architectural enhancements proposed in Section 3.3.

The selection of this particular system is predicated on several practical and technical considerations. As an open-source repository released under the MIT License, it provides both the transparency and extensibility necessary for architectural modification and empirical analysis. Furthermore, Langchain's established platform infrastructure facilitates agent deployment and experimentation. Developing a novel deep research system from first principles would have extended beyond the scope of this thesis; instead, systematic enhancement of an existing, advanced architecture enables focused investigation of specific architectural components.

Released in July 2025, Open Deep Research represents a state-of-the-art implementation of a Deep Research system that combines multi-agent coordination, iterative information gathering, and sophisticated report synthesis. Understanding the structure, capabilities, and operational mechanisms of this baseline system is therefore essential for contextualising both the evaluation results and the rationale for the proposed architectural modifications.

The Open Deep Research System workflow comprises three sequential phases: Scope, Research, and Write [13]. The system begins by analysing user queries to determine whether additional clarification is needed before proceeding. Once the scope is clear, it generates a structured research brief that guides subsequent investigation. The agent then employs a hierarchical coordination approach where a supervisor agent delegates focused research tasks to multiple parallel sub-researchers, each investigating specific aspects of the broader question. All accumulated findings are then synthesised into a cohesive final report. This architecture is illustrated in Figure 3.1.

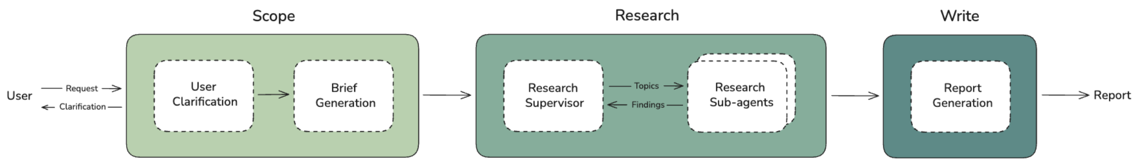


FIGURE 3.1: The Langchain Open Deep Research architecture.

Scope Phase

The scope definition phase establishes clear research parameters before investigation begins. The system first analyses incoming user messages using structured output generation to determine whether the research scope requires clarification. The analysis considers factors such as ambiguous terminology, undefined acronyms, or insufficient contextual information. If clarification is needed, the system requests additional information from the user through targeted questions. Once sufficient information is gathered, the system verifies its understanding and proceeds to the next stage. This clarification step can be configured to be optional based on deployment requirements. As such, it is is disabled for the evaluation within this thesis as sufficiently articulated tasks are provided.

Following clarification, the system transforms the conversational user messages into a formal, structured research brief. This brief serves as the primary directive for the research phase and includes detailed specifications about what information should be gathered, which sources should be prioritised, and any constraints or preferences expressed by the user. The research brief initialisation also establishes the appropriate system prompts and operational constraints for the supervisor agent that will coordinate the investigation.

Research Phase

The research execution phase employs a two-level hierarchical structure consisting of a supervisor agent and multiple subordinate researcher agents. This hierarchical approach enables sophisticated task decomposition and parallel information gathering while maintaining coherent overall coordination.

The supervisor agent operates at the strategic level, analysing the research brief and determining how to decompose the investigation into manageable subtasks/topics. It makes decisions about whether research can be parallelised across multiple independent dimensions or whether a single focused investigation is more appropriate. The supervisor has access to several capabilities: it can delegate research tasks to subordinate agents, engage in strategic reflection to assess progress, and signal when sufficient information has been gathered. The supervisor's operation is bounded by configurable limits on the number of research iterations it can perform, preventing excessive resource consumption while ensuring thorough investigation.

When executing tool calls, the supervisor manages parallel research delegation up to a configurable maximum number of concurrent research units, processes its own strategic reflections, and aggregates the findings returned from completed research tasks. The system includes robust error handling for cases where resource constraints, such as token limits, are exceeded during coordination.

Individual researcher agents conduct focused investigation on specific topics assigned by the supervisor. Each researcher is given a specific topic to investigate, operates independently, and has access to multiple information-gathering tools. These include web search capabilities, which is implemented through a third-party search API by Tavily for this thesis. Researchers can also access external integrations via MCP, enabling interaction with specialised data sources or domain-specific tools, though these are not used in this thesis. Additionally, researchers can engage in strategic reflection between searches to assess their findings and plan subsequent information-gathering steps.

Using the concrete example from the Open Deep Research paper [13], the question *"Compare the approaches of OpenAI vs Anthropic vs Google DeepMind to AI safety. I want to understand their different philosophical frameworks, research priorities, and how they're thinking about the alignment problem."* may lead the supervisor to spin up three agents to each explore OpenAI, Anthropic, and Google DeepMind respectively, as depicted in Figure 3.2.

The research execution for each subordinate agent involves parallel tool execution, where multiple searches or tool invocations can be performed simultaneously. The system monitors progress and determines whether to continue the research loop or proceed to the compression stage based on configurable limits on the number of tool-calling iterations. This prevents individual researchers from consuming excessive resources while ensuring they gather sufficient information.

Once a researcher completes its investigation, its findings undergo a compression and synthesis process. This process takes the accumulated research messages, including tool outputs and intermediate reasoning, and distills them into a concise, structured summary. The compression maintains all critical information and preserves source citations while removing redundancy and improving readability. This compression

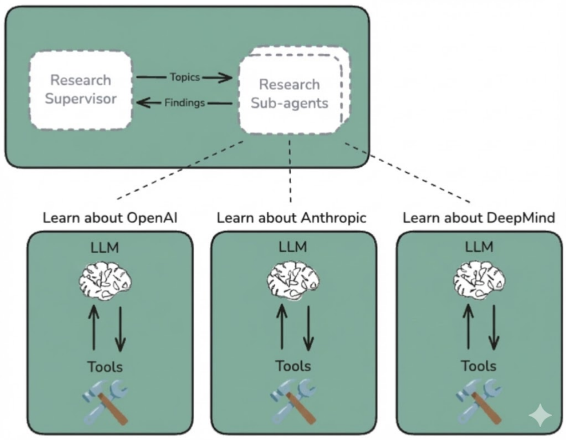


FIGURE 3.2: The multi-agent approach of researcher agents [13].

step implements retry logic with progressive message truncation to handle cases where the volume of accumulated research exceeds model token limits, ensuring that findings can be successfully summarised even in resource-constrained scenarios.

Following the compression and synthesis of each researcher's findings, the results are returned to the supervisor, which then evaluates the collective progress and determines the subsequent course of action. The supervisor receives the compressed research summaries as tool outputs and can assess whether the accumulated findings adequately address the original research brief. At this decision point, the supervisor has three options: it can delegate additional research tasks by invoking further researcher agents to investigate unexplored dimensions or gather supplementary information; it can engage in strategic reflection to evaluate the current state of knowledge and identify remaining gaps; or it can signal completion of the research phase when satisfied that sufficient information has been gathered. This iterative evaluation continues until one of three termination conditions is met: the supervisor explicitly signals research completion, the maximum number of research iterations is reached, or the supervisor determines no further tool invocations are necessary.

Write Phase

The final write phase generates the comprehensive research report by combining the research brief, original user context, and all accumulated findings from the parallel research efforts. This stage employs a dedicated model optimised for report generation that analyses the complete information set and produces a well-structured, coherent document addressing the original research question.

The synthesis process implements a progressive truncation strategy with retry logic to handle token limit constraints. If the complete set of findings exceeds the model's context window, the system progressively reduces the input size while attempting to preserve the most critical information. This adaptive approach maximises report quality while operating within computational constraints, ensuring that reports can be generated even when research has accumulated extensive information.

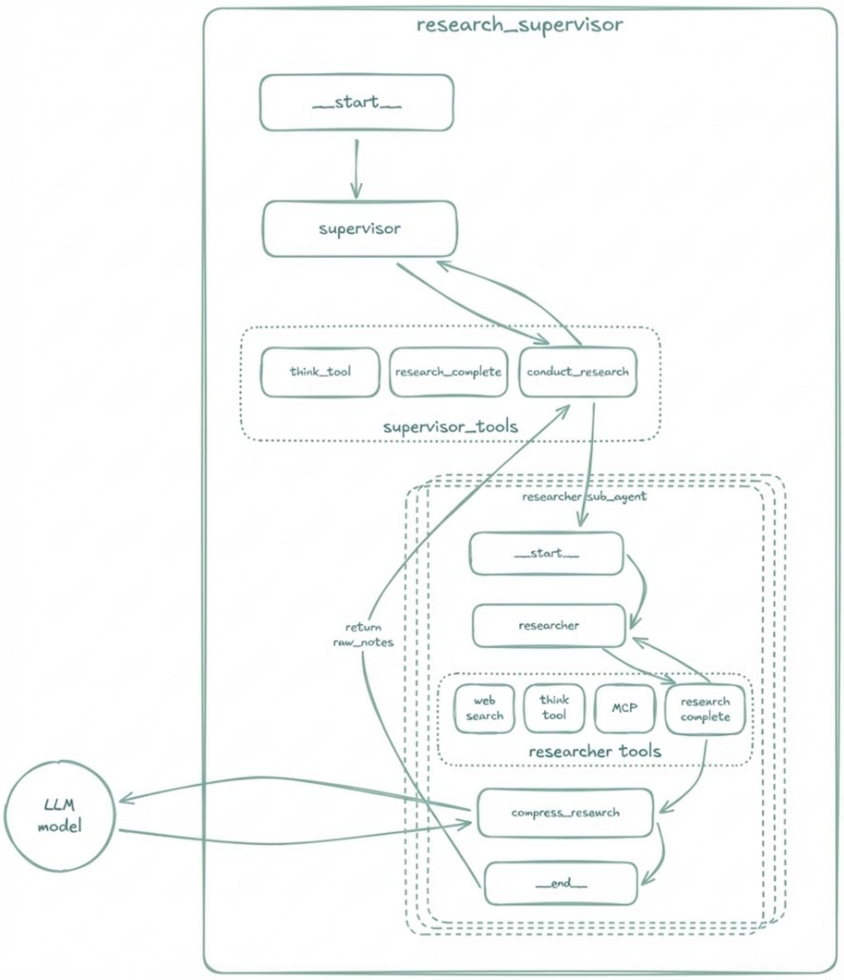


FIGURE 3.3: The Research Phase overview [14].

Model Specialisation Strategy

The architecture employs four specialised language models, each optimised for distinct tasks within the research workflow. The research model powers both the supervisor and individual researcher agents, handling complex reasoning tasks such as strategic planning, task decomposition, and iterative information gathering. The summarisation model focuses specifically on condensing webpage content retrieved from search results, enabling efficient information extraction from lengthy web documents. The compression model specialises in synthesising and structuring the findings accumulated by individual researchers, producing concise summaries that preserve critical information while improving coherence. Finally, the final report model handles the integration of all research findings into a comprehensive, well-structured document that addresses the user's original query. This specialisation approach allows each component to be optimised for its specific function, enabling the use of different model architectures, sizes, and configurations based on the requirements of each task. The system prompts of each model are presented in Appendix C.1.

3.2 Open Deep Research Architectural Analysis

This section conducts a systematic analysis of the architecture to identify the critical limitations that constrain research quality, citation accuracy, and cost-efficiency. The analysis focuses on three fundamental architectural constraints: the absence of quality validation mechanisms in the research workflow, the absence of verification following the report generation, and the inflexibility of static model assignment. Each of these limitations presents a specific opportunity for targeted architectural enhancement, which will be addressed through the proposed modifications detailed in Section 3.3.

3.2.1 Absence of Research Quality Validation

The first limitation of the current architecture regarding research quality lies in its handling of research outputs. Every research result is immediately wrapped in a ToolMessage and added to the supervisor's context without validation, quality checking, or the possibility of rejection. This direct pass-through mechanism prevents any form of quality assurance before research findings are incorporated into the supervisor's decision-making process. The supervisor agent's limited toolset further constrains its ability to perform quality control. Specifically, it has access to only three tools: `ConductResearch` for delegating new research tasks, `ResearchComplete` for signalling completion, and `think_tool` for strategic thinking. Notably absent is any mechanism to reject, revise, or filter research outputs, effectively eliminating the possibility of iterative refinement or quality-based feedback loops within the research phase. This architectural constraint creates a rigid, unidirectional flow where low-quality or incomplete research findings cannot be identified and corrected before being integrated into the final report. However, the supervisor's ability to decide whether to continue or complete research serves as a limited form of quality validation.

3.2.2 Absence of Post-Generation Verification

Furthermore, research reliability is compromised by a limitation that manifests in the final stage of the research workflow, where the synthesised report is delivered directly to the end-user without any form of post-generation validation. The system immediately returns the generated report content upon successful model invocation, creating an unmediated pipeline from the report agent output to system output. The architecture provides no mechanism to cross-reference generated claims against source findings to verify citation accuracy. Consequently, factual inconsistencies or unsupported assertions propagate into the final deliverable without detection, undermining the reliability of the research output.

3.2.3 Static Model Assignment

Beyond the quality validation, a fundamental constraint lies in the system's approach to model selection. The current implementation employs a static model assignment paradigm wherein the research model is predetermined at configuration time and uniformly applied across all researcher agents, irrespective of the semantic complexity or cognitive demands of individual research sub-tasks. The `research_model` parameter is defined as a fixed string field with a singular default value, offering no mechanism for dynamic model selection based on task characteristics.

This architectural constraint presents a fundamental trade-off that directly impacts both cost and performance: selecting a high-capacity model (e.g., GPT-4o, Claude Sonnet) ensures robust performance across complex reasoning tasks but incurs unnecessary computational cost for straightforward queries, whereas selecting a lightweight model (e.g., GPT-4o-mini) optimises for cost-efficiency but may yield suboptimal results when confronted with nuanced analytical requirements. Consequently, this inflexibility forces a compromise where either quality or cost-efficiency must be sacrificed, rather than enabling adaptive optimisation based on the specific demands of each research subtask.

Summary

Collectively, these two architectural limitations—the absence of quality validation and the inflexibility of static model assignment—represent fundamental constraints that limit both the quality and cost-efficiency of the baseline Open Deep Research system. Addressing these limitations requires targeted architectural modifications that introduce: (1) verification and correction mechanisms to ensure research quality, (2) enhanced agent capabilities to enable iterative refinement, and (3) adaptive model selection strategies to optimise the cost-performance trade-off. The following section 3.3 presents a proposed architecture that systematically addresses each of these constraints through the integration of ...

3.3 Proposed Deep Research Architecture

To address the limitations identified in Section 3.2, this section introduces three architecture amendments: a Research Reviewer, a Research Verifier, and an Adaptive Model Selection. Each amendment is implemented upon the baseline architecture shown in Figure 3.4, which depicts the Langchain Open Deep Research architecture configured for DeepResearch Bench evaluation with user clarification disabled. Implementation details are provided in Sections 3.4, 3.5, and 3.6, all of which adhere to the system constraints outlined in Section 3.3.1.

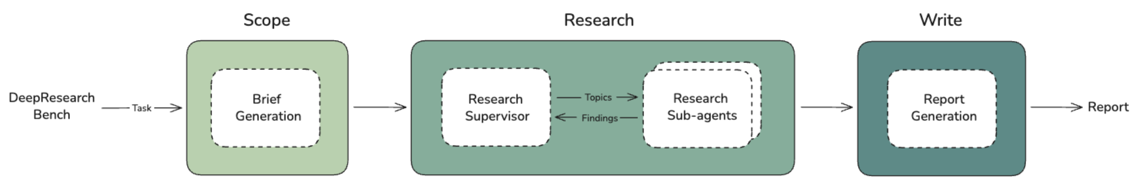


FIGURE 3.4: The Langchain Open Deep Research architecture configured for DeepResearch Bench evaluation.

3.3.1 System Constraints

The implementation of all three architectural enhancements within this thesis is deliberately scoped to function as augmentations rather than fundamental redesigns of the proven Langchain Open Deep Research framework. The existing architecture embodies established design principles that have been empirically validated through prior deployment. Consequently, the implementations must preserve the core research workflow while introducing their respective mechanisms at minimal intervention points.

This constraint precludes more ambitious approaches such as training models and reinforcement learning. Specifically, the implementation relies exclusively on proprietary pre-trained LMs accessed via commercial APIs, with all architectural enhancements achieved through prompt engineering, workflow orchestration, and introducing new nodes rather than fine-tuning or custom model development. This constraint-aware approach ensures practical applicability to real-world enterprise deployments where organisations typically lack the computational resources, specialised expertise, or proprietary datasets required for large-scale model training.

The detailed system constraints for each implementation are presented in Sections 3.4.3, 3.5.1, 3.6.3 respectively. These constraints emerge from both technical limitations of the existing system and deliberate scoping decisions to maintain architectural coherence while introducing systematic quality control and cost optimisation mechanisms.

3.3.2 Research Reviewer

The lack of quality validation mechanisms in the baseline architecture permits research outputs to propagate directly to the supervisor without assessment, eliminating opportunities for iterative refinement of inadequate or incomplete findings. The Research Reviewer introduces a quality assurance layer between individual researcher agents and the supervisor, implementing a verification-and-correction loop that systematically evaluates compressed research outputs across four dimensionsComprehensiveness, Insight, Instruction Following, and Factualitybefore permitting integration into the supervisor's context. This architectural enhancement aims to elevate the overall quality of research reports by ensuring that only outputs meeting explicit quality thresholds contribute to the final synthesis, thereby reducing the propagation of deficient research into the final deliverable.

3.3.3 Research Verifier

The baseline architecture delivers synthesised reports directly to end-users without post-generation validation, providing no mechanism to detect factual inconsistencies or unsupported assertions that may have been introduced during report synthesis. The Research Verifier introduces a verification stage following final report generation that systematically cross-references generated claims against the accumulated source findings to validate citation accuracy and remove statements lacking evidentiary support. This architectural enhancement aims to improve research reliability by detecting and correcting citation inaccuracies before the report reaches the end-user, thereby ensuring that all claims in the final deliverable are properly grounded in the source material.

3.3.4 Adaptive Model Selection

The static model assignment paradigm of the baseline architecture uniformly applies a single predetermined model to all researcher agents regardless of sub-task complexity, forcing a compromise between deploying computationally expensive high-capacity models for all tasks or accepting suboptimal performance on complex analytical requirements. Adaptive Model Selection introduces a routing-based mechanism wherein the supervisor assesses the complexity of each research sub-task at delegation time and dynamically assigns models from a tiered pool, ranging from a lightweight model for straightforward queries to frontier models for nuanced analytical tasks. This architectural enhancement aims to optimise the cost-performance trade-off by allocating computational resources proportionally to task requirements, thereby reducing aggregate system costs without sacrificing output quality on complex research problems.

3.4 Research Reviewer Implementation

3.4.1 The Quality Validation Paradox

The implementation of a reviewer-based quality assurance mechanism introduces a fundamental paradox: employing a language model to validate and refine the output of another language model inherently incurs additional computational costs and latency overhead. This paradoxical situation arises because the very act of quality validation requires inference time that could alternatively be allocated to deploying a more capable model for the initial generation task. Furthermore, the reviewer model itself is susceptible to introducing evaluation errors, inconsistencies in assessment criteria, or misidentification of valid outputs as inadequate, thereby potentially propagating rather than eliminating quality deficiencies.

The economic viability of this multi-stage architecture depends critically on whether the aggregate cost of research generation, quality evaluation, and iterative refinement remains below the cost of simply utilising a superior model for the initial generation. Formally, the reviewer-based approach is justified only when the following inequality holds:

$$\text{Cost(research)} + \text{Cost(review)} + \text{Cost(refinement)} < \text{Cost(superior\_model)} \tag{3.1}$$

This constraint reveals a delicate balance: if the combined overhead of the reviewer mechanism approaches or exceeds the cost differential between the research model and more capable alternatives, the architectural complexity fails to deliver meaningful economic advantage. The challenge is particularly acute given that refinement may require multiple iterations to achieve acceptable quality, with each iteration compounding the cumulative cost. Consequently, the practical implementation of a reviewer-based system necessitates careful selection of model pairings wherein the cost asymmetry between the research and review models is substantial, yet the reviewer retains sufficient sophistication to perform meaningful quality assessment without introducing excessive false positive or false negative evaluations.

3.4.2 The Correctability Assumption

A critical assumption underlying iterative refinement architectures is that quality deficiencies in generated outputs stem primarily from correctable factors such as insufficient instruction clarity, ambiguous task specifications, or suboptimal information synthesis, rather than from fundamental limitations in the model's reasoning capabilities. This assumption posits that a reviewer's refinement prompts can successfully guide the research model toward higher-quality outputs as per the thesis hypothesis. However, the validity of this assumption is contingent upon the research model possessing the latent capability to produce improved results when provided with more precise guidance. If a model generates inadequate research due to inherent reasoning capacity constraints rather than merely unclear instructions, no amount of iterative refinement can elevate the output quality beyond the model's intrinsic performance ceiling.

For research models with insufficient capabilities, the reviewer's feedback becomes essentially ineffectual, as the model lacks the cognitive architecture to implement the suggested improvements. Conversely, when deploying highly capable research models, the reviewer may generate false negative assessments, incorrectly flagging sophisticated outputs as inadequate simply because the reviewer itself lacks the analytical depth to recognise valid reasoning patterns or nuanced argumentation. This asymmetry introduces a substantial implementation challenge: the reviewer must be sufficiently capable to evaluate the research model's outputs without being so inferior as to misjudge quality, yet not so sophisticated that its deployment negates the economic advantages of the multi-stage architecture.

Empirical observations during preliminary testing revealed substantial performance disparities across model generations, with outputs from advanced frontier models demonstrating qualitatively superior reasoning and synthesis capabilities compared to earlier-generation systems. This observation underscored that refinement effectiveness would depend critically on whether identified deficiencies originated from correctable factors or from fundamental reasoning capacity limitations that lie beyond the scope of prompt-based refinement. Based on these considerations, the implementation employed GPT-4o Mini as the research model and Google Gemini 2.0 Flash as the reviewer model. This pairing was selected to optimise the cost-capability trade-off: Gemini 2.0 Flash provides specialised reasoning capabilities at a 33% cost reduction compared to GPT-4o Mini, while maintaining sufficient evaluative sophistication to meaningfully assess GPT-4o Mini's outputs. The research model, despite being designated as a "mini" variant, possesses robust general knowledge and versatile reasoning abilities that enable it to respond effectively to refinement guidance, thereby satisfying the Correctability Assumption within the economic constraints of the system.

3.4.3 System Constraints

The Research Reviewer must operate within these architectural and operational constraints:

(Constraint 1) **Fixed Three-Stage Sequential Pipeline**: The workflow follows a rigid Scope Research

Write workflow where each stage must complete before the next begins. The pipeline structure cannot be dynamically modified or reordered based on query complexity or type.

(Constraint 2) **Stateless Session-Based Execution**: Each research session starts from scratch without persistent memory of previous queries, findings, or user preferences. The system cannot build on prior research sessions or maintain long-term knowledge accumulation across interactions.

(Constraint 3) **Token Limit Handling via Progressive Truncation**: The system handles context window limits through reactive truncation rather than proactive chunking or retrieval-augmented generation. When token limits are exceeded, the system progressively removes older messages or truncates findings by 10% per retry (reducing to 90% of the previous limit each time) until the content fits, potentially losing earlier research context.

(Constraint 4) **Structured Output Dependency for Agent Communication**: Inter-agent communication relies on Pydantic-defined structured outputs parsed from LLM responses. This creates a tight coupling between prompt engineering and schema definitionsif the LLM fails to produce valid JSON conforming to the schema after a configurable amount of retry attempts, the entire node fails.

3.4.4 Design Decisions

Placement of the Research Reviewer Layer

The fundamental architectural question concerns the optimal placement of the quality validation checkpoint within the multi-agent hierarchy to enable refinement loops without disrupting the existing supervisor-to-sub-researcher delegation flow. Three implementation options were considered: (i) positioning the reviewer between researcher sub-agents to validate intermediate outputs, (ii) embedding the reviewer within the researcher subgraph after compression but before returning control to the supervisor, or (iii) implementing review as a separate post-processing stage after all sub-agents have completed their execution.

The design decision draws theoretical support from Wei et al.'s AlignRAG framework, which reconceptualises RAG not merely as a retrieval-based generation process, but as a "reasoning trajectory" susceptible to specific misalignment phases at three critical junctures: relevance assessment, query-evidence mapping, and evidence-integrated justification [91]. Their Critique-Driven Alignment (CDA) mechanism iteratively refines these trajectories by identifying breakdowns in inductive bias and logical consistency before the final answer is materialised. This insight directly informs the decision to embed the reviewer within the researcher subgraph, as this placement enables intervention during the "Research" stage of the system's fixed three-stage pipeline. By intercepting the reasoning trajectory after compression but before supervisor handoff, the reviewer can correct evidence mapping errors and logical inconsistencies that would otherwise propagate to the final output. This is a capability that a post-processing approach inherently lacks due to its separation from the reasoning process.

The implemented solution positions the reviewer as a node within the researcher subgraph, specifically after the compression operation and before the subgraph's end state. This placement achieves architectural modularity by encapsulating the entire quality-assurance loop within the researcher subgraph boundary, thereby maintaining the supervisor's abstraction layer. The supervisor remains agnostic to refinement iterations, receiving only validated outputs that satisfy the established quality criteria. This design preserves the clean contractual interface wherein the supervisor delegates a research topic and receives back a quality-assured synthesis, eliminating the need to restructure the supervisor's orchestration logic.

The reviewer is registered as a node within the `researcher_builder` StateGraph via `add_node("research_reviewer", research_reviewer)`, with an unconditional edge from `compress_research`. The `research_reviewer()` function returns a `Command` object that either terminates the subgraph (`goto=END`) to pass findings to the supervisor, or loops back to the researcher node (`goto="researcher"`) for refinement, all without modifying the parent `supervisor_subgraph`.

```
researcher_builder.add_node("compress_research", compress_research)
researcher_builder.add_node("research_reviewer", research_reviewer)
researcher_builder.add_edge("compress_research", "research_reviewer")
researcher_subgraph = researcher_builder.compile()
```

FIGURE 3.5: Research reviewer positioned as post-compression gate within the researcher subgraph.

Review Trigger Strategy

The second architectural decision addresses the selection of which researcher sub-agents findings should undergo quality review. Three review trigger strategies were evaluated: (i) triggering reviews conditionally when outputs fail a pre-screening heuristic, (ii) enforcing universal review of all compressed research outputs regardless of preliminary quality indicators, or (iii) probabilistically sampling a subset of outputs for review (e.g., 50% random selection).

Cattan et al.'s work on conflicting sources in search-augmented LLMs establishes a taxonomy of knowledge conflicts, including Freshness, Conflicting Opinions, and Misinformation, and demonstrates that different conflict types necessitate distinct model behaviors, such as prioritising recent dates for freshness conflicts versus neutralising opinions for perspective conflicts [92]. Critically, their findings validate that LMs can accurately predict these conflict types given a query and search results, suggesting the feasibility of selective review triggered by pre-screening heuristics. This insight would theoretically support implementing a lightweight classification step using an LLM to detect specific conflict types within sub-researcher findings, thereby selectively triggering the resource-intensive VCL loop only when necessary.

However, the implemented solution adopts universal review, wherein every compressed research output passes through the research reviewer unconditionally. This decision prioritises consistent quality assurance over computational efficiency, recognising that each sub-researcher processes a distinct topic delegated by the supervisor, and that inconsistent quality across these outputs would fundamentally undermine the final report's coherence. While the selective approach informed by Cattan et al. offers theoretical efficiency gains, it introduces variance based on heuristic accuracy and sampling decisions, creating the possibility of quality blind spots where problematic outputs evade review. The overhead associated with universal review is deemed acceptable given two mitigating factors: review occurs once per research unit rather than per tool call, and the reviewer employs a cost-optimised model (Google Gemini 2.0 Flash) to minimise per-review expenses. This design trades modest computational overhead for deterministic quality guarantees across all research outputs.

Universal review is enforced through the unconditional edge `add_edge("compress_research", "research_reviewer")` in the `researcher_builder` graph definition. No conditional routing or sampling logic exists; every invocation of `compress_research()` proceeds directly to `research_reviewer()`, ensuring all sub-researcher outputs undergo quality validation before returning to the supervisor.

```
# No conditional edges
researcher_builder.add_edge(START, "researcher")
researcher_builder.add_edge("compress_research", "research_reviewer")
```

FIGURE 3.6: Unconditional edge enforcing universal review of all compressed research outputs.

Quality Assessment Schema

The third design decision concerns the representation of research quality: should it be expressed as a single holistic measure or decomposed into multiple evaluable dimensions to enable both clear accept/reject decisions and actionable refinement guidance? Research quality can be represented as a single aggregate numerical score on a continuous scale (e.g., 0–10), a multi-dimensional binary PASS/FAIL evaluation across defined criteria, or multi-dimensional numerical scores with weighted aggregation to derive a final composite metric.

Two complementary frameworks inform this decision space. Zheng et al.'s CRAVE framework compels LMs to reason from conflicting perspectives (supporting versus refuting) across four specific evaluation dimensions: Direct Evidence Analysis, Semantic Features, Linguistic Patterns, and Logical Reasoning [93]. This structured decomposition prevents evaluative confusion arising from opposing viewpoints and ensures comprehensive evidence assessment. CRAVE's multi-dimensional approach directly supports binary PASS/FAIL evaluation, where dimensions such as Logical Reasoning can be realised as boolean criteria within the Pydantic schema, aligning with Constraint 4's requirement for tight coupling between prompt engineering and structured output definitions. Conversely, Asai et al.'s SELF-RAG framework suggests an alternative approach wherein the LLM-as-a-judge outputs numerical scores (0–10 or probabilities) for dimensions such as Relevance, Support, and Utility, with programmatic weight application to derive a final quality metric [94]. This weighted numerical approach would theoretically enable more nuanced acceptance thresholds than binary classification.

The implemented solution adopts a four-dimensional binary schema encompassing Relevance, Depth, Evidence, and Completeness, where all dimensions must pass for research acceptance. This decision follows CRAVE's multi-dimensional decomposition principle while rejecting SELF-RAG's numerical scoring approach based on empirical observations during initial prototyping. Preliminary experiments with 1–10 numerical scales (mapping 0–7 to fail and 8–10 to pass) revealed that LMs exhibit systematic leniency bias in numerical scoring, particularly when evaluation rubrics lack precision. The binary pass/fail representation creates unambiguous decision boundaries and enables targeted feedback: when refinement is triggered, failed dimensions directly map to specific deficiencies the researcher must address. A single aggregate score would obscure which quality aspects require improvement, while weighted numerical scores introduce threshold-tuning complexity and obscure the decision boundary. The strict AND-logic (all dimensions must pass) prioritises consistent baseline quality over permissive averaging that could mask critical deficiencies in individual dimensions. Operationally, the LLM evaluates the research output against each dimension and returns independent binary judgments accompanied by natural language feedback, while programmatic logic aggregates these judgments through strict conjunction and applies deterministic overrides (e.g., failing an output on factuality when inline citations are absent). The complete Research Reviewer system prompt and refinement system prompt are provided in Appendix C.2.

The `QualityAssessmentScores` Pydantic model in `state.py` defines four boolean fields (`comprehensiveness_pass`, `insight_pass`, `instruction_following_pass`, `factuality_pass`) that the LLM populates via structured output. The `research_reviewer()` function then constructs a `QualityAssessment` object with programmatically computed `failed_dimensions` and `all_pass` fields derived from these boolean scores.

```
class QualityAssessmentScores(BaseModel):
    comprehensiveness_pass: bool
    insight_pass: bool
    instruction_following_pass: bool
    factuality_pass: bool
    feedback: str
```

FIGURE 3.7: Multi-dimensional binary schema for structured quality assessment.

Refinement Control Strategy

The fourth design decision addresses termination control: how should the system handle persistent quality failures to prevent infinite refinement loops while maximising opportunities for improvement? Three control mechanisms were evaluated: (i) unlimited refinement iterations continuing until all quality dimensions pass, (ii) a fixed maximum number of refinements with graceful degradation (accepting suboptimal outputs with appropriate disclaimers), or (iii) a cost-based budget that decrements with each refinement attempt, terminating when resources are exhausted.

Tie et al.'s CorrectBench benchmark provides empirical evidence that while iterative self-correction improves accuracy, it incurs substantial time costs and exhibits diminishing returns; additional correction steps frequently yield minimal gains or even degrade performance for strong reasoning models [95]. Their introduction of the "Efficiency Rank" metric, which balances accuracy improvements against token consumption and API call overhead, theoretically supports implementing a cost-based budget that tracks resource expenditure per refinement attempt. Given the system's progressive token truncation limitations (Constraint 3), an unlimited refinement approach would risk exceeding context window limits.

Therefore, the implemented solution adopts a configurable maximum refinement limit with a fallback mechanism, departing from CorrectBench's cost-based approach. This decision prioritises simplicity and deterministic termination guarantees over dynamic resource management. The fixed iteration cap aligns with existing termination controls throughout the system architecture, providing operators with predictable worst-case execution bounds. When the refinement limit is reached without achieving full quality compliance, the system accepts the research output but attaches a disclaimer explicitly noting which quality dimensions failed validation. This fallback mechanism prevents pipeline blockage due to persistently problematic outputs while maintaining transparency about quality limitations. The approach also embodies the system's broader design philosophy of preferring potentially incomplete results with known limitations over indefinite execution or complete pipeline failure.

The `max_research_reviewer_refinements` configuration field (default: 2) in `Configuration` class controls the refinement budget. Within `research_reviewer()`, the check `if refinement_attempts >= configurable.max_research_reviewer_refinements` triggers graceful degradation, accepting the research with a disclaimer message appended to `compressed_research` noting the failed dimensions.

```
if refinement_attempts >= configurable.max_research_reviewer_refinements:
    disclaimer_message = f"[Research Reviewer: Max Refinements Reached]\n\n"

                        f"{compressed_research}\n\n"
                        f"[Note: Failed dimensions: {', '.join(
    failed_dimensions)}]"
    return Command(goto=END, update={"compressed_research":
    disclaimer_message})
```

FIGURE 3.8: Fallback mechanism when refinement budget is exhausted.

Decision Authority Separation

The fifth design decision concerns the allocation of decision-making authority: should the language model determine both the quality assessment and the accept/refine routing decision, or should these responsibilities be divided between model inference and programmatic logic? Three authority distribution models were considered: (i) the LLM making the final accept/refine decision directly as part of its generation, (ii) the LLM providing dimensional assessments while programmatic logic calculates the routing decision, or (iii) a hybrid approach wherein the LLM recommends a decision subject to deterministic rule overrides.

Wang et al.'s CIBER framework employs "Multi-Aspect Interrogation," probing language models with diverse queries (Agree versus Conflict probes) and resolving final truthfulness verdicts through mathematical fusion strategies such as Weighted Information Gain or Dempster-Shafer Belief Update [96]. CIBER explicitly separates the LLM's raw response generation from the final verdict determination, delegating fusion logic to deterministic algorithms. This architectural separation directly supports the division of responsibilities wherein the LLM provides structured evaluation data while programmatic logic computes the final decision. Rather than tasking the model with inherently unstable binary accept/reject determinations, CIBER's approach suggests prompting the model to output structured assessments for individual evaluation dimensions, then applying deterministic aggregation rules to derive the routing decision. This separation ensures robustness and reproducibility by addressing the stochastic variability inherent in LLM decision-making.

The implemented solution adopts strict separation in line with CIBER: the LLM returns only boolean pass/fail judgments for each dimension accompanied by natural language feedback, while the accept/refine decision is computed deterministically through programmatic logic. This separation between evaluation and decision-making ensures predictable, reproducible behavior, which is a critical requirement given that LMs exhibit inconsistency in binary decision-making even with identical inputs, as empirically observed during the quality assessment schema prototyping discussed in Section 3.4.4. If this inconsistency is permitted to influence routing decisions directly, it could potentially create unpredictable refinement loops with non-deterministic termination properties. By constraining the LLM's role to dimensional assessment and feedback generation while delegating branching logic to code, the system achieves determinism without sacrificing the model's evaluative capabilities.

The LLM returns only `QualityAssessmentScores`, while the accept/refine decision is computed deterministically in `research_reviewer()` through `all_pass = len(failed_dimensions) == 0` and `decision = "accept" if all_pass else "refine"`. Additionally, a programmatic citation check can override `scores.factuality_pass` to `False` when no inline citations are detected in substantial text, ensuring deterministic failure conditions bypass LLM judgment.

```
scores = await research_reviewer_model.ainvoke([HumanMessage(content=
    prompt_content)])

if len(citations_in_text) == 0 and len(compressed_research) > 500:
    scores.factuality_pass = False  # Deterministic override

all_pass = len(failed_dimensions) == 0
decision = "accept" if all_pass else "refine"  # Programmatic decision
```

FIGURE 3.9: LLM evaluation separated from deterministic decision logic.

3.4.5 Design Changes and Rationale

After evaluating the initial implementation, a second Research Reviewer variation was developed. The first modification involved refining the Research Reviewer system prompt while preserving its overall structure. The evaluation dimensions were revised to Comprehensiveness, Insight, Instruction Following, and Factuality, aligning the assessment criteria with the DeepResearch Bench RACE framework. The research refinement system prompt remained unchanged.

The second modification introduced structured feedback mechanisms. When research output fails to satisfy any evaluation criterion, the reviewer generates structured refinement directives comprising three components: `missing_subtopics`, `required_evidence_types`, and `action`. These directives are returned as structured objects rather than unstructured free-text feedback, enabling systematic processing by downstream components.

TABLE 3.1: Research Reviewer v1 vs v2 Implementation Comparison.

Area	v1	v2
Review Dimensions	Relevance, Depth, Evidence, Completeness	Comprehensiveness, Insight, Instruction Following, Factuality
Assessment Process	Reviewer outputs PASS/FAIL for each dimension, plus free-text feedback.	Reviewer outputs PASS/FAIL for each dimension, plus a structured feedback object.
Guidance/Feedback	Free-text feedback, plus actionable recommendations for improvement.	Structured directives for missing subtopics, evidence types, and actions.

3.4.6 Implementation Overview

The Research Reviewer is implemented as a node within the researcher subgraph that intercepts compressed research outputs before they return to the supervisor, as illustrated in Figures 3.10 and 3.11. Upon receiving compressed research, the reviewer invokes Google Gemini 2.0 Flash to evaluate the output against four quality dimensions, returning structured boolean assessments that programmatic logic aggregates to determine whether to accept the research or trigger refinement. When refinement is required, the system loops back to the researcher node, with a specialised system prompt targeting refinemeent, with feedback guiding targeted improvements, continuing until all dimensions pass or the configurable refinement limit is reached. This architecture maintains supervisor abstraction by encapsulating the entire quality-assurance cycle within the subgraph boundary, ensuring the supervisor receives only validated outputs without requiring modifications to the existing delegation logic.

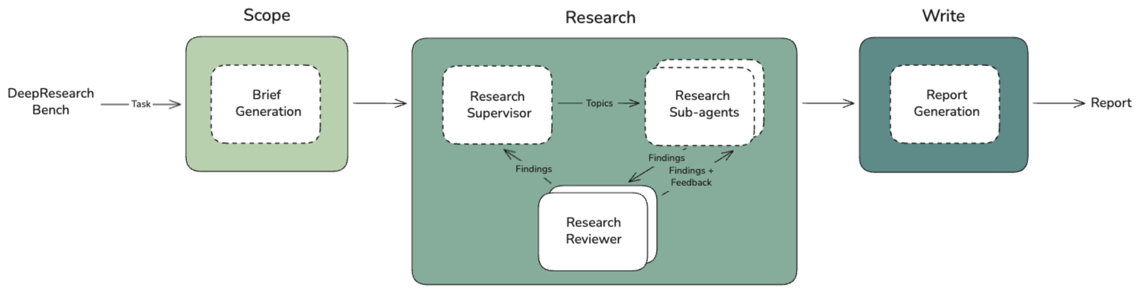


FIGURE 3.10: The Langchain Open Deep Research architecture with an integrated Research Reviewer component.

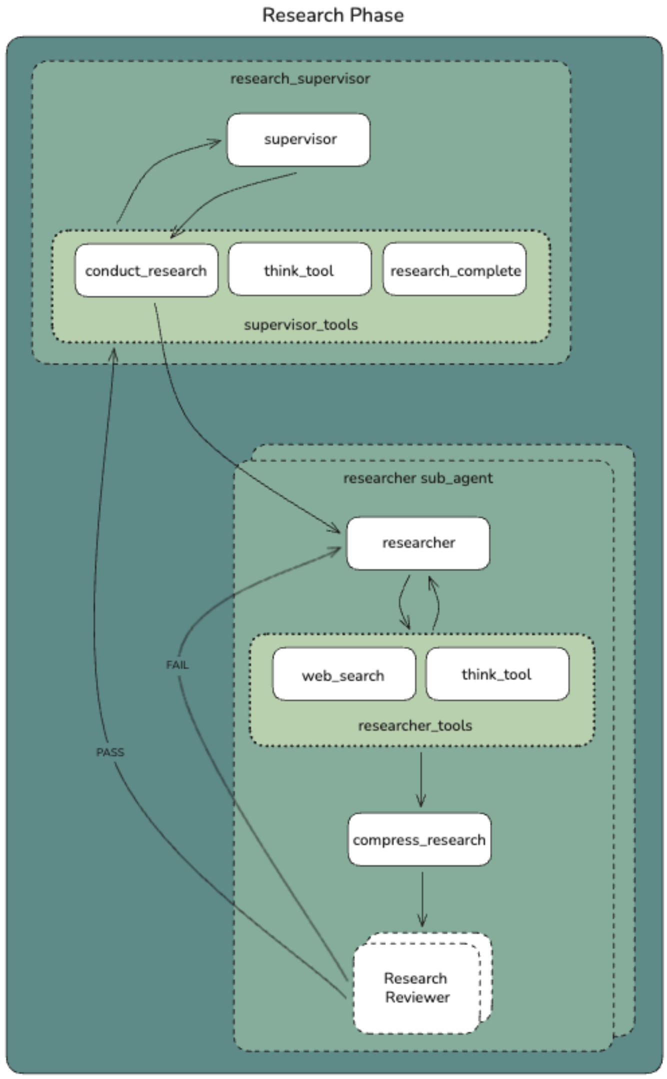


FIGURE 3.11: The Research Reviewer architecture within the research phase.

3.5 Research Verifier Implementation

TODO: complete

3.5.1 System Constraints

3.5.2 Design Decisions

3.5.3 Implementation Overview

The Research Verifier is a post-processing component responsible for validating citation accuracy before final report delivery as depicted in Figure 3.12. The implementation adapts the FACT evaluation methodology from the DeepResearch Bench benchmark, originally designed for benchmarking, into a real-time verification step. The verification process operates in four sequential stages, in line with the corrected logic described in Section 3.9.2. First, an LLM extracts discrete factual claims paired with their cited source URLs from the generated report. Second, equivalent claims citing the same source are deduplicated to avoid redundant verification. Third, the system retrieves the textual content of each cited webpage using the Jina AI Reader API. Fourth, an LLM compares each extracted claim against the retrieved source content and issues a support judgment. Upon completion, the verifier reconstructs the report by preserving only those claims determined to be supported by their cited sources, while maintaining the original structure and wording. Unsupported claims are removed without introducing new content or considerably restructuring the document.

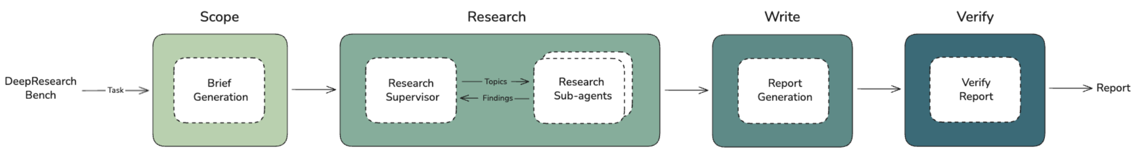


FIGURE 3.12: The Langchain Open Deep Research architecture with an integrated Research Verifier component.

TODO: add architecture diagram

3.6 Adaptive Model Selection Implementation

3.6.1 Cascading vs. Routing

While minimal literature exists on the notion of adaptive model selection within agentic AI systems, existing research positions model routing and model cascading as two distinct strategies for optimising the cost-performance trade-off in LM inference. The cascade approach employs sequential execution ($M_{small} \rightarrow M_{large}$), where the routing decision occurs after the smaller model generates output, contingent upon confidence or consistency thresholds. In contrast, the routing approach performs selective execution (M_{opt}) through an upfront prediction, wherein the optimal model is determined before any inference occurs based on prompt features or estimated task complexity.

Zhu et al.'s seminal work on model multiplexing provides empirical grounding for this architectural choice [97]. Their framework defines model multiplexing as a policy $\pi(q)$ that selects between small and large models based on expected cost and accuracya formulation consistent with the routing paradigm. While the authors acknowledge that cascade approaches may outperform inaccurate routers when smaller models are substantially cheaper, their experimental comparisons between "LEC + selector" (routing) and "LEC + cascade" demonstrate that routing systematically outperforms cascading when the selector achieves sufficient classification accuracy.

The routing-based architecture is therefore adopted due to its superior efficiency characteristics in agentic AI contexts, where agents frequently invoke models across multi-turn interactions. Cascading imposes unavoidable computational and latency overheads by mandating smaller model invocation on every query, irrespective of whether this output is ultimately retained or discarded. Routing eliminates this redundancy through direct allocation to the appropriate model tier, a distinction that becomes critical in multi-agent workflows where sequential inference costs compound across interaction chains. Moreover, routing aligns inherently with the proactive assessment paradigm required in agentic systems: task complexity can be evaluated through structured prompt analysis prior to model invocation, enabling intelligent resource allocation without incurring preliminary inference costs. While cascading provides a fail-safe mechanism under conditions of low router confidence, empirical evidence demonstrates that well-designed routing systems consistently achieve superior cost-efficiency and reduced latency while maintaining comparable accuracy [97].

Moreover, the system constraints enumerated in 3.6.3 collectively necessitate a routing-based architecture rather than a cascading approach. The inability to perform task re-routing (Constraint 6) eliminates iterative refinement strategies that cascading systems depend upon for error correction. The requirement for complexity assessment at delegation time (Constraint 1) combined with prompt-based classification (Constraint 2) aligns precisely with routing's upfront prediction paradigm, wherein task complexity features are extracted prior to model invocation. Parallel sub-researcher execution (Constraint 3) further disadvantages cascading, as sequential $M_{small} \rightarrow M_{large}$ evaluation would introduce compounding latency across concurrent research branches. Fixed iteration and token budgets (Constraint 4) render the redundant inference costs of cascading prohibitive, particularly when preliminary outputs from smaller models are discarded.

3.6.2 The Meta-Optimisation Dilemma

Introducing an adaptive model selection algorithm presents a fundamental dilemma: evaluating the difficulty of a given task necessitates the use of a LM, yet the primary objective of adaptive model selection is to reduce overall computational costs. Deploying an additional model to assess the complexity of numerous sub-tasks introduces overhead that may counteract the intended savings. This constitutes a genuine paradox in meta-optimisation, wherein the cost of determining which model to use could exceed the savings realised by selecting a less expensive model. Formally, the overhead of an evaluator is justified only when the following inequality holds:

$$\text{Cost}(\text{selected\_model}) + \text{Cost}(\text{evaluator}) < \text{Cost}(\text{default\_model}) \tag{3.2}$$

Piggybacking on Existing LLM Calls

The most elegant solution to circumventing the LLM overhead involves integrating the complexity assessment on existing LM calls within the system architecture. Specifically, the research supervisor agent, which already executes prior to spawning individual researcher agents, can be augmented to output an estimated complexity level as part of its existing planning step. The key insight underlying this approach is that the supervisor must already reason about the research task in order to generate the `research_topic` parameter for each sub-agent. Extending the output schema to include a `complexity_assessment` field adds negligible cognitive load to the model and incurs zero additional API cost, as the complexity assessment becomes a side-effect of work that is already performed. The specific invocation point is discussed in detail in 3.6.4.

The advantages of this approach are substantial: the marginal token cost is minimal, no additional API call is required, and the complexity assessment leverages the supervisor's contextual understanding of the overall research task. However, potential limitations include variability in the accuracy of the supervisor's estimates and increased complexity in the output schema, which may require additional validation logic to handle edge cases where the complexity field is malformed or absent.

3.6.3 System Constraints

The AMS algorithm must operate within these architectural and operational constraints:

(Constraint 1) **Complexity Assessment at Delegation Time**: The supervisor must assess sub-task complexity before delegating to sub-researchers. Once a sub-researcher is invoked with a specific model, that model cannot be changed mid-task.

(Constraint 2) **Prompt-Based Complexity Classification**: Complexity assessment must be performed by the supervisor model itself using prompt engineering and structured output schemas. This involves outputting a complexity level alongside each delegation.

(Constraint 3) **Parallel Sub-Researcher Execution**: Multiple sub-researchers run concurrently. Model selection decisions must be made independently for each sub-task without knowledge of other concurrent tasks' outcomes.

(Constraint 4) **Fixed Iteration and Token Budgets**: Sub-researchers have hard limits on tool call iterations and token consumption. Model selection must consider whether a task can complete within these constraints.

(Constraint 5) **Configuration-Driven Model Pool**: Available models are defined in a configuration schema. The algorithm must select from this predefined pool rather than dynamically discovering models.

(Constraint 6) **No Task Re-Routing or Escalation**: The initial sub-task complexity classification final for that sub-task. If a simpler model struggles with a task, the system cannot automatically escalate to a more capable model.

3.6.4 Design Decisions

The six core design decisions outlined in the subsequent sections are informed by principles extracted from the model multiplexing, cascading, and routing literature, adapted to accommodate the constraint environment defined in 3.6.3. The three-tier complexity classification scheme draws from MDAgents' hierarchical assessment mechanism [98] and MoT's dual-modality detection principles [99], both of which operate within prompt-based classification paradigms without requiring runtime learning. Model pool definition synthesises Hybrid LLM's quality gap analysis [100] with LEC's reliability-weighted optimisation [97], constrained by the fixed model inventory available through OpenRouter APIs. Prompt modifications incorporate LLMRank's feature extraction methodology [101] to enable complexity assessment at delegation time (Constraint 1), while default behaviour policies adapt early abstention principles [102] to environments prohibiting task re-routing. Token limit handling considerations, though informed by FrugalGPT's pre-estimation strategies [103], ultimately defer to existing framework mechanisms due to implementation complexity trade-offs. Each design decision thus represents a constrained optimisation problem: extracting applicable principles from the literature while respecting architectural boundaries that preclude more sophisticated approaches such as reinforcement learning, dynamic re-routing, and custom model training.

Complexity Classification Scheme

The categorisation of task complexity constitutes a foundational design decision that directly influences model assignment granularity and classification reliability. Three principal approaches were considered: discrete tier classification (low/medium/high), numeric scoring (1–10), and binary classification (simple/complex). While numeric scoring offers fine-grained control and binary classification provides maximal simplicity, a discrete three-tier scheme was adopted based on empirical precedent and practical considerations of model pool mapping.

The MDAgents framework provides strong empirical support for this architectural choice through its hierarchical complexity assessment mechanism [98]. Within this medical diagnosis system, an evaluator agent categorises incoming queries into three distinct complexity levels: low-complexity queries requiring single-agent resolution, moderate-complexity queries necessitating multi-agent consultation, and high-complexity scenarios demanding extensive inter-agent coordination. The framework instantiates this classification through explicit prompt engineering, instructing the moderator to "decide the difficulty/complexity of the medical query" among three discrete options "1) low... 2) moderate... 3) high" with a one-sentence definition for each tier. This tripartite structure enables clear model-to-tier mapping while maintaining sufficient granularity to differentiate between routine, standard, and demanding research sub-tasks.

The integration of "Mixture of Thought" (MoT) principles further informs complexity classification heuristics within this scheme [99]. MoT demonstrates that complex reasoning tasks benefit significantly from the synergy between narrative reasoning (Chain-of-Thought) and symbolic execution (Program-of-Thought). This observation translates to a specific classification criterion: sub-tasks requiring both qualitative reasoning and precise calculation or code execution should be classified as high-complexity, as mid-tier models typically excel at one modality but not both simultaneously. This dual-modality detection serves as a reliable signal for escalation to top-tier models.

The three-tier complexity classification is implemented through the `ComplexityAssessment` Pydantic model defined in `state.py`, which captures discrete tier assignment alongside structured feature extraction. The `ModelSelector._check_safety_escalation()` method implements MoT-based detection by identifying tasks requiring both narrative reasoning and symbolic execution, automatically escalating such tasks to the high tier regardless of supervisor classification.

```
class ComplexityAssessment(BaseModel):
    tier: Literal["low", "mid", "high"]
    estimated_confidence: int
    failure_risk: Literal["low", "medium", "high"]
    features: Optional[FeatureExtraction] = None
    context_estimation: Optional[ContextEstimation] = None
    quality_gap_prediction: Literal["negligible", "moderate", "significant"
    ]
    rationale: str
```

FIGURE 3.13: Three-tier complexity classification schema with MoT detection.

Model Pool Definition

The definition of which models constitute each complexity tier represents the fundamental challenge in adaptive model selection: determining which models demonstrate superior performance for specific task categories. No consensus methodology exists for this allocation problem, necessitating a synthesis of theoretical principles from model multiplexing literature with practical constraints imposed by the OpenRouter API's available model inventory.

The Hybrid LLM framework introduces the concept of the "quality gap"—the performance differential between small and large models for specific query types—as a guiding principle for low-tier model definition [100]. The framework demonstrates empirically that for approximately 20% of queries, smaller models achieve response quality identical to larger models, indicating negligible quality gaps for certain task categories such as summarisation or structured extraction. This observation establishes a critical criterion for low-tier model assignment: models should be allocated to this tier not solely on the basis of cost efficiency, but specifically when deployed against task types exhibiting historically negligible quality gaps. The supervisor should therefore route to low-tier models only when task characteristics predict near-zero quality degradation, thereby achieving cost reduction without performance compromise.

The principles underlying the Least Expected Cost (LEC) algorithm further refine model pool definition by introducing reliability-weighted cost optimisation [97]. While the LEC framework addresses caching and query frequency, its core insight—optimising for expected cost rather than inference cost alone—proves applicable to tier definition. Expected cost incorporates both the probability of model failure and the downstream cost of that failure, which in multi-agent research systems manifests as failed research branches and wasted computational resources. This formulation implies that model pools should not be structured linearly by cost (cheap/medium/expensive), but rather defined by reliability thresholds. Specifically, the mid-tier model must exhibit the highest reliability-to-cost ratio to serve as the primary workhorse for the majority of standard research tasks, balancing cost efficiency with acceptable failure rates.

Constrained by the available model inventory on OpenRouter, the following tier definitions and criteria guided model selection:

- **Low-tier**: Models optimised for fast execution, minimal cost, and routine or deterministic tasks requiring limited reasoning. These models must be substantially cheaper and less capable than the mid-tier baseline.
- **Mid-tier**: Models suited for moderate reasoning demands, light abstraction, structured workflows, and efficiency-sensitive operations. This tier serves as the default allocation for standard research sub-tasks.
- **Top-tier**: Models designed for deep reasoning, multi-constraint problem solving, creativity, extended context analysis, and open-ended inference. These models must demonstrate superior capability relative to the mid-tier baseline.

Using models from a single provider family ensures consistent API behaviour, output formatting, and evaluation comparability, while the version numbering establishes an unambiguous capability progression that aligns with increasing per-token costs. The final model pool comprises:

- **Low-tier**: GPT-4o Mini, OpenAI's most advanced small model, offers significant cost savings of over 60% cheaper than GPT-3.5 Turbo, while maintaining state-of-the-art intelligence [104].
- **Mid-tier**: GPT-4.1 Mini, a mid-sized model that delivers performance competitive with GPT-4o while operating at substantially lower latency and cost [105].
- **Top-tier**: GPT-o1, OpenAI's most advanced reasoning model offering major improvements in step-by-step reasoning, code quality, and accuracy in high-stakes use cases, with support for test-time routing and advanced prompt understanding [106].

The selection of GPT-4o Mini as the low-tier model—identical to the baseline system's research model—is deliberate. This equivalence establishes a controlled comparison: when the adaptive selector routes simple tasks to the low-tier, it operates under identical conditions to the baseline, isolating the contribution of intelligent routing. Cost optimisation is hypothetically achieved not through cheaper low-tier models, but through the token efficiency of higher-tier models on complex tasks. More capable models may require fewer iterations, generate more concise outputs, and execute fewer tool calls to achieve equivalent or superior results, potentially offsetting their higher per-token costs through reduced total token consumption. The model cost and context window sizes are presented in Table 3.2.

TABLE 3.2: Model costs and context window sizes for the AMS model pool.

Role	**Model**	**Input Cost** ($/M tokens)	**Output Cost** ($/M tokens)	**Context Window**
Tier 1 Model	OpenAI GPT-4o Mini	$0.15	$0.60	128K tokens
Tier 2 Model	OpenAI GPT-4.1 Mini	$0.40	$1.60	1.05M tokens
Tier 3 Model	OpenAI GPT-5	$1.25	$10.00	400K tokens

Token Limit Handling

The interaction between adaptive model selection and varying context window capacities across model tiers presents a secondary consideration in routing logic. Three approaches were evaluated: implementing fail-and-retry mechanisms that escalate to larger-context models upon capacity exhaustion, pre-estimating context requirements and incorporating this estimation into initial model selection, or maintaining existing truncation and retry logic without modification.

The FrugalGPT framework's emphasis on prompt adaptation and query concatenation strategies for managing token costs and limitations suggests that pre-estimation represents the theoretically optimal approach [103]. Within a fixed-budget system such as the Deep Research framework, this principle would translate to supervisor-based context load estimation: if anticipated retrieval or contextual requirements exceed the low-tier model's context window—typically smaller in cost-optimised models—the routing logic should automatically escalate to mid-tier or top-tier models possessing larger context capacities, independent of the task's semantic complexity. This pre-emptive escalation would prevent failures due to context overflow.

However, this approach was ultimately rejected due to implementation complexity considerations. The research supervisor's system prompt already encompasses extensive instructions for sub-task generation, research planning, and synthesis coordination. The addition of token load estimation heuristics would introduce substantial prompt verbosity and cognitive load without commensurate benefits, given that the existing framework already implements truncation and retry mechanisms. Consequently, the adaptive model selection system maintains existing token limit handling logic unchanged, relegating context window management to the framework's established error recovery procedures rather than incorporating it into upfront routing decisions.

Context window constraints are enforced through `ModelSelector._enforce_context_window_constraint()`, which calculates estimated token requirements from `ContextEstimation` attributes and applies multiplicative factors for retrieval breadth and context accumulation. When estimated context exceeds a tier's capacity adjusted by the safety margin, the method returns a tuple containing the upgraded tier identifier and justification string, triggering tier escalation in the parent `resolve_tier()` method.

Prompt Modifications

The method by which the supervisor assesses sub-task complexity directly determines classification accuracy and, consequently, routing effectiveness. The MDAgents framework provides a baseline approach through its moderator agent, which receives explicit instructions to classify queries into complexity tiers accompanied by single-sentence definitions for each tier [98]. While this approach establishes the viability of prompt-based classification, its reliance on implicit model reasoning without structured feature extraction represents a limitation in reliability and interpretability.

The implementation developed for this thesis enhances this baseline through the introduction of explicit complexity metrics that characterise task attributes and enable systematic classification. This design was informed by the feature extraction methodology presented in LLMRank, which demonstrates that explicit feature identification—including task type indicators (e.g., reasoning versus retrieval), scenario complexity attributes (ambiguity, contextual difficulty), and reasoning pattern requirements—enables substantially more accurate prediction of model utility [101]. Rather than instructing the supervisor to directly output a complexity tier based on intuition, the modified prompt requires the supervisor to first evaluate the sub-task against specific dimensions (e.g., reasoning depth, retrieval scope) and subsequently derive the complexity classification from these measured attributes. This metrics-driven approach increases classification consistency and provides greater transparency regarding the rationale underlying routing decisions.

The supervisor prompt incorporates structured complexity assessment instructions that require explicit feature evaluation before tier classification. The `lead_researcher_prompt` template specifies classification criteria organized by task type, reasoning requirements, and quality gap considerations, with the `ConductResearch` tool schema enforcing structured `ComplexityAssessment` output.

Default Behaviour

The system's response to classification failures or ambiguous assessments constitutes a critical design parameter that balances cost efficiency against task completion reliability. Two principal strategies were considered: defaulting to mid-tier models as a balanced compromise between cost and capability, or defaulting to top-tier models to maximise safety at the expense of efficiency.

The concept of early abstention from cascade-based model selection systems provides theoretical guidance for this decision [102]. In cascade systems, smaller models are trained to recognise when they lack sufficient capability to handle a query and abstain from responding, thereby triggering escalation to larger models. This principle, when adapted to the constraint environment of this thesis where re-routing is prohibited once a model is assigned (Constraint 6), suggests an inversion of the abstention logic: if the supervisor cannot affirmatively classify a task as low-complexity with high confidence, it should interpret this uncertainty as an implicit abstention signal. The correlation between classification uncertainty and potential model failure implies that ambiguous cases represent elevated risk scenarios.

Consequently, the system implements a conservative default policy: classification failures or low-confidence assessments trigger automatic assignment to mid-tier models at minimum, with provision for escalation to top-tier models if uncertainty signals are particularly strong. This approach prioritises task completion reliability over cost optimisation in edge cases, recognising that the penalty for routing a complex task to an inadequate model, a failed research branch within the fixed iteration budget, substantially exceeds the marginal cost of conservative model selection. By treating classification uncertainty as a risk factor rather than a neutral outcome, the default behaviour aligns with the system's inability to implement corrective re-routing mechanisms.

Conservative escalation is implemented through confidence threshold checks in `ModelSelector._check_safety_escalation()`, which interprets low-confidence assessments as abstention signals requiring tier upgrade. The `_resolve_tier_for_task()` fallback mechanism defaults to mid-tier when complexity assessment parsing fails or adaptive selection is disabled.

Invocation Point

The architectural placement of adaptive selection logic within the existing codebase determines both implementation complexity and maintainability. Three candidate locations were evaluated: integration within the `researcher()` function to read complexity classifications from state, implementation within `supervisor_tools()` to pass model assignments via configuration overrides, or creation of a dedicated utility function `select_model_for_complexity()` within the utilities module.

The `researcher()` function approach offers the advantage of centralising model selection at the point of sub-researcher instantiation, enabling direct access to state information and straightforward parameter passing. However, this approach conflates sub-researcher creation logic with model selection heuristics, potentially reducing code modularity. The `supervisor_tools()` implementation provides earlier intervention in the workflow by embedding selection logic within the supervisor's tool invocation mechanism, allowing model assignments to be determined before sub-researcher creation. This approach maintains clearer separation of concerns by positioning routing decisions as a supervisory responsibility rather than a sub-researcher configuration detail.

The utility function approach offers maximal modularity by encapsulating all complexity-to-model mapping logic within a dedicated, testable function that can be invoked from multiple locations as needed. This design facilitates future modifications to selection heuristics without requiring changes to core workflow components and enables straightforward unit testing of routing logic in isolation. Given the experimental nature of adaptive model selection and the likelihood of iterative refinement based on performance observations, the utility function approach was adopted to maximise code maintainability and support rapid experimentation with alternative selection strategies.

Adaptive model selection is invoked within `supervisor_tools()` through the `_resolve_tier_for_task()` helper function, which extracts complexity assessments from `ConductResearch` tool calls and delegates tier resolution to `ModelSelector`. The resolved tier configuration is passed to the researcher subgraph via state variables (`selected_model`, `selected_max_tokens`, `selected_tier`), where `researcher()` applies the tier-specific model configuration.

```
async def supervisor_tools(state, config):
    model_selector = ModelSelector(configurable.adaptive_model_config)
    ...

    research_tasks = []
    for tool_call in allowed_conduct_research_calls:
        tier_decision = _resolve_tier_for_task(
            tool_call, model_selector, configurable
        )

        task = researcher_subgraph.ainvoke({
            "selected_model": tier_decision.model_config.model,
            "selected_max_tokens": tier_decision.model_config.max_tokens,
            "selected_tier": tier_decision.tier
            ...
        }, config)
        research_tasks.append(task)
```

FIGURE 3.14: Invocation point in supervisor tools with tier resolution.

```
# Supervisor passing complexity assessment in the ConductResearch tool call
ConductResearch(
    research_topic="...",
    complexity_assessment={
        "tier": "mid",
        "estimated_confidence": 85,
        "failure_risk": "low",
        ...
    }
)
```

FIGURE 3.15: Code excerpt of a `conduct_research` tool call passing complexity assessment.

3.6.5 Design Changes and Rationale

The initial implementation yielded no allocations to the low-tier model, as the supervisor's complexity assessments consistently classified sub-tasks above the low-tier threshold. To enforce tier diversity, the research supervisor prompt was revised to generate exactly three sub-tasks ranked by complexity, which were then programmatically mapped to the low, mid, and high tiers respectively.

3.6.6 Implementation Overview

The adaptive model selection mechanism is integrated into the Langchain Open Deep Research architecture as illustrated in Figure 3.16, where the `model_selector` component operates within the research phase to mediate between the research supervisor and sub-agent instantiation. As detailed in Figure 3.17, the supervisor invokes `conduct_research` tool calls containing structured complexity assessments, which are intercepted by `supervisor_tools()` and passed to the `ModelSelector` service for tier resolution. The resulting tier decision determines the model configuration supplied to each research sub-agent, enabling

TABLE 3.3: AMS v1 vs v2 Implementation Comparison.

Area	v1	v2
Research Supervisor Prompt	Flexible sub-task count (at least 1). No ordering requirement. Supervisor classifies each task tier (low/mid/high).	Fixed 3 sub-tasks. Ordered by complexity (simplest to most complex). Supervisor assigns complexity rank (1, 2, 3).
Model Tier Assignment	Supervisor explicitly sets `tier` field. Model selector can escalate or downgrade tiers based on confidence and quality gap.	Programmatic mapping: position 0→low, 1→mid, 2→high.. Uses `complexity_rank` field. Only escalation allowed, no downgrades.

heterogeneous model deployment across parallel research tasks without requiring modifications to the researcher or supervisor node implementations. This architecture preserves the existing state management and tool execution semantics while introducing adaptive model selection as an orthogonal concern.

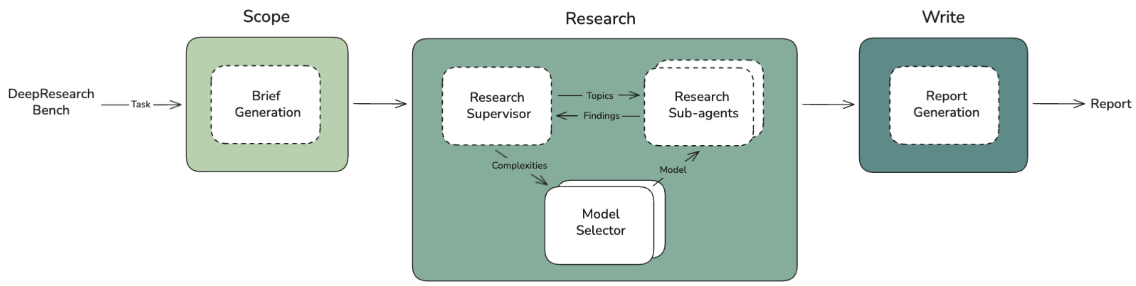


FIGURE 3.16: The Langchain Open Deep Research architecture with integrated Adaptive Model Selection.

Algorithm 1 Adaptive Model Selection

Require: Complexity assessment A with tier t, confidence c, failure risk r, features F, context estimation C
Ensure: Tier decision D with model configuration

```
t_orig ← A.tier                                          ▷ Store original tier recommendation
t_current ← A.tier
Stage 1: Safety Escalation
if F.narrative_reasoning ∧ F.symbolic_execution then     ▷ MoT signal
    t_current ← HIGH                                     ▷ Escalate for synergistic reasoning
else if c < θ_mid then                                   ▷ Low confidence threshold (default: 70%)
    t_current ← HIGH                                     ▷ Safety escalation
else if r = HIGH then
    t_current ← HIGH                                     ▷ Failure risk escalation
else if F.context_difficulty = contradictory then
    t_current ← HIGH                                     ▷ Ambiguous context requires high-tier
else if t_current = LOW ∧ c < θ_low then                 ▷ Low-tier threshold (default: 85%)
    t_current ← MID                                      ▷ Escalate from low to mid
end if
Stage 2: Context Window Constraint
ctx_est ← (C.input_tokens + C.output_tokens) × β_breadth × β_accum
if t_current = LOW then
    limit_low ← CONTEXT_WINDOW_LOW × α_safety
    if ctx_est > limit_low then
        limit_mid ← CONTEXT_WINDOW_MID × α_safety
        if ctx_est > limit_mid then
            t_current ← HIGH                             ▷ Exceeds mid-tier capacity
        else
            t_current ← MID                              ▷ Exceeds low-tier capacity
        end if
    end if
else if t_current = MID then
    limit_mid ← CONTEXT_WINDOW_MID × α_safety
    if ctx_est > limit_mid then
        t_current ← HIGH                                 ▷ Exceeds mid-tier capacity
    end if
end if
Stage 3: Cost Optimization Downgrade
if A.quality_gap = NEGLIGIBLE ∧ r = LOW then
    if t_current = MID ∧ c ≥ 95 then                     ▷ Mid-to-low requires 95% confidence
        if F.task_type = retrieval ∧ F.reasoning ∈ {single-hop, none} then
            t_current ← LOW                              ▷ Cost optimisation: simple retrieval
        end if
    else if t_current = HIGH ∧ c ≥ 90 then               ▷ High-to-mid requires 90% confidence
        if ¬(F.narrative_reasoning ∧ F.symbolic_execution) then   ▷ No MoT
            if F.context_difficulty ≠ contradictory then
                if F.task_type = synthesis ∧ F.context_difficulty = clear then
                    t_current ← MID                      ▷ Cost optimisation: clear synthesis
                end if
            end if
        end if
    end if
end if
return TierDecision(t_current, MODEL_CONFIG[t_current], t_orig)
```

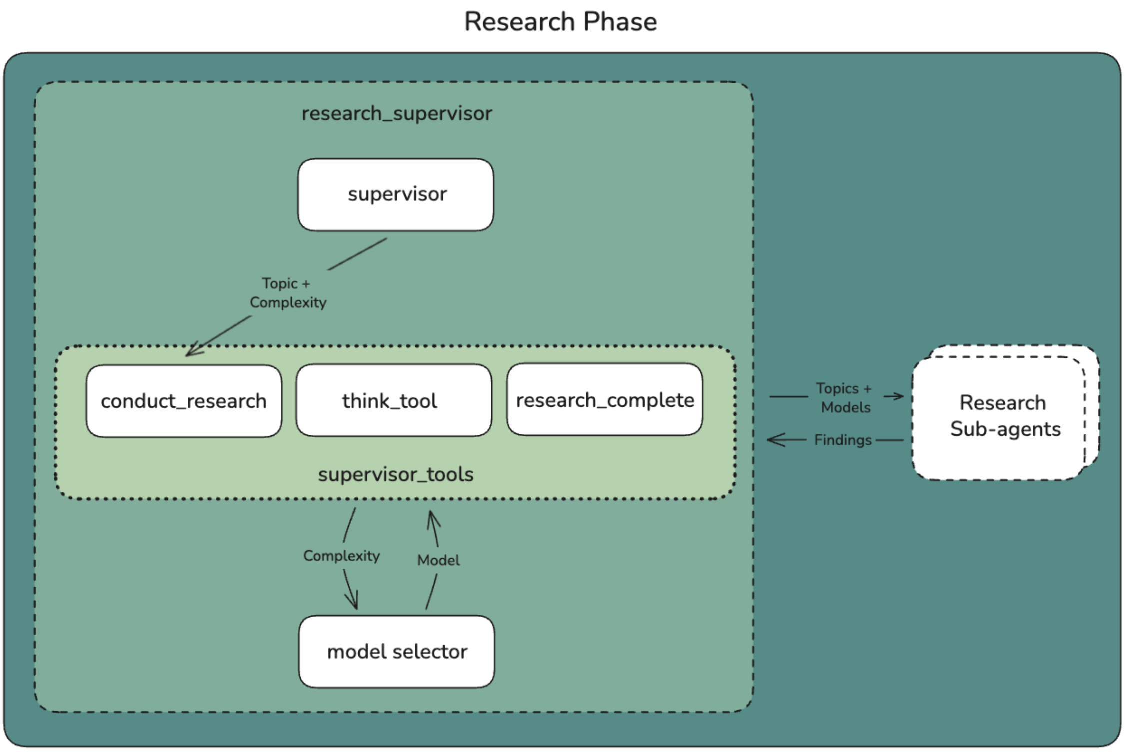


FIGURE 3.17: The AMS architecture within the research phase.

3.7 Baseline System Configuration Setup

The establishment of a baseline system for the Open Deep Research system was driven by the need to balance computational cost with research quality, given the resource-intensive nature of multi-agent deep research workflows. Each research task in such systems typically involves numerous LLM API calls, parallel web searches, and iterative refinement cycles, which collectively generate substantial inference costs. To achieve an economically viable baseline while maintaining acceptable performance standards, a systematic configuration process was undertaken involving three key dimensions: modification of agent system prompts, adjustment of behavioural parameters governing agent interactions, and strategic selection of cost-efficient models across all phases of the research pipeline.

The efficacy of these optimisations was evaluated against a set of 50 deep research tasks, as detailed in Section 3.8. The combined effect of system prompt modifications and behavioural parameter adjustments alone yielded significant cost reductions: the total number of Tavily search API calls decreased from 2,580 to 1,754 (a 32% reduction), while total inference costs declined from $24.91 to $18.32 (a 26.5% reduction). Furthermore, strategic model selection produced substantial additional savings. For context, three benchmark submissions by LangChain against the DeepResearch Bench of all 100 questions (see Section 3.8.1) incurred total costs of $45.98, $187.09, and $87.83 [17], highlighting the considerable expense associated with unconstrained deep research configurations.

3.7.1 Researcher Agent System Prompt

The researcher agent's system prompt includes a Hard Limits guideline, which specifies explicit limits on the number of search tool calls permitted: for simple queries, the agent is allowed a maximum of 2-3 search calls, while for complex queries, up to 5 calls are permitted, as shown in Figure 3.18. These constraints were revised to restrict simple queries to 1-2 search calls and complex queries to a maximum of 3. The researcher agent's complete system prompt is presented in Appendix C.1.

```
""" <Hard Limits>
**Tool Call Budgets** (Prevent excessive searching):
- **Simple queries**: Use 2-3 search tool calls maximum
- **Complex queries**: Use up to 5 search tool calls maximum
- **Always stop**: After 5 search tool calls if you cannot find the right
    sources
...
</Hard Limits>
"""
```

FIGURE 3.18: Code excerpt of the hard limits guideline in the researcher agent's system prompt.

3.7.2 Behavioural Parameters

The three behavioural limits governing the research phase were also significantly tightened. First, `max_concurrent_research_units`, which is injected into the research supervisors system prompt and determines how many times it may invoke `ConductResearch` (thereby spawning researcher agents) within a single iteration, was reduced from 10 to 3. Second, `max_researcher_iterations`, referring to the number of research cycles in which researcher agents are assigned tasks and return their findings before the supervisor loop terminates, was lowered from 6 to 3. Finally, `max_react_tool_calls`, which restricts the total number of ReAct-style tool invocations available to each researcher agent—including task completion signals (`ResearchComplete`), reflective planning steps (`think_tool`), and web search calls (`tavily_search`)—was decreased from 10 to 6. This reduction nearly halves the original allowance while still providing twice the number of permissible search calls, since reflective calls typically follow search actions and the system prompt now enforces a maximum of three searches per research task. Figure 3.19 presents a side-by-side comparison of the original and reduced behavioural limits for the research phase configuration. Section 3.10 *Experiment Design* later refers to the original configuration as the `Maxed` behaviour configuration, and the reduced configuration as the `Baseline` behaviour configuration.

Task Decomposition Bounds

The research supervisor's capacity for subtopic generation is governed by two primary configuration parameters: the maximum number of concurrent research units, denoted $k_{\text{concurrent}}$, and the maximum number

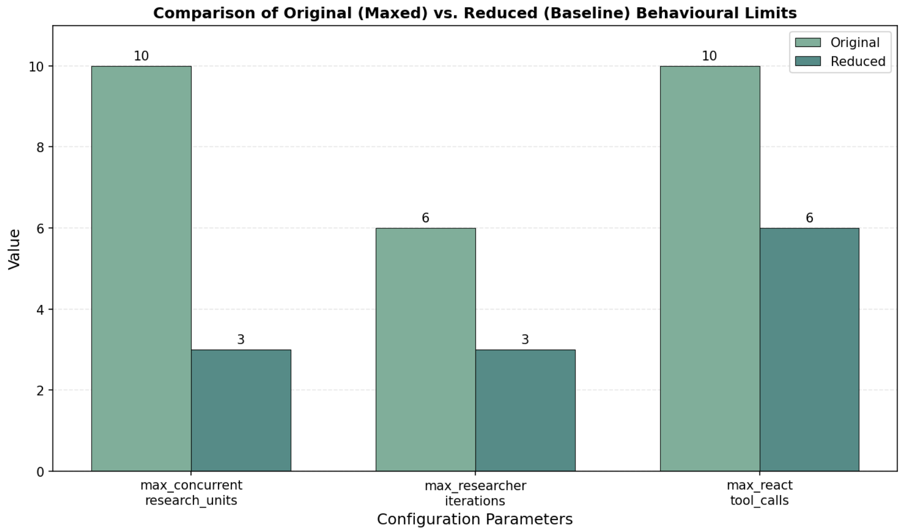


FIGURE 3.19: Comparison of original and reduced behavioural limits for the research phase configuration.

of supervisor iterations, denoted $n_{\text{iterations}}$. Under the specified configuration where $k_{\text{concurrent}} = 3$ and $n_{\text{iterations}} = 3$, the iteration counter i is incremented upon each invocation of the supervisor function, with the termination condition evaluated as $i > n_{\text{iterations}}$. Consequently, the supervisor executes at most $n_{\text{iterations}}$ complete cycles before forced termination occurs at iteration $i = n_{\text{iterations}} + 1$. Within each iteration, the system enforces an upper bound of $k_{\text{concurrent}}$ parallel `ConductResearch` invocations, yielding a theoretical maximum of $n_{\text{iterations}} \times k_{\text{concurrent}} = 9$ subtopics. Since the supervisor's primary function is to delegate research tasks, it will in practice generate at least one subtopic before signaling completion. Thus, the number of generated subtopics s satisfies $1 \leq s \leq 9$, with the distribution influenced by task complexity: simple fact-finding queries typically require $s \in [1, 3]$ subtopics within a single iteration, whereas comparative analyses spanning multiple entities may approach the upper bound across successive iterations.

3.7.3 Model Selection

As aforementioned in the 3.1 *Model Specialisation Strategy*, the Open Deep Research system utilises four distinct models, each serving a specialised function within the research workflow. The models are ordered by their estimated cost contribution as follows:

1. Research Model: Powers both the supervisor and researcher sub-agents in planning and executing research tasks. This shared model handles the entire research workflow: the supervisor uses it to decompose complex questions, plan research strategies, and delegate tasks, while the researcher sub-agents use the same model to conduct focused research and make iterative decisions. It requires robust reasoning capabilities alongside support for parallel tool calls, structured JSON outputs, and web search integration via the configured Tavily search API. Its large context window (128K+ tokens) and high usage volume make it the primary driver of computational cost, as each research task can entail 10-60 API calls with inputs ranging from 5K to 20K tokens.

2. Summarisation Model: Focuses on extracting structured summaries from raw webpage content retrieved by Tavily searches, compressing key information while preserving citations and key excerpts. Operating under a 64K+ token context window, it prioritises speed to meet a 60-second timeout per summary, processing multiple searches in parallel and serving as the second-largest cost driver due to high call volume.

3. Compression Model: Consolidates the complete outputs of each researcher sub-agent, synthesising search results, tool outputs, and conversation histories into structured summaries while ensuring citation integrity, deduplication, and elimination of hallucinations. It handles large context windows (128K+ tokens) and is invoked once per researcher. Despite being mandatory in the workflow, its relatively short execution time and moderate context size keep its cost contribution manageable.

4. Final Report Model: Integrates all compressed and reviewed research outputs into a coherent, high-quality final report, demonstrating advanced synthesis and complex reasoning capabilities, resolving contradictions, and identifying patterns. Although only one call per research task is required, it manages extremely large contexts (200K+ tokens) and represents the highest quality requirement, as its output is the ultimate output of the research workflow and is consumed by end users, justifying a higher computational expense.

To achieve a balance between cost and performance, the baseline agent was configured to use the following models:[1]

1. Research Model — OpenAI GPT-4o Mini ($0.15/M input tokens, $0.60/M output tokens): GPT-4o Mini was chosen as the research model for its exceptional structured output fidelity and robust tool-calling capabilities. The model supports a 128K token context window and can generate outputs of up to 16.4K tokens, which is sufficient for the iterative reasoning required during research task decomposition and strategy planning. OpenAI models are particularly well-suited for agentic workflows due to their reliable adherence to structured JSON output schemas and consistent parallel tool call execution, which is critical when coordinating multiple Tavily search queries and managing complex multi-step research plans. While slightly more expensive than some alternatives, GPT-4o Mini's superior instruction-following and reasoning capabilities justify its selection for the high-stakes research coordination role, where reliability directly impacts downstream workflow quality.

2. Summarisation Model — Google Gemini 2.5 Flash Lite ($0.10/M input tokens, $0.40/M output tokens): Gemini 2.5 Flash Lite was selected as the summarisation model primarily for its exceptional throughput and massive context window (1.05M tokens) at the lowest cost among the models used. Its architecture is optimised for fast token generation and low-latency processing, which is essential for meeting the 60-second timeout constraint per summary when processing multiple web pages in parallel. The model's ability to rapidly ingest large volumes of raw webpage content and extract structured summaries makes it ideal for the high-volume summarisation workload, which represents the second-largest cost driver in the system. By leveraging Gemini 2.5 Flash Lite's speed and cost-efficiency for this parallelisable task, the

[1] Note that all models were accessed via OpenRouter, a platform that provides centralised access to a wide range of LLM APIs from multiple providers. Accordingly, the context windows and pricing reported correspond to OpenRouter's offerings, which may not match the models' native specifications and may evolve over time. All pricing is in USD.

system can process numerous search results simultaneously without incurring prohibitively high token costs or exceeding time constraints.

3. Compression Model — Google Gemini 2.0 Flash Lite ($0.075/M input tokens, $0.30/M output tokens): Gemini 2.0 Flash Lite was chosen to consolidate outputs from the researcher sub-agents due to its rapid time to first token and cost-efficient pricing. Although it handles a large context of 1.05M tokens, it is invoked only once per researcher, and its reduced execution time ensures that compression does not become a major cost factor. The model is capable of producing high-quality structured summaries with reliable deduplication and citation integrity, closely matching the performance of larger Gemini Pro models. This makes it a practical choice that balances quality with minimal computational overhead.

4. Final Report Model — OpenAI GPT-5 Mini ($0.25/M input tokens, $2/M output tokens): GPT-5 Mini was selected for final report generation to ensure the highest fidelity output with advanced synthesis and reasoning capabilities. Its context window of 400K tokens and maximum output of 128K tokens are sufficient to integrate compressed research from multiple sub-agents. While it is more expensive per token compared to the other models, the one call per research task limits total cost, making it a justified investment for producing the ultimate research deliverable. GPT-5 Mini's instruction-following, safety-tuning, and reduced latency relative to larger GPT-5 variants ensure that the final report meets quality standards without incurring unnecessary computational expense.

Collectively, this model configuration leverages lightweight but capable models for high-volume operations (research, summarisation, compression) and reserves a higher-cost, high-performance model for the critical final report. This approach optimally balances the competing demands of performance, context capacity, speed, and cost across the full research workflow. A summary of the model pricing and characteristics is presented in Table 3.4.

TABLE 3.4: Model costs and context window sizes for the baseline system configuration.

Role	**Model**	**Input Cost** ($/M tokens)	**Output Cost** ($/M tokens)	**Context Window**
Research Model	OpenAI GPT-4o Mini	$0.15	$0.60	128K tokens
Summarisation Model	Google Gemini 2.5 Flash Lite	$0.10	$0.40	1.05M tokens
Compression Model	Google Gemini 2.0 Flash Lite	$0.075	$0.30	1.05M tokens
Final Report Model	OpenAI GPT-5 Mini	$0.25	$2.00	400K tokens

3.8 Evaluation Framework

To quantitatively assess whether the proposed architectural modifications discussed in Section 3.3 genuinely improve research quality rather than merely shifting behaviour, the system must be evaluated under controlled, repeatable conditions. Qualitative case studies are insufficient on their own: deep research agents operate over long horizons, consume substantial context, and interact with external tools, making it necessary to measure factual accuracy, reasoning robustness, and citation faithfulness using standardised tasks and metrics. A principled benchmark also enables comparison against state-of-the-art commercial and open-source systems, providing an external reference point for both performance and cost-efficiency. For this thesis, DeepResearch Bench [15] was selected as the quantitative evaluation framework.

3.8.1 DeepResearch Bench

DeepResearch Bench consists of 100 complex, PhD-level research tasks across 22 domains, meticulously curated by experts to reflect real user needs in real-world scenarios [15]. DeepResearch Bench is the first specialised benchmark for evaluating DRAs; unlike traditional LLM benchmarks that assess isolated capabilities such as simple question answering or code generation, it targets the complex, end-to-end workflow of autonomous research. This includes multi-step web exploration, information retrieval, and the synthesis of analyst-grade reports. The tasks were collected by first determining a topic-distributed target number of 96,000 real-world user queries, then compressing this distribution to 100 high-quality, challenging tasks (50 Chinese, 50 English) proposed by domain experts and manually vetted for complexity and authenticity.

The DeepResearch Bench leaderboard [16] reports comparative performance scores for DeepMind, OpenAI, Anthropic, Grok, and Perplexity's proprietary deep research products. It has attracted increasing attention within the deep research community by also incorporating submissions from both proprietary and open-source implementations developed by organisations such as Salesforce, Moonshot AI, and Alibaba Cloud. LangChain's Open Deep Research appears on the leaderboard with three separate submissions corresponding to different model and search API configurations.

3.9 Evaluation Methodology

Evaluating the output of DRAs is challenging due to the open-ended nature of research reports and the lack of a single "ground truth." To address this, DeepResearch Bench introduces two human-aligned evaluation methodologies that assess complementary aspects of agent performance:

1. **RACE (Reference-based Adaptive Criteria-driven Evaluation):** A framework focused on assessing the textual quality, depth, and structure of the generated research reports.
2. **FACT (Framework for Factual Abundance and Citation Trustworthiness):** A framework focused on evaluating the agent's ability to retrieve relevant information and accurately cite sources.

3.9.1 RACE Framework

The RACE framework utilises an LLM-as-a-Judge approach to evaluate the quality of long-form research reports. It addresses the limitations of static rubrics by dynamically generating task-specific criteria and

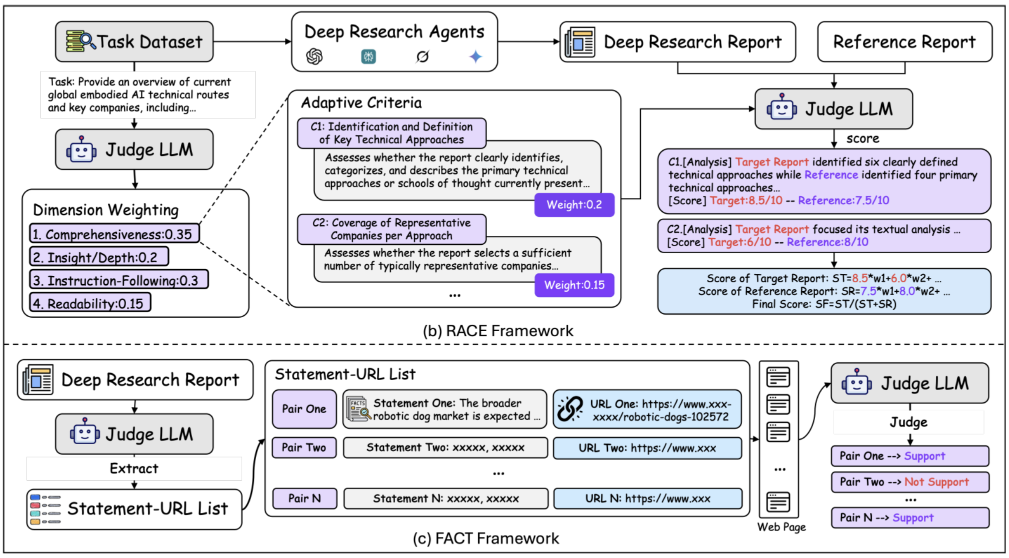


FIGURE 3.20: Overview of RACE and FACT Evaluation Methodologies [15].

employing a reference-based scoring system.

Evaluation Dimensions

RACE evaluates reports across four top-level orthogonal dimensions:

- **Comprehensiveness (COMP):** Assesses coverage of key areas and overall understanding.
- **Insight/Depth (DEPTH):** Evaluates the analysis of causes, impacts, and trends.
- **Instruction-Following (INST):** Checks adherence to the research topic and specific user constraints.
- **Readability (READ):** Reviews structure, language fluency, and clarity.

Dynamic Weighting and Adaptive Criteria

To align evaluation with the specific intent of a task t, the Judge LLM first derives task-specific dimension weights. For each dimension $d \in \{COMP, DEPTH, INST, READ\}$, the weight W_d is calculated by averaging weights $w_d^{(j)}$ obtained over T trials:

$$W_d = \frac{1}{T} \sum_{j=1}^{T} w_d^{(j)}$$

Subsequently, for each dimension, the Judge LLM generates a set of tailored criteria $\{c_{d,k}\}$ and corresponding weights $\{w_{d,k}\}$ (where $\sum_{k=1}^{K_d} w_{d,k} = 1$) to ensure the evaluation is specific to the unique requirements of the research prompt.

Reference-Based Scoring Calculation

RACE mitigates score inflation by comparing the target agent's report (R_{tgt}) against a high-quality reference report (R_{ref}). The scoring process proceeds as follows:

1. **Criterion scoring.** The Judge LLM evaluates both R_{tgt} and R_{ref} against the generated criteria, yielding scores $s_{R,c_{d,k}}$.

2. **Dimension calculation.** Dimension-level scores $S_d(R)$ are calculated by weighting the criterion-level scores.

3. **Intermediate scoring.** These scores are aggregated using the dimension weights W_d to produce an intermediate overall score $S_{\text{int}}(R)$.

4. **Final relative score.** The final quality score $S_{\text{final}}(R_{tgt})$ is computed as a relative value to normalise results:
$$S_{\text{final}}(R_{tgt}) = \frac{S_{\text{int}}(R_{tgt})}{S_{\text{int}}(R_{tgt}) + S_{\text{int}}(R_{ref})}$$

The RACE evaluation produces five scores for each task in the output `race_result.jsonl`: individual dimension scores for Comprehensiveness, Insight/Depth, Instruction-Following, and Readability, each normalised relative to the reference report, alongside an `overall_score`.

Perception of RACE Scores

The RACE score should be interpreted as a relative indicator of proficiency rather than an absolute, additive value unit. Since the scoring formula derives the final value by comparing a target report against a high-quality reference standard, the result represents a ratio of performance rather than a cumulative grade [15]. As a result, the scores are best understood through rankings and proportional differences; for example, small numerical gaps often signify meaningful distinctions in quality that align closely with human expert consensus. Scores of 45 and 50 can be compared to human scores of 90 and 100 [15]. If the target report is equally as good as the high-quality reference, the formula becomes $X/(X + X) = 0.5$, resulting in a score of 50. To score significantly higher than 50, the target report would have to be vastly superior to the state-of-the-art reference. Given that the reference is already a top-tier output, scores significantly exceeding 50 are rare in practice. Although the metric is grounded in qualitative assessments, it functions as a linear grading on a curve, where the value reflects the report's proximity to an expert baseline.

Mitigating High Variance in LLM Judges

To counteract the inherent instability and high variance associated with LLM-based evaluations, the RACE framework employs a comparative anchoring mechanism. When AI models evaluate texts in isolation, they often default to uniformly high scores, failing to discriminate effectively between varying levels of quality. RACE resolves this by forcing the model to evaluate the target report alongside a reference report simultaneously, ensuring that any bias affects both texts equally and cancels out. By decomposing the evaluation into specific, granular criteria and averaging the dimension weights across multiple trials, the system stabilises the scoring process, ultimately achieving consistency rates that are highly linearly correlated with human evaluations.

3.9.2 FACT Framework

The FACT framework is designed to assess the factual grounding of report content and the reliability of web retrieval, also using a LLM-as-a-Judge approach. It automates the verification of citations to measure how effectively an agent utilises web-based information.

Methodology

The evaluation process involves three primary steps:

1. **Extraction and Deduplication:** A Judge LLM extracts discrete Statement-URL pairs from the generated report. Multiple statements citing the same URL for the same fact are deduplicated to ensure unique factual claims are counted only once.
2. **Content Retrieval:** The textual content of the cited webpage is retrieved using Jina AI's reader API, which extractis the core content from a URL and converts it into clean, LLM-friendly text.
3. **Support Judgment:** A Judge LLM compares the statement against the retrieved webpage content. It renders a binary judgment of *support* or *not support*, determining if the source provides sufficient evidence for the claim.

Scoring Metrics

FACT calculates two key metrics to quantify performance:

Citation Accuracy (C. Acc.): This metric measures the precision of the agent's citations. For a given task t, let $N_{u,t}$ be the total number of unique statement-URL pairs, and $N_{s,t}$ be the number of pairs judged as 'support'. The accuracy for task t, denoted as Acc_t, is:

$$Acc_t = \begin{cases} \frac{N_{s,t}}{N_{u,t}} & \text{if } N_{u,t} > 0 \\ 0 & \text{if } N_{u,t} = 0 \end{cases}$$

The overall Citation Accuracy is the average across all tasks in the benchmark set T:

$$C.Acc. = \frac{1}{|T|} \sum_{t \in T} Acc_t$$

Average Effective Citations (E. Cit.): This metric quantifies the volume of valuable, verified information retrieved. It is calculated by summing the total number of supported pairs across all tasks and dividing by the total number of tasks:

$$E.Cit. = \frac{\sum_{t \in T} N_{s,t}}{|T|}$$

The FACT evaluation outputs three aggregate metrics in `fact_result.txt`: `total_citations` representing the mean number of unique citation URLs per task, `total_valid_citations` indicating the average count of citations with at least one supported claim per task, and `valid_rate` computed as the ratio of valid to total citations across all evaluated tasks.

Corrected Evaluation Logic

During the course of this research, a critical flaw was identified in the original FACT evaluation implementation that systematically penalised agents with more granular claim extraction. The original logic incorrectly counted individual claim-level validation results (each statement within a citation) rather than unique citation URLs, thereby inflating the denominator in the accuracy calculation for agents that extracted multiple verifiable statements from a single source. This bias is evidenced by the DeepResearch Bench leaderboard, where deep research agents such as the Langchain Open Deep Research GPT-5 submission achieved anomalously low Citation Accuracy scores of 34.74%, while LLM-with-search approaches such as Claude 3.5 Sonnet attained 94.06% [16]. The corrected implementation adopts citation-level evaluation, wherein each unique URL is counted once and deemed valid if any associated claim is supported by the retrieved content, reducing the evaluation bias.

3.9.3 Configuration

While the original implementation utilises Gemini-2.5-pro as the Judge LLM for RACE evaluation tasks and Gemini-2.5-flash for the FACT framework's statement-URL pair extraction and support judgment, this thesis employs Gemini-2.5-flash for both the RACE and FACT frameworks to reduce computational costs as shown in Table 3.5. This decision is justified by the substantial cost differential between the two models, with Gemini-2.5-pro priced at $1.25 per million input tokens and $10.00 per million output tokens, compared to Gemini-2.5-flash at $0.30 per million input tokens and $2.50 per million output tokens. Given that Gemini-2.5-flash demonstrates sufficient capabilities for the token-intensive citation verification tasks whilst offering greater economic efficiency, it represents a pragmatic choice for large-scale evaluation. Furthermore, the tasks are configured to utilise only the English subset of the dataset, comprising 50 of the 100 total queries, by setting the `ONLY_EN="--only_en"` parameter. This achieves an approximate half reduction in costs while maintaining a sufficient sample size.

Conclusion

The dual framework approach of RACE and FACT enables targeted evaluation of the proposed architectural enhancements by measuring distinct aspects of the final research report. The Research Reviewer component,

TABLE 3.5: Model costs and context window sizes for the DeepResearch Bench evaluation models.

Role	Model	Input Cost ($/M tokens)	Output Cost ($/M tokens)	Context Window
RACE Model	Google Gemini 2.5 Flash	$0.30	$2.50	1.05M tokens
FACT Model	Google Gemini 2.5 Flash	$0.30	$2.50	1.05M tokens

designed to enhance report quality through iterative self-refinement, is principally evaluated via the RACE framework, as improvements in comprehensiveness, depth, instruction-following, and readability directly reflect the component's capacity to elevate research output quality through critical analysis and structured revision. Conversely, the Research Verifier component is assessed through the FACT framework, as this framework directly measures the alignment between stated claims and their supporting evidence through automated verification of statement-URL pairs against retrieved webpage content, thereby providing a quantitative measure of the component's core function. Finally, the Adaptive Model Selection mechanism requires evaluation across both frameworks: given that its objective is to optimise resource allocation without compromising output integrity, both RACE and FACT are necessary to ensure that strategic model substitution preserves fidelity across the dual concerns of report quality and citation reliability, with any degradation in either dimension indicating suboptimal model selection policies that sacrifice correctness for efficiency.

3.10 Experiment Design

The experimental evaluation is structured to rigorously assess the performance determinants and architectural contributions of the proposed Deep Research system. This design comprises two distinct components: a two-factor factorial study to explore the effects of scaling behaviour and model intelligence, and an ablation study to isolate the impact of each architectural component from Section 3.3. Together, these analyses provide a comprehensive understanding of how operational parameters and structural innovations drive research reliability and efficiency.

First, a two-factor factorial study is employed to investigate the independent and interactive effects of behavioural scaling and model capability enhancement on research reliability and cost-efficiency. This approach addresses a fundamental question in Deep Research architecture: whether improvements in output quality and citation accuracy are primarily driven by operational parameters that govern agent behaviour or by the capabilities of the underlying language models themselves. By evaluating four distinct system configurations that permute these two dimensions, the research design enables decomposition of performance outcomes into contributions attributable to behavioural orchestration versus raw model intelligence, while simultaneously revealing potential interaction effects wherein behavioural scaling may amplify or weaken the benefits of more capable models. These findings are essential for establishing evidence-based guidelines for resource allocation in production deployments.

Table 3.6 details the four primary system configurations employed in this evaluation. The Base configuration establishes a reference deployment with conservative operational constraints and a model hierarchy emphasising computational efficiency through lightweight variants. The Base Maxed Behaviour configuration isolates the effect of behavioural scaling by increasing operational limits while maintaining the Base model hierarchy. Conversely, the Base Maxed Models configuration isolates the effect of enhanced model capabilities by upgrading to more sophisticated GPT models but keeping the Base behavioural parameters. Finally, the Base Maxed Behaviour & Models configuration combines both scaling dimensions, operating with maximal behavioural parameters and the most capable model hierarchy to establish an upper performance bound.

TABLE 3.6: Experimental system configurations and their defining parameters.

	System Configuration			
Parameter	**Base**	**Base Maxed Behaviour**	**Base Maxed Models**	**Base Maxed Behaviour & Models**
Max Concurrent Research Units	3	10	3	10
Max Researcher Iterations	3	6	3	6
Max ReAct Tool Calls	6	10	6	10
Summarisation Model	`Gemini 2.5 Flash-Lite`	`Gemini 2.5 Flash-Lite`	`GPT-4.1 Mini`	`GPT-4.1 Mini`
Research Model	`GPT-4o Mini`	`GPT-4o Mini`	`GPT-5`	`GPT-5`
Compression Model	`Gemini 2.0 Flash-Lite`	`Gemini 2.0 Flash-Lite`	`GPT-4.1`	`GPT-4.1`
Final Report Model	`GPT-5 Mini`	`GPT-5 Mini`	`GPT-5`	`GPT-5`

Table 3.7 provides the associated costs and context window sizes for the models employed in the maxed model configurations.

TABLE 3.7: Model costs and context window sizes for the maxed models and maxed system configurations.

Role	**Model**	**Input Cost ($/M tokens)**	**Output Cost ($/M tokens)**	**Context Window**
Research Model	OpenAI GPT-5	$1.25	$10.00	400K tokens
Summarisation Model	OpenAI GPT-4.1 Mini	$0.40	$1.60	1.05M tokens
Compression Model	OpenAI GPT-4.1	$2.00	$8.00	1.05M tokens
Final Report Model	OpenAI GPT-5	$1.25	$10.00	400K tokens

Second, an ablation methodology evaluates the individual contributions of each proposed architectural enhancement. The experimental setup comprises four distinct system configurations: the baseline system without modifications, the baseline augmented with Research Reviewer only, the baseline augmented with Research Verifier only, and the baseline augmented with Adaptive Model Selection only. This design enables precise attribution of performance changes to each specific architectural contribution while controlling for confounding variables. Table 3.8 details the ablation configurations, wherein behavioural parameters and model selections remain constant across all variants while the architectural components are selectively activated. This systematic approach ensures that observed improvements in report quality, citation accuracy, or cost-efficiency can be confidently attributed to the specific architectural component under investigation.

Each system configuration is evaluated against the DeepResearch Bench benchmark, with each research workflow executed once and each research evaluation conducted twice and averaged. While increased

TABLE 3.8: Ablation study configurations. All configurations maintain identical behavioural parameters and model selections, differing only in the presence of architectural enhancements.

	System Configuration			
Parameter	**Baseline**	**+ Research Reviewer**	**+ Research Verifier**	**+ Adaptive Model Selection**
Behavioural Parameters				
Max Concurrent Research Units	3	3	3	3
Max Researcher Iterations	3	3	3	3
Max ReAct Tool Calls	6	6	6	6
Model Configuration				
Summarisation Model	`Gemini 2.5 Flash-Lite`	`Gemini 2.5 Flash-Lite`	`Gemini 2.5 Flash-Lite`	`Gemini 2.5 Flash-Lite`
Research Model	`GPT-4o Mini`	`GPT-4o Mini`	`GPT-4o Mini`	`GPT-4o Mini`
Compression Model	`Gemini 2.0 Flash-Lite`	`Gemini 2.0 Flash-Lite`	`Gemini 2.0 Flash-Lite`	`Gemini 2.0 Flash-Lite`
Final Report Model	`GPT-5 Mini`	`GPT-5 Mini`	`GPT-5 Mini`	`GPT-5 Mini`
Architectural Components				
Research Reviewer	–	✓	–	–
Research Verifier	–	–	✓	–
Adaptive Model Selection	–	–	–	✓

replication is desirable, the large costs of running single benchmark evaluations make this impractical.

3.11 Expected Results

This thesis posits that integrating a Research Reviewer for quality enhancement, a Research Verifier for citation validation, and an Adaptive Model Selection algorithm will meaningfully enhance the reliability and efficiency of Deep Research agents. This can be translated to quantifiable improvements in the RACE and FACT scores of the DeepResearch Bench benchmark.

This section expresses the anticipated outcomes of the proposed architectural modifications based on their theoretical foundations and targeted limitations. Each modification is expected to demonstrate measurable improvements aligned with its primary design objective while maintaining performance across other evaluation dimensions.

Research Reviewer

The Research Reviewer component is expected to demonstrate measurable improvements in RACE dimensions, particularly in the Comprehensiveness and Insight scores, as the iterative review-and-revision cycle should enhance analytical rigor and coverage breadth. These improvements should manifest as an increased overall RACE score relative to the baseline configuration, validating that self-critique mechanisms can systematically elevate report quality.

Research Verifier

The Research Verifier component is expected to demonstrate a substantial increase in Citation Accuracy by systematically detecting and eliminating unsupported claims before the report is finalised, while maintaining the Effective Citation volume, through a rigorous source validation process. This dual outcome would demonstrate that citation verification can enhance factual reliability without sacrificing the informational richness of the research output.

Adaptive Model Selection

The Adaptive Model Selection mechanism is expected to achieve comparable RACE and FACT scores to the baseline configuration while demonstrating significant cost reductions through strategic allocation of less expensive models to subtasks that do not require maximum reasoning capacity and increased token usage by more advanced models. This outcome would validate that intelligent model routing can preserve output quality across both report coherence and citation reliability dimensions whilst substantially reducing the computational expenditure per research task.

CHAPTER 4

Results

This chapter presents the empirical findings from the experimental design outlined in Section **??**. The evaluation comprises 50 English-language research tasks spanning 22 domains from the DeepResearch Bench benchmark, with each system being assessed using the RACE framework for report quality, the FACT framework for citation accuracy, or both. All RACE and FACT scores represent averages across two independent evaluation runs to ensure measurement reliability, as detailed in Section 4.9.1. While tables present RACE scores in their native 0–1 range, figures display scores scaled to 0–100 for visual clarity.

The results are organised to reflect the two-part experimental design. The first part addresses the factorial configuration study, which decomposes performance outcomes into contributions attributable to behavioural orchestration versus model capability. [1] Section 4.1 establishes the Base configuration as the reference deployment, followed by Sections 4.2–4.4 which present results for the three experimental configurations:

1. **Maxed Behaviour:** isolating the effect of behavioural scaling;

2. **Maxed Models:** isolating the effect of model capability enhancement; and

3. **Maxed Behaviour & Models:** combining both scaling dimensions to establish an upper performance bound.

The second part addresses the ablation study, evaluating the individual contributions of each proposed architectural enhancement. Sections 4.5–4.7 present results for each ablation configuration, wherein behavioural parameters and model selections remain constant while architectural components are selectively activated. A comprehensive comparative analysis is then presented in Section 4.8, synthesising findings across both experimental tracks. The Research Reviewer considers only RACE scores for report quality, the Research Verifier considers only FACT scores for citation accuracy, and the Adaptive Model Selection considers both dimensions.

Cost metrics provide insight into the computational efficiency of each configuration. Total cost is calculated as the sum of model API cost and search API cost. Model API costs are determined by tracking OpenRouter credits before and after report generation, while search API costs are determined by tracking Tavily credits before and after report generation, with the credit difference multiplied by the cost per credit on the Bootstrap plan ($0.0067 USD per credit). Token counts were extracted from the Langsmith Studio, a

[1] For brevity, system configurations are referred to as "agents", though each represents a complete multi-agent system rather than an individual agent component.

specialised AI agent interactive development environment used to deploy the configured and implemented systems.

4.1 Baseline Agent

The baseline agent configuration, established through systematic optimisation of system prompts, behavioural parameters, and cost-efficient model selection as described in Section 3.7, serves as the foundational reference point for all subsequent comparative analyses.

Table 4.1 presents the aggregate RACE scores across all dimensions. The baseline agent achieved an overall score of 0.4697, with notable variation across dimensions. Instruction Following attained the highest score at 0.4899, while Insight recorded the lowest at 0.4603. Comprehensiveness and Readability scored 0.4618 and 0.4753 respectively, positioning between these extremes. These scores establish the baseline performance envelope for report quality against which all subsequent modifications are measured.

TABLE 4.1: Baseline Agent RACE Scores

Metric	Comp	Ins	Inst	Read	Overall
Value	0.4618	0.4603	0.4899	0.4753	0.4697

The per-task performance distribution, presented in Table 4.2, reveals variability in agent performance across the 50 evaluation tasks. Overall scores ranged from a minimum of 0.417 (Task 3) to a maximum of 0.505 (Task 14), spanning approximately 0.088 points.

Table 4.3 provides detailed statistical characterisation of the baseline agent's score distributions across all RACE dimensions. Standard deviations ranged from 0.0123 (Instruction Following) to 0.0262 (Comprehensiveness), indicating that Instruction Following exhibited the most consistent performance while Comprehensiveness demonstrated the greatest variability. Median values aligned closely with means across all dimensions, with maximum deviations of 0.0032, suggesting approximately symmetric distributions without substantial skew. The interquartile ranges further quantify the central tendency, with Instruction Following exhibiting the tightest IQR of 0.0139 and Insight the widest at 0.0381.

Table 4.4 reports the FACT evaluation metrics assessing citation quality and source retrieval effectiveness. The baseline agent generated an average of 15.06 unique citations per task, of which 11.53 were validated as supporting their associated claims, yielding a valid rate of 76.66%. This citation accuracy baseline provides the reference standard for evaluating whether architectural modifications impact the agent's ability to ground research outputs in verifiable sources.

The operational metrics in Table 4.5 quantify the computational resources consumed by the baseline configuration. The agent executed 1,774 Tavily search API calls across the 50-task evaluation set, averaging

TABLE 4.2: Per-Task RACE Scores (Baseline Agent)

ID	Comp	Ins	Inst	Read	Ovr	ID	Comp	Ins	Inst	Read	Ovr
1	0.449	0.456	0.497	0.506	0.472	26	0.471	0.464	0.486	0.477	0.473
2	0.426	0.427	0.500	0.479	0.447	27	0.451	0.451	0.494	0.473	0.462
3	0.369	0.410	0.480	0.451	0.417	28	0.470	0.451	0.499	0.479	0.474
4	0.482	0.476	0.501	0.503	0.486	29	0.447	0.444	0.475	0.482	0.457
5	0.464	0.466	0.495	0.477	0.473	30	0.460	0.474	0.500	0.481	0.479
6	0.481	0.468	0.500	0.492	0.481	31	0.455	0.469	0.491	0.474	0.471
7	0.481	0.481	0.500	0.460	0.482	32	0.486	0.482	0.489	0.469	0.483
8	0.468	0.459	0.490	0.474	0.469	33	0.483	0.498	0.491	0.475	0.488
9	0.450	0.439	0.476	0.451	0.451	34	0.438	0.452	0.498	0.471	0.462
10	0.472	0.501	0.500	0.470	0.488	35	0.439	0.433	0.461	0.465	0.445
11	0.516	0.496	0.494	0.485	0.499	36	0.491	0.458	0.502	0.445	0.476
12	0.436	0.426	0.473	0.467	0.444	37	0.454	0.467	0.498	0.488	0.473
13	0.479	0.455	0.500	0.481	0.473	38	0.466	0.437	0.485	0.489	0.462
14	0.511	0.504	0.515	0.485	0.505	39	0.478	0.479	0.493	0.464	0.481
15	0.476	0.492	0.501	0.479	0.490	40	0.502	0.487	0.515	0.492	0.498
16	0.475	0.474	0.491	0.476	0.479	41	0.439	0.395	0.468	0.471	0.451
17	0.466	0.479	0.492	0.476	0.477	42	0.439	0.435	0.496	0.463	0.453
18	0.464	0.463	0.493	0.483	0.473	43	0.461	0.459	0.490	0.486	0.470
19	0.452	0.435	0.456	0.488	0.453	44	0.475	0.466	0.490	0.471	0.475
20	0.468	0.479	0.493	0.476	0.478	45	0.394	0.439	0.459	0.459	0.435
21	0.463	0.464	0.483	0.498	0.472	46	0.473	0.470	0.500	0.467	0.478
22	0.474	0.449	0.486	0.480	0.470	47	0.497	0.494	0.495	0.486	0.494
23	0.479	0.507	0.495	0.493	0.495	48	0.431	0.440	0.485	0.454	0.450
24	0.451	0.440	0.480	0.437	0.451	49	0.428	0.435	0.487	0.458	0.448
25	0.454	0.444	0.476	0.478	0.459	50	0.459	0.447	0.485	0.485	0.463

TABLE 4.3: RACE Score Distribution Statistics (Baseline Agent)

Dimension	Mean	Median	Std Dev	Min	Max	Q1	Q3	IQR
Comprehensiveness	0.4618	0.4650	0.0262	0.3685	0.5156	0.4498	0.4779	0.0281
Insight	0.4603	0.4612	0.0245	0.3949	0.5071	0.4404	0.4785	0.0381
Instruction Following	0.4899	0.4924	0.0123	0.4560	0.5150	0.4847	0.4986	0.0139
Readability	0.4753	0.4762	0.0142	0.4371	0.5056	0.4668	0.4847	0.0179
Overall Score	**0.4697**	**0.4729**	**0.0177**	**0.4168**	**0.5054**	**0.4572**	**0.4807**	**0.0235**

TABLE 4.4: Baseline Agent FACT Scores

Metric	Total citations	Total valid citations	Valid rate (%)
Value	15.06	11.53	76.66

35.48 searches per task. Token consumption totalled 62.28M (million) tokens, with input tokens comprising 58.50M and output tokens 3.78M. The total cost of \$29.95 was distributed between model API costs (\$18.06) and search API costs (\$11.89), establishing the baseline cost reference for subsequent efficiency comparisons.

4.2 Maxed Behaviour Agent

The Maxed Behaviour configuration isolates the impact of relaxing behavioural constraints while maintaining the baseline model selection. This variant increased the maximum concurrent research units from 3 to 10, the maximum researcher iterations from 3 to 6, and the maximum ReAct tool calls per researcher from 6 to 10, effectively expanding the agent's capacity for parallel research delegation and iterative information

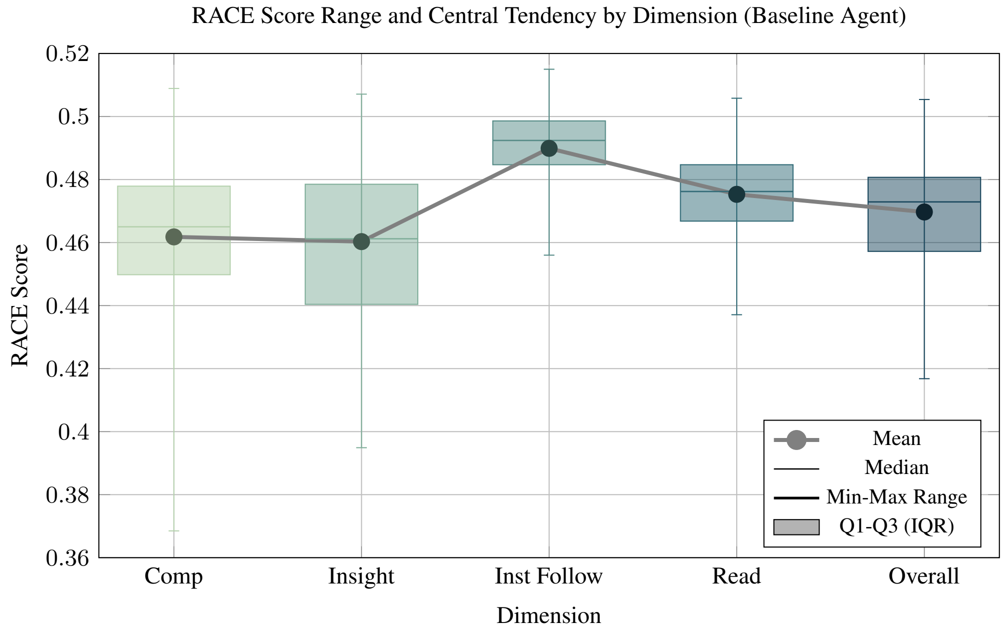


FIGURE 4.1: Statistical summary visualisation showing mean, median, IQR, and full range for each RACE dimension (Baseline Agent).

TABLE 4.5: Operational Metrics (Baseline Agent)

Metric	**Value**
Total Tavily search calls	1,774
Average searches per task	35.48
Total tokens	62.28M
Total input tokens	58.50M
Total output tokens	3.78M
Model API cost (USD)	$18.06
Search API cost (USD)	$11.89
Total cost (USD)	**$29.95**

gathering.

Table 4.6 presents the RACE scores for this configuration. The agent achieved an overall score of 0.4731, representing a 0.0034 increase over the baseline's 0.4697. Dimension-level performance exhibited modest gains: Comprehensiveness increased to 0.4661 (+0.0043), Insight to 0.4661 (+0.0058), Readability to 0.4773 (+0.0020), while Instruction Following remained essentially unchanged at 0.4900 (+0.0001).

TABLE 4.6: Maxed Behaviour Agent RACE Scores

Metric	**Comp**	**Ins**	**Inst**	**Read**	**Overall**
Value	0.4661	0.4661	0.4900	0.4773	0.4731

Per-task performance distribution is detailed in Table 4.7, with overall scores spanning from 0.432 (Task 3) to 0.500 (Task 4), a range comparable to the baseline configuration.

TABLE 4.7: Per-Task RACE Scores (Maxed Behaviour Agent)

ID	Comp	Ins	Inst	Read	Ovr	ID	Comp	Ins	Inst	Read	Ovr
1	0.441	0.450	0.491	0.483	0.463	26	0.461	0.466	0.483	0.469	0.470
2	0.441	0.449	0.499	0.485	0.461	27	0.481	0.476	0.497	0.490	0.483
3	0.397	0.420	0.483	0.472	0.432	28	0.460	0.433	0.467	0.467	0.456
4	0.491	0.507	0.501	0.499	0.500	29	0.480	0.482	0.489	0.495	0.485
5	0.454	0.459	0.492	0.469	0.466	30	0.444	0.482	0.500	0.487	0.478
6	0.462	0.472	0.491	0.479	0.474	31	0.478	0.486	0.502	0.484	0.488
7	0.480	0.491	0.498	0.490	0.489	32	0.489	0.478	0.501	0.477	0.486
8	0.486	0.459	0.490	0.483	0.476	33	0.488	0.501	0.486	0.441	0.484
9	0.474	0.454	0.496	0.464	0.470	34	0.436	0.463	0.482	0.485	0.463
10	0.450	0.461	0.492	0.482	0.467	35	0.445	0.426	0.488	0.469	0.451
11	0.451	0.464	0.487	0.461	0.465	36	0.499	0.491	0.507	0.462	0.493
12	0.447	0.444	0.487	0.474	0.457	37	0.475	0.493	0.498	0.492	0.489
13	0.453	0.429	0.500	0.453	0.452	38	0.463	0.469	0.489	0.495	0.474
14	0.468	0.470	0.482	0.475	0.473	39	0.468	0.463	0.484	0.456	0.469
15	0.491	0.504	0.500	0.478	0.498	40	0.478	0.476	0.518	0.486	0.486
16	0.518	0.493	0.500	0.481	0.500	41	0.476	0.451	0.491	0.487	0.481
17	0.478	0.495	0.488	0.479	0.487	42	0.459	0.450	0.495	0.457	0.463
18	0.470	0.467	0.498	0.489	0.478	43	0.443	0.448	0.486	0.467	0.457
19	0.448	0.443	0.465	0.489	0.456	44	0.475	0.468	0.484	0.462	0.473
20	0.446	0.439	0.474	0.466	0.454	45	0.447	0.448	0.487	0.462	0.460
21	0.483	0.499	0.501	0.501	0.495	46	0.477	0.481	0.500	0.493	0.487
22	0.461	0.449	0.430	0.470	0.451	47	0.500	0.485	0.495	0.481	0.492
23	0.460	0.493	0.475	0.502	0.484	48	0.438	0.426	0.481	0.484	0.451
24	0.476	0.478	0.497	0.445	0.476	49	0.454	0.453	0.500	0.475	0.467
25	0.477	0.451	0.496	0.483	0.474	50	0.488	0.466	0.476	0.488	0.477

Distribution statistics in Table 4.8 indicate that standard deviations remained comparable to baseline, with the overall score exhibiting a standard deviation of 0.0152 compared to baseline's 0.0177, demonstrating marginally reduced variability in aggregate performance.

TABLE 4.8: RACE Score Distribution Statistics (Maxed Behaviour Agent)

Dimension	Mean	Median	Std Dev	Min	Max	Q1	Q3	IQR
Comprehensiveness	0.4661	0.4680	0.0211	0.3970	0.5177	0.4497	0.4798	0.0301
Insight	0.4661	0.4661	0.0221	0.4204	0.5065	0.4503	0.4822	0.0319
Instruction Following	0.4900	0.4912	0.0133	0.4297	0.5177	0.4840	0.4992	0.0152
Readability	0.4773	0.4803	0.0141	0.4406	0.5022	0.4670	0.4874	0.0203
Overall Score	**0.4731**	**0.4738**	**0.0152**	**0.4324**	**0.5004**	**0.4627**	**0.4860**	**0.0233**

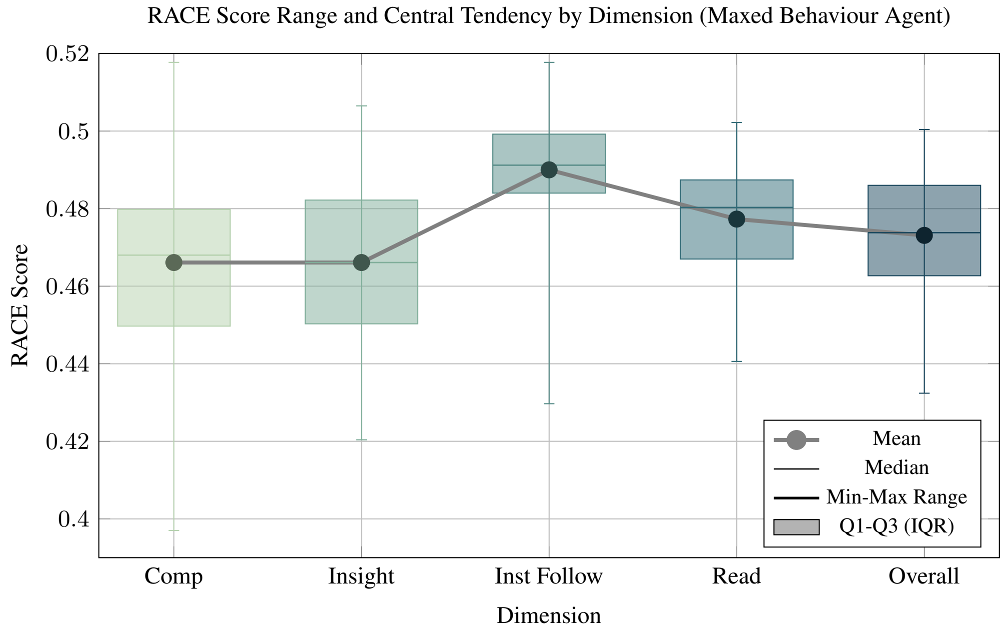


FIGURE 4.2: Statistical summary visualisation showing mean, median, IQR, and full range for each RACE dimension (Maxed Behaviour Agent).

The FACT evaluation results in Table 4.9 reveal the citation behaviour under expanded behavioural parameters. The configuration generated an average of 18.12 total citations per task, an increase of 3.06 citations (20.3%) over the baseline's 15.06. Valid citations increased to 13.66, a gain of 2.13 citations (18.5%), though the valid rate decreased marginally to 75.39% from the baseline's 76.66%. Therefore, there is a greater citation volume with a slight reduction in citation precision.

TABLE 4.9: Maxed Behaviour Agent FACT Scores

Metric	**Total citations**	**Total valid citations**	**Valid rate (%)**
Value	18.12	13.66	75.39

The operational cost of expanded behavioural parameters is quantified in Table 4.10. Search API calls increased substantially to 2,471 (averaging 49.42 per task), representing a 39.3% increase over baseline's 1,774 calls. Total token consumption rose to 87.92M tokens, a 41.2% increase from baseline's 62.28M. The total cost reached $40.14, distributed as $23.58 for model API costs and $16.56 for search API costs, representing a 34% cost increase relative to baseline for a 0.72% improvement in overall RACE score.

4.3 Maxed Models Agent

The Maxed Models configuration isolates the the impact of model quality by upgrading to higher-capacity models while maintaining baseline behavioural constraints. This variant employed GPT-5 as the research model (replacing GPT-4o Mini), GPT-4.1 Mini for summarisation (replacing Gemini 2.5 Flash Lite), GPT-4.1

TABLE 4.10: Operational Metrics (Maxed Behaviour Agent)

Metric	Value
Total Tavily search calls	2,471
Average searches per task	49.42
Total tokens	87.92M
Total input tokens	82.70M
Total output tokens	5.20M
Model API cost (USD)	$23.58
Search API cost (USD)	$16.56
Total cost (USD)	**$40.14**

for compression (replacing Gemini 2.0 Flash Lite), and GPT-5 for final report generation (replacing GPT-5 Mini).

Table 4.11 presents substantial performance improvements across most dimensions. The overall score reached 0.4810, a 0.0113 increase (+2.4%) over baseline's 0.4697. Comprehensiveness demonstrated the largest absolute gain at 0.4782 (+0.0164), while Insight improved to 0.4752 (+0.0149). Readability increased to 0.4800 (+0.0047), and Instruction Following remained stable at 0.4899 (+0.0000). These improvements represent the largest quality gains observed among the agent configuration variants.

TABLE 4.11: Maxed Models Agent RACE Scores

Metric	Comp	Ins	Inst	Read	Overall
Value	0.4782	0.4752	0.4899	0.4800	0.4810

Task 23 was excluded from the per-task analysis presented in Table 4.12 due to an early termination event that resulted in an anomalously low overall score of 0.23, and has consequently been omitted from all relevant calculations. The per-task scores for the remaining 49 tasks ranged from 0.417 (Task 33) to 0.535 (Task 15).

TABLE 4.12: Per-Task RACE Scores (Maxed Models Agent)

ID	Comp	Ins	Inst	Read	Ovr	ID	Comp	Ins	Inst	Read	Ovr
1	0.476	0.477	0.508	0.472	0.483	26	0.453	0.441	0.475	0.453	0.455
2	0.435	0.437	0.470	0.470	0.446	27	0.476	0.478	0.497	0.500	0.484
3	0.468	0.467	0.497	0.483	0.475	28	0.454	0.460	0.474	0.505	0.471
4	0.517	0.529	0.503	0.505	0.517	29	0.510	0.507	0.492	0.498	0.503
5	0.486	0.507	0.501	0.499	0.499	30	0.477	0.510	0.500	0.493	0.497
6	0.422	0.449	0.471	0.451	0.446	31	0.471	0.497	0.483	0.492	0.486
7	0.439	0.456	0.501	0.479	0.466	32	0.516	0.526	0.508	0.507	0.516
8	0.474	0.482	0.494	0.490	0.483	33	0.409	0.362	0.423	0.490	0.417
9	0.508	0.492	0.500	0.467	0.494	34	0.460	0.459	0.501	0.451	0.469
10	0.493	0.495	0.505	0.480	0.494	35	0.503	0.479	0.500	0.491	0.492
11	0.529	0.521	0.500	0.459	0.510	36	0.510	0.488	0.497	0.474	0.495
12	0.446	0.458	0.488	0.484	0.464	37	0.472	0.507	0.502	0.495	0.495
13	0.475	0.475	0.501	0.472	0.479	38	0.464	0.449	0.479	0.463	0.461
14	0.491	0.469	0.500	0.479	0.483	39	0.532	0.482	0.514	0.480	0.503
15	0.529	0.568	0.513	0.497	0.535	40	0.517	0.518	0.522	0.503	0.516
16	0.495	0.481	0.497	0.472	0.488	41	0.491	0.446	0.493	0.502	0.489
17	0.529	0.517	0.503	0.498	0.515	42	0.467	0.467	0.508	0.483	0.477
18	0.493	0.493	0.501	0.492	0.495	43	0.495	0.490	0.493	0.498	0.493
19	0.470	0.449	0.475	0.490	0.467	44	0.487	0.484	0.500	0.465	0.486
20	0.474	0.443	0.503	0.458	0.469	45	0.481	0.481	0.493	0.481	0.484
21	0.504	0.488	0.498	0.494	0.496	46	0.486	0.493	0.500	0.494	0.494
22	0.473	0.456	0.475	0.477	0.469	47	0.515	0.520	0.501	0.471	0.508
23	-	-	-	-	-	48	0.472	0.480	0.497	0.491	0.483
24	0.520	0.502	0.499	0.456	0.501	49	0.484	0.468	0.497	0.475	0.480
25	0.477	0.446	0.487	0.473	0.468	50	0.492	0.483	0.486	0.499	0.488

Distribution statistics in Table 4.13 reveal that standard deviations ranged from 0.0170 (Readability) to 0.0320 (Insight), indicating Readability exhibited the most consistent performance while Insight demonstrated the greatest variability. The overall score standard deviation of 0.0240 represents moderate variability compared to other agent configurations. Median values aligned closely with means across all dimensions, with maximum deviations of 0.0025, suggesting approximately symmetric distributions.

TABLE 4.13: RACE Score Distribution Statistics (Maxed Models Agent)

Dimension	Mean	Median	Std Dev	Min	Max	Q1	Q3	IQR
Comprehensiveness	0.4782	0.4825	0.0280	0.4090	0.5319	0.4700	0.5045	0.0345
Insight	0.4752	0.4815	0.0320	0.3620	0.5676	0.4590	0.4975	0.0385
Instruction Following	0.4899	0.4985	0.0220	0.4230	0.5221	0.4880	0.5015	0.0135
Readability	0.4800	0.4830	0.0170	0.4510	0.5074	0.4725	0.4950	0.0225
Overall Score	**0.4810**	**0.4860**	**0.0240**	**0.4170**	**0.5347**	**0.4700**	**0.4960**	**0.0260**

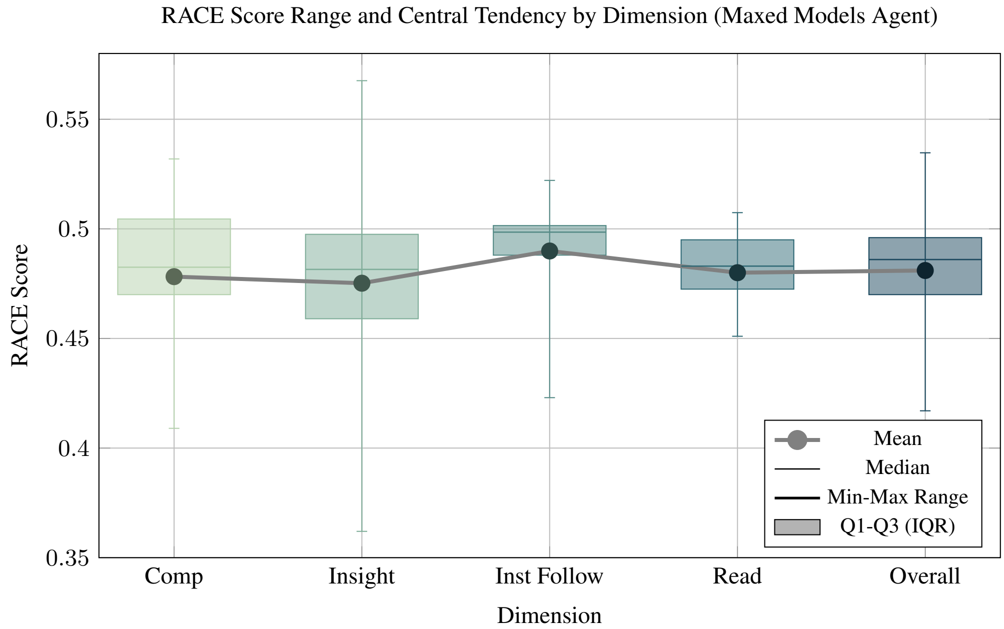


FIGURE 4.3: Statistical summary visualisation showing mean, median, IQR, and full range for each RACE dimension (Maxed Models Agent).

FACT evaluation metrics in Table 4.14 reveal that higher-capacity models yielded the highest citation valid rate observed across all agent configurations. The configuration generated 16.16 total citations per task, producing 12.80 valid citations with a valid rate of 79.21%, surpassing both baseline (76.66%) and Maxed Behaviour (75.39%).

TABLE 4.14: Maxed Models Agent FACT Scores

Metric	**Total citations**	**Total valid citations**	**Valid rate (%)**
Value	16.16	12.80	79.21

Operational metrics in Table 4.15 reveal a distinct resource utilisation pattern compared to prior configurations. Total Tavily search calls decreased to 1,286 (averaging 25.72 per task), representing a 27.5% reduction from baseline's 1,774 calls. Total token consumption decreased substantially to 39.47M tokens, a 36.6% reduction from baseline's 62.28M. However, model API costs increased significantly to $43.18, reflecting the higher per-token costs of premium models, while search API costs decreased to $8.62. The total cost of $51.80 represents a 73.0% increase over baseline, yielding a 2.4% improvement in overall RACE score at an effective model API cost of $1.09/M tokens compared to baseline's $0.29/M tokens.

4.4 Maxed Agent

The Maxed Agent configuration represents the simultaneous application of both maxed behavioural parameters and maxed model selection, combining the expanded search capacity of the Maxed Behaviour

TABLE 4.15: Operational Metrics (Maxed Models Agent)

Metric	**Value**
Total Tavily search calls	1,286
Average searches per task	25.72
Total tokens	39.47M
Total input tokens	36.50M
Total output tokens	3.00M
Model API cost (USD)	$43.18
Search API cost (USD)	$8.62
Total cost (USD)	**$51.80**

variant with the enhanced reasoning capabilities of the Maxed Models variant. This configuration provides an empirical upper bound on performance achievable within the experimental design space defined in Section **??**.

Table 4.16 demonstrates that combining both enhancements yielded the highest performance across all agent configuration variants. The overall score reached 0.4886, representing a 0.0189 improvement (+4.0%) over baseline and a 0.0076 improvement (+1.6%) over Maxed Models alone. Comprehensiveness achieved 0.4883 (+0.0265 over baseline), Insight reached 0.4842 (+0.0239), Instruction Following attained 0.4968 (+0.0069), and Readability measured 0.4837 (+0.0084). These results demonstrate additive effects from the two enhancement dimensions, with the combined configuration surpassing either individual enhancement.

TABLE 4.16: Maxed Agent RACE Scores

Metric	**Comp**	**Ins**	**Inst**	**Read**	**Overall**
Value	0.4883	0.4842	0.4968	0.4837	0.4886

TABLE 4.17: Per-Task RACE Scores (Maxed Agent)

ID	Comp	Ins	Inst	Read	Ovr	ID	Comp	Ins	Inst	Read	Ovr
1	0.466	0.474	0.478	0.460	0.470	26	0.476	0.467	0.500	0.452	0.476
2	0.460	0.444	0.500	0.485	0.465	27	0.462	0.461	0.492	0.484	0.470
3	0.463	0.455	0.501	0.484	0.470	28	0.498	0.495	0.510	0.508	0.502
4	0.484	0.492	0.503	0.488	0.492	29	0.497	0.496	0.495	0.504	0.497
5	0.506	0.505	0.502	0.482	0.502	30	0.486	0.539	0.503	0.485	0.508
6	0.496	0.517	0.509	0.517	0.510	31	0.499	0.507	0.516	0.510	0.507
7	0.470	0.472	0.501	0.478	0.480	32	0.504	0.482	0.501	0.499	0.495
8	0.486	0.487	0.495	0.496	0.489	33	0.519	0.473	0.502	0.483	0.499
9	0.501	0.475	0.503	0.472	0.487	34	0.484	0.460	0.502	0.474	0.479
10	0.474	0.487	0.504	0.481	0.486	35	0.500	0.486	0.500	0.484	0.493
11	0.560	0.543	0.511	0.460	0.529	36	0.499	0.457	0.482	0.461	0.476
12	0.456	0.476	0.479	0.473	0.470	37	0.459	0.513	0.502	0.499	0.494
13	0.459	0.448	0.500	0.423	0.456	38	0.493	0.480	0.496	0.506	0.491
14	0.497	0.481	0.500	0.482	0.490	39	0.522	0.522	0.530	0.472	0.519
15	0.533	0.569	0.514	0.518	0.539	40	0.513	0.508	0.516	0.487	0.509
16	0.503	0.494	0.500	0.486	0.497	41	0.486	0.421	0.497	0.506	0.487
17	0.526	0.463	0.501	0.480	0.492	42	0.487	0.483	0.500	0.473	0.486
18	0.472	0.468	0.495	0.494	0.479	43	0.495	0.475	0.492	0.486	0.487
19	0.442	0.429	0.467	0.484	0.449	44	0.469	0.470	0.500	0.447	0.473
20	0.453	0.459	0.493	0.479	0.468	45	0.413	0.478	0.396	0.458	0.439
21	0.496	0.491	0.495	0.497	0.494	46	0.466	0.477	0.500	0.483	0.481
22	0.495	0.477	0.455	0.483	0.477	47	0.511	0.531	0.507	0.482	0.513
23	0.504	0.521	0.503	0.493	0.506	48	0.476	0.475	0.493	0.500	0.484
24	0.514	0.510	0.507	0.464	0.504	49	0.496	0.497	0.500	0.498	0.497
25	0.490	0.466	0.497	0.497	0.484	50	0.500	0.455	0.496	0.495	0.481

Table 4.18 reveals that standard deviations ranged from 0.0182 (Readability) to 0.0285 (Insight), indicating Readability exhibited the most consistent performance while Insight demonstrated the greatest variability. The overall score standard deviation of 0.0187 represents relatively low variability compared to other agent configurations. Median values aligned closely with means across all dimensions, with maximum deviations of 0.0055, suggesting approximately symmetric distributions. Instruction Following notably exhibited the tightest IQR of 0.0075.

TABLE 4.18: RACE Score Distribution Statistics (Maxed Agent)

Dimension	Mean	Median	Std Dev	Min	Max	Q1	Q3	IQR
Comprehensiveness	0.4883	0.4938	0.0252	0.4126	0.5603	0.4703	0.5007	0.0304
Insight	0.4842	0.4788	0.0285	0.4209	0.5688	0.4666	0.4970	0.0305
Instruction Following	0.4968	0.5000	0.0187	0.3960	0.5300	0.4951	0.5026	0.0075
Readability	0.4837	0.4837	0.0182	0.4228	0.5177	0.4742	0.4970	0.0228
Overall Score	**0.4886**	**0.4882**	**0.0187**	**0.4392**	**0.5386**	**0.4775**	**0.4988**	**0.0213**

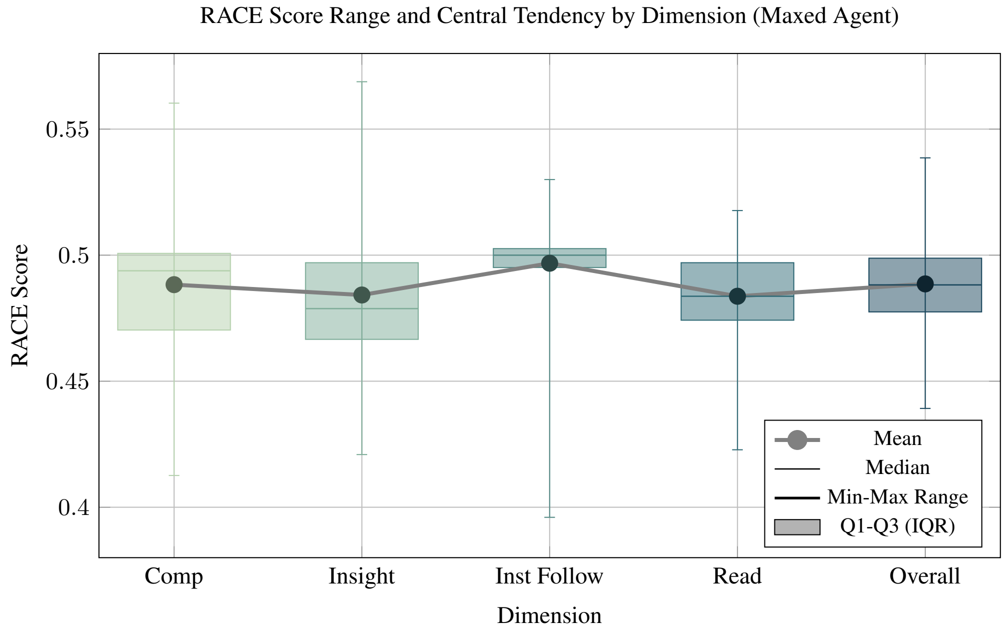


FIGURE 4.4: Statistical summary visualisation showing mean, median, IQR, and full range for each RACE dimension (Maxed Agent).

Citation metrics in Table 4.19 reflect the interaction between expanded search capacity and enhanced model quality. The configuration generated 18.83 total citations per task, the highest volume observed across all variants, with 14.85 valid citations. The valid rate of 78.86% positioned between baseline (76.66%) and Maxed Models (79.21%), suggesting that the citation precision benefits of higher-capacity models partially offset the precision reduction associated with expanded search volume observed in the Maxed Behaviour configuration.

TABLE 4.19: Maxed Agent FACT Scores

Metric	**Total citations**	**Total valid citations**	**Valid rate (%)**
Value	18.83	14.85	78.86

The operational cost structure in Table 4.20 reflects the combined impact of both enhancement dimensions. Search API calls reached 2,496 (averaging 49.92 per task), comparable to Maxed Behaviour's 2,471 and substantially exceeding baseline's 1,774. Total token consumption measured 82.37M tokens, approaching Maxed Behaviour's 87.92M but substantially exceeding Maxed Models' 39.47M and baseline's 62.28M. Model API costs totalled $73.88, the highest observed across all configurations, while search API costs reached $16.72. The total cost of $90.60 represents a 202.5% increase over baseline, yielding a 4.0% improvement in overall RACE score at an effective cost of $0.90/M tokens.

TABLE 4.20: Operational Metrics (Maxed Agent)

Metric	Value
Total Tavily search calls	2,496
Average searches per task	49.92
Total tokens	82.37M
Total input tokens	77.30M
Total output tokens	5.10M
Model API cost (USD)	$73.88
Search API cost (USD)	$16.72
Total cost (USD)	**$90.60**

4.5 Research Reviewer

The Research Reviewer system spawned 166 researcher sub-agents across the 50-task evaluation set, representing an average of 3.32 sub-tasks per task. This resulted in 25 refinement loops for a 15.1% refinement rate (calculated as refinements divided by sub-agents: 25/166). As shown in Figure 4.5, Relevance and Evidence dimensions achieved 100% pass rates, while Depth achieved 92.17% (13 failed evaluations) and Completeness exhibited the lowest pass rate at 85.54% (24 failed evaluations). Completeness was flagged in 96.0% of refinement loops triggered, with Depth co-occurring in 52.0% of cases, demonstrating inadequate research scope frequently coincided with insufficient analytical depth. The Completeness dimension's 14.46% failure rate confirms it as the major bottleneck for the Research Reviewer.

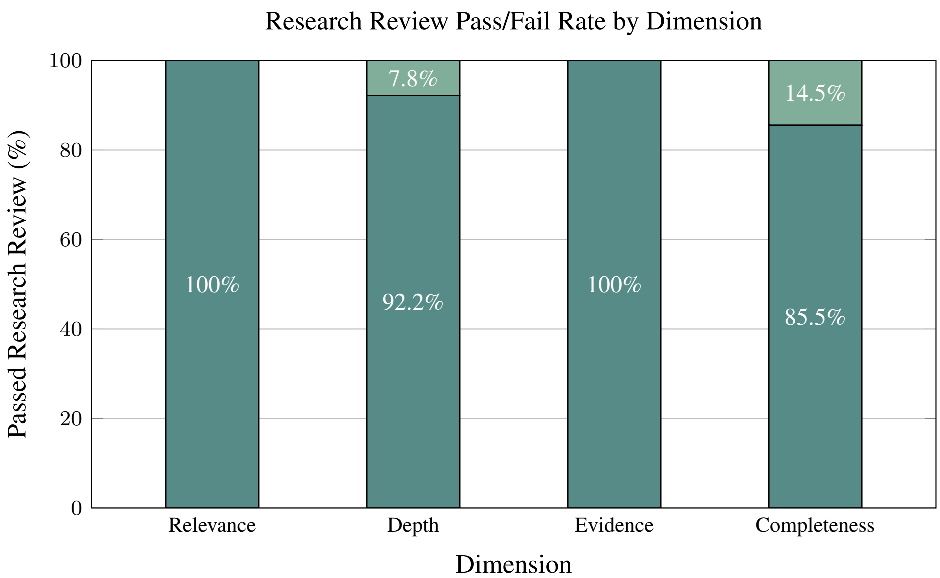


FIGURE 4.5: LLM judge evaluation pass/fail rates across metrics. Total evaluations: 166 per metric.

Under the revised configuration with the redefined dimensions detailed in Section 3.4.5, 160 researcher sub-agents were spawned across the 50-task evaluation set, triggering 13 refinement loops for an 8.1% refinement rate (13/160)—a notable reduction from the baseline 15.1%. As shown in Figure 4.6, Instruction Following achieved a 100% pass rate, Factuality 99.38%, and Comprehensiveness 98.13%, while Insight exhibited the lowest pass rate at 91.88% (13 failed evaluations). Insight was flagged in all 13 refinement loops triggered, confirming analytical depth as the primary quality bottleneck under the revised criteria. The shift from Completeness failures (14.46% in baseline) to Insight failures (8.12% in revised) indicates that the modified criteria successfully redirected quality assessment focus from scope coverage to analytical depth.

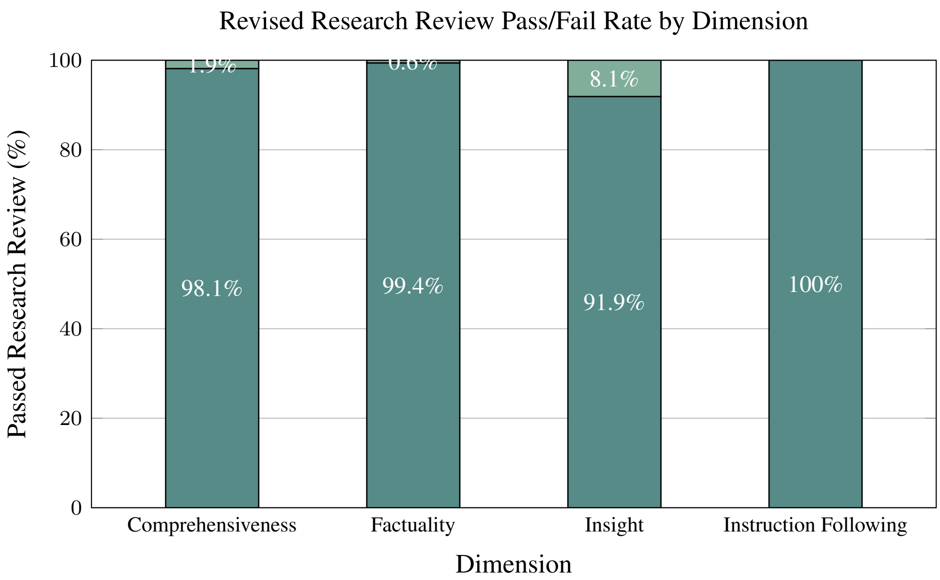


FIGURE 4.6: Revised LLM judge evaluation pass/fail rates across metrics with stricter assessment criteria. Total evaluations: 160 per metric.

Table 4.21 presents the RACE performance comparison between the two Research Reviewer (RR) variations. RR v2 demonstrated modest improvements over RR v1 across all dimensions, with the overall score increasing from 0.4615 to 0.4645 (+0.65%). The most substantial gains appeared in Instruction Following (+0.0043) and Readability (+0.0049), while Comprehensiveness and Insight exhibited minimal improvements (+0.0027 and +0.0007 respectively). Both variations underperformed relative to the baseline agent (0.4697 overall), with RR v1 showing a 1.7% deficit and RR v2 narrowing this gap to 1.1%.

TABLE 4.21: Research Reviewer RACE Scores

	Metric				
Iteration	**Comprehensiveness**	**Insight**	**Instruction Following**	**Readability**	**Overall Score**
RR v1	0.4524	0.4527	0.4808	0.4691	0.4615
RR v2	0.4551	0.4534	0.4851	0.4740	0.4645

Compared to the baseline agent (Table 4.5), both Research Reviewer versions exhibit higher costs for lower scores, incurring increased costs while delivering reduced performance. RR v1 performed approximately 12% more searches (1,987 vs. 1,774) and consumed 19% more tokens (74.3M vs. 62.3M), while RR v2 achieved greater efficiency with only 2% more searches (1,805 vs. 1,774) and 8% more tokens (67.2M vs. 62.3M). The total costs of $31.57 (RR v1) and $30.51 (RR v2) represent 5.4% and 1.9% increases over baseline ($29.95), yet both configurations underperform the baseline in RACE scores (0.4615 and

0.4645 vs. 0.4697).

TABLE 4.22: Operational Metrics (Research Reviewer)

Metric	RR v1	RR v2
Total Tavily search calls	1,987	1,805
Average searches per task	39.74	36.10
Total tokens	74.30M	67.20M
Total input tokens	70.10M	63.20M
Total output tokens	4.30M	4.00M
Model API cost (USD)	$18.26	$18.42
Search API cost (USD)	$13.31	$12.09
Total cost (USD)	**$31.57**	**$30.51**

4.6 Research Verifier

Table 4.23 presents the FACT scores for the Research Verifier configuration. The verifier achieved a valid rate of 90.3%, representing a 13.6 percentage point improvement over the baseline's 76.7%. This improvement comes at the cost of reduced citation volume: average citations per task decreased from 15.06 to 11.34 (24.7% reduction) as the verification stage removed unsupported claims. Valid citations decreased modestly from 11.53 to 10.24 (11.2% reduction).

TABLE 4.23: Research Verifier FACT Scores

Metric	**Total citations**	**Total valid citations**	**Valid rate (%)**
Value	11.34	10.24	90.30

Per-task performance is detailed in Table 4.24. Ten tasks achieved 100% citation accuracy, while the lowest accuracy observed was 78.6% (Task 19, with 3 invalid citations out of 14). The majority of tasks achieved accuracy rates at or above 90%.

TABLE 4.24: Per-Task FACT Scores (Research Verifier)

ID	#Cit.	#Valid	Acc. (%)	ID	#Cit.	#Valid	Acc. (%)
1	10	10	100.0	26	11	9	81.8
2	11	10	90.9	27	11	10	90.9
3	12	10	83.3	28	11	11	100.0
4	12	12	100.0	29	12	11	91.7
5	12	11	91.7	30	10	9	90.0
6	10	9	90.0	31	14	12	85.7
7	11	9	81.8	32	11	10	90.9
8	9	9	100.0	33	12	12	100.0
9	11	10	90.9	34	12	11	91.7
10	13	12	92.3	35	11	10	90.9
11	13	11	84.6	36	10	8	80.0
12	11	11	100.0	37	15	12	80.0
13	10	9	90.0	38	12	11	91.7
14	12	11	91.7	39	10	9	90.0
15	13	13	100.0	40	12	10	83.3
16	10	8	80.0	41	10	10	100.0
17	11	10	90.9	42	11	10	90.9
18	10	9	90.0	43	12	11	91.7
19	14	11	78.6	44	11	9	81.8
20	12	11	91.7	45	11	10	90.9
21	11	10	90.9	46	10	9	90.0
22	10	10	100.0	47	11	11	100.0
23	11	10	90.9	48	13	11	84.6
24	12	11	91.7	49	11	10	90.9
25	10	9	90.0	50	12	10	83.3

Table 4.25 presents the distribution statistics for per-task FACT metrics. The mean accuracy of 90.5% represents the macro-average (arithmetic mean of individual task accuracies), which differs from the micro-average of 90.3% reported in Table 4.23. The micro-average weights tasks by citation count, answering "what proportion of all citations are valid?", whereas the macro-average treats each task equally, answering "what is the average validity rate per task?". The slightly lower micro-average indicates that tasks with more citations tend to exhibit marginally lower accuracy rates. Note that micro-average values are used as the primary metric for cross-configuration FACT comparisons throughout this chapter to maintain consistency.

TABLE 4.25: FACT Score Distribution Statistics (Research Verifier)

Metric	Mean	Median	Std Dev	Min	Max	Q1	Q3	IQR
Total Citations	11.34	11.00	1.21	9	15	10	12	2
Valid Citations	10.24	10.00	1.09	8	13	9	11	2
Accuracy (%)	**90.49**	**90.90**	**6.11**	**78.6**	**100.0**	**85.4**	**91.7**	**6.3**

Table 4.26 summarises the operational costs. The verification stage consumed 12.11M Jina Reader API tokens to retrieve source content for claim validation, incurring an additional cost of \$0.61[2]. The total cost of \$31.30 represents a 4.5% increase over the baseline (\$29.95), an additional cost for the 13.6 percentage point improvement in citation validity.

[2]The Jina AI Reader API cost is calculated at \$50 USD per 1 billion tokens on the Jina AI Search Foundation API plan, equating to \$0.05 USD /M tokens.

TABLE 4.26: Operational Metrics (Research Verifier)

Metric	Value
Total Tavily search calls	1,832
Average searches per task	36.64
Total tokens	66.16M
Total input tokens	62.44M
Total output tokens	3.72M
Total Jina reader tokens	12.11M
Model API cost (USD)	$18.42
Search API cost (USD)	$12.27
Reader API cost (USD)	$0.605
Total cost (USD)	**$31.30**

4.7 Adaptive Model Selection

The model tier distribution, visualised in Figure 4.7, reveals the AMS system's routing behaviour across the 166 researcher sub-agent invocations. The mid-tier model (GPT-4.1 Mini) was selected for 89.2% of sub-tasks, establishing it as the predominant allocation. The high-tier model (GPT-5) was assigned to 10.8% of sub-tasks, escalation for complex analytical requirements. Notably, the low-tier model (GPT-4o Mini) received zero allocations, revealing that the supervisor's complexity assessment classified no sub-tasks as sufficiently routine to warrant lightweight model assignment.

The second implementation, detailed in Section 3.6.5, spawned 171 researcher sub-agents. Thus, enforcing 3 sub-tasks effectively limited the research iterations, as only 7 tasks required the second iteration.

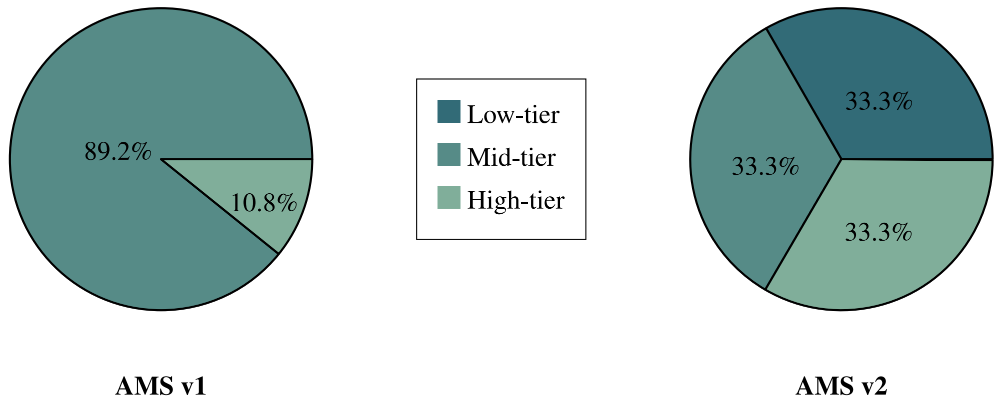


FIGURE 4.7: AMS Model Tier Distribution

Table 4.27 presents the RACE scores for both AMS implementations. AMS v1 achieved an overall score of 0.4792, closely approaching the Maxed Models' performance of 0.4810, while AMS v2 recorded a slightly lower overall score of 0.4765. In dimension-level performance, AMS v1 recorded 0.4770 for Comprehensiveness and 0.4740 for Insight, surpassing AMS v2's scores of 0.4710 and 0.4690 respectively. Both implementations maintained strong Instruction Following, with AMS v1 at 0.4899 and AMS v2 at 0.4890.

TABLE 4.27: AMS RACE Scores

	Metric				
Iteration	**Comprehensiveness**	**Insight**	**Instruction Following**	**Readability**	**Overall Score**
AMS v1	0.4770	0.4740	0.4899	0.4790	0.4792
AMS v2	0.4710	0.4690	0.4890	0.4770	0.4765

The FACT evaluation results in Table 4.28 demonstrate the validation efficiency of adaptive routing. AMS v1 achieved a valid rate of 78.80%, nearly matching the Maxed Models' 79.21%. AMS v2 yielded a lower valid rate of 77.40%. Specifically, AMS v1 generated 16.00 total citations per task with 12.61 valid citations. AMS v2 produced a higher volume of 16.50 total citations but a comparable count of 12.77 valid citations.

TABLE 4.28: AMS FACT Scores

	Metric		
Iteration	**Total citations**	**Total valid citations**	**Valid rate (%)**
AMS v1	16.00	12.61	78.80
AMS v2	16.50	12.77	77.40

Table 4.29 highlights the significant cost-efficiency gains achieved through adaptive selection. AMS v1 reduced total token consumption to 44.50 million and total cost to $36.80, achieving 99.6% of the Maxed Model Agent's performance at only 71% of the cost. In contrast, AMS v2 incurred higher costs ($44.15) and token usage (53.20 million).

TABLE 4.29: Operational Metrics (AMS)

Metric	**AMS v1**	**AMS v2**
Total Tavily search calls	1,350	1,550
Average searches per task	27.00	31.00
Total tokens	44.50M	53.20M
Total input tokens	41.00M	49.00M
Total output tokens	3.50M	4.20M
Model API cost (USD)	$27.75	$33.75
Search API cost (USD)	$9.05	$10.40
Total cost (USD)	**$36.80**	**$44.15**

4.8 Comparative Analysis

4.8.1 Agent Configuration Comparison

The comparative analysis synthesises performance metrics across all evaluated agent configurations to identify systematic patterns in the relationship between architectural parameters, research quality, and computational cost. The four baseline configurations—Baseline, Maxed Behaviour, Maxed Models, and Maxed—establish the performance envelope achieved through variations in behavioural limits and model selection, providing reference points for evaluating the proposed architectural enhancements.

Figure 4.8 presents a side-by-side comparison of RACE scores across all dimensions for the four baseline configurations. The visualisation reveals that model quality (Maxed Models, Maxed) exerted a stronger influence on score improvements than behavioural parameter expansion (Maxed Behaviour) alone. The Maxed configuration achieved the highest scores across all dimensions, demonstrating cumulative benefits from combining both enhancement approaches.

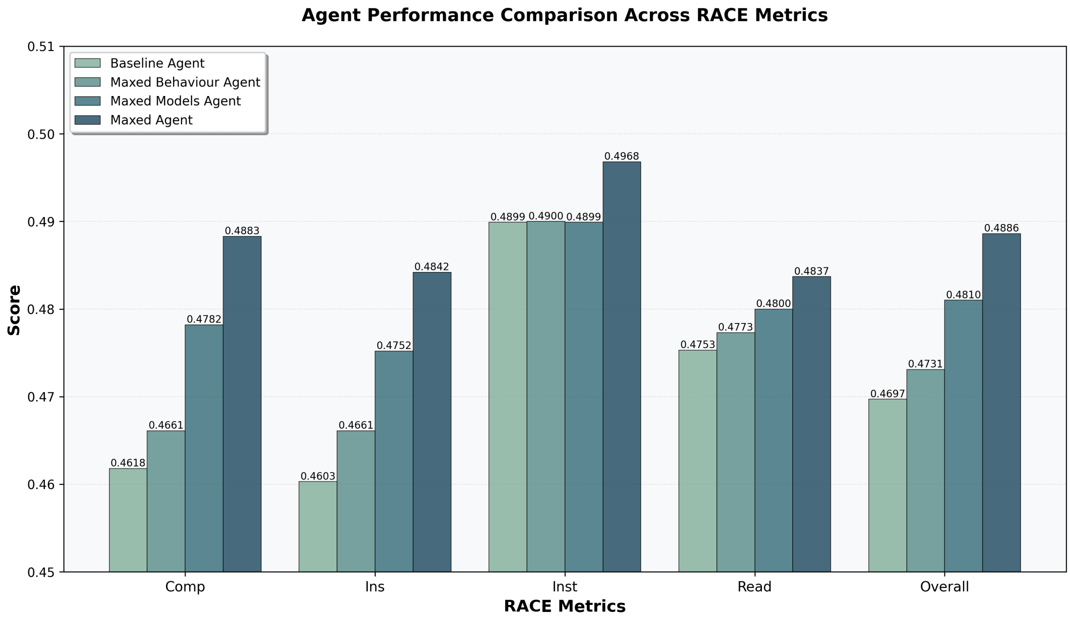


FIGURE 4.8: Clustered bar graph comparing agent performance across RACE metrics.

Citation performance comparisons across configurations are presented in Figure 4.9. The FACT metrics reveal distinct trade-offs between citation volume and validation precision. Configurations with expanded behavioural parameters (Maxed Behaviour, Maxed) generated the highest total citation counts. The Maxed Behaviour configuration exhibited a slightly reduced valid rate compared to baseline (75.39% vs. 76.66%), while the Maxed configuration maintained a higher valid rate (78.86%) despite its increased citation volume. The Maxed Models configuration achieved the highest valid rate (79.21%) with moderate citation volume.

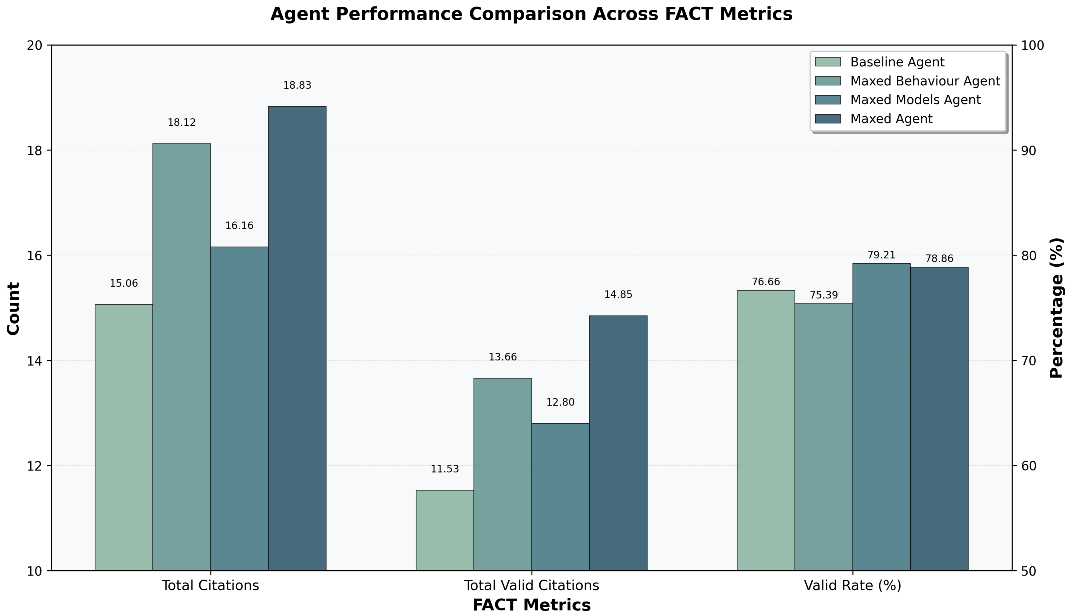


FIGURE 4.9: Clustered bar graph comparing agent performance across FACT metrics.

The performance-cost relationship across configurations is visualised in Figure 4.10, where bubble positions represent the trade-off between total cost (x-axis) and overall RACE score (y-axis), with bubble size proportional to total token consumption. The visualization reveals a general positive correlation between cost and performance. The Baseline configuration occupied the lower-left position ($29.95, 0.4697), offering the most cost-efficient option. Maxed Behaviour shifted rightward to $40.14 with a modest score increase to 0.4731, demonstrating poor cost-efficiency (34.1% cost increase for 0.72% score improvement). Maxed Models positioned at $51.80 with 0.4810 score, achieving the best score-to-cost improvement ratio (73.0% cost increase for 2.4% score improvement). The Maxed configuration occupies the upper-right extreme ($90.60, 0.4886), representing the performance ceiling at the highest computational cost.

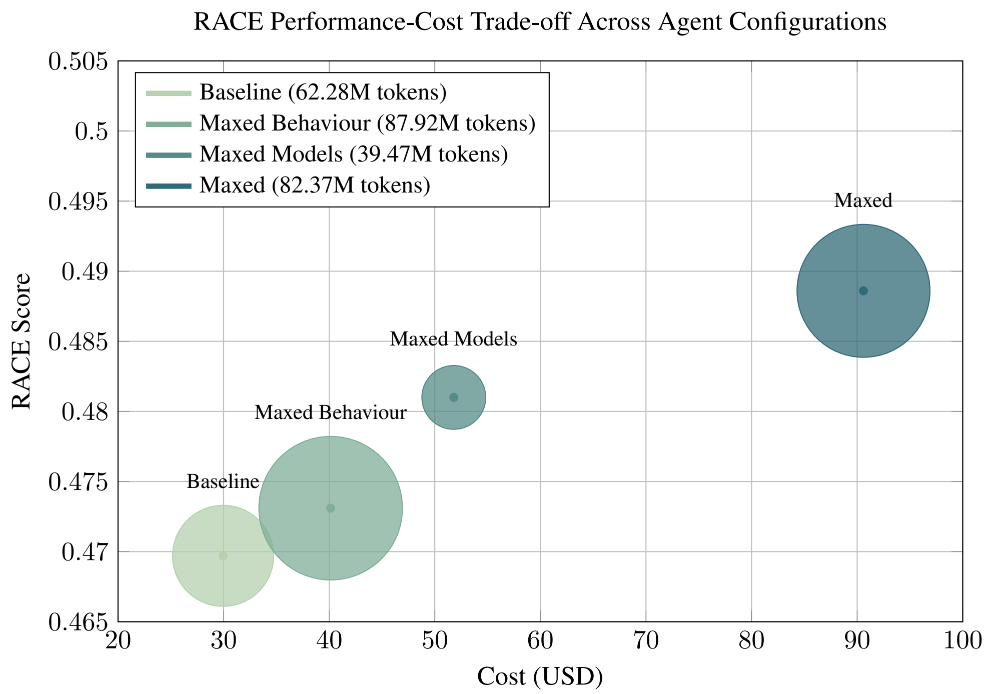


FIGURE 4.10: RACE Performance-cost trade-off comparison across agent configurations. Bubble size represents total token usage.

Figure 4.11 provides a breakdown of operational efficiency by plotting model API cost against total token usage. The radial lines indicate the effective model API cost/M tokens, revealing two distinct efficiency tiers. The Baseline and Maxed Behaviour agents operate in a lower cost-per-token regime (\$0.27–\$0.29/M), leveraging cost-effective models even at higher token volumes. Conversely, the Maxed Models and Maxed agents function in a higher cost-per-token regime (\$0.90–\$1.09/M), where the integration of premium models results in a steeper cost trajectory relative to token consumption.

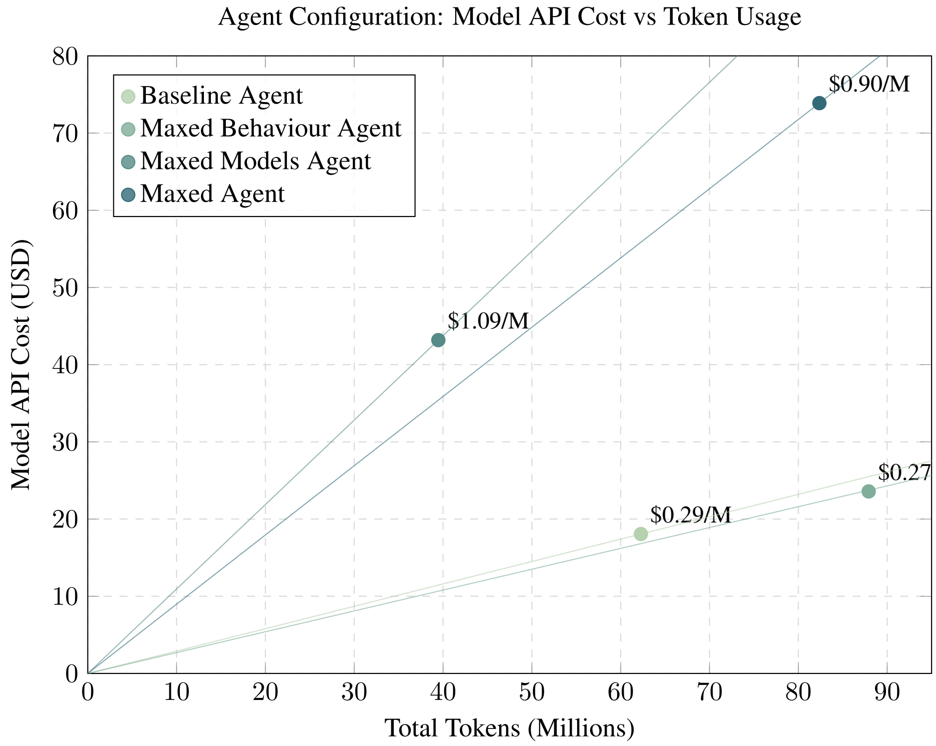


FIGURE 4.11: Comparison of operational metrics across agent configurations. The solid lines from origin indicate model API cost efficiency (steeper slope = higher cost/M tokens). Labels show effective model API cost/M tokens ($/M).

4.8.2 Research Reviewer & Agent Configurations Comparison

Figure 4.12 compares the performance of Research Reviewer v1 and v2 against the Baseline Agent across RACE metrics. Both Research Reviewer iterations underperform the Baseline Agent in all categories. While Research Reviewer v2 demonstrates consistent improvements over v1, particularly in Instruction Following and Readability, it fails to match the Baseline's overall effectiveness (0.4645 vs. 0.4697), indicating that the current implementation of the review layer degrades rather than enhances report quality.

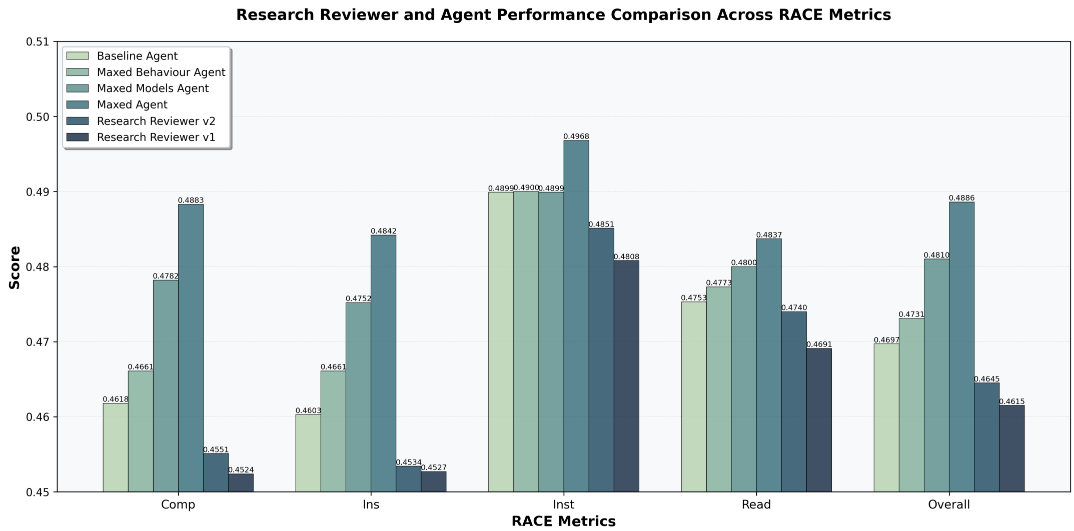


FIGURE 4.12: Clustered bar graph comparing agent configurations and research reviewer performance across RACE metrics.

Figure 4.13 illustrates the cost-performance trade-off for Research Reviewer configurations relative to the Baseline Agent. The Baseline Agent achieves the highest RACE score (0.4697) at the lowest cost ($29.95), establishing the efficiency frontier. Research Reviewer v1 incurs the highest operational cost ($31.57) and token usage (74.30M) while yielding the lowest performance (0.4615). Research Reviewer v2 improves upon v1 by reducing cost ($30.51) and token consumption (67.20M) while increasing performance (0.4645), yet still remains suboptimal compared to the Baseline, demonstrating a negative return on investment for the additional computational resources.

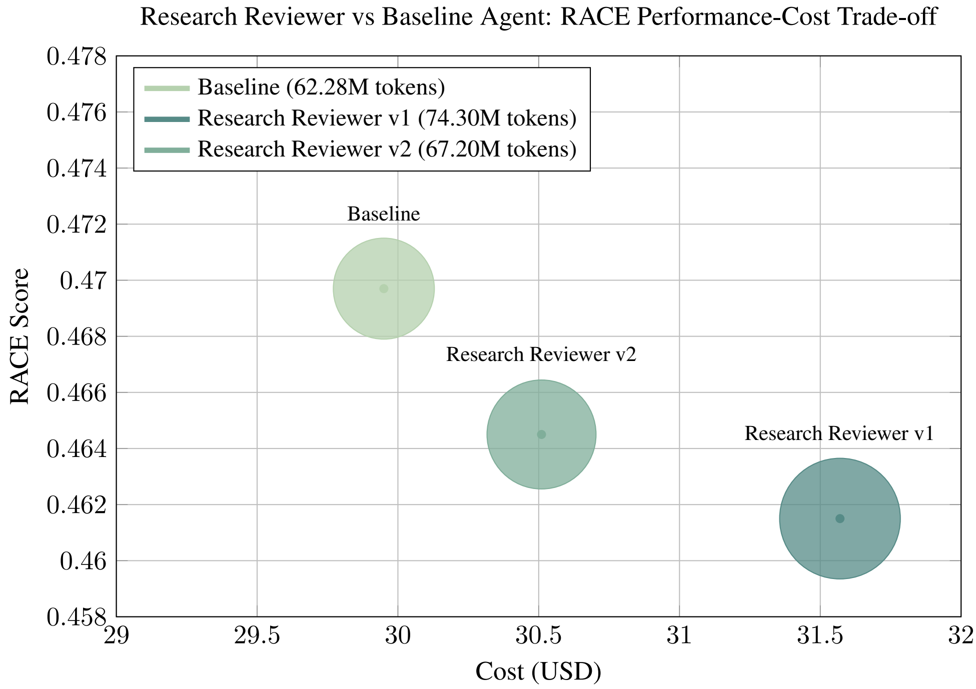


FIGURE 4.13: Research Reviewer vs Baseline Agent: RACE Performance-Cost Trade-off. Bubble size represents total token usage.

4.8.3 Research Verifier & Agent Configurations Comparison

Figure 4.14 compares the FACT performance of the Research Verifier against other agent configurations. The Research Verifier achieves a substantially higher valid rate (90.30%) compared to the Baseline (76.66%) and other configurations, demonstrating its effectiveness in filtering hallucinated and unsupported claims. However, this precision comes at the cost of recall: the Research Verifier produces the fewest total citations (11.34) and valid citations (10.24).

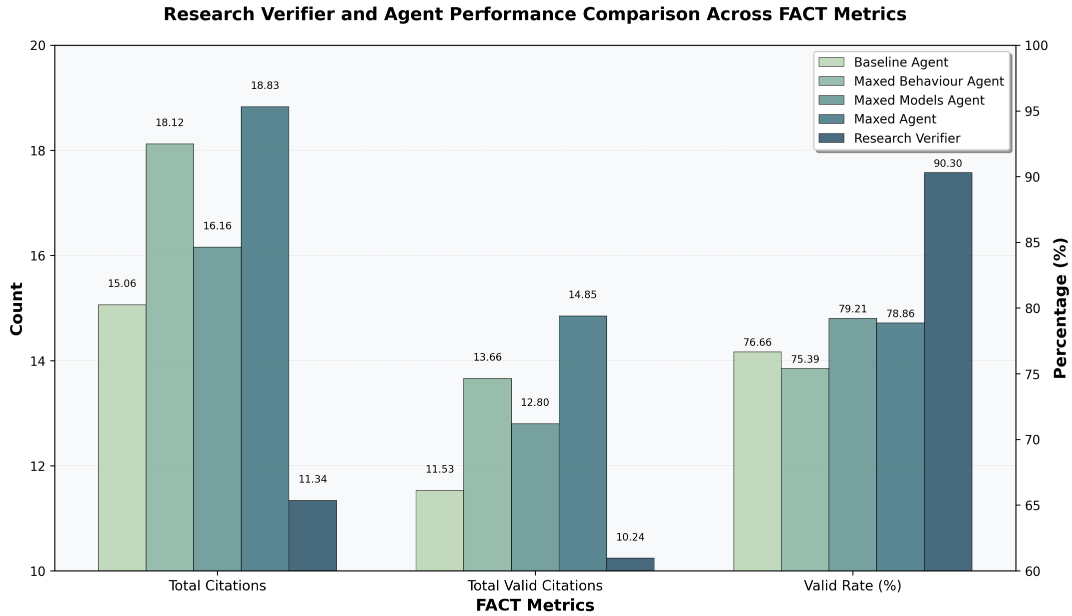


FIGURE 4.14: Clustered bar graph comparing agent configurations and research verifier performance across FACT metrics.

Figure 4.15 visualises the trade-off between citation quantity and quality, positioning the Research Verifier in a league of its own. While all other agent configurations cluster in the ”Lower Quality / High Quantity” quadrant with valid rates between 75% and 80%, the Research Verifier occupies the ”High Quality / Low Quantity” quadrant, achieving a 90.3% valid rate. This separation underscores its distinct operational profile as a precision-focused filter rather than a volume-focused generator.

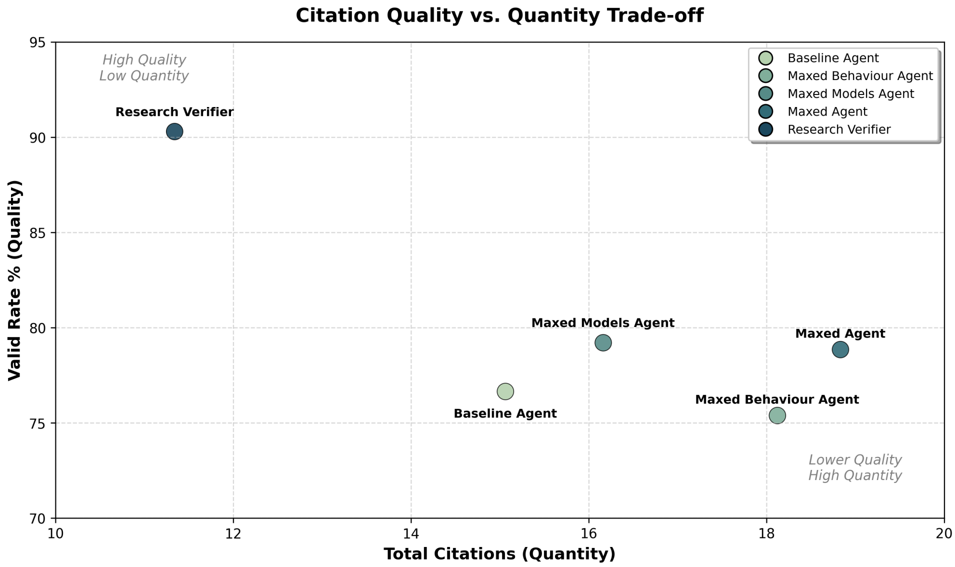


FIGURE 4.15: Research Verifier against agent configurations: citation quality vs. quantity trade-off.

4.8.4 AMS & Agent Configurations Comparison

Figure 4.16 compares the RACE performance of AMS configurations against the Baseline and Maxed Models agents. Both AMS implementations outperform the Baseline agent across all metrics. AMS v1 achieves an overall score of 0.4792, effectively matching the performance of the significantly more expensive Maxed Models agent (0.4810), while AMS v2 follows closely at 0.4765.

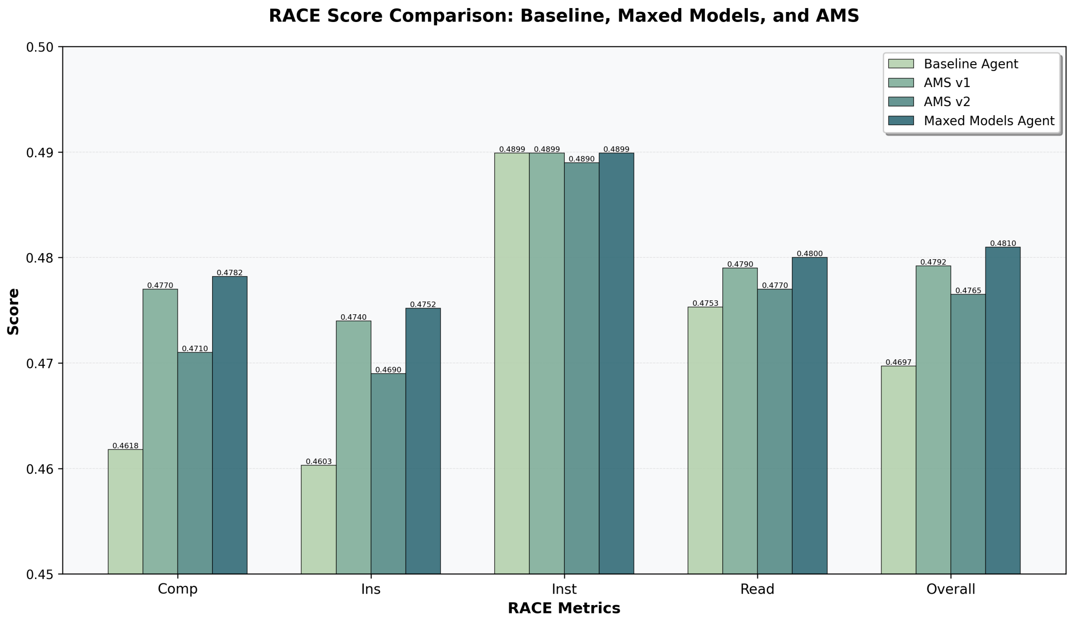


FIGURE 4.16: RACE score comparison between Baseline, Maxed Models Agent, AMS v1, and AMS v2 across all metrics.

Figure 4.17 illustrates the FACT performance across configurations. The AMS agents demonstrate improved citation validity compared to the Baseline (76.66%). AMS v1 achieves a valid rate of 78.80%, nearly equaling the Maxed Models agent (79.21%), albeit with lower citation volume. AMS v2 offers a balanced profile with a 77.40% valid rate and higher citation counts than v1.

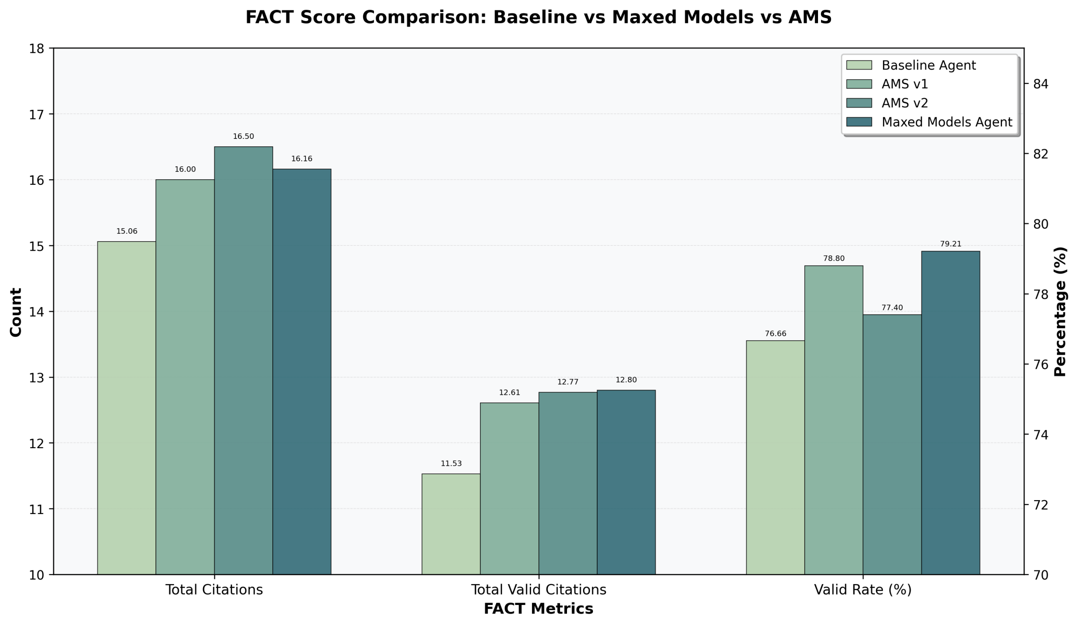


FIGURE 4.17: FACT score comparison between Baseline, Maxed Models Agent, AMS v1, and AMS v2 across all metrics.

Figure 4.18 illustrates the RACE performance-cost trade-off for AMS configurations. Both AMS versions offer performance improvements over the Baseline agent but at increased cost. AMS v1 provides an intermediate option, achieving near-Maxed performance (0.4792 vs. 0.4810) at a significantly lower cost ($36.80 vs. $51.80) and token usage (44.5M vs. 39.5M) than the Maxed Models agent.

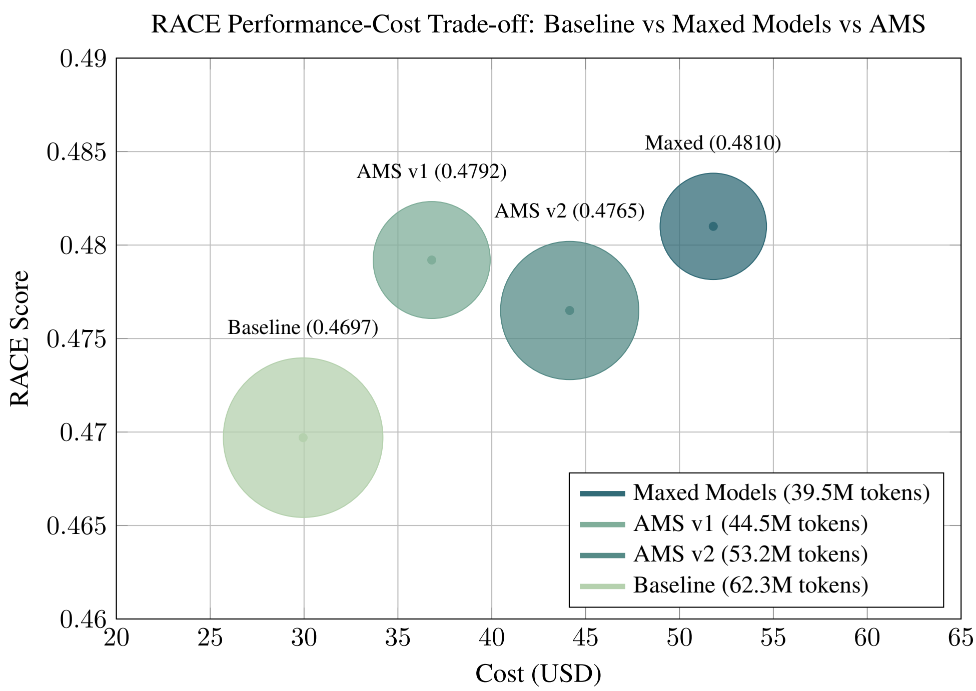


FIGURE 4.18: RACE Performance-cost trade-off comparison between Baseline, Maxed Models Agent and AMS implementations. Bubble size represents total token usage.

4.9 Robustness & Validity Checks

4.9.1 Run Consistency Analysis

The reliability of the RACE evaluation framework was assessed through duplicate evaluation runs for each agent configuration. All configurations were evaluated twice against the 50-task English subset of DeepResearch Bench, with final scores computed as arithmetic means of the two independent runs. This redundancy enables quantification of evaluation variance and validates the stability of the LLM-as-a-judge assessment methodology employed by the RACE framework.

Table 4.30 presents the standard errors and 95% confidence intervals for all configurations. Standard errors were computed as the absolute difference between runs divided by $\sqrt{2}$, and confidence intervals calculated using the standard $\pm 1.96 \times SE$ formula. The narrow confidence intervals across all metrics (RACE: ±0.001–0.007; FACT: ±0.5–2.8 percentage points) demonstrate high measurement reliability.

TABLE 4.30: Evaluation Consistency: Standard Errors and 95% Confidence Intervals

Configuration	RACE (Overall Score)			FACT (Valid Rate %)		
	Mean	**SE**	**95% CI**	**Mean**	**SE**	**95% CI**
Baseline	0.4697	0.0017	±0.0033	76.7	0.34	±0.7
Maxed Behaviour	0.4731	0.0018	±0.0036	75.4	0.33	±0.6
Maxed Models	0.4810	0.0017	±0.0033	79.2	0.38	±0.7
Maxed	0.4886	0.0018	±0.0036	78.9	0.23	±0.4
Research Reviewer v2	0.4645	0.0034	±0.0067	—	—	—
Research Verifier	—	—	—	90.3	0.25	±0.5
AMS v1	0.4792	0.0006	±0.0011	78.8	1.42	±2.8

Baseline Agent - Individual Run Scores

Tables 4.31 and 4.32 show the individual runs and averages for the Baseline Agent's RACE and FACT scores, respectively.

TABLE 4.31: Baseline Agent RACE Scores - Individual Runs and Average

Run	Comp	Ins	Inst	Read	Overall
Run 1	0.4631	0.4615	0.4887	0.4768	0.4709
Run 2	0.4605	0.4591	0.4911	0.4738	0.4685
Average	0.4618	0.4603	0.4899	0.4753	0.4697

TABLE 4.32: Baseline Agent FACT Scores - Individual Runs and Average

Run	Total citations	Total valid citations	Valid rate (%)
Run 1	15.32	11.70	76.42
Run 2	14.80	11.36	76.90
Average	15.06	11.53	76.66

Maxed Behaviour Agent - Individual Run Scores

Tables 4.33 and 4.34 show the individual runs and averages for the Maxed Behaviour Agent's RACE and FACT scores, respectively.

TABLE 4.33: Maxed Behaviour Agent RACE Scores - Individual Runs and Average

Run	Comp	Ins	Inst	Read	Overall
Run 1	0.4674	0.4649	0.4914	0.4761	0.4744
Run 2	0.4648	0.4673	0.4886	0.4785	0.4718
Average	0.4661	0.4661	0.4900	0.4773	0.4731

TABLE 4.34: Maxed Behaviour Agent FACT Scores - Individual Runs and Average

Run	Total citations	Total valid citations	Valid rate (%)
Run 1	18.38	13.84	75.16
Run 2	17.86	13.48	75.62
Average	18.12	13.66	75.39

Maxed Models Agent - Individual Run Scores

Tables 4.35 and 4.36 show the individual runs and averages for the Maxed Models Agent's RACE and FACT scores, respectively.

TABLE 4.35: Maxed Models Agent RACE Scores - Individual Runs and Average

Run	Comp	Ins	Inst	Read	Overall
Run 1	0.4795	0.4739	0.4913	0.4814	0.4822
Run 2	0.4769	0.4765	0.4885	0.4786	0.4798
Average	0.4782	0.4752	0.4899	0.4800	0.4810

TABLE 4.36: Maxed Models Agent FACT Scores - Individual Runs and Average

Run	Total citations	Total valid citations	Valid rate (%)
Run 1	16.42	13.02	79.48
Run 2	15.90	12.58	78.94
Average	16.16	12.80	79.21

Maxed Agent - Individual Run Scores

Tables 4.37 and 4.38 show the individual runs and averages for the Maxed Agent's RACE and FACT scores, respectively.

TABLE 4.37: Maxed Agent RACE Scores - Individual Runs and Average

Run	Comp	Ins	Inst	Read	Overall
Run 1	0.4897	0.4829	0.4981	0.4851	0.4899
Run 2	0.4869	0.4855	0.4955	0.4823	0.4873
Average	0.4883	0.4842	0.4968	0.4837	0.4886

TABLE 4.38: Maxed Agent FACT Scores - Individual Runs and Average

Run	Total citations	Total valid citations	Valid rate (%)
Run 1	19.08	15.06	79.02
Run 2	18.58	14.64	78.70
Average	18.83	14.85	78.86

Research Reviewer - Individual Run Scores

Tables 4.39 and 4.40 show the individual runs and averages for the Research Reviewer v1 and v2 RACE scores, respectively.

TABLE 4.39: Research Reviewer v1 RACE Scores - Individual Runs and Average

Run	Comp	Ins	Inst	Read	Overall
Run 1	0.4548	0.4551	0.4832	0.4715	0.4639
Run 2	0.4500	0.4503	0.4784	0.4667	0.4591
Average	0.4524	0.4527	0.4808	0.4691	0.4615

TABLE 4.40: Research Reviewer v2 RACE Scores - Individual Runs and Average

Run	Comp	Ins	Inst	Read	Overall
Run 1	0.4575	0.4558	0.4875	0.4764	0.4669
Run 2	0.4527	0.4510	0.4827	0.4716	0.4621
Average	0.4551	0.4534	0.4851	0.4740	0.4645

Research Verifier - Individual Run Scores

Table 4.41 shows the individual runs and averages for the Research Verifier's FACT scores.

TABLE 4.41: Research Verifier FACT Scores - Individual Runs and Average

Run	Total citations	Total valid citations	Valid rate (%)
Run 1	11.58	10.46	90.12
Run 2	11.10	10.02	90.48
Average	11.34	10.24	90.30

AMS - Individual Run Scores

Tables 4.42 through 4.45 show the individual runs and averages for AMS v1 and v2 RACE and FACT scores.

TABLE 4.42: AMS v1 RACE Scores - Individual Runs and Average

Run	Comp	Ins	Inst	Read	Overall
Run 1	0.4765	0.4735	0.4895	0.4785	0.4788
Run 2	0.4775	0.4745	0.4903	0.4795	0.4796
Average	0.4770	0.4740	0.4899	0.4790	0.4792

TABLE 4.43: AMS v1 FACT Scores - Individual Runs and Average

Run	Total citations	Total valid citations	Valid rate (%)
Run 1	15.43	12.00	77.77
Run 2	16.57	13.22	79.78
Average	16.00	12.61	78.80

TABLE 4.44: AMS v2 RACE Scores - Individual Runs and Average

Run	Comp	Ins	Inst	Read	Overall
Run 1	0.4700	0.4680	0.4880	0.4760	0.4755
Run 2	0.4720	0.4700	0.4900	0.4780	0.4775
Average	0.4710	0.4690	0.4890	0.4770	0.4765

TABLE 4.45: AMS v2 FACT Scores - Individual Runs and Average

Run	Total citations	Total valid citations	Valid rate (%)
Run 1	16.17	12.52	77.43
Run 2	16.83	13.02	77.36
Average	16.50	12.77	77.40

4.9.2 External Validation

External validity of the evaluation methodology and agent configurations can be assessed through comparison against reference implementations submitted to the DeepResearch Bench leaderboard by Langchain. This validates that the RACE and FACT scores achieved are reasonable.

Figure 4.19 compares the four baseline configurations against the Langchain reference submissions. This positioning validates that the systematically optimised baseline configuration established in Section 3.7 achieved comparable performance to the references.

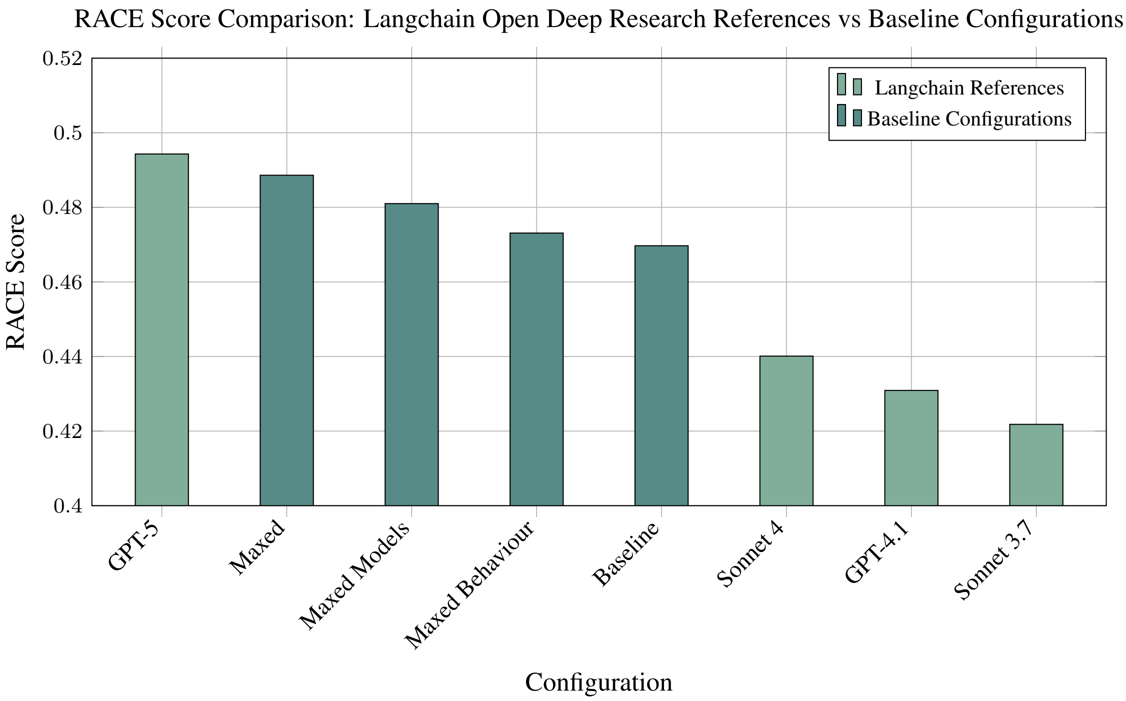


FIGURE 4.19: RACE Score Comparison: Langchain Open Deep Research References vs Baseline Configurations.

Figure 4.20 benchmarks the proposed architectural enhancements against the Langchain references. The AMS implementations achieve comparable scores, outperforming the Sonnet 4 and GPT-4.1 references and bridging the gap towards the state-of-the-art GPT-5 result. The Research Reviewer configurations, while lower scoring, still maintain parity with standard reference models.

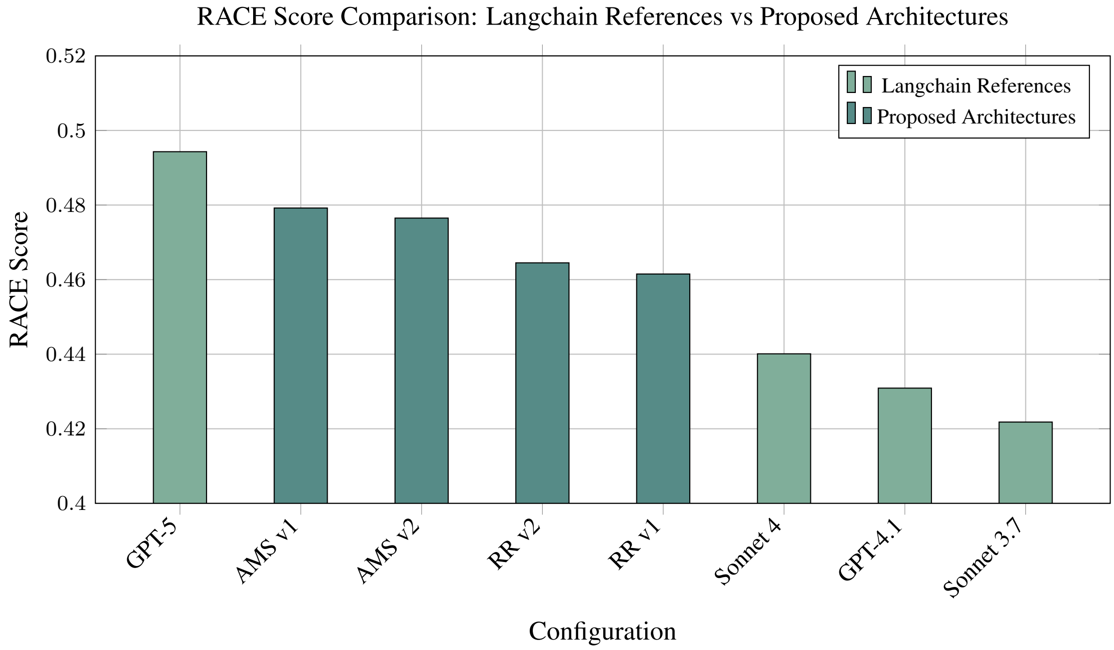


FIGURE 4.20: RACE Score Comparison: Langchain Open Deep Research References vs Proposed Architectures.

Given the modifications to the FACT evaluation framework, the metric of Effective Citations serves as a more reliable indicator of citation utility. The three Langchain Open Deep Research reference configurations achieved scores ranging from 21.06 to 29.49, whereas the Baseline configurations recorded scores between 13.05 and 16.73, as shown in Figure 4.21. This reduction in citation volume is consistent with the strategic revisions made to the Research Agent system prompt across all configurations, as detailed in Section 3.7.1, which reduced the number of search tool calls.

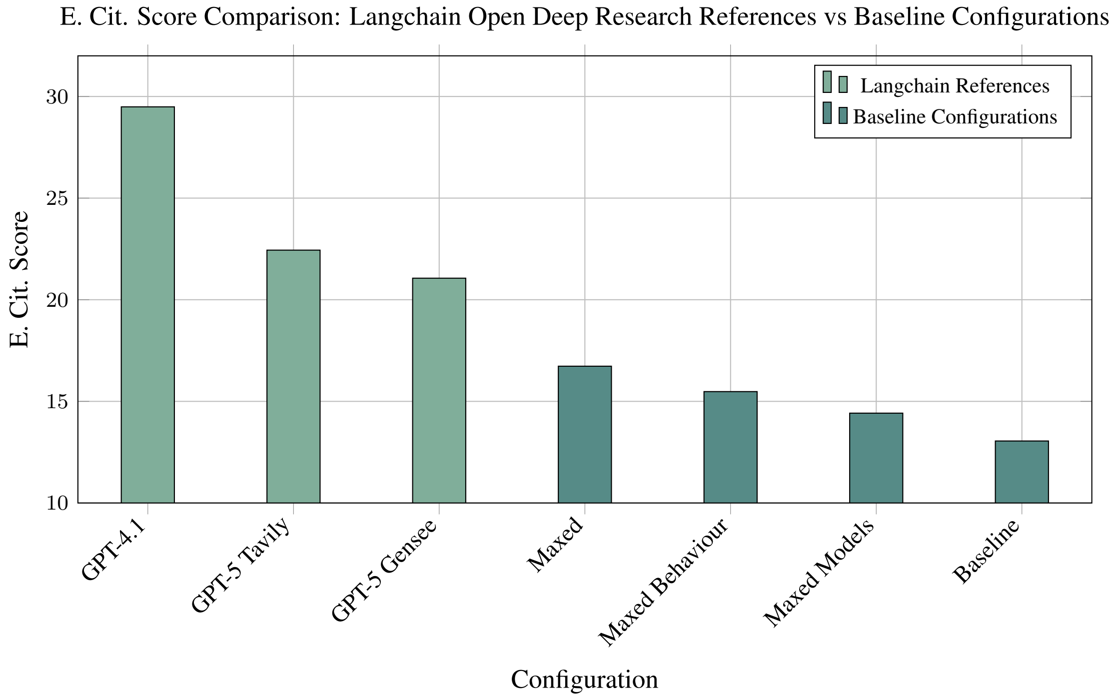


FIGURE 4.21: E. Cit. Score Comparison: Langchain Open Deep Research References vs Baseline Configurations.

Figure 4.22 compares the Effective Citation scores. The proposed architectures yield lower scores (10.24–12.24) compared to the Langchain references (21.06–29.49). This deviation highlights the trade-off inherent in the Research Verifier and AMS designs: strictly enforcing citation validity and reducing hallucination (as seen in Figure 4.14) naturally reduces the aggregate count of effective citations compared to systems optimised for generation volume.

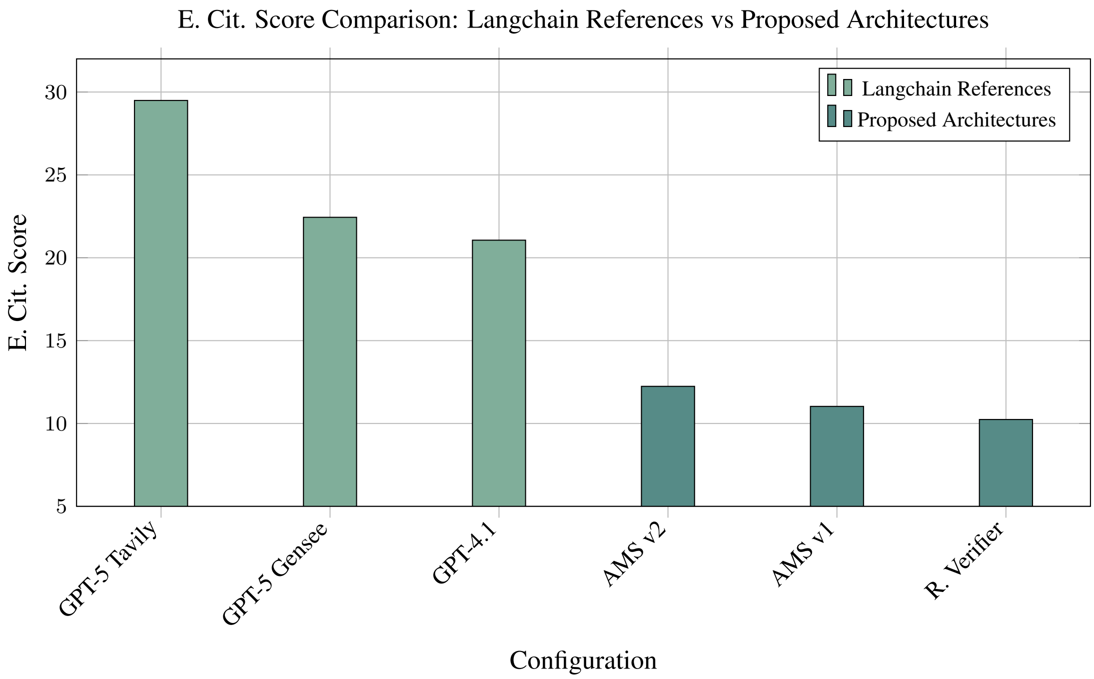


FIGURE 4.22: E. Cit. Score Comparison: Langchain Open Deep Research References vs Proposed Architectures.

Figure 4.23 presents the DeepResearch Bench leaderboard in which only five submissions have achieved a RACE score exceeding 50, highlighting the challenge of the benchmark. The results discussed in this chapter fall within the competitive 45–50 range, placing them among the top-tier performing systems.

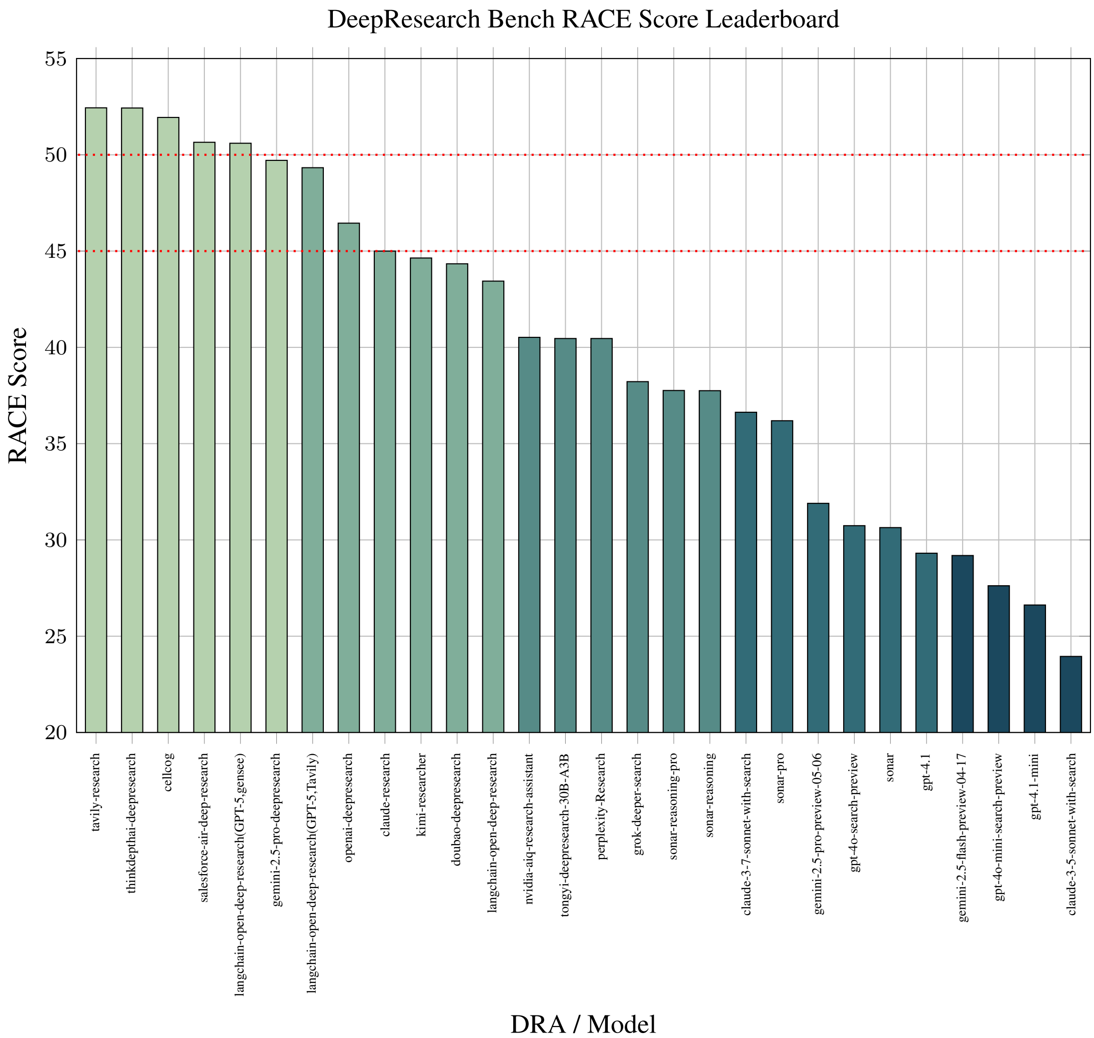


FIGURE 4.23: DeepResearch Bench RACE score leaderboard [16].

Figure 4.24 compares the performance of specialised DRAs against standard LLMs equipped with search. The trend clearly indicates that DRAs consistently outperform LLMs with search, validating the superior capability of agentic workflows for complex research tasks.

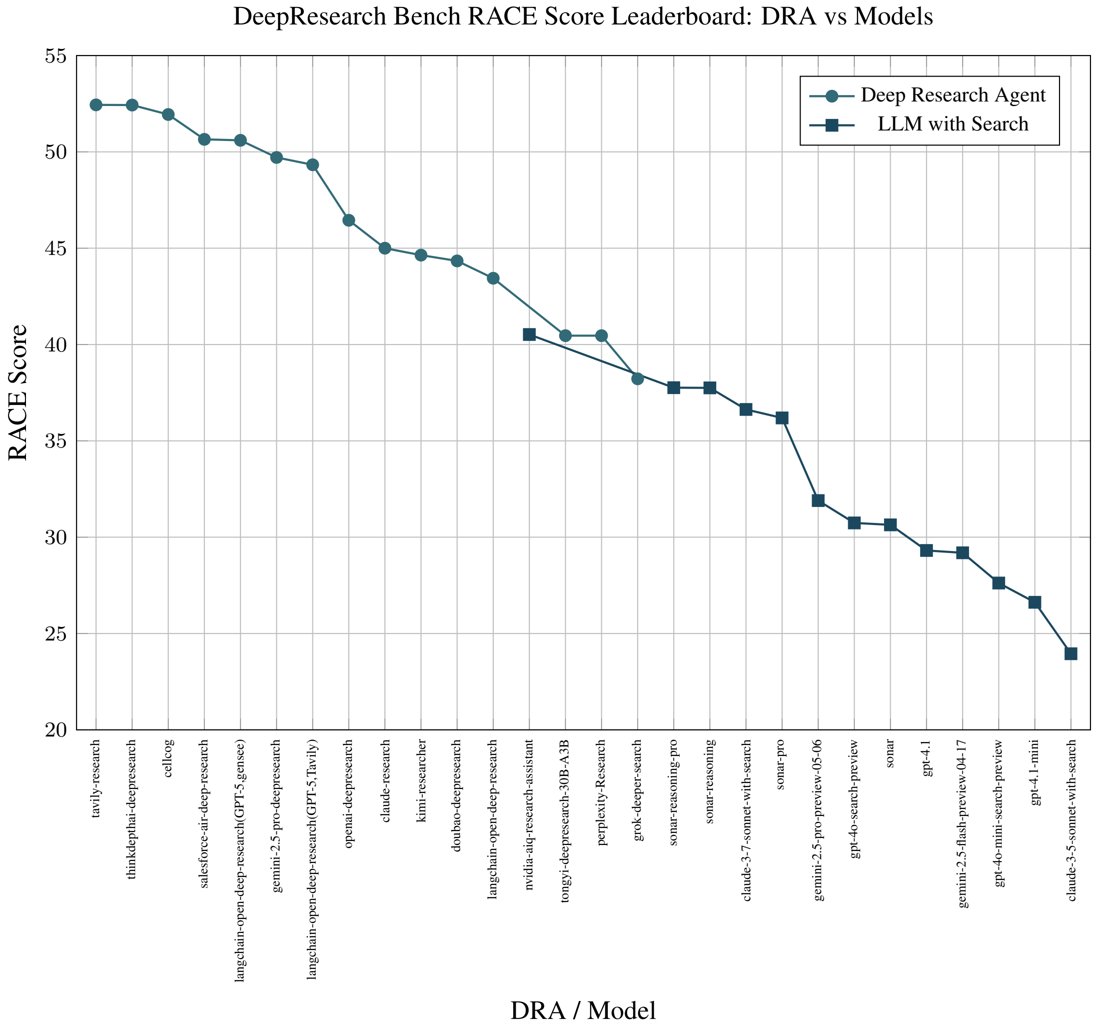
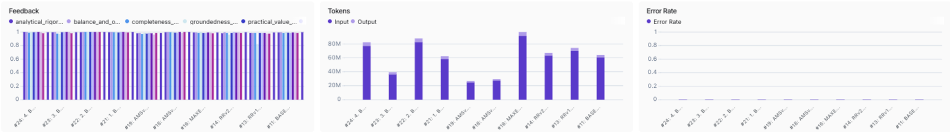


FIGURE 4.24: DeepResearch Bench RACE score leaderboard differentiating between Deep Research Agents and LLMs with Search [16].

4.9.3 Experiment Execution Record

The experimental runs were tracked and managed using LangSmith. Figures 4.25 and 4.26 present snapshots of the experiment execution record from the LangSmith studio dashboard, showing the configuration parameters for behavioural constraints and model strings, respectively.



	Experiment Name	Splits	Run Count	Error Rate	Prompt Tokens	Output Tokens	Total Tokens	max_concurrent_research_...	max_react_tool_calls	max_researcher_iterations
#24	4. Base maxed behaviour & ...	base	50	0%	77,270,653	5,094,397	82,365,050	10	10	6
#23	3. Base maxed models: OP...	base	50	0%	36,495,593	2,976,718	39,472,311	3	6	3
#22	2. Base maxed behaviour: ...	base	50	0%	82,723,721	5,197,563	87,921,284	10	10	6
#21	1. Base: OPEN_DEEP_RESE...	base	50	0%	58,501,670	3,782,154	62,283,824	3	6	3
#19	AMSv2: OPEN_DEEP_RESE...	base	50	0%	49,038,455	4,209,361	53,247,816	3	6	3
#18	AMSv1: OPEN_DEEP_RESE...	base	50	0%	41,092,611	3,425,292	44,517,903	3	6	3
#16	MAXED: OPEN_DEEP_RESE...	base	50	0%	91,677,737	5,531,176	97,208,913	10	10	6
#14	RRv2: OPEN_DEEP_RESEAR...	base	50	0%	63,236,742	3,975,225	67,211,967	3	6	3
#13	RRv1: OPEN_DEEP_RESEAR...	base	50	0%	70,027,202	4,280,023	74,307,225	3	6	3
#11	BASE: OPEN_DEEP_RESEA...	base	50	0%	60,569,116	3,762,169	64,331,285	3	6	3

FIGURE 4.25: The Langchain experiment execution record with behaviour columns.

	Experiment Name	Splits	Run Count	Error Rate	Prompt Tokens	Output Tokens	Total Tokens	compression_model	final_report_model	research_model	summarization_model
#24	4. Base maxed behaviour ...	base	50	0%	77,270,653	5,094,397	82,365,050	openai:gpt-4.1	openai:gpt-5	openai:gpt-5	openai:gpt-4.1-mini
#23	3. Base maxed models: O...	base	50	0%	36,495,593	2,976,718	39,472,311	openai:gpt-4.1	openai:gpt-5	openai:gpt-5	openai:gpt-4.1-mini
#22	2. Base maxed behaviour: ...	base	50	0%	82,723,721	5,197,563	87,921,284	openai:google/gemini-2.0-flash-	openai:gpt-5-mini	openai:gpt-4o-mini	openai:google/gemini-2.5-flash-lit
#21	1. Base: OPEN_DEEP_RES...	base	50	0%	58,501,670	3,782,154	62,283,824	openai:google/gemini-2.0-flash-	openai:gpt-5-mini	openai:gpt-4o-mini	openai:google/gemini-2.5-flash-lit
#19	AMSv2: OPEN_DEEP_RES...	base	50	0%	49,038,455	4,209,361	53,247,816	openai:google/gemini-2.0-flash-	openai:gpt-5-mini	openai:gpt-4o-mini	openai:google/gemini-2.5-flash-lit
#18	AMSv1: OPEN_DEEP_RES...	base	50	0%	41,092,611	3,425,292	44,517,903	openai:google/gemini-2.0-flash-	openai:gpt-5-mini	openai:gpt-4o-mini	openai:google/gemini-2.5-flash-lit
#16	MAXED: OPEN_DEEP_RES...	base	50	0%	91,677,737	5,531,176	97,208,913	openai:google/gemini-2.0-flash-	openai:gpt-5-mini	openai:gpt-4o-mini	openai:google/gemini-2.5-flash-lit
#14	RRv2: OPEN_DEEP_RESEA...	base	50	0%	63,236,742	3,975,225	67,211,967	openai:google/gemini-2.0-flash-	openai:gpt-5-mini	openai:gpt-4o-mini	openai:google/gemini-2.5-flash-lit
#13	RRv1: OPEN_DEEP_RESEA...	base	50	0%	70,027,202	4,280,023	74,307,225	openai:google/gemini-2.0-flash-	openai:gpt-5-mini	openai:gpt-4o-mini	openai:google/gemini-2.5-flash-lit
#11	BASE: OPEN_DEEP_RESE...	base	50	0%	60,569,116	3,762,169	64,331,285	openai:google/gemini-2.0-flash-	openai:gpt-5-mini	openai:gpt-4o-mini	openai:google/gemini-2.5-flash-lit

FIGURE 4.26: The Langchain experiment execution record with models columns.

CHAPTER 5

Discussion

5.1 Primacy of Internal Reasoning

The advantage of scaling model intelligence over system behaviour is strongly supported by the findings of this thesis. The Maxed Models agent, which upgraded the underlying LLMs to frontier models such as GPT-5, yielded a 0.0113 increase in overall RACE score compared to the baseline. In stark contrast, the Maxed Behaviour configuration, which aggressively scaled search iterations and depth, resulted in only a 0.0034 improvement. This disparity highlights that simply increasing the volume of information retrieval and synthesis offers diminishing returns compared to enhancing the cognitive engine that synthesises that information.

Furthermore, the Maxed Agent, which combined both model and behavioural enhancements, achieved the highest absolute performance (0.4886). However, the marginal gain over the Maxed Models configuration (0.4810) was minimal (0.0076) relative to the extreme cost increase of $51.80 to $90.60. Figure 5.1 reinforces this idea by isolating the impact of model choice on the DeepResearch Bench, showing a clear performance hierarchy where superior models like GPT-5 consistently outperform capable but smaller models like GPT-4.1 and Sonnet 4, regardless of the surrounding architecture. This suggests that the ceiling for agentic performance is currently dictated more by the fundamental reasoning capabilities of the model than by the complexity of the workflow.

Complex reasoning tasks, such as multi-step arithmetic or symbolic manipulation, only become solvable once a model reaches a critical threshold of scale, typically exceeding 10^{23} training FLOPs[1] or 100 billion parameters [107]. Below this threshold, extrinsic interventions are often not worthwhile; for instance, Chain-of-Thought (CoT) prompting degrades performance in smaller models while enabling significant gains in larger models [108]. This indicates that the reasoning engine—the LLM itself—must possess the inherent capacity before behavioural parameters can be leveraged effectively.

Recent advancements in self-correction further underscore the importance of model intelligence. Extrinsic closed-loop paradigms that rely on external prompts to trigger refinement often yield minimal improvement and even performance degradation if the model lacks sufficient verification capabilities [109]. Conversely, systems like SPOC (Spontaneous Self-Correction) demonstrate that performance is maximised when the model internalises the roles of both proposer and verifier within a single inference pass, relying solely on

[1] FLOPs (floating point operations) quantify the total amount of computation used during training, serving as a primary metric for measuring model scale alongside the number of parameters [107].

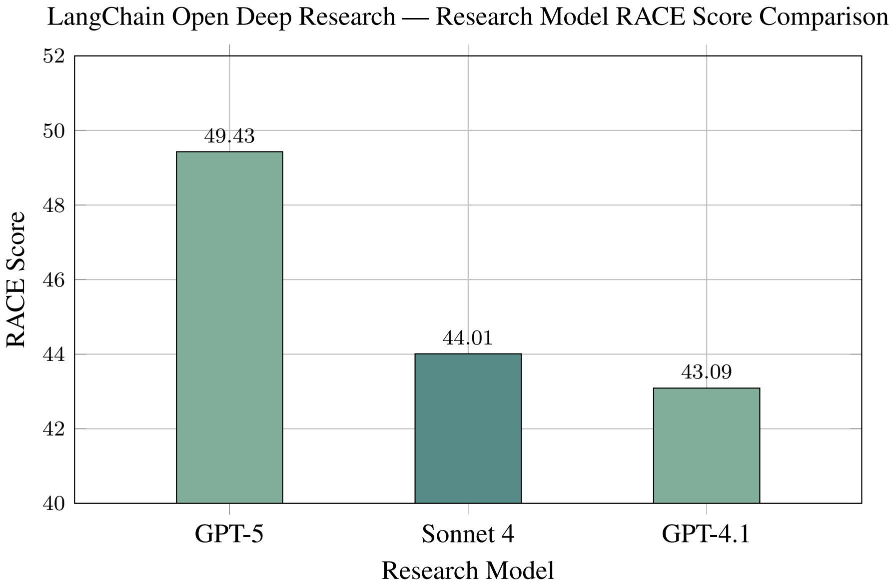


FIGURE 5.1: RACE score comparison between different research models against DeepResearch Bench [17].

inherent capabilities to dynamically guide generation [109]. Ultimately, while internal reasoning is the engine of performance, the extent to which it outweighs the system scaffolding depends on the architecture and likely varies by research task.

5.2 Reasoning in High-Capability LLMs

TODO: complete (LLMs performing better individually without layering)

5.3 Reactive versus Emergent Systems

TODO: Beyond just LLM capabilities, RR is redundant because of the overall architecture.

The Research Reviewer results revealed a counter-intuitive finding: both iterations, though explicitly designed to enforce quality standards, degraded performance relative to the baseline. Research Reviewer v1, which triggered 25 refinement loops (15.1% rate), achieved the lowest overall RACE score of 0.4615. Research Reviewer v2, with refined criteria that triggered fewer loops (13 loops, 8.1% rate), improved slightly to 0.4645 but still failed to match the baseline's 0.4697. This negative correlation between refinement frequency and report quality suggests that forced, prompt-driven review steps may be actively harmful when not intrinsically required. Rather than refining the output, this dedicated review layer introduced a bottleneck, incurring higher computational costs while yielding lower-quality reports. This failure underscores a fundamental limitation of reactive architectures: imposing static, post-hoc quality gates interrupts the continuous reasoning trajectory required for complex inquiry.

This observation aligns with a broader architectural evolution making dedicated review agents redundant. This section discusses how this concept is applied by five DeepResearch Bench submissions, namely

ThinkDepth.ai Deep Research, Salesforce Enterprise Deep Research, Claude Research, Kimi-Researcher, and Tongyi Deep Research, all of which are either open-source or propriety with a public technical paper.

The development of Deep Research arcitectures has seen a decisive shift from reactive, workflow-based toward emergent, dynamic systems. Early iterations of agentic AI relied on rigid, pre-defined plans often involving assigning specialised roles and coordinating them via fixed prompt-based workflows, as seen in the LangChain Open Deep Research system. However, state-of-the-art architectures are abandoning these constraints in favour of real-time adaptability [110]. The unpredictability inherent in Deep Research renders linear, hardcoded paths obsolete, as the investigative process is "dynamic and path-dependent" [111]. Consequently, the industry is witnessing the obsolescence of static review steps, replaced by agents that pivot strategies instantly upon discovering new information.

The limitation of traditional workflow-based systems lies in their inability to accommodate the "unknown unknowns" of complex inquiry. In these older paradigms, a planner outlines a report structure and executes searches sequentially, a method that often fails to maintain global context or adapt to shifting information landscapes [112]. In contrast, emerging architectures leverage end-to-end reinforcement learning (RL) to cultivate holistic reasoning capabilities. For instance, Kimi-Researcher utilizes RL to learn planning, perception, and tool use simultaneously, allowing the agent to explore strategies and learn from full trajectories rather than adhering to hand-crafted rules [112]. This approach enables emergent behaviors, such as self-correction and iterative hypothesis refinement, which arise naturally without specific programming [111, 112].

In systems like ThinkDepth.ai Deep Research, the model is granted the autonomy to "add or remove structures or even create its own steps" provided they satisfy high-level self-balancing constraints [112]. This aligns with the notion that imposing human-centric structures like rigid planning steps eventually inhibits performance as compute power scales; instead, systems should be free to leverage search and learning to discover optimal paths [112]. Since the agent explicitly reasons about information gaps and pivots its strategy in real-time to bridge them, the need for a pre-computation planning phase is diminished.

Furthermore, the implementation of dynamic replanning and reflection mechanisms allows for instantaneous strategic pivots. In the Enterprise Deep Research framework built by Salesforce, a Master Research Agent performs adaptive query decomposition that integrates new knowledge gaps identified from prior iterations, permitting adjustments based on intermediate results rather than a static initial plan [73]. Similarly, Anthropic's multi-agent system demonstrates that agents using Interleaved Thinking can evaluate tool results and refine their next queries autonomously, cutting research time by up to 90% compared to sequential searches [111].

Ultimately, the trajectory of deep research agents is moving toward environmental interaction as the primary driver of decision-making, subsequently making steps like the Research Reviewer counter-productive. [113]. Whether through the strategic context management of Tongyi DeepResearch or the iterative draft refinement of ThinkDepth.ai Deep Research, the consensus is clear that pre-defined plans are brittle [112,113]. True agency emerges from dynamically navigating the open web, treating research as fluid exploration rather than a linear checklist [73]. The Langchain Open Deep Research system blog itself acknowledges that the optimal way to handle a research task cannot be known beforehand, as different requests demand different research approaches and depths of investigation [13].

5.4 The Necessity of Verification

The Research Verifier results (Section 4.6) provide empirical evidence for the necessity of post-generation citation verification in Deep Research systems. The baseline configuration's valid rate of 76.7% indicates that nearly one in four citations contained factual inaccuracies or unsupported assertions. Critically, simply upgrading the models doesn't address this issue and only provides minimal gains in citation accuracy: the Maxed Models configuration achieved only 79.2% validity (a 2.5 percentage point improvement), whilst the Maxed Agent combining both enhanced models and expanded behaviour yielded 78.9% (a 2.2 percentage point improvement). These reliability thresholds are unacceptable for research applications where factual accuracy is paramount. Without verification mechanisms, users receive synthesised reports containing invalid citations that may misrepresent sources or present unsupported claims as established facts, undermining the credibility of the research output.

The verification component's 13.6 percentage point improvement over the baseline to 90.3% validity demonstrates the impressive effectiveness of systematic claim validation against source material. This enhancement comes at a negligible operational cost increase of 4.5% as well as an expected reduction in citation volume from 15.06 to 11.34 per task, reflecting the removal of unsupported assertions. Critically, valid citation count decreased only marginally from 11.53 to 10.24 (11.2% reduction) compared to the baseline, indicating that the verifier primarily eliminates invalid citations rather than indiscriminately reducing coverage. This trade-off of fewer total citations but substantially higher reliability aligns with research integrity principles where accuracy supersedes quantity.

The per-task distribution reveals persistent challenges: ten tasks achieved perfect accuracy whilst one task's 78.6% validity indicates continued vulnerabilities in the synthesis process. Even the improved 90.3% aggregate rate means approximately one in ten citations remains invalid, suggesting that verification mechanisms, whilst necessary and effective, require further refinement to approach the reliability standards expected of human-produced research outputs. It is estimated that the remaining 10% gap stems from multiple factors: source content retrieval failures when websites employ paywalls or access restrictions that prevent the Jina Reader API from extracting verification content; the inherent non-determinism of LLM-as-Judge evaluation where marginal cases may be inconsistently classified; and edge cases where factual claims require domain expertise or multi-hop reasoning beyond the verifier's capabilities. Addressing these limitations through enhanced retrieval mechanisms, deterministic fallback strategies for marginal judgments, and multi-stage verification processes could incrementally approach the reliability standards expected of human-curated research. However, these technical challenges still partially reflect deeper fundamental issues regarding LLM factual reliability.

These limitations are well-documented in the broader literature on LLM truthfulness. Current LLMs frequently generate imitative falsehoods—errors where models reproduce misconceptions present in their training data [114]. The TruthfulQA benchmark reveals an inverse scaling phenomenon in which larger models often display lower truthfulness than smaller ones because they are more capable of learning and mimicking these deceptive patterns from the training distribution [114]. This finding directly explains why simply increasing model parameters fails to address inaccuracies, as the training objectives focus on text imitation rather than veracity [114]. This pattern is clearly reflected in the minimal FACT score improvements observed across the Maxed Models and Maxed Agent configurations.

The challenge extends beyond generation to verification itself. Benchmarks like FEVER demonstrate that automated claim verification against textual sources remains computationally difficult, with early pipeline systems achieving only 31.87% accuracy when strict evidence retrieval is required [115]. This explains why even the Research Verifier, despite its substantial improvements, cannot achieve perfect accuracy: the verification task itself requires sophisticated reasoning capabilities that current systems struggle to provide consistently. These findings collectively underscore that LLMs cannot currently be relied upon as standalone truth-tellers, necessitating the integration of external retrieval and rigorous post-generation verification mechanisms to ensure research integrity [114].

5.5 Efficiency via Adaptive Selection

TODO: complete

5.6 The Cost-Performance Trade-off

TODO: complete

Figure 5.2 presents a comparison of the cost, token usage, and RACE score between GPT-4.1 and Sonnet 4 models against the DeepResearch Bench. All variables were held constant except for the research model.

5.7 Limitations

This section addresses the constraints and shortcomings of both the research methodology and the proposed architectural components. The methodological limitations primarily stem from resource constraints that affected experimental design and evaluation rigor. On the other hand, the architectural limitations reflect fundamental design trade-offs that constrain system reliability and adaptability.

5.7.1 Limitations of Methodology

The methodological constraints described below derive from the high operational costs of agentic AI systems, which is pronounced in Deep Research due to the involvement of web search API calls.

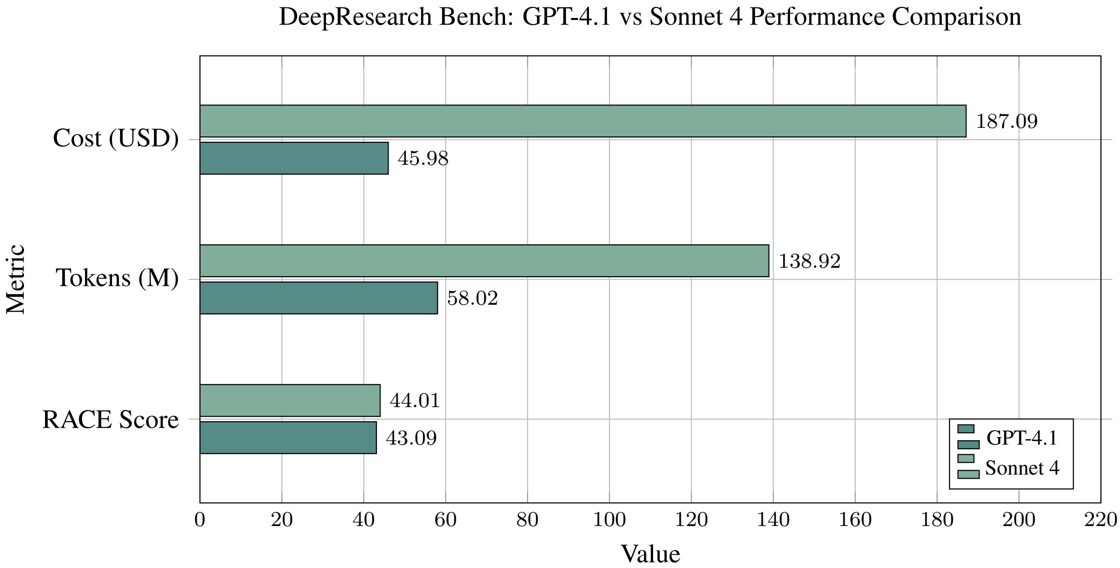


FIGURE 5.2: Cost, token usage, and RACE score comparison between GPT-4.1 and Sonnet 4 models against DeepResearch Bench.

Single-Pass Execution of Research Workflows

The primary limitation of the methodology is the report generation against the DeepResearch Bench only being run once for each agent configuration and architectural component of the proposed architecture. Yet, the stochastic nature of LLMs means that a single run per task cannot establish statistical significance. Performance variations may reflect random seed fluctuations rather than genuine architectural differences, limiting the reliability of comparative conclusions. However, this was unavoidable given the high cost of up to \$90.60 per execution.

Reliance on a Lightweight Judge Model

The DeepResearch Bench evaluation framework officially uses Gemini 2.5 Pro, though this thesis used the less capableGemini 2.5 Flash as a cost-saving measure. This introduces a risk of evaluation compression, where the judge may lack sufficient reasoning depth to discriminate between high-quality and superior-quality outputs or to detect subtle hallucinations. However, given the structured nature of both evaluation frameworks and non-extreme difference between the two models, this remains a minor limitation.

5.7.2 Limitations of Proposed Architecture

The architectural components proposed in this thesis face two fundamental constraints that affect their reliability and adaptability in production environments.

Circular Reasoning Without Ground Truth

The Research Reviewer uses an LLM (Gemini 2.0 Flash) to validate another LLM's (GPT-4o Mini) output, creating a validation paradox. The reviewer model itself may introduce evaluation errors, apply inconsistent assessment criteria, or misclassify valid outputs as inadequate. Without a mechanism to validate the reviewer's own judgments, the system risks iteratively refining toward the reviewer's biases rather than objective quality. False negatives waste computational resources on unnecessary refinement cycles, while false positives allow flawed research outputs to pass through unchecked.

No Corrective Escalation in Adaptive Model Selection

The AMS component makes a single routing decision at delegation time without provision for mid-task correction. If a low-tier model is assigned but proves inadequate, the system cannot escalate to a higher-tier model during execution. This inflexibility creates a critical point of failure: incorrect complexity assessments lead to inevitable quality degradation, as the architecture cannot recover from suboptimal routing decisions.

5.8 Summary of Results

TODO: complete

CHAPTER 6

Conclusions & Future Work

6.1 Conclusions

6.2 Contributions & Implications

6.3 Future Work

Deep Agents

Using deep agents.

Training Models

Train a model for the adaptive model selection.

Application

Apply in a real enterprise setting.

Increased Evaluation

Introducing new benchmarks.

References

[1] Y. Huang, Y. Chen, H. Zhang, K. Li, H. Zhou, M. Fang, L. Yang, X. Li, L. Shang, S. Xu, J. Hao, K. Shao, and J. Wang, "Deep research agents: A systematic examination and roadmap," *arXiv preprint*, 2025.

[2] C.-P. Hsieh, S. Sun, S. Kriman, S. Acharya, D. Rekesh, F. Jia, Y. Zhang, and B. Ginsburg, "Ruler: What's the real context size of your long-context language models?," *arXiv preprint*, 2024.

[3] N. F. Liu, K. Lin, J. Hewitt, A. Paranjape, M. Bevilacqua, F. Petroni, and P. Liang, "Lost in the middle: How language models use long contexts," *arXiv preprint*, 2023.

[4] Y. Gao, Y. Xiong, X. Gao, K. Jia, J. Pan, Y. Bi, Y. Dai, J. Sun, M. Wang, and H. Wang, "Retrieval-augmented generation for large language models: A survey," *arXiv preprint*, 2024.

[5] H. Derouiche, Z. Brahmi, and H. Mazeni, "Agentic ai frameworks: Architectures, protocols, and design challenges," *arXiv preprint*, 2025.

[6] M. Vaccaro, A. Almaatouq, and T. Malone, "When are combinations of humans and ai useful: A systematic review and meta-analysis," *arXiv preprint*, 2024.

[7] D. Dellermann, A. Calma, N. Lipusch, T. Weber, S. Weigel, and P. Ebel, "The future of human-ai collaboration: a taxonomy of design knowledge for hybrid intelligence systems," *arXiv preprint*, 2021.

[8] Y. Wang and Y. Lu, "Interaction, process, infrastructure: A unified architecture for human-agent collaboration," *a*, 2025.

[9] Y. B. et al., "Constitutional ai: Harmlessness from ai feedback," *arXiv preprint*, 2022.

[10] P. G. et al., "Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context," *arXiv preprint*, 2024.

[11] R. Xu and J. Peng, "A comprehensive survey of deep research: Systems, methodologies, and applications," *arXiv preprint*, 2025.

[12] X. Li, J. Jin, G. Dong, H. Qian, Y. Wu, J.-R. Wen, Y. Zhu, and Z. Dou, "Webthinker: Empowering large reasoning models with deep research capability," *arXiv preprint*, 2025.

[13] LangChain, "Open deep research." `https://blog.langchain.com/open-deep-research/`, 2025.

[14] S. Bolshchikov, "Langchain open deep research internals: A step-by-step architecture guide." `https://www.bolshchikov.com/p/open-deep-research-internals-a-step`, 2025.

[15] M. Du, B. Xu, C. Zhu, X. Wang, and Z. Mao, "Deepresearch bench: A comprehensive benchmark for deep research agents," *arXiv preprint*, 2025.

[16] M. AI, "Deepresearch bench leaderboard." https://huggingface.co/spaces/muset-ai/DeepResearch-Bench-Leaderboard, 2025.

[17] LangChain, "Open deep research." GitHub repository, https://github.com/langchain-ai/open_deep-research, 2025.

[18] R. Sapkota, K. I. Roumeliotis, and M. Karkee, "Ai agents vs. agentic ai: A conceptual taxonomy, applications and challenges," *arXiv preprint*, 2025.

[19] S. Mehta, "Beyond accuracy: A multi-dimensional framework for evaluating enterprise agentic ai systems," *arXiv preprint*, 2025.

[20] M. Y. Jaradeh and S. Auer, "Deep research in the era of agentic ai: Requirements and limitations for scholarly research," in *CEUR Workshop Proceedings*, vol. 4065, 2025.

[21] S. M. Mousavi, S. Alghisi, and G. Riccardi, "Llms as repositories of factual knowledge: Limitations and solutions," *arXiv preprint*, 2025.

[22] A. Alansari and H. Luqman, "Large language models hallucination: A comprehensive survey," *arXiv preprint*, 2025.

[23] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Kttler, M. Lewis, W. tau Yih, T. Rocktschel, S. Riedel, and D. Kiela, "Retrieval-augmented generation for knowledge-intensive nlp tasks," *arXiv preprint*, 2021.

[24] Y. Huang and J. Huang, "The survey on retrieval-augmented text generation for large language models," *arXiv preprint*, 2024.

[25] T. Yu, W. Zhou, L. Leiyang, A. Shukla, M. Mmadugula, P. Gundecha, N. Burnett, A. Xu, V. Viseth, T. Tbar, R. Akkiraju, and V. Zhang, "Ekrag: Benchmark rag for enterprise knowledge question answering," *ACL Anthology*, 2025.

[26] T. Bruckhaus, "Rag does not work for enterprises," *arXiv preprint*, 2024.

[27] D. Cohen, L. Burg, S. Pykhnivskyi, H. Gur, S. Kovynov, O. Atzmon, and G. Barkan, "Wixqa: A multi-dataset benchmark for enterprise retrieval-augmented generation," *arXiv preprint*, 2025.

[28] D. B. Acharya, K. Kuppan, and B. Divya, "Agentic ai: Autonomous intelligence for complex goalsa comprehensive survey," *IEEE Access*, 2025.

[29] V.Botti, "Agentic ai and multiagentic: Are we reinventing the wheel?," *arXiv preprint*, 2025.

[30] T. Petrova, B. Bliznioukov, A. Puzikov, and R. State, "From semantic web and mas to agentic ai: A unified narrative of the web of agents," *arXiv preprint*, 2025.

[31] Amazon Web Services, "Building serverless architectures for agentic ai on aws," tech. rep., AWS Prescriptive Guidance, 2025.

[32] F. Tian, A. Luo, J. Du, X. Xian, R. Specht, G. Wang, X. Bi, J. Zhou, A. Kundu, J. Srinivasa, C. Fleming, R. Zhang, Z. Liu, M. Hong, and J. Ding, "An outlook on the opportunities and challenges of multi-agent ai systems," *arXiv preprint*, 2025.

[33] W. Cai, T. Zhu, J. Niu, R. Hu, L. Li, T. Wang, X. Dai, W. Shen, and L. Zhang, "Lightagent: Production-level open-source agentic ai framework," *arXiv preprint*, 2025.

[34] D. Panchal, "Simpliflow: A lightweight open-source framework for rapid creation and deployment of generative agentic ai workflows," *arXiv preprint*, 2025.

[35] S. Murugesan, "The rise of agentic ai: Implications, concerns, and the path forward," *IEEE Access*, 2025.

[36] A. Roychoudhury, C. Pasareanu, M. Pradel, and B. Ray, "Agentic ai software engineers: Programming with trust," *arXiv preprint*, 2025.

[37] R. Zhang, G. Liu, Y. Liu, C. Zhao, J. Wang, Y. Xu, D. Niyato, J. Kang, Y. Li, S. Mao, S. Sun, X. Shen, and D. I. Kim, "Toward edge general intelligence with agentic ai and agentification: Concepts, technologies, and future directions," *arXiv preprint*, 2025.

[38] H. Axelsen, V. Licht, and J. Damsgaard, "Agentic ai for financial crime compliance," *arXiv preprint*, 2025.

[39] A. K. Pati, "Agentic ai: A comprehensive survey of technologies, applications, and societal implications," *IEEE Access*, 2025.

[40] E. Iyidogan and A. I. Ozkes, "Agentic ai and hallucinations," *arXiv preprint*, 2025.

[41] N. Balic, "Will agents replace us? perceptions of autonomous multi-agent ai," *arXiv preprint*, 2025.

[42] M. Shukla, "Adaptive monitoring and real-world evaluation of agentic ai systems," *arXiv preprint*, 2025.

[43] Q. Zheng, "The evolution and optimization of game ai: From rule-driven to deep reinforcement learning," *Applied and Computational Engineering*, 2025.

[44] V. Vassilev, "Is the rule-based order in ai dead, or is still kicking?," *Journal of Robotics and Automation Research*, 2025.

[45] Y. Gui, D. Tang, Z. Haihua, and Y. Zhang, "Dynamic scheduling for flexible job shop using a deep reinforcement learning approach," *Computers and Industrial Engineering*, 2023.

[46] T. H. Tan, "Rule-based vs. ai-driven: Comparing polyaqg framework and generative ai models," *preprint*, 2025.

[47] D. T. K. Ng, C. W. Tan, and J. K. L. Leung, "Empowering student selfregulated learning and science education through chatgpt: A pioneering pilot study," *British Journal of Educational Technology*, 2024.

[48] V. Vats, M. B. Nizam, M. Liu, Z. Wang, R. Ho, M. S. Prasad, V. Titterton, S. V. Malreddy, R. Aggarwal, Y. Xu, L. Ding, J. Mehta, N. Grinnell, L. Liu, S. Zhong, D. N. Gandamani, X. Tang, R. Ghosalkar, C. Shen, R. Shen, N. Hussain, K. Ravichandran, and J. Davis, "A survey on human-ai collaboration with large foundation models," *arXiv preprint*, 2025.

[49] J. Senoner, S. Schallmoser, B. Kratzwald, S. Feuerriegel, and T. Netland, "Explainable ai improves task performance in human-ai collaboration," *arXiv preprint*, 2024.

[50] F. Krause, H. Paulheim, E. Kiesling, K. Kurniawan, M. C. Leva, H. D. Estrada-Lugo, G. Stbl, N. K. re, J. Dominguez-Ledo, M. Khan, P. Demolder, H. Gaux, B. Heinzl, T. Hoch, J. Martinez-Gil, A. Silvina, and B. A. Moser, "Managing human-ai collaborations within industry 5.0 scenarios via knowledge graphs: key challenges and lessons learned," *Frontiers in Artificial Intelligence*, 2024.

[51] A. Ait, J. L. C. Izquierdo, and J. Cabot, "Towards modeling human-agentic collaborative workflows: A bpmn extension," *arXiv preprint*, 2025.

[52] D. B. Kim, M. S. Bajestani, G. Shao, and A. T. Jones, "Conceptual architecture of digital twin with human-in-the-loop-based smart manufacturing," *Advanced Manufacturing*, 2024.

[53] C. Spera and G. Agrawal, "Reversing the paradigm: Building ai-first systems with human guidance," *arXiv preprint*, 2025.

[54] K. Tallam, "From autonomous agents to integrated systems, a new paradigm: Orchestrated distributed intelligence," *arXiv preprint*, 2025.

[55] OpenAI, "Introducing deep research." https://openai.com/index/introducing-deep-research/, 2025.

[56] T. Verge, "Anthropic launches research tool and google workspace integration." https://www.theverge.com/ai-artificial-intelligence/648595/anthropic-claude-research-google-workplace, 2025.

[57] OpenAI, "Webgpt: Improving the factual accuracy of language models through web browsing." https://openai.com/index/webgpt/, 2021.

[58] R. Nakano, J. Hilton, S. Balaji, J. Wu, L. Ouyang, C. Kim, C. Hesse, S. Jain, V. Kosaraju, W. Saunders, X. Jiang, K. Cobbe, T. Eloundou, G. Krueger, K. Button, M. Knight, B. Chess, and J. Schulman, "Webgpt: Browser-assisted question-answering with human feedback," *arXiv preprint*, 2021.

[59] OpenAI, "Chatgpt plugins." https://openai.com/index/chatgpt-plugins/, 2023.

[60] Anthropic, "Cintroducing 100k context windows." https://www.anthropic.com/news/100k-context-windows, 2023.

[61] OpenAI, "Gpt-4." https://openai.com/index/gpt-4-research/, 2023.

[62] OpenAI, "Function calling and other api updates." https://openai.com/index/function-calling-and-other-api-updates/, 2023.

[63] W. Zhang, Y. Li, Y. Bei, J. Luo, G. Wan, L. Yang, C. Xie, Y. Yang, W.-C. Huang, C. Miao, H. P. Zou, X. Luo, Y. Zhao, Y. Chen, C. Chan, P. Zhou, X. Zhang, C. Zhang, J. Shang, M. Zhang, Y. Song, I. King, and P. S. Yu, "From web search towards agentic deep research: Incentivizing search with reasoning agents," *arXiv preprint*, 2025.

[64] W. Zhang, X. Li, Y. Zhang, P. Jia, Y. Wang, H. Guo, Y. Liu, and X. Zhao, "Deep research: A survey of autonomous research agents," *arXiv preprint*, 2025.

[65] Z. Shi, Y. Chen, H. Li, W. Sun, S. Ni, and Y. Lyu, "Deep research: A systematic survey," *arXiv preprint*, 2025.

[66] R. Wang, C. Zhang, J.-Y. Ma, J. Zhang, H. Wang, Y. Chen, B. Xue, T. Fang, Z. Zhang, H. Zhang, H. Mi, D. Yu, and K.-F. Wong, "Explore to evolve: Scaling evolved aggregation logic via proactive online exploration for deep research agents," *arXiv preprint*, 2025.

[67] K. Dong, S. Huang, F. Ye, W. Han, Z. Zhang, D. Li, W. Li, Q. Yang, G. Wang, Y. Wang, C. Zhang, and Y. Liu, "Doc-researcher: A unified system for multimodal document parsing and deep research," *arXiv preprint*, 2025.

[68] H. Yen, A. Paranjape, M. Xia, T. Venkatesh, J. Hessel, D. Chen, and Y. Zhang, "Lost in the maze: Overcoming context limitations in long-horizon agentic search," *arXiv preprint*, 2025.

[69] W. Fan, W. Yao, Z. Li, F. Yao, X. Liu, L. Qiu, Q. Yin, Y. Song, and B. Yin, "Deepplanner: Scaling planning capability for deep research agents via advantage shaping," *arXiv preprint*, 2025.

[70] Y. Deng, G. Wang, Z. Ying, X. Wu, J. Lin, W. Xiong, Y. Dai, S. Yang, Z. Zhang, Q. Wang, Y. Qin, Y. Wang, Q. Zha, S. Dai, and C. Meng, "Atom-searcher: Enhancing agentic deep research via fine-grained atomic thought reward," *arXiv preprint*, 2025.

[71] Y. Wan, J. Wang, L. Li, J. Liu, R. Zhu, and Z. Zhu, "Pokeeresearch: Effective deep research via reinforcement learning from ai feedback and robust reasoning scaffold," *arXiv preprint*, 2025.

[72] L. Ye, X. Cai, X. Wang, J. Wang, X. Hu, J. Su, Y. Nan, S. Wang, B. Zhang, X. Fan, J. Luo, Y. Zheng, T. Xu, D. Fu, Y. Wu, P. Lu, Z. Wang, Y. Qin, Z. Huang, Y. Ma, Z. Hu, H. Zou, T. Mi, Y. Ye, E. Chern, and P. Liu, "Interaction as intelligence: Deep research with human-ai partnership," *arXiv preprint*, 2025.

[73] A. Prabhakar, R. Ram, Z. Chen, S. Savarese, F. Wang, C. Xiong, H. Wang, and W. Yao, "Enterprise deep research: Steerable multi-agent deep research for enterprise analytics," *arXiv preprint*, 2025.

[74] J. Jin, Y. Zhang, Y. Xu, H. Qian, Y. Zhu, and Z. Dou, "Finsight: Towards real-world financial deep research," *arXiv preprint*, 2025.

[75] S. Chen, Z. Li, Z. Han, B. He, T. Liu, H. Chen, G. Groh, P. Torr, V. Tresp, and J. Gu, "Deep research brings deeper harm," *arXiv preprint*, 2025.

[76] A. Wang, A. Singh, J. Michael, F. Hill, O. Levy, and S. R. Bowman, "Glue: A multi-task benchmark and analysis platform for natural language understanding," *arXiv preprint*, 2018.

[77] A. Wang, Y. Pruksachatkun, N. Nangia, A. Singh, J. Michael, F. Hill, O. Levy, and S. R. Bowman, "Superglue: A stickier benchmark for general-purpose language understanding systems," *arXiv preprint*, 2020.

[78] D. Hendrycks, C. Burns, S. Basart, A. Zou, M. Mazeika, D. Song, and J. Steinhardt, "Measuring massive multitask language understanding," *arXiv preprint*, 2021.

[79] A. S. et al., "Beyond the imitation game: Quantifying and extrapolating the capabilities of language models," *arXiv preprint*, 2022.

[80] L. P. et al., "Humanity's last exam," *arXiv preprint*, 2025.

[81] C. Rosset, H.-L. Chung, G. Qin, E. Chau, Z. Feng, A. Awadallah, J. Neville, and N. Rao, "Researchy questions: A dataset of multi-perspective, decompositional questions for deep research," *Association for Computing Machinery*, 2025.

[82] A. Java, A. Khandelwal, S. Midigeshi, A. Halfaker, A. Deshpande, N. Goyal, A. Gupta, N. Natarajan, and A. Sharma, "Characterizing deep research: A benchmark and formal definition," *arXiv preprint*, 2025.

[83] J. Wang, Y. Ming, R. Dulepet, Q. Chen, A. Xu, Z. Ke, F. Sala, A. Albarghouthi, C. Xiong, and S. Joty, "Liveresearchbench: A live benchmark for user-centric deep research in the wild," *arXiv preprint*, 2025.

[84] N. I. Bosse, J. Evans, R. G. Gambee, D. Hnyk, P. Mhlbacher, L. Phillips, D. Schwarz, and J. Wildman, "Deep research bench: Evaluating ai web research agents," *arXiv preprint*, 2025.

[85] Z. Chen, X. Ma, S. Zhuang, P. Nie, K. Zou, , A. Liu, J. Green, K. Patel, R. Meng, M. Su, S. Sharifymoghaddam, Y. Li, H. Hong, X. Shi, X. Liu, N. Thakur, C. Zhang, L. Gao, W. Chen, and J. Lin, "Browsecomp-plus: A more fair and transparent evaluation benchmark of deep-research agent," *arXiv preprint*, 2025.

[86] I. A. Azime, T. D. Belay, and A. L. Tonja, “Evaluation sheet for deep research: A use case for academic survey writing,” *arXiv preprint*, 2025.

[87] P. Chandrahasan, J. Jin, Z. Zhang, T. Wang, A. Tang, L. Mo, M. Ziyadi, L. F. Ribeiro, Z. Qiu, M. Dreyer, A. Asai, and C. Xiong, “Deep research comparator: A platform for fine-grained human annotations of deep research agents,” *arXiv preprint*, 2025.

[88] P. K. Choubey, X. Peng, S. Bhagavath, K.-H. Huang, C. Xiong, and C.-S. Wu, “Benchmarking deep search over heterogeneous enterprise data,” *arXiv preprint*, 2025.

[89] B. Townsend, M. May, K. Mackowiak, and C. Wells, “Realkie: Five novel datasets for enterprise key information extraction,” *arXiv preprint*, 2025.

[90] A. Abaskohi, T. Chen, M. Muoz-Mrmol, C. Fox, A. V. Ramesh, tienne Marcotte, X. H. L, N. Chapados, S. Gella, C. Pal, A. Drouin, and I. H. Laradji, “Drbench: A realistic benchmark for enterprise deep research,” *arXiv preprint*, 2025.

[91] J. Wei, H. Zhou, X. Zhang, D. Zhang, Z. Qiu, W. Wei, J. Li, W. Ouyang, and S. Sun, “Alignrag: An adaptable framework for resolving misalignments in retrieval-aware reasoning of rag,” *arXiv preprint*, 2025.

[92] A. Cattan, A. Jacovi, O. Ram, J. Herzig, R. Aharoni, S. Goldshtein, E. Ofek, I. Szpektor, and A. Caciularu, “Dragged into conflicts: Detecting and addressing conflicting sources in search-augmented llms,” *arXiv preprint*, 2025.

[93] Y. Zheng, X. Liu, P. Wu, and L. Pan, “Crave: A conflicting reasoning approach for explainable claim verification using llms,” *arXiv preprint*, 2025.

[94] A. Asai, Z. Wu, Y. Wang, A. Sil, and H. Hajishirzi, “Self-rag: Learning to retrieve, generate, and critique through self-reflection,” *arXiv preprint*, 2023.

[95] G. Tie, Z. Yuan, Z. Zhao, C. Hu, T. Gu, R. Zhang, S. Zhang, J. Wu, X. Tu, M. Jin, Q. Wen, L. Chen, P. Zhou, and L. Sun, “Can llms correct themselves? a benchmark of self-correction in llms,” *arXiv preprint*, 2025.

[96] S. Wang, J. R. Foulds, M. O. Gani, and S. Pan, “Llm-based corroborating and refuting evidence retrieval for scientific claim verification,” *arXiv preprint*, 2025.

[97] B. Zhu, Y. Sheng, L. Zheng, C. Barrett, M. I. Jordan, and J. Jiao, “On optimal caching and model multiplexing for large model inference,” *arXiv preprint*, 2023.

[98] Y. Kim, C. Park, and H. Jeong, “Mdagents: An adaptive collaboration of llms for medical decision-making,” *arXiv preprint*, 2024.

[99] M. Yue, J. Zhao, M. Zhang, L. Du, and Z. Yao, "Large language model cascades with mixture of thoughts representations for cost-efficient reasoning," *arXiv preprint*, 2024.

[100] D. Ding, A. Mallick, and C. Wang, "Hybrid llm: Cost-efficient and quality-aware query routing," *arXiv preprint*, 2024.

[101] S. Agrawal and P. Gupta, "Llmrank: Understanding llm strengths for model routing," *arXiv preprint*, 2025.

[102] M. J. Zellinger, R. Liu, and M. Thomson, "Cost-saving llm cascades with early abstention," *arXiv preprint*, 2025.

[103] L. Chen, M. Zaharia, and J. Zou, "Frugalgpt: How to use large language models while reducing cost and improving performance," *arXiv preprint*, 2023.

[104] OpenRouter, "Openai: Gpt-4o-mini." https://openrouter.ai/openai/gpt-4o-mini, 2025.

[105] OpenRouter, "Openai: Gpt-4.1 mini." https://openrouter.ai/openai/gpt-4.1-mini, 2025.

[106] OpenRouter, "Openai: Gpt-5." https://openrouter.ai/openai/gpt-5, 2025.

[107] J. Wei, Y. Tay, and R. Bommasani, "Emergent abilities of large language models," *arXiv preprint*, 2022.

[108] J. Wei, X. Wang, and D. Schuurmans, "Chain-of-thought prompting elicits reasoning in large language models," *arXiv preprint*, 2023.

[109] X. Zhao, T. Xu, and X. Wang, "Boosting llm reasoning via spontaneous self-correction," *arXiv preprint*, 2025.

[110] M. AI, "Kimi-researcher." https://moonshotai.github.io/Kimi-Researcher/, 2025.

[111] Anthropic, "How we built our multi-agent research system." https://www.anthropic.com/engineering/multi-agent-research-system, 2025.

[112] P. Lin, "Self-balancing agentic ai: Test-time diffusion and context engineering re-imagined for deep research." https://paichunlin.substack.com/p/self-balancing-agentic-ai-test-time, 2025.

[113] B. Li, B. Zhang, and D. Zhang, "Tongyi deepresearch technical report," *arXiv preprint*, 2025.

[114] S. Lin, J. Hilton, and O. Evans, "Truthfulqa: Measuring how models mimic human falsehoods," *arXiv preprint*, 2022.

[115] J. Thorne, A. Vlachos, C. Christodoulopoulos, and A. Mittal, “Fever: a large-scale dataset for fact extraction and verification,” *arXiv preprint*, 2018.

APPENDIX A

Work Health and Safety

This appendix outlines the Work Health and Safety (WHS) management plan implemented during the completion of this thesis at Constantinople. Given the primarily office-based nature of the project, the risk assessment focuses on hazards specific to this environment, including mental wellbeing, cybersecurity, computer use, electrical safety, and workplace aggression. All operations were conducted in adherence to the relevant legislative framework, specifically the *Work Health and Safety Act 2011* (NSW), the national *Work Health and Safety Act 2011* (Cth), and the *Work Health and Safety Regulation 2017* (NSW). To systematically assess the identified hazards, the Risk Rating Matrix shown in Figure A.1 was utilised. This matrix determines risk levels through the product of likelihood and consequence, producing ratings that range from Low (1-3) to Extreme (15-25).

		Impact				
		Negligible 1	**Minor 2**	**Moderate 3**	**Major 4**	**Catastrophic 5**
Likelihood	**Almost certain 5**	**Moderate** 5	**High** 10	**Extreme** 15	**Extreme** 20	**Extreme** 25
	Likely 4	**Moderate** 4	**High** 8	**High** 12	**Extreme** 16	**Extreme** 20
	Possible 3	**Low** 3	**Moderate** 6	**High** 9	**High** 12	**Extreme** 15
	Unlikely 2	**Low** 2	**Moderate** 4	**Moderate** 6	**High** 8	**High** 10
	Rare 1	**Low** 1	**Low** 2	**Low** 3	**Moderate** 4	**Moderate** 5

FIGURE A.1: Risk Rating Matrix

A.1 Mental Wellbeing

The primary hazards identified in this domain relate to the psychosocial stressors inherent in balancing high-level academic requirements with professional workplace obligations. Specifically, this includes the potential for conflicting deadlines between thesis milestones and project deliverables. Additionally, the sedentary nature of office-based work and the potential for social isolation during remote working days were identified as contributing factors to mental and physical fatigue.

Failure to manage these hazards effectively could result in psychological distress, including anxiety, burnout, and stress-related fatigue. Physical impacts may include musculoskeletal strain from prolonged sedentary periods. These outcomes could negatively affect both academic performance, specifically thesis quality and submission timeliness, and professional efficacy.

Given the dual pressures of the placement and the academic thesis, the emergence of stress-related issues is a plausible scenario without intervention, leading to a likelihood assessment of Possible (3). The consequences of unmanaged stress could moderately impact the timely completion of the thesis and overall wellbeing, resulting in an impact assessment of Moderate (3). This yields an initial risk rating of High (9).

Several controls were implemented to mitigate these risks. Regular contact was maintained with two workplace managers and a product manager who remained approachable and supportive. A flexible approach allowed for the prioritization of thesis work over professional tasks as deadlines approached. Continuous communication with the academic supervisor via Slack and engagement with a peer network of students provided advice and reassurance. A standard 38-hour work week was strictly adhered to, with no expectation of overtime, ensuring adequate recovery time. The office environment was leveraged to prevent isolation, with daily social interactions such as team lunches. Remote work isolation was framed positively as focused time, preventing feelings of loneliness. Regular breaks were taken to walk around the office, hydrate, and interact with colleagues, counteracting the sedentary nature of the role.

The comprehensive support system and flexible working arrangements effectively reduced the likelihood of significant stress to a rare occurrence (1) with minor potential impacts (2). Consequently, the residual risk rating is assessed as Low (2).

A.2 Cybersecurity & Data Protection

The primary hazards in this domain involved the handling of sensitive internal data, specifically employee meeting transcripts and chatbot interactions. There was a risk of unauthorized access to personal transcripts or exposure of confidential employee data through the Model Context Protocol (MCP) server or the internal chatbot. Additionally, the mismanagement of API credentials posed a security risk.

A failure in data protection could lead to significant privacy breaches, unauthorized disclosure of confidential business information, and non-compliance with data privacy regulations. This would result in reputational damage to the sponsor company and potential legal consequences.

Without robust controls, the likelihood of a data breach or access error during development was considered Possible (3). The consequences of such a breach were assessed as Moderate (3), resulting in an initial risk rating of High (9).

To mitigate these risks, strict access controls and architectural safeguards were implemented. User isolation was enforced at both the application and API levels to ensure employees could only access their own data. AWS Bedrock services were utilized to securely manage chatbot data, and API credentials were stored exclusively in AWS Secrets Manager. Access to production environments was governed by

AWS role-based access control (RBAC), requiring specific approval for any data modifications. Work was conducted on a company-issued MacBook Pro with enforced security protocols. Furthermore, mandatory completion of "General Security Training & Policy Acceptance" and "Privacy Compliance Training" ensured awareness of compliance obligations.

These measures effectively reduced the likelihood of a security incident to Rare (1). With the potential impact remaining Moderate (3), the residual risk rating is assessed as Low (3).

A.3 Computer Use

The primary hazards associated with this domain relate to ergonomic risks inherent in prolonged computer use. These include sustained sedentary postures, repetitive strain from keyboard and mouse usage, and extended periods of screen time. The lack of a structured break schedule and potential dehydration were also identified as contributing factors to physical discomfort.

Potential impacts include musculoskeletal disorders such as back, neck, and wrist strain, as well as eye strain and headaches. Specifically, headaches were noted as a symptom, often exacerbated by insufficient hydration. These physical ailments could reduce productivity, increase fatigue, and negatively impact overall health and wellbeing.

Given the requirement for 7-8 hours of daily computer use with extended periods of continuous screen time, the likelihood of experiencing ergonomic-related discomfort was assessed as Likely (4). The impact of such issues, while manageable, could be Minor (2), resulting in an initial risk rating of High (8).

To control these risks, a comprehensive ergonomic setup was utilized. The office environment provided adjustable chairs, adjustable monitor stands, and external peripherals, with occasional access to standing desks. Remote work was conducted using a personal standing desk, ensuring consistent ergonomic standards. Although a strict break schedule was not followed, frequent ad-hoc breaks were taken to stand, stretch, and interact with colleagues, breaking up sedentary periods. Hydration was actively managed to address headaches, serving as an immediate corrective action.

The implementation of ergonomic equipment and spontaneous physical activity effectively mitigated the risks. The likelihood of persistent injury was reduced to Unlikely (2), and the impact remained Minor (2). Consequently, the residual risk rating is assessed as Low (4).

A.4 Electrical Safety Hazards

Hazards in this domain were primarily associated with the office environment's high density of electrical equipment, including numerous computer workstations with dual monitors. The sheer volume of cabling presented potential risks for tripping and electrical faults. Additionally, the possibility of equipment overheating, particularly given the local climate conditions, was identified as a potential environmental hazard.

The potential impacts of these hazards range from minor injuries due to trips and falls to more severe consequences such as electric shock or fire. Equipment failure due to overheating could also result in data loss and significant disruption to work continuity.

Given the modern and well-maintained nature of the office, the likelihood of an electrical incident was assessed as Possible (3). The potential impact of an electric shock or fire is considered Major (4), resulting in an initial risk rating of High (12).

Control measures focused on effective cable management and environmental regulation. Cables were systematically secured and routed under desks to eliminate trip hazards and protect connections. The office climate was controlled to prevent equipment overheating, mitigating the risk posed by external weather conditions. Furthermore, a clear reporting protocol was established, requiring immediate notification of colleagues and the Workplace Experience Manager in the event of an electrical hazard. Although no incidents occurred, this procedure ensured a rapid response capability.

These controls significantly reduced the probability of an incident. The likelihood is re-assessed as Rare (1), with the potential impact remaining Major (4) should an event occur. The residual risk rating is therefore Low (4).

A.5 Workplace Aggression

Hazards in this domain were restricted to internal interactions, as the role did not involve contact with external customers or the public. The primary risks included potential bullying, verbal abuse, or interpersonal conflict with colleagues or supervisors. While the initial impression of the workplace suggested a low probability of such events, the inherent nature of professional environments necessitates the consideration of human behavioral risks.

The potential impacts of workplace aggression include significant psychological distress, such as anxiety and depression, leading to reduced morale and productivity. In severe cases, this could result in resignation or long-term reputational damage to the organization.

Identifying the inherent risk in a standard office setting, the likelihood of interpersonal conflict was initially assessed as Possible (3). The potential psychological impact was considered Moderate (3), resulting in an initial risk rating of High (9).

Control measures were deeply embedded in the organizational structure and culture. A Code of Conduct was signed upon commencement, explicitly prohibiting workplace aggression. This was reinforced by an exceptionally positive and inclusive workplace culture, facilitated by a demographic peer group that ensured a welcoming integration. Furthermore, clear and accessible reporting channels were established; two direct managers and a product manager were available to address and escalate any concerns.

These cultural and procedural controls effectively mitigated the risk. No incidents of aggression were witnessed or experienced. The likelihood is re-assessed as Rare (1), with the potential impact remaining Moderate (3). The residual risk rating is therefore Low (3).

A.6 Summary

A summary of the identified hazards and their respective risk assessments is presented in Figure A.2. As demonstrated, the implementation of targeted control measures effectively reduced the residual risk for all categories to a 'Low' rating, ensuring a safe and compliant research environment.

Hazard	**Likelihood**	**Impact**	**Initial Risk**	**Residual Risk**
Mental Wellbeing	Possible (3)	Moderate (3)	**High (9)**	**Low (2)**
Cybersecurity & Data	Possible (3)	Moderate (3)	**High (9)**	**Low (3)**
Computer Use	Likely (4)	Minor (2)	**High (8)**	**Low (4)**
Electrical Safety	Possible (3)	Major (4)	**High (12)**	**Low (4)**
Workplace Aggression	Possible (3)	Moderate (3)	**High (9)**	**Low (3)**

FIGURE A.2: Risk Register Summary

Appendix B

UoS Case Studies

B.1 COMP3927: Algorithm Design (Advanced)

This unit provides an introduction to the design techniques that are used to find efficient algorithmic solutions for given problems. The techniques covered include greedy, divide-and-conquer, dynamic programming, and adjusting flows in networks. Students will extend their skills in algorithm analysis. The unit also provides an introduction to the concepts of computational complexity and reductions between problems.[1]

This project addressed seven of the nine COMP3927 learning objectives.

LO1 Produce a clear account of an algorithm, that would allow others to understand and implement it.

LO2 Learn about new algorithms by searching for descriptions in textbooks or online.

LO3 Read, understand, analyze and modify a given algorithm, as well as design efficient algorithmic solutions for given problems and evaluate the proposal.

LO4 Demonstrate basic experience of implementing algorithms.

LO5 Analyze the complexity of a given algorithm.

LO6 Demonstrate knowledge of fundamental algorithms for several problems, especially graph problems, testing graph properties and solving optimization problems on graphs, as well as knowledge of fundamental general algorithmic design techniques, such as greedy, dynamic programming and divide-and-conquer.

LO7 Understand the fundamental concepts of computational hardness.

[1] Unit of study description copied verbatim from the official outline.

LO1: Produce a clear account of an algorithm, that would allow others to understand and implement it.

Addressable.

LO2: Learn about new algorithms by searching for descriptions in textbooks or online.

Addressable.

LO3: Read, understand, analyze and modify a given algorithm, as well as design efficient algorithmic solutions for given problems and evaluate the proposal.

Addressable.

LO4: Demonstrate basic experience of implementing algorithms.

Addressable.

LO5: Analyze the complexity of a given algorithm.

Addressable.

LO6: Demonstrate knowledge of fundamental algorithms for several problems, especially graph problems, testing graph properties and solving optimization problems on graphs, as well as knowledge of fundamental general algorithmic design techniques, such as greedy, dynamic programming and divide-and-conquer.

Not Addressable.

LO7: Understand the fundamental concepts of computational hardness.

Not Addressable.

B.2 COMP4328: Advanced Machine Learning

Machine learning models explain and generalise data. This course introduces some fundamental machine learning concepts, learning problems and algorithms to provide understanding and simple answers to many questions arising from data explanation and generalisation. For example, why do different machine learning models work? How to further improve them? How to adapt them to different purposes?[1]

This project addressed six of the eight COMP4328 learning objectives.

LO1 Present the design and evaluation of a machine learning algorithm, describing the design processes and evaluation.

LO3 Understand and analyse some machine learning algorithms and have some knowledge to further improve them.

LO4 Understand and analyse some machine learning problems and have some knowledge to adapt the existing machine learning models to different purposes.

LO5 Implement machine learning algorithms from peer-reviewed papers.

LO6 Understand the nature of the statistical foundations of designing or adapting learning algorithms.

LO7 Demonstrate knowledge of the introduced machine learning models and the relative strengths and weaknesses of each and their most appropriate uses.

LO1: Present the design and evaluation of a machine learning algorithm, describing the design processes and evaluation.

Addressable.

LO3: Understand and analyse some machine learning algorithms and have some knowledge to further improve them.

Addressable.

LO4: Understand and analyse some machine learning problems and have some knowledge to adapt the existing machine learning models to different purposes.

Addressable.

LO5: Implement machine learning algorithms from peer-reviewed papers.

Addressable.

LO6: Understand the nature of the statistical foundations of designing or adapting learning algorithms.

”On Optimal Caching and Model Multiplexing for Large Model Inference” provides a rigorous theoretical analysis of learning algorithms for caching and model selection in both offline and online settings

”Cost-Saving LLM Cascades with Early Abstention” (Zellinger et al.) explores statistical foundations through probabilistic modeling to tune cascade thresholds.

”Hybrid LLM: Cost-Efficient and Quality-Aware Query Routing” (Ding et al.) addresses the statistical nature of uncertainty in LLM responses to adapt learning algorithms.

LO7: Demonstrate knowledge of the introduced machine learning models and the relative strengths and weaknesses of each and their most appropriate uses.

Addressable.

APPENDIX C

Agent System Prompts

C.1 Langchain Open Deep Research System Prompts

```
transform_messages_into_research_topic_prompt = """You will be given a set
    of messages that have been exchanged so far between yourself and the
    user.
Your job is to translate these messages into a more detailed and concrete
    research question that will be used to guide the research.

The messages that have been exchanged so far between yourself and the user
    are:
<Messages>
{messages}
</Messages>

Today's date is {date}.

You will return a single research question that will be used to guide the
    research.

Guidelines:
1. Maximize Specificity and Detail
- Include all known user preferences and explicitly list key attributes or
    dimensions to consider.
- It is important that all details from the user are included in the
    instructions.

2. Fill in Unstated But Necessary Dimensions as Open-Ended
- If certain attributes are essential for a meaningful output but the user
    has not provided them, explicitly state that they are open-ended or
    default to no specific constraint.

3. Avoid Unwarranted Assumptions
- If the user has not provided a particular detail, do not invent one.
- Instead, state the lack of specification and guide the researcher to
    treat it as flexible or accept all possible options.
"""
```

LISTING C.1: Research supervisor research brief generation system prompt (part 1 of 2).

```
# Continuation of transform_messages_into_research_topic_prompt:
"""
4. Use the First Person
- Phrase the request from the perspective of the user.

5. Sources
- If specific sources should be prioritized, specify them in the research
    question.
- For product and travel research, prefer linking directly to official or
    primary websites (e.g., official brand sites, manufacturer pages, or
    reputable e-commerce platforms like Amazon for user reviews) rather
    than aggregator sites or SEO-heavy blogs.
- For academic or scientific queries, prefer linking directly to the
    original paper or official journal publication rather than survey
    papers or secondary summaries.
- For people, try linking directly to their LinkedIn profile, or their
    personal website if they have one.
- If the query is in a specific language, prioritize sources published in
    that language.
"""
```

LISTING C.2: Research supervisor research brief generation system prompt (part 2 of 2).

```
lead_researcher_prompt = """You are a research supervisor. Your job is to
    conduct research by calling the "ConductResearch" tool. For context,
    today's date is {date}.

<Task>
Your focus is to call the "ConductResearch" tool to conduct research
    against the overall research question passed in by the user.
When you are completely satisfied with the research findings returned from
    the tool calls, then you should call the "ResearchComplete" tool to
    indicate that you are done with your research.
</Task>

<Available Tools>
You have access to three main tools:
1. **ConductResearch**: Delegate research tasks to specialized sub-agents
2. **ResearchComplete**: Indicate that research is complete
3. **think_tool**: For reflection and strategic planning during research

**CRITICAL: Use think_tool before calling ConductResearch to plan your
    approach, and after each ConductResearch to assess progress. Do not
    call think_tool with any other tools in parallel.**
</Available Tools>

<Instructions>
Think like a research manager with limited time and resources. Follow these
     steps:

1. **Read the question carefully** - What specific information does the
    user need?
2. **Decide how to delegate the research** - Carefully consider the
    question and decide how to delegate the research. Are there multiple
    independent directions that can be explored simultaneously?
3. **After each call to ConductResearch, pause and assess** - Do I have
    enough to answer? What's still missing?
</Instructions>

<Hard Limits>
**Task Delegation Budgets** (Prevent excessive delegation):
- **Bias towards single agent** - Use single agent for simplicity unless
    the user request has clear opportunity for parallelization
- **Stop when you can answer confidently** - Don't keep delegating research
     for perfection
- **Limit tool calls** - Always stop after {max_researcher_iterations} tool
     calls to ConductResearch and think_tool if you cannot find the right
    sources

**Maximum {max_concurrent_research_units} parallel agents per iteration**
</Hard Limits>
"""
```

LISTING C.3: Research supervisor research delegation system prompt (part 1 of 2).

```
# Continuation of lead_researcher_prompt:
"""
<Show Your Thinking>
Before you call ConductResearch tool call, use think_tool to plan your
    approach:
- Can the task be broken down into smaller sub-tasks?

After each ConductResearch tool call, use think_tool to analyze the results:

- What key information did I find?
- What's missing?
- Do I have enough to answer the question comprehensively?
- Should I delegate more research or call ResearchComplete?
</Show Your Thinking>

<Scaling Rules>
**Simple fact-finding, lists, and rankings** can use a single sub-agent:
- *Example*: List the top 10 coffee shops in San Francisco  Use 1 sub-agent

**Comparisons presented in the user request** can use a sub-agent for each
    element of the comparison:
- *Example*: Compare OpenAI vs. Anthropic vs. DeepMind approaches to AI
    safety  Use 3 sub-agents
- Delegate clear, distinct, non-overlapping subtopics

**Important Reminders:**
- Each ConductResearch call spawns a dedicated research agent for that
    specific topic
- A separate agent will write the final report - you just need to gather
    information
- When calling ConductResearch, provide complete standalone instructions -
    sub-agents can't see other agents' work
- Do NOT use acronyms or abbreviations in your research questions, be very
    clear and specific
</Scaling Rules>
"""
```

LISTING C.4: Research supervisor research delegation system prompt (part 2 of 2).

```
research_system_prompt = """You are a research assistant conducting
    research on the user’s input topic. For context, today’s date is {date}.


<Task>
Your job is to use tools to gather information about the user’s input topic.

You can use any of the tools provided to you to find resources that can
    help answer the research question. You can call these tools in series or
     in parallel, your research is conducted in a tool-calling loop.
</Task>

<Available Tools>
You have access to two main tools:
1. **tavily_search**: For conducting web searches to gather information
2. **think_tool**: For reflection and strategic planning during research
{mcp_prompt}

**CRITICAL: Use think_tool after each search to reflect on results and plan
     next steps. Do not call think_tool with the tavily_search or any other
     tools. It should be to reflect on the results of the search.**
</Available Tools>

<Instructions>
Think like a human researcher with limited time. Follow these steps:

1. **Read the question carefully** - What specific information does the
    user need?
2. **Start with broader searches** - Use broad, comprehensive queries first
3. **After each search, pause and assess** - Do I have enough to answer?
    What’s still missing?
4. **Execute narrower searches as you gather information** - Fill in the
    gaps
5. **Stop when you can answer confidently** - Don’t keep searching for
    perfection
</Instructions>

<Hard Limits>
**Tool Call Budgets** (Prevent excessive searching):
- **Simple queries**: Use 2-3 search tool calls maximum
- **Complex queries**: Use up to 5 search tool calls maximum
- **Always stop**: After 5 search tool calls if you cannot find the right
    sources

**Stop Immediately When**:
- You can answer the user’s question comprehensively
- You have 3+ relevant examples/sources for the question
- Your last 2 searches returned similar information
</Hard Limits>

<Show Your Thinking>
After each search tool call, use think_tool to analyze the results:
- What key information did I find?
- What’s missing?
- Do I have enough to answer the question comprehensively?
- Should I search more or provide my answer?
</Show Your Thinking>
"""
```

LISTING C.5: Researcher agent system prompt.

```
compress_research_system_prompt = """You are a research assistant that has
    conducted research on a topic by calling several tools and web searches.
     Your job is now to clean up the findings, but preserve all of the
    relevant statements and information that the researcher has gathered.
    For context, today's date is {date}.

<Task>
You need to clean up information gathered from tool calls and web searches
    in the existing messages.
All relevant information should be repeated and rewritten verbatim, but in
    a cleaner format.
The purpose of this step is just to remove any obviously irrelevant or
    duplicative information.
For example, if three sources all say "X", you could say "These three
    sources all stated X".
Only these fully comprehensive cleaned findings are going to be returned to
     the user, so it's crucial that you don't lose any information from the
     raw messages.
</Task>

<Guidelines>
1. Your output findings should be fully comprehensive and include ALL of
    the information and sources that the researcher has gathered from tool
    calls and web searches. It is expected that you repeat key information
    verbatim.
2. This report can be as long as necessary to return ALL of the information
     that the researcher has gathered.
3. In your report, you should return inline citations for each source that
    the researcher found.
4. You should include a "Sources" section at the end of the report that
    lists all of the sources the researcher found with corresponding
    citations, cited against statements in the report.
5. Make sure to include ALL of the sources that the researcher gathered in
    the report, and how they were used to answer the question!
6. It's really important not to lose any sources. A later LLM will be used
    to merge this report with others, so having all of the sources is
    critical.
</Guidelines>

<Output Format>
The report should be structured like this:
**List of Queries and Tool Calls Made**
**Fully Comprehensive Findings**
**List of All Relevant Sources (with citations in the report)**
</Output Format>

<Citation Rules>
- Assign each unique URL a single citation number in your text
- End with ### Sources that lists each source with corresponding numbers
- IMPORTANT: Number sources sequentially without gaps (1,2,3,4...) in the
    final list regardless of which sources you choose
- Example format:
  [1] Source Title: URL
  [2] Source Title: URL
</Citation Rules>

Critical Reminder: It is extremely important that any information that is
    even remotely relevant to the user's research topic is preserved
    verbatim (e.g. don't rewrite it, don't summarize it, don't paraphrase
    it).
"""
```

LISTING C.6: Compression model system prompt

```
final_report_generation_prompt = """Based on all the research conducted,
    create a comprehensive, well-structured answer to the overall research
    brief:
<Research Brief>
{research_brief}
</Research Brief>

For more context, here is all of the messages so far. Focus on the research
     brief above, but consider these messages as well for more context.
<Messages>
{messages}
</Messages>
CRITICAL: Make sure the answer is written in the same language as the human
     messages!
For example, if the user's messages are in English, then MAKE SURE you
    write your response in English. If the user's messages are in Chinese,
    then MAKE SURE you write your entire response in Chinese.
This is critical. The user will only understand the answer if it is written
     in the same language as their input message.

Today's date is {date}.

Here are the findings from the research that you conducted:
<Findings>
{findings}
</Findings>

Please create a detailed answer to the overall research brief that:
1. Is well-organized with proper headings (# for title, ## for sections,
    ### for subsections)
2. Includes specific facts and insights from the research
3. References relevant sources using [Title](URL) format
4. Provides a balanced, thorough analysis. Be as comprehensive as possible,
     and include all information that is relevant to the overall research
    question. People are using you for deep research and will expect
    detailed, comprehensive answers.
5. Includes a "Sources" section at the end with all referenced links

You can structure your report in a number of different ways. Here are some
    examples:

To answer a question that asks you to compare two things, you might
    structure your report like this:
1/ intro
2/ overview of topic A
3/ overview of topic B
4/ comparison between A and B
5/ conclusion

To answer a question that asks you to return a list of things, you might
    only need a single section which is the entire list.
1/ list of things or table of things
Or, you could choose to make each item in the list a separate section in
    the report. When asked for lists, you don't need an introduction or
    conclusion.
1/ item 1
2/ item 2
3/ item 3
"""
```

LISTING C.7: Final report model system prompt (part 1 of 2).

```
# Continuation of final_report_generation_prompt:
"""
To answer a question that asks you to summarize a topic, give a report, or
    give an overview, you might structure your report like this:
1/ overview of topic
2/ concept 1
3/ concept 2
4/ concept 3
5/ conclusion

If you think you can answer the question with a single section, you can do
    that too!
1/ answer

REMEMBER: Section is a VERY fluid and loose concept. You can structure your
     report however you think is best, including in ways that are not
    listed above!
Make sure that your sections are cohesive, and make sense for the reader.

For each section of the report, do the following:
- Use simple, clear language
- Use ## for section title (Markdown format) for each section of the report
- Do NOT ever refer to yourself as the writer of the report. This should be
     a professional report without any self-referential language.
- Do not say what you are doing in the report. Just write the report
    without any commentary from yourself.
- Each section should be as long as necessary to deeply answer the question
     with the information you have gathered. It is expected that sections
    will be fairly long and verbose. You are writing a deep research report,
     and users will expect a thorough answer.
- Use bullet points to list out information when appropriate, but by
    default, write in paragraph form.

REMEMBER:
The brief and research may be in English, but you need to translate this
    information to the right language when writing the final answer.
Make sure the final answer report is in the SAME language as the human
    messages in the message history.

Format the report in clear markdown with proper structure and include
    source references where appropriate.

<Citation Rules>
- Assign each unique URL a single citation number in your text
- End with ### Sources that lists each source with corresponding numbers
- IMPORTANT: Number sources sequentially without gaps (1,2,3,4...) in the
    final list regardless of which sources you choose
- Each source should be a separate line item in a list, so that in markdown
     it is rendered as a list.
- Example format:
  [1] Source Title: URL
  [2] Source Title: URL
- Citations are extremely important. Make sure to include these, and pay a
    lot of attention to getting these right. Users will often use these
    citations to look into more information.
</Citation Rules>
"""
```

LISTING C.8: Final report model system prompt (part 2 of 2).

```
summarize_webpage_prompt = """You are tasked with summarizing the raw
    content of a webpage retrieved from a web search. Your goal is to
    create a summary that preserves the most important information from the
     original web page. This summary will be used by a downstream research
    agent, so it's crucial to maintain the key details without losing
    essential information.

Here is the raw content of the webpage:

<webpage_content>
{webpage_content}
</webpage_content>

Please follow these guidelines to create your summary:

1. Identify and preserve the main topic or purpose of the webpage.
2. Retain key facts, statistics, and data points that are central to the
    content's message.
3. Keep important quotes from credible sources or experts.
4. Maintain the chronological order of events if the content is time-
    sensitive or historical.
5. Preserve any lists or step-by-step instructions if present.
6. Include relevant dates, names, and locations that are crucial to
    understanding the content.
7. Summarize lengthy explanations while keeping the core message intact.

When handling different types of content:

- For news articles: Focus on the who, what, when, where, why, and how.
- For scientific content: Preserve methodology, results, and conclusions.
- For opinion pieces: Maintain the main arguments and supporting points.
- For product pages: Keep key features, specifications, and unique selling
    points.

Your summary should be significantly shorter than the original content but
    comprehensive enough to stand alone as a source of information. Aim for
     about 25-30 percent of the original length, unless the content is
    already concise.

Present your summary in the following format:

```
{{
  "summary": "Your summary here, structured with appropriate paragraphs or
      bullet points as needed",
  "key_excerpts": "First important quote or excerpt, Second important quote
       or excerpt, Third important quote or excerpt, ...Add more excerpts
      as needed, up to a maximum of 5"
}}
```
"""
```

LISTING C.9: Summarisation model system prompt (part 1 of 2).

```
# Continuation of summarize_webpage_prompt:
"""
Here are two examples of good summaries:

Example 1 (for a news article):
```json
{{
  "summary": "On July 15, 2023, NASA successfully launched the Artemis II
      mission from Kennedy Space Center. This marks the first crewed
      mission to the Moon since Apollo 17 in 1972. The four-person crew,
      led by Commander Jane Smith, will orbit the Moon for 10 days before
      returning to Earth. This mission is a crucial step in NASA's plans to
       establish a permanent human presence on the Moon by 2030.",
  "key_excerpts": "Artemis II represents a new era in space exploration,
      said NASA Administrator John Doe. The mission will test critical
      systems for future long-duration stays on the Moon, explained Lead
      Engineer Sarah Johnson. We're not just going back to the Moon, we're
      going forward to the Moon, Commander Jane Smith stated during the pre-
      launch press conference."
}}
```

Example 2 (for a scientific article):
```json
{{
  "summary": "A new study published in Nature Climate Change reveals that
      global sea levels are rising faster than previously thought.
      Researchers analyzed satellite data from 1993 to 2022 and found that
      the rate of sea-level rise has accelerated by 0.08 mm/year over the
      past three decades. This acceleration is primarily attributed to
      melting ice sheets in Greenland and Antarctica. The study projects
      that if current trends continue, global sea levels could rise by up
      to 2 meters by 2100, posing significant risks to coastal communities
      worldwide.",
  "key_excerpts": "Our findings indicate a clear acceleration in sea-level
      rise, which has significant implications for coastal planning and
      adaptation strategies, lead author Dr. Emily Brown stated. The rate
      of ice sheet melt in Greenland and Antarctica has tripled since the
      1990s, the study reports. Without immediate and substantial
      reductions in greenhouse gas emissions, we are looking at potentially
       catastrophic sea-level rise by the end of this century, warned co-
      author Professor Michael Green."
}}
```

Remember, your goal is to create a summary that can be easily understood
    and utilized by a downstream research agent while preserving the most
    critical information from the original webpage.

Today's date is {date}.
"""
```

LISTING C.10: Summarisation model system prompt (part 2 of 2).

C.2 Verification-Correction Loop System Prompts

```
research_refinement_prompt = """You are a research assistant tasked with
    refining previous research that did not meet quality standards. For
    context, today's date is {date}.

<Task>
Your job is to use tools to gather information to address specific gaps
    identified by a research reviewer.
You can use any of the tools provided to you to find resources that can
    help answer the research question. You can call these tools in series or
     in parallel, your research is conducted in a tool-calling loop.
</Task>

<Available Tools>
You have access to two main tools:
1. **tavily_search**: For conducting web searches to gather information
2. **think_tool**: For reflection and strategic planning during research
{mcp_prompt}

**CRITICAL: Use think_tool after each search to reflect on results and plan
     next steps. Do not call think_tool with the tavily_search or any other
     tools. It should be to reflect on the results of the search.**
</Available Tools>

<Instructions>
You are NOT starting from scratch. Follow these steps:

1. **Review the feedback and guidance** - Understand exactly what gaps need
     to be filled
2. **Do NOT repeat previous searches** - Build on existing work, don't
    duplicate
3. **Conduct targeted searches** - Focus on the specific deficiencies
    mentioned
4. **Prioritize depth over breadth** - Add detail, analysis, and evidence
    to weak areas
5. **Address each gap systematically** - Work through the refinement
    guidance point by point
6. **Stop when gaps are addressed** - Don't over-research, just fill what's
     missing
</Instructions>
"""
```

LISTING C.11: Research refinement system prompt (part 1 of 2).

```
# Continuation of research_refinement_prompt:
"""
<Hard Limits>
**Tool Call Budgets** (Prevent excessive searching):
- Use 1-3 search tool calls to address specific gaps
- Focus searches on reviewer's identified deficiencies
- Stop after finding information that addresses the gaps

**Stop Immediately When**:
- You have addressed all specific gaps mentioned in the refinement guidance
- You have found sufficient depth/evidence/examples for weak areas
- Your searches are returning information you already have
</Hard Limits>

<Show Your Thinking>
After each search tool call, use think_tool to analyze the results:
- Did this search address the identified gap?
- What specific information did I find to strengthen the research?
- Which refinement points have I now addressed?
- What gaps remain from the reviewer's guidance?
</Show Your Thinking>
"""
```

LISTING C.12: Research refinement system prompt (part 2 of 2).

```
research_reviewer_prompt = """You are a research evaluator assessing
    research output before it is submitted to the research supervisor.

<Task>
Your role is to evaluate whether research output adequately addresses the
    research topic provided below.
IMPORTANT: The research topic below is a focused research area that fits
    within a broader research scope. Evaluate whether the research output
    comprehensively addresses THIS SPECIFIC TOPIC, not whether it answers
    broader related questions.
Provide scores and detailed feedback, which will be used to inform
    decisions about whether the research meets quality standards.
</Task>

Today's date is {date}.

<Research Topic>
{research_topic}
</Research Topic>

<Research Output>
{research_output}
</Research Output>

<Evaluation Criteria>
Evaluate each dimension with a binary PASS/FAIL assessment. Research must
    PASS all 4 dimensions to be accepted.

1. **Relevance** (PASS/FAIL): Does the research directly address the
    research topic?
  - **PASS**: Directly addresses the topic with strong alignment to all
      requested aspects
  - **FAIL**: Off-topic, partially relevant, significant gaps, or drift
      from core topic

2. **Depth** (PASS/FAIL): How detailed and substantive is the information?
  - **PASS**: Includes comprehensive detail with analysis, mechanisms,
      causal explanations, and concrete examples
  - **FAIL**: Surface-level descriptions, basic information without deep
      analysis, or missing explanations

3. **Evidence** (PASS/FAIL): Are all claims supported with sufficient
    authoritative sources?
  - **PASS**: At least 10 unique authoritative sources with proper
      citations throughout, all major claims supported by evidence
  - **FAIL**: Fewer than 10 unique sources, OR poor citation coverage, OR
      unreliable references, OR unsupported claims

4. **Completeness** (PASS/FAIL): Does the research cover all key aspects of
     the topic?
  - **PASS**: Covers all key aspects and dimensions of the topic with no
      significant gaps
  - **FAIL**: Missing important aspects, dimensions, or details of the
      topic
</Evaluation Criteria>
"""
```

LISTING C.13: Research Reviewer v1 system prompt (part 1 of 2).

```
# Continuation of research_reviewer_prompt:
"""
<Evaluation Guidelines>
- Use strict standards: when in doubt between PASS/FAIL, choose FAIL
- ALL 4 dimensions must PASS for research to be accepted
- If even 1 dimension fails, research will be sent for refinement
- Be explicit about which dimensions failed and why
</Evaluation Guidelines>

<Output Requirements>
1. Provide PASS or FAIL for each of the 4 criteria
2. Write detailed feedback explaining which dimensions passed/failed and
    why
3. For any FAILED dimensions: provide specific, actionable refinement
    guidance on what needs to be added or improved
</Output Requirements>
"""
```

LISTING C.14: Research Reviewer v1 system prompt (part 2 of 2).

```
research_reviewer_prompt = """You are a research evaluator assessing
    research output before it is submitted to the research supervisor.

<Task>
Your role is to evaluate whether the research output adequately addresses
    the research topic provided below.
You must ensure the report is of the highest quality by evaluating it
    against four critical dimensions: **Comprehensiveness**, **Insight**, **
    Instruction Following**, and **Factuality**. Your goal is to ensure the
     report is thorough, deeply analytical, strictly adherent to
    instructions, and rigorously evidenced.

Evaluate whether the research output comprehensively addresses THIS
    SPECIFIC TOPIC, not whether it answers broader related questions.
</Task>

Today's date is {date}.

<Research Topic>
{research_topic}
</Research Topic>

<Research Output>
{research_output}
</Research Output>

<Evaluation Criteria>
Evaluate each dimension with a binary PASS/FAIL assessment. Research must
    PASS all 4 dimensions to be accepted.

1. **Comprehensiveness** (PASS/FAIL):
  - Does the research cover *multiple perspectives*? (e.g., conflicting
     views, diverse sources)
  - Does it answer ALL implicit and explicit sub-questions?
  - **FAIL** if it misses a key angle or is too narrow.

2. **Insight** (PASS/FAIL):
  - Does the research provide *causal reasoning*? (Explains *why* things
     happen, not just *what* happened)
  - Does it connect disparate facts to form a conclusion?
  - **FAIL** if it is just a list of surface-level facts without analysis.

3. **Instruction Following** (PASS/FAIL):
  - Does the research follow ALL negative constraints? (e.g., "Do not
     include X")
  - Does the research focus on the specific geography/timeframe requested?
  - **FAIL** if it ignored any specific instruction from the user.

4. **Factuality & Evidence** (PASS/FAIL):
  - **CITATION CHECK:** Are there at least 5-10 unique authoritative
     sources?
  - **HALLUCINATION CHECK:** Do the citations in the text actually appear
     in the Sources list?
  - **FAIL** if citation density is low or if citations are mismatched/
     broken.
"""
```

LISTING C.15: Research Reviewer v2 system prompt (part 1 of 2).

```
# Continuation of research_reviewer_prompt:
"""
<Evaluation Guidelines>
- Use strict standards: when in doubt between PASS/FAIL, choose FAIL.
- ALL 4 dimensions must PASS for research to be accepted.
- If even 1 dimension fails, research will be sent for refinement.
</Evaluation Guidelines>

<Output Requirements>
1. Provide PASS or FAIL for each of the 4 criteria.
2. Provide structured **Refinement Guidance** for any failed dimensions:
  - **missing_subtopics**: List specific things to search for.
  - **required_evidence_types**: List types of data needed (stats, quotes).
  - **action**: "search_specific_queries", "verify_citations", etc.
"""
```

LISTING C.16: Research Reviewer v2 system prompt (part 2 of 2).