



Multi-Agent Retrieval-Augmented Generation for Academic Literature Review Synthesis

A Comparative Evaluation Against
Single LLM Baselines

Seminar Paper
Elson Felix Mendes Filho

CAS Generative AI
Fall Semester 2025

Supervised by
Dr. Jannis Vamvas

Submitted on 10.02.2026

Table of Content

1. Introduction	3
1.1 Problem Statement	3
1.2 Research Question	3
1.3 Thesis Overview	4
2. Literature Review	4
2.1 Retrieval-Augmented Generation	4
2.2 Multi-Agent AI Systems	4
2.3 Current Practice	5
2.4 LLM-as-Judge for Evaluation	5
3. System Design	6
3.1 Multi-Agent RAG Architecture	6
3.2 Comparison Framework	6
3.3 Design Rationale	6
4. Methodology	7
4.1 Research Design & Evaluation Metrics	7
4.2 System Architecture	7
4.3 Evaluation	9
5. Implementation	10
5.1 Technical Stack	10
5.2 Corpus and RAG Infrastructure	10
5.3 Baseline and Multi-Agent Infrastructure	12
5.4 Evaluation Infrastructure	13
6. Results	15
6.1 Data Collection Outcomes	15
6.2 RAG Corpus Construction Results	15
6.3 Evaluation Execution and Comparative Performance Results	15
7. Discussion	18
8. Conclusion	19
9. Summary	21
References	22
Declaration of Authorship	23
Appendix	24

1. Introduction

1.1 Problem Statement

Academic research faces a fundamental challenge. Over 2.5 million papers are published annually, with this volume doubling approximately every 12 years (Bornmann & Mutz, 2015). This growth creates an overwhelming information landscape where researchers struggle to maintain comprehensive awareness of their fields.

Researchers face interconnected challenges: identifying relevant prior work across vast literature bases, understanding emerging trends and paradigm shifts, discovering cross-disciplinary applications, and avoiding narrow specialization that misses important developments. Traditional manual approaches such as keyword searches, abstract reading, and citation chaining become increasingly inefficient as publication volumes grow. The cognitive burden of synthesizing insights across hundreds of papers delays research progress and risks incomplete literature coverage.

While tools such as Google Scholar and Semantic Scholar provide discovery mechanisms through web-scale search and citation tracking (Ammar et al., 2018), they fundamentally lack synthesis capabilities. These systems return ranked lists of publications, leaving users to analyse abstracts, identify cross-study relationships, and synthesize insights. Consequently, the search-synthesis gap remains at least partially unresolved.

More recently, researchers have turned to large language models (LLMs), such as ChatGPT (OpenAI, 2025) and Claude (Anthropic, 2024), for literature synthesis. However, pure parametric models exhibit critical limitations: knowledge constrained by training cutoff dates, fabricating citations, misattributing findings, and generating unverifiable claims (Ji et al., 2023). Without grounding mechanisms or citation validation, these tools risk propagating misinformation in academic contexts where credibility is essential.

Emerging AI-powered tools such as Elicit, Consensus, Scispace, and Scite.ai combine large language models with curated databases to retrieve papers, extract claims, and generate evidence-backed summaries, benefiting from reliable source attribution and transparent citation linking. However, they largely remain limited to extractive synthesis, producing reasonable but superficial literature review drafts, with restricted ability for higher-order integration such as thematic synthesis, methodological comparison, and critical evaluation across studies, and typically rely on single-pass reasoning without multi-agent coordination.

Recent advances in Retrieval-Augmented Generation (RAG) and multi-agent AI systems present a promising opportunity to address these limitations (Lewis et al., 2020; Wu et al., 2023). By combining external knowledge retrieval, coordinated multi-agent reasoning, and explicit citation validation, such approaches enable literature synthesis while maintaining factual grounding and traceability.

Beyond academic literature, this architectural approach applies to diverse domains requiring document-based synthesis with verifiable claims, including legal case analysis, pharmaceutical research documentation, financial literature review, and corporate knowledge management.

1.2 Research Question

This thesis investigates: '**How does a multi-agent RAG architecture compare to a single LLM approach for academic literature synthesis in terms of response quality?**'. Specifically, it examines whether architectural complexity combining specialized retrieval agents, multi-source validation, and coordinated synthesis delivers measurable improvements over parametric language models alone.

The comparison is methodologically significant for three reasons. First, it isolates architectural impact by using the same base model (GPT-5.2) in both systems, ensuring differences reflect design rather than model capability. Second, it focuses on response quality (comprehensiveness, accuracy, citation reliability) rather than generic performance metrics. Third, it evaluates whether development and deployment complexity is justified by quality improvements.

1.3 Thesis Overview

This thesis presents a comparative evaluation of two approaches for literature synthesis in the artificial intelligence domain: a multi-agent Retrieval-Augmented Generation (RAG) system with specialized agents and citation validation, and a single-LLM baseline relying on parametric knowledge alone. The evaluation is conducted across a diverse set of research queries using an LLM-as-judge framework complemented by statistical analysis.

Results demonstrate that the proposed system achieves significant improvements across all quality metrics. The largest improvements appear in citation quality and factual grounding, validating that increased response latency is an acceptable trade-off for superior quality.

The remainder of this thesis is structured as follows. Chapter 2 reviews relevant literature on RAG and multi-agent AI systems. Chapter 3 describes the system design and architectural decisions. Chapter 4 outlines the evaluation methodology. Chapter 5 details the system implementation. Chapter 6 presents the results, Chapter 7 discusses findings and limitations, and Chapter 8 concludes with contributions and future research directions.

2. Literature Review

This chapter reviews related work spanning foundational contributions and recent advances relevant to the multi-agent RAG architecture developed in this thesis. Section 2.1 examines retrieval-augmented generation approaches and their limitations for academic synthesis. Section 2.2 discusses multi-agent AI systems and their advantages for complex tasks. Section 2.3 surveys current practice in academic literature reviews. Section 2.4 establishes the LLM-as-judge evaluation employed in this work.

2.1 Retrieval-Augmented Generation

LLMs generate plausible but incorrect information when queried beyond their training data, a phenomenon known as hallucination (Ji et al., 2023). Retrieval-Augmented Generation (RAG) addresses this challenge by combining external knowledge retrieval with language model generation to ground responses in verifiable sources. Lewis et al. (2020) established the foundational RAG paradigm, demonstrating that it significantly improves performance on knowledge-intensive tasks by grounding generation in verifiable sources. The approach combines parametric memory (learned model parameters) with non-parametric memory (a retrievable document index), enabling models to access specific information rather than relying solely on memorized patterns. This hybrid approach addresses parametric models' inability to update knowledge without retraining and their tendency to hallucinate when uncertain. Gao et al. (2024) categorize RAG systems into Naive RAG (simple retrieval and generation), Advanced RAG (with retrieval reranking), and Modular RAG (with decomposition and filtering). While these approaches improve factual accuracy, they struggle with multi-hop reasoning, retrieval noise, and domain adaptation.

For academic literature specifically, L'ala et al. (2023) developed PaperQA, a RAG system for scientific literature that substantially outperformed GPT-4 on question answering benchmarks. Critically, PaperQA could recognize when retrieved evidence was insufficient to answer a query and abstain from generating a response, thereby avoiding hallucinated answers that plagued single-LLM baselines.

However, academic literature synthesis requires capabilities beyond standard RAG systems. While RAG provides grounded citations from retrieved documents, comprehensive literature review demands multi-source synthesis across numerous papers, temporal analysis to track research evolution, and multi-hop reasoning to connect findings across dependencies. Existing single-agent RAG approaches lack the specialized agents needed to orchestrate retrieval, synthesis, and validation across multiple sources simultaneously.

2.2 Multi-Agent AI Systems

Multi-agent systems decompose complex tasks into specialized subtasks handled by coordinated agents, offering advantages over monolithic models (Chen et al., 2024). Park et al. (2023) demonstrated that generative agents with specialized roles produce more believable and coherent behavior than single models when

simulating human interactions, as agents could maintain consistent personas while coordinating complex social dynamics. This architectural pattern addresses fundamental limitations of single LLMs. Chang et al. (2024) identified that multi-agent RAG architectures outperform traditional single-agent approaches by 2-11% on complex benchmarks through specialized decomposition, filtering, and decision agents that mitigate attention bias in monolithic retrieval systems.

For retrieval-augmented generation specifically, multi-agent architectures coordinate specialized retrieval, synthesis, and validation capabilities. Liu et al. (2025) developed a 4-agent system for automated survey generation that achieved 71% improvement over the AutoSurvey baseline by coordinating search, clustering, writing, and quality evaluation, demonstrating the value of specialized agent roles for comprehensive synthesis. Current systems typically draw from a single knowledge source, either a document corpus or parametric model knowledge, rather than integrating multiple sources. Comprehensive literature review requires combining corpus papers and parametric knowledge with validated citations into coherent, well-structured analyses that track research evolution over time. Achieving this demands specialized coordination mechanisms to integrate retrieval, validation, and synthesis agents that combine information from multiple sources into comprehensive, well-structured text, representing an important opportunity to enhance multi-agent architectures for academic literature synthesis.

2.3 Current Practice

Academic literature reviews traditionally require manual search, reading, and synthesis across hundreds of papers, a time- and effort-intensive process. The emergence of LLMs has introduced new approaches: researchers increasingly use general-purpose systems like ChatGPT, Gemini, and Claude for synthesis tasks, while academic search platforms like Google Scholar and Semantic Scholar (Ammar et al., 2018) have added recommendations and citation context analysis.

Novel specialized AI-powered research tools have emerged to address specific literature review subtasks. Tools like Elicit and Consensus AI focus on search and evidence aggregation across large paper databases, while Scite.ai provides citation context analysis to evaluate how papers support or contradict each other. SciSpace offers integrated workflows combining literature search with reading assistance and manuscript preparation. These platforms represent a shift toward automation of individual research tasks, though each addresses distinct workflow components rather than end-to-end synthesis.

These approaches each address different aspects of the literature review challenge with distinct limitations. While formal comparative studies of these commercial platforms remain limited, this fragmented landscape leaves researchers combining multiple specialized tools or accepting general-purpose LLM limitations. Given this context, this work evaluates a multi-agent RAG architecture against a single LLM baseline using GPT-5.2, representing how researchers typically employ general-purpose models for synthesis. This comparison establishes a controlled point for measuring the value of specialized architectural components.

2.4 LLM-as-Judge for Evaluation

Recent work has established LLM-as-judge methodologies for evaluating AI system outputs. Zheng et al. (2023) demonstrated that strong LLM judges can match human preferences with over 80% agreement, providing a scalable and explainable alternative to expensive human evaluation. Though this approach approximates rather than replaces human judgment, it enables rigorous comparative assessment at scale.

This work employs GPT-5.2 as judge with systematic rubrics across three dimensions: overall quality, citation quality, and factual grounding. Following empirically-validated design principles (Yamauchi et al., 2024), each evaluation provides explicit score descriptions for a 1-5 Likert scale and requires the judge to provide detailed reasoning for its assessments. Statistical validation through Wilcoxon signed-rank tests and effect size analysis ensures rigorous comparative analysis between the multi-agent system and baseline.

3. System Design

3.1 Multi-Agent RAG Architecture

The multi-agent system integrates corpus retrieval with the model's parametric knowledge, balancing curated paper depth with training-data breadth while enforcing citation accuracy through external source verification.

The corpus component offers full-text access to curated papers, enabling retrieval of technical details, experimental results, and methodological comparisons. It emphasizes recent publications to cover novel work and supports targeted filtering by topic, though any fixed corpus inevitably leaves gaps in coverage and perspective.

The LLM parametric component addresses corpus limitations by providing breadth from its training data. It supplies contextual knowledge, historical perspective, and conceptual frameworks that connect findings across research areas. This proves particularly valuable for foundational concepts, cross-domain links, or topics underrepresented in the corpus. However, reliance on parametric knowledge risks hallucinations and requires safeguards to ground citations in verifiable papers.

The external validation component mitigates hallucination by verifying citations against established academic databases. This component ensures that cited papers actually exist and that attributions remain accurate, preventing the fabrication of references that can occur with pure parametric generation.

This strategy combines these components systematically: corpus papers provide grounded technical details, parametric knowledge supplies broader context, and external validation ensures citation accuracy. This design reflects the principle that high-quality synthesis requires depth, breadth, recency, and verifiable grounding, which no single knowledge source can provide independently.

3.2 Comparison Framework

The evaluation compares the multi-agent RAG architecture against a single LLM baseline to isolate the value contributed by retrieval augmentation, agent specialization, and citation validation. Both systems employ the same foundation model with optimized prompting, isolating architectural differences rather than confounding results with model capacity variations. This comparison design addresses two fundamental questions: is multi-agent RAG better than baseline? and does architectural complexity justify its implementation costs?

This framing has practical relevance: the baseline represents what users can already access through ChatGPT or similar interfaces, while the multi-agent system can demonstrate incremental quality value.

This comparison tests whether quality improvements justify the increased complexity and latency overhead.

3.3 Design Rationale

Three architectural decisions distinguish this implementation from conventional RAG approaches.

Parallel dual-source retrieval executes corpus search and LLM parametric synthesis simultaneously rather than conditionally routing queries to one source or the other. This design reflects the finding that corpus papers and LLM knowledge provide complementary rather than redundant information, with corpus offering specific technical details while LLM supplies contextual connections.

Multi-stage citation validation employs a fallback strategy across multiple academic databases with different coverage characteristics and access constraints. This staged approach prioritizes speed for common cases while ensuring comprehensive verification for difficult citations, optimizing validation coverage while managing practical constraints.

Automated evaluation using LLM-as-Judge addresses the practical impossibility of manual expert evaluation at scale. Comprehensive comparative testing across multiple quality dimensions would demand prohibitive expert time and introduce consistency challenges across human evaluators. This design choice prioritizes scalability and consistency over the gold standard of human expert judgment.

4. Methodology

4.1 Research Design & Evaluation Metrics

This research employs a within-subjects experimental design in which both the multi-agent and the single LLM baseline respond to identical academic synthesis queries. Both systems are tested on diverse queries spanning multiple AI research topics, with varying complexity and corpus coverage levels. To enable systematic and consistent assessment at scale, the study adopts LLM-as-Judge methodology using detailed rubrics and 5-point Likert scales for each quality dimension.

The experimental design controls for confounding variables through two key constraints. First, both systems utilize identical base LLM capabilities (GPT-5.2), ensuring observed differences stem from architectural features rather than model capacity. Second, generation parameters (temperature, token limits) and computational conditions are held constant across both systems.

The evaluation procedure follows a systematic process: both systems generate responses to each query, after which an LLM judge assesses each response on three quality dimensions. Two secondary metrics track citation validation accuracy and overall win rates. Statistical analysis via non-parametric methods determines whether observed quality differences are significant and quantifies practical impact.

Primary metrics assess response quality on three dimensions.

- Overall Quality evaluates comprehensiveness, coherence, and technical accuracy of responses.
- Citation Quality measures whether citations are relevant, correctly attributed, and properly support claims.
- Factual Grounding assesses the degree to which claims are verifiable, specific, and represent diverse perspectives.

Secondary metrics provide complementary assessment.

- Citation Validation Accuracy measures the percentage of citations successfully verified through external sources, applicable only to the multi-agent system as the baseline performs no validation.
- Win Rate captures the proportion of queries where each system produces superior responses, with ties recorded separately when both systems perform equivalently.

Statistical analysis employs non-parametric methods appropriate for ordinal Likert scale data. The Wilcoxon signed-rank test assesses whether paired quality differences are significant ($\alpha = 0.05$), while Cohen's d quantifies effect magnitude (0.2/0.5/0.8 for small/medium/large effects). The binomial test validates whether win rates exceed the 50% null hypothesis. This approach addresses both statistical significance and practical impact of observed differences.

4.2 System Architecture

The baseline system employs a pure GPT-5.2 LLM without retrieval augmentation, external tools, or citation validation. This architecture relies entirely on parametric knowledge encoded in the model's training data. The system receives queries through a designed prompt that encourages academic synthesis, requests specific details, and asks the model to cite key papers using standard academic format. The prompt explicitly instructs the model to synthesize dominant research directions, compare approaches, and highlight trade-offs while maintaining formal academic prose suitable for publication.

This baseline represents the default experience researchers encounter when using LLMs for literature synthesis: broad knowledge from training data without verification mechanisms. The system produces responses quickly but performs no citation validation, creating risk of hallucinated references. Despite these limitations, the baseline provides a fair comparison point. It uses the same LLM family as the multi-agent system, with equivalent generation parameters and an optimized system prompt for academic synthesis. This design isolates architectural differences as the sole source of performance variation between systems. The key distinction lies in the presence or absence of retrieval infrastructure, specialized agents, and validation mechanisms rather than differences in underlying model capabilities.

The multi-agent RAG system comprises six specialized agents coordinated through a unified workflow architecture. Each agent addresses a specific aspect of academic literature synthesis, as shown in Table 1. The Coordinator Agent orchestrates the overall process, the Query Decomposition Agent handles complex multi-faceted questions, the Retrieval Agent searches the curated corpus of papers, the Parametric Agent accesses LLM for broader context, the Citation Validator Agent ensures reference accuracy through external verification, and the Synthesis Agent combines information from the multiple sources into a coherent academic prose.

Agent	Primary Role	Key Capabilities
1. Coordinator	Workflow orchestration	Phase management, routing, parallel execution
2. Query Decomposition	Complex query handling	Temporal analysis, multi-aspect breakdown, sub-query generation
3. Retrieval	Corpus search	Semantic search, hybrid retrieval (dense and sparse), reranking, relevance filtering, full-text access
4. Parametric	LLM knowledge access	GPT-5.2 synthesis, reference extraction, contextual integration
5. Citation Validator	Reference verification	Multiple source verification, two-tier validation
6. Synthesis	Multi-source integration	Combines Retrieval and Parametric sources into natural prose, with source-tagged citations

Table 1: Multi-Agent System Agents

The curated corpus focuses on recent AI research developments published between 2020 and 2025, spanning multiple research areas including Computer Vision, Natural Language Processing, Edge Computing, Reinforcement Learning, and related fields. Papers were selected from major Swiss and chosen international research institutions to constrain corpus size while ensuring quality and relevance. The corpus text is divided into embedded chunks stored in a vector database, enabling semantic search over complete paper content. The full-text access provides the foundation for detailed, grounded synthesis.

The system processes queries through a coordinator-managed workflow comprising five phases. Phase 1 evaluates query complexity through the Query Decomposition Agent: simple questions proceed directly to retrieval, while complex queries involving temporal comparisons or multi-faceted analysis are decomposed into manageable sub-questions. Phase 2 activates the Retrieval Agent to perform hybrid search over the corpus, identifying and extracting relevant papers with complete text and metadata. Phase 3 engages the Parametric Agent to access LLM training data, synthesizing broader contextual knowledge and extracting key paper references in structured citation formats. Phase 4 implements citation validation through a two-tier approach: the Citation Validator Agent verifies corpus citations through metadata checks and validates parametric citations through a progressive fallback strategy across multiple external academic databases. Phase 5 concludes with the Synthesis Agent integrating all validated findings from corpus papers and parametric knowledge into coherent academic prose, maintaining distinct source attribution through tagged citations and delivering the final response.

This system incorporates the following distinctive features: Adaptive complexity management through the Query Decomposition Agent allows the system to scale its approach based on query characteristics: straightforward questions receive direct processing while complex questions are decomposed into sub-questions that each proceed through the complete workflow. Hybrid retrieval in the Retrieval Agent combines the precision of semantic vector search with the recall benefits of keyword matching, balancing the strengths of dense and sparse retrieval methods. The Parametric Agent provides access to broad contextual knowledge encoded in LLM training data, complementing the depth of corpus-specific information. The progressive fallback validation strategy in the Citation Validator Agent maximizes citation verification coverage, ensuring

robust reference checking across multiple academic databases. The Synthesis Agent leverages GPT-5.2's reasoning capabilities to integrate disparate sources into coherent narratives, maintaining natural academic prose while preserving distinct source attribution.

GPT-5.2 accessed through OpenAI's API was chosen for both baseline and multi-agent implementations. This model represents current state-of-the-art capabilities in natural language understanding and generation, with robust reasoning performance. The API provides consistent model behavior and computational reliability, eliminating infrastructure variability that could confound experimental results. While open-source alternatives were considered for cost reduction and transparency, they would introduce implementation complexity and potential performance inconsistencies that could obscure the comparison, central to this research.

4.3 Evaluation

The evaluation employs a diverse set of queries designed to test system performance across multiple dimensions. Queries span the major research areas in the AI field, ensuring comprehensive topic coverage. Each query is characterized along three dimensions that can influence system behavior and expected performance: topic area, query complexity, and expected corpus coverage.

Query complexity varies from simple single-aspect questions to complex multi-faceted queries. Simple queries typically request straightforward information about specific approaches or methods, while complex queries involve temporal evolution, approach comparisons, multi-aspect analysis, or causal relationships possibly requiring decomposition and synthesis across multiple perspectives. Expected corpus coverage represents the degree to which the corpus contains relevant papers for each query. High-coverage queries address topics well-represented in the collected papers, medium-coverage queries have partial representation, and low-coverage queries test the system's ability to leverage parametric knowledge when corpus information is limited.

The test set was developed through a hybrid few-shot prompting approach combining manual crafting with LLM generation: initial queries were manually crafted to represent typical researcher information needs, then used as examples to prompt a LLM to generate additional queries while maintaining balance across topics, complexity levels, and coverage categories. This approach ensured both authentic representation of research questions and systematic coverage of the evaluation dimensions.

Response quality assessment employs LLM-as-Judge methodology. This approach enables systematic, scalable evaluation across the complete test set while maintaining consistency through detailed and structured prompts. The judge model receives each query along with both system responses and evaluates them against three quality dimensions using five-point Likert scales. The judge operates with consistent parameters across all evaluations to ensure comparable assessments.

The evaluation rubric provides explicit criteria for each quality dimension and rating level. Overall Quality assesses comprehensiveness of coverage, coherence of presentation, and technical accuracy of claims. A rating of five indicates comprehensive coverage with clear organization and accurate technical content, a rating of three reflects adequate but incomplete coverage, and a rating of one indicates poor quality with significant errors. Citation Quality evaluates the relevance of cited papers to the claims being made, accuracy of citation information, and proper attribution of ideas to sources. Factual Grounding assesses whether claims are backed by verifiable sources, the specificity and concreteness of provided details, and representation of diverse perspectives rather than narrow viewpoints.

For each query comparison, the judge provides numerical ratings across all three dimensions for both responses, declares an overall winner based on holistic assessment rather than simple score summation, and provides detailed reasoning explaining the decision. Ties are permitted when both responses demonstrate equivalent quality, ensuring the evaluation captures genuine performance parity rather than forcing artificial distinctions.

5. Implementation

5.1 Technical Stack

The system was implemented in Python 3.11, leveraging modular frameworks and APIs across functional layers: Knowledge Infrastructure, Multi-Agent Orchestration, Citation Validation and Evaluation Infrastructure.

Knowledge Infrastructure. ArXiv was used as the main source for the corpus, targeting specific subject categories of the AI research field. ChromaDB served as the vector database, storing embedded chunks from papers with HNSW indexing and cosine similarity metrics. Text embeddings were generated using OpenAI's text-embedding-3-small model. PDF text extraction combined PyPDF2 with pdfminer.six as a fallback for complex documents.

Multi-Agent Orchestration. The LangChain-core library supplied standardized abstractions for agent interfaces and response handling. LangGraph provided workflow coordination and state management, enabling conditional routing between specialized agents. Python's asyncio facilitated parallel execution of retrieval operations, reducing overall query processing time.

Citation Validation. External validation integrated three academic APIs: arXiv's official Python library for recent preprints, OpenAlex's RESTful API for broad coverage without rate limits, and the scholarly library for Google Scholar access as a comprehensive fallback.

Evaluation Infrastructure. The system employed GPT-3.5 as the primary language model for all generation tasks, including agent reasoning, baseline, synthesis and evaluation judging. Statistical analysis employed scipy.stats for non-parametric testing (Wilcoxon signed-rank) and numpy for effect size calculations. Gradio powered user interfaces, with Plotly generating interactive performance visualizations. Data management utilized JSON for processed corpora and pandas for CSV-based analytics.

5.2 Corpus and RAG Infrastructure

5.2.1 Corpus Collection and Processing

A custom Gradio-based interface (*paper_collector_ui.py*) was developed to facilitate and automate paper selection, corpus collection and the processing pipeline (see Appendix A, Figure A.1). The interface enabled systematic gathering of papers from arXiv by targeting specific subject categories (cs.AI, cs.CV, cs.CL, cs.RO, cs.LG, cs.HC, stat.ML, cs.IR) and filtering by institutional affiliations, with metadata enrichment, including citation counts from Semantic Scholar.

The collection workflow proceeded through five automated steps. First, filters were configured by selecting target institutions—Swiss institutions (University of Zurich, ETH Zurich, EPFL, IDSIA, USI, University of Bern, University of Basel, University of Geneva, ZHAW) and international institutions (University of São Paulo, Massachusetts Institute of Technology, Stanford University, University of Toronto, University of Oxford, University of Cambridge, and Max Planck Institute)—alongside arXiv categories and publication year ranges (2020-2025). Second, the system retrieved papers via the arXiv API, presenting summary statistics including papers per institution and category distribution. Third, an interactive review table enabled manual selection refinement, displaying paper titles, institutions, publication years, abstracts, citation counts, and categories, with bulk selection controls. Fourth, the selection was confirmed with a preview of chosen papers. Fifth, the system executed full PDF text extraction using PyPDF2 and pdfminer.six, reporting success rates and generating enriched JSON files.

The interface produced *papers_with_fulltext_YYYYMMDD_HHMMSS.json* files containing 17 metadata fields per paper (that could also be exported as a CSV file), critically including full extracted text rather than only abstracts. This comprehensive extraction proved essential for subsequent tasks. The final corpus stored in *(data/processed/)* for ingestion.

5.2.2 RAG Setup - Text Processing and Vector Storage

Following corpus collection, a second Gradio interface (`rag_setup_ui.py`) was implemented to enable interactive dataset selection and to automate the ensuing text processing workflow, comprising four sequential stages: chunking, embedding generation, vector storage, and validation (see Appendix B, Figure B.1).

Text Chunking. The TextChunker class segmented full-text papers into semantically coherent chunks of 800 tokens with 100-token overlap, using tiktoken for accurate token counting. The chunking algorithm prioritized semantic boundaries, first attempting splits at paragraph breaks (`\|n\|n`), then sentence delimiters (`.!?`), and finally spaces as a last resort. This strategy preserved contextual integrity while maintaining processable chunk sizes. Each chunk retained complete metadata linking it to its source paper.

Embedding Generation. The EmbeddingGenerator class processed chunks through OpenAI's text-embedding-3-small model, producing dimensional vectors. Batch processing included progress tracking, rate limiting to avoid API throttling, and comprehensive error handling with exponential backoff (three retry attempts per failure). These safeguards proved essential given the long processing duration, ensuring reliable completion despite occasional API timeouts. The interface displayed real-time progress, showing chunk counts, token estimates, processing time, and per-embedding success rates.

Vector Storage. Embeddings were persisted in ChromaDB, configuring the database with cosine similarity metrics for semantic search. The VectorStore class managed insertion of the embeddings with associated metadata, creating a persistent collection (stored in `data/chromadb/`). The database remained compact, enabling fast similarity search without requiring reload on system restart. Post-processing statistics confirmed successful storage of all chunks with proper metadata preservation and embedding dimension validation.

The complete pipeline transformed raw PDFs into a queryable vector database, with the interface displaying comprehensive statistics upon completion to confirm the system was ready for retrieval operations.

5.2.3 Preliminary Retrieval Tests and Validation

Following RAG infrastructure completion, the system underwent validation to verify retrieval accuracy and identify corpus coverage limitations. Validation proceeded through automated unit testing and interactive interface testing.

Unit tests verified chunking accuracy (token counting, overlap logic, metadata preservation), embedding generation (dimension validation, batch processing), and vector storage operations (insertion, retrieval, filtering), with all tests passing.

End-to-end functionality was validated using a Gradio-based query interface, implemented as a preliminary evaluation environment in which users could submit arbitrary natural-language queries while adjusting parameters such as chunk count and generation temperature. The interface returned synthesized responses together with structured source metadata, including authors, publication year, arXiv identifier, and relevance scores. Test runs with diverse queries enabled an initial qualitative and quantitative assessment of system behavior, confirming successful semantic retrieval from ChromaDB with response times on the order of 3 seconds and relevance scores typically exceeding 0.5 for corpus-covered topics.

While test runs confirmed successful semantic retrieval, this preliminary evaluation also revealed corpus coverage limitations. Queries targeting well-represented topics produced coherent, grounded responses with consistently high relevance scores. In contrast, queries concerning broader foundational concepts underrepresented or absent from the corpus resulted in incomplete answers and notably lower relevance scores, often below 0.35. These findings motivated the development of the subsequent multi-agent architecture that integrates LLM parametric knowledge with external citation validation, thereby addressing corpus gaps while upholding verifiable source grounding.

5.3 Baseline and Multi-Agent Infrastructure

5.3.1 Baseline System

The baseline system implemented reflected the typical approach researchers employ: a single LLM queried directly. This system used GPT-5.2, relying entirely on the model's parametric knowledge. The implementation included no retrieval components, external tool access, or citation validation mechanisms. Another limitation of this component is its fixed training horizon. GPT-5.2 has a training data cutoff in August 2025, meaning that more recent publications and events are not represented in its parametric knowledge (OpenAI, 2025).

The baseline configuration set temperature to 0.5, balancing consistency with diversity to enable broader coverage across the model's parametric knowledge, including foundational concepts and cross-domain connections. The implementation used the OpenAI API with standard GPT-5.2 model access. Queries were processed through a single synchronous API call.

The system prompt instructed the model to produce approximately 600-word responses, identify dominant research directions, compare methodological approaches, and cite four to six key papers. The prompt required exact citation formatting and a formal References section listing all cited works in standard academic format.

The baseline implementation operated through a single module (*baseline_agent.py*) managing OpenAI API interactions and response formatting. Average response times were approximately five seconds per query, with responses generated synchronously through direct model access. This streamlined architecture established the performance baseline against which the multi-agent system's capabilities and latency were compared.

5.3.2 Multi-Agent Architecture

The multi-agent system consisted of six specialized agents orchestrated through the LangGraph framework for stateful multi-agent workflows (see Table 1 in Section 4.2.2). LangGraph's explicit state management enabled tracking of intermediate results as data flowed between agents, with standardized AgentResponse objects ensuring consistent communication. All agents inherited from a common base class (*base_agent.py*) providing standardized logging, configuration handling, and response formatting through the AgentResponse dataclass. Agents communicated through these standardized AgentResponse objects containing content, metadata, success status, and optional error messages.

The Coordinator Agent (*coordinator_agent.py*) managed the workflow structuring every query response. The Coordinator controlled agent configuration, including optional decomposition enabling, and orchestrated phase transitions. Phase Zero conditionally invoked the Query Decomposition Agent when complexity detection identified multi-faceted queries. Phase One executed parallel retrieval through both the Retrieval Agent and Parametric Agent for each query or sub-query, implemented via asyncio concurrent execution that reduced latency by approximately 50% compared to sequential processing. Phase Two invoked the Citation Validation Agent to verify citations through external academic databases. Phase Three passed all validated results to the Synthesis Agent.

The Query Decomposition Agent (*query_decomposition_agent.py*) analyzed query complexity through heuristic checks detecting temporal keywords (evolution, from X to Y), comparison phrases (compare A vs B), and multi-aspect structure through pattern matching. When decomposition was warranted, the agent invoked GPT-5.2 to generate two to five focused sub-queries processable independently. Simple what/how questions without complexity markers bypassed decomposition entirely. The agent enforced explicit limits preventing over-decomposition, with sub-query counts capped at five maximum.

The Retrieval Agent (*retrieval_agent.py*) wrapped the existing ChromaDB vector store containing the embedded chunks from the papers. The Coordinator configured the agent with top-k retrieval of five papers and a relevance threshold of 0.35. The agent converted distance metrics to relevance scores using the formula $(1 - \text{distance}/2)$, applied the threshold filter, and returned the top papers and metadata. This hybrid retrieval approach combined dense vector similarity with BM25 sparse matching, using reciprocal rank fusion to merge results from both strategies.

The Parametric Agent (*research_agent.py*) accessed GPT-5.2's parametric knowledge using essentially the same configuration as the baseline system: temperature 0.5, system prompt requesting synthesis of research directions with citation of four to six key papers. The agent prompted the model for synthesis, then extracted structured paper references using pattern matching and tagged these references as parametric source [LLM].

The Citation Validation Agent (*citation_validation_agent.py*) implemented two-tier validation. Level 1 applied to corpus citations, verifying correspondence to retrieved papers and metadata (title, authors, year) matched. Level 2 targeted LLM-generated citations by extracting paper references from the Parametric Agent's output, capturing author, year, and title metadata. The agent validated each extracted reference through a three-stage fallback chain: ArXiv search first, OpenAlex fallback for failed searches, and Google Scholar as the final fallback source. All validation sources applied a ± 2 year tolerance to handle publication date variations across databases. The agent logged source attribution and extracted metadata (title, authors, venue, citation count) for each validated citation.

The Synthesis Agent (*synthesis_agent.py*) used GPT-5.2 with temperature 0.1 and maximum completion tokens of 5,000 to combine corpus papers (from the Retrieval Agent) and LLM parametric knowledge (from the Parametric Agent) into approximately 600-word literature synthesis. The agent's system prompt specified detailed citation guidelines requiring proper academic format, and mandating a References section at the end listing all cited papers with [CORPUS] or [LLM] source tags. The synthesis prompt provided structured inputs separated by source type: corpus papers with full metadata and content excerpts, and LLM parametric synthesis. The prompt instructed natural academic prose without meta-commentary or headers, prioritizing corpus papers due to semantic relevance, recency of publications, curated quality, and full-text access, and targeted 8-10 citations.

Where the baseline relied solely on GPT-5.2's parametric knowledge, the multi-agent RAG system integrated six specialized agents with dual-source retrieval and triple-fallback citation validation. This architectural complexity, comprising parallel execution, specialized processing, and multi-stage validation, required systematic comparative evaluation to determine whether quality improvements justified the added latency and implementation overhead.

5.4 Evaluation Infrastructure

5.4.1 Evaluation Interface

The evaluation infrastructure implemented a seven-step Gradio web interface (*evaluation_ui.py*) guiding the user through systematic evaluation (see Appendix C).

Step 1 upload a CSV file containing test queries (a set of files was prepared in advance as described in Section 4.3) and validated file format, required columns (*query_id*, *query_text*, *topic_category*, *expected_coverage*, *segment*), unique identifiers, and empty field detection.

Step 2 displayed query distribution across topics and corpus coverage levels through interactive Plotly visualizations (charts for topics and coverage), confirming balanced test sets before evaluation.

Step 3 executed automated evaluation by sequentially processing each query through both systems (multi-agent and baseline), invoking the Judge for quality assessment, and saving results incrementally to prevent data loss during extended runs. Each evaluation collected twenty-two data fields including query metadata, system responses, citation counts with source breakdown, judge scores across the three quality dimensions, the winner, and its reasoning. The interface provided real-time progress tracking throughout the evaluation.

Step 4 presented results analysis through multiple interactive components. The interface displayed summary statistics including total queries evaluated, win distribution across both systems, and average quality scores. Four interactive visualizations enabled dynamic exploration: win distribution, average response latencies, performance metrics by topic, and winner counts per topic. The interface provided detailed query-by-query views, showing side-by-side response comparisons, complete judge evaluations with dimension-specific scores, and winner determination reasoning.

Step 5 executed statistical significance testing, invoking the `StatisticalTests` class to perform Wilcoxon Signed-Rank Tests for the three quality dimensions and a Binomial Test for win rate analysis. The interface displayed formatted results including test statistics, p-values, effect sizes (Cohen's d), significance indicators, and interpretive text explaining whether differences reached statistical significance at $\alpha=0.05$.

Step 6 provided CSV export functionality for complete evaluation results, and Step 7 offered downloadable statistical test summaries. The complete workflow enabled reproducible comparative evaluation from query upload through statistical validation.

5.4.2 LLM-as-Judge

The evaluation employed GPT-5.2 as a judge to assess response quality across three dimensions using 1-5 Likert scales. The Judge class (`judge.py`) configured the model with temperature 0.0 for consistent evaluation and implemented detailed scoring rubrics for each dimension. Overall Quality assessed comprehensiveness (coverage of main approaches and recent developments), structure (organization and coherence), depth (beyond surface-level description), and synthesis (comparison rather than enumeration). Citation Quality evaluated relevance of citations to specific claims, recency (preferring more recent papers for fast-moving AI topics), balance between foundational and recent work, specificity (author names and years versus vague references), and appropriate citation quantity. Factual Grounding measured specificity of claims with concrete details, attribution to known or validated sources, technical accuracy, and verifiable statements.

The prompts explicitly instructed the judge to use the full 1-5 scale rather than clustering around middle values. The judge returned structured JSON output containing six-dimension scores, winner determination, and detailed reasoning citing specific examples from both responses.

5.4.3 Statistical Analysis Methods

The evaluation framework implemented statistical testing through the `StatisticalTests` class to assess significance of performance differences (`statistical_tests.py`). The implementation applied Wilcoxon Signed-Rank Tests for hypotheses H1 through H3, comparing paired quality scores across the three evaluation dimensions (overall quality, citation quality, factual grounding). This non-parametric test suited ordinal Likert scale data without assuming normal distribution. Hypothesis H4 employed a Binomial Test to assess whether the multi-agent system's win rate differed significantly from the 50% null hypothesis representing no systematic preference. All tests used significance level $\alpha = 0.05$ as the threshold for rejecting null hypotheses. The framework calculated Cohen's d effect sizes for each comparison, interpreting magnitudes as negligible ($|d|<0.2$), small ($|d|<0.5$), medium ($|d|<0.8$), or large ($|d|\geq0.8$). Statistical results included test statistics, p-values, effect sizes with interpretations, mean differences, and significance indicators displayed in the interface.

5.4.4 Results Visualization and Reporting

The `ResultsAnalyzer` (`results_analyzer.py`) generated performance summaries from evaluation data. The implementation calculated summary statistics including win counts and percentages for both systems, average response latencies, average citation counts, citation validation accuracy for the multi-agent system, and average quality scores for both systems. Four interactive visualizations enabled data exploration, displaying win distribution, response latency comparison, query distribution by topic, and winner counts per topic.

Query-by-query detailed view displayed side-by-side response comparisons with full text, judge scores for all three dimensions, winner determination with detailed reasoning, and citation metadata including source breakdown (CORPUS or LLM) and validation status. The interface provided CSV export functionality delivering complete evaluation results with all twenty-two collected data fields, enabling reproducible analysis and integration with external statistical tools.

6. Results

This chapter presents the empirical results obtained from the complete evaluation framework, comprising data collection, corpus construction, and comparative evaluation of the proposed multi-agent system. The results are reported descriptively, without interpretation. Analysis and discussion of these findings are deferred to Chapter 7.

6.1 Data Collection Outcomes

This section reports the outcomes of the data collection and filtering process used to construct the document corpus for the experimental evaluation.

An initial set of 336 academic papers was collected from arXiv by selecting all relevant artificial intelligence categories and all institutions available in the collection interface (a mix of major Swiss universities and selected international institutions). The collection period spanned from 2020 to 2025, and citation enrichment was enabled using Semantic Scholar.

During the paper review phase, 36 papers were manually excluded due to relevance or suitability criteria. In the subsequent step, 4 additional papers failed either full-text PDF extraction or citation enrichment.

As a result, the final corpus consisted of 296 full-text papers, all of which were successfully processed and retained for further checks and processing. The data was stored in a structured JSON format (*papers_with_fulltext_20251223_123143.json*) for subsequent use in vector embedding generation and corpus construction. An overview of the data collection and filtering process is provided in Appendix A (Figure A.1).

6.2 RAG Corpus Construction Results

This section reports the outcomes of constructing the retrieval corpus.

The final dataset of 296 full-text papers was used as input for corpus construction. Each document was segmented into smaller textual units to support efficient retrieval. This process resulted in a total of 12,109 text chunks across the corpus.

Vector embeddings were generated for all chunks and stored in a persistent vector database. The resulting vector store was successfully created and saved using ChromaDB, enabling similarity-based retrieval during the evaluation phase. The vector database was stored locally to ensure consistency across evaluation runs.

An overview of the corpus construction process and storage structure is provided in Appendix B (Figure B.1).

6.3 Evaluation Execution and Comparative Performance Results

This section presents the results of the evaluation process, including dataset characteristics, execution outcomes, and comparative performance between the baseline and multi-agent systems.

The evaluation was conducted using a predefined test set consisting of 100 queries, stored in a structured CSV file (*test_queries_100.csv*), see Appendix C (Figure C.1). Analysis of the query set identified 12 distinct topical categories. Queries were segmented into four complexity levels, with 25 queries per segment. Query set statistics and distribution charts are shown in Appendix C (Figure C.2). Complete characteristics and a representative sample of this query file are provided in Appendix D (Table D.1).

Following the query validation, the evaluation was executed using both the baseline system and the proposed multi-agent system, as shown in Appendix C (Figure C.3). The evaluation completed successfully without system interruptions or failures. The execution time for the complete processing the set of 100 queries was 2 hours, 6 minutes, and 42 seconds.

The win distribution across all 100 queries is presented in Table 2. The multi-agent system achieved wins in 83% of cases, the baseline in 8%, and 9% resulted in ties. Additionally, both systems were scored on the three quality dimensions: Overall Quality, Citation Quality, and Factual Grounding, using a 5-point Likert scale. Mean scores are presented in Table 3. These results are summarized visually in Appendix C (Figures C.4 and C.5).

System	Wins	Percentage
Multi-Agent	83	83%
Baseline	8	8%
Ties	9	9%

Table 2: Overall Win Distribution

Metric	Baseline	Multi-Agent RAG	Difference (Δ)
Overall Quality	3.53	3.96	+0.43
Citation Quality	3.20	4.37	+1.17
Factual Grounding	3.66	3.91	+0.25

Table 3: Quality Metrics Comparison (Mean Scores, 1-5 Scale)

Statistical significance tests were conducted on the 91 non-tie queries ($\alpha = 0.05$). Results are presented in Table 4.

Test	Metric	Test Statistic	p-value	Cohen's d
Wilcoxon Signed-Rank Test	Overall Quality	W = 135.0	< 0.001	0.73 (medium)
Wilcoxon Signed-Rank Test	Citation Quality	W = 137.5	< 0.001	1.43 (large)
Wilcoxon Signed-Rank Test	Factual Grounding	W = 416.0	0.0016	0.33 (small)
Binomial Test	Win Rate (83/91 (91.2%))	-	< 0.001	-

Table 4: Statistical significance tests

Response generation times were recorded for both systems across all queries. Results are reported in Table 5.

System	Mean Latency (seconds)	Standard Deviation	Latency Ratio
Multi-Agent	76.0	18.3	3.0 x
Baseline	25.6	4.2	1.0 x

Table 5: Response Latency Statistics

Win rates were analyzed across the three corpus coverage levels. Results are presented in Table 6.

Coverage	Queries (n)	Baseline Wins	Multi-Agent Wins	Ties	Multi-Agent Win Rate
High	27	4	21	2	77.8%
Medium	54	4	46	4	85.2%
Low	19	0	16	3	84.2%

Table 6: Performance by Corpus Coverage Level

Win rates were calculated for each of the 12 topic categories. Results are presented in Table 7.

Topic	Queries (n)	Baseline Wins	Multi-Agent Wins	Ties	Multi-Agent Win Rate
Optimization	36	3	31	2	86.1%
Computer Vision	24	2	19	3	79.2%
NLP	10	2	8	0	80.0%
Adversarial	8	1	6	1	75.0%
Edge Computing	8	0	6	2	75.0%
Reinforcement Learning	4	0	4	0	100%
Multimodal	3	0	3	0	100%
Robotics	2	0	2	0	100%
Explainability	2	0	2	0	100%
Healthcare	1	0	1	0	100%
Fairness	1	0	1	0	100%
Privacy	1	0	0	1	0%

Table 7: Win Distribution by Topic Category

To illustrate the comparative evaluation process¹, a representative query example is presented below with judge scores and reasoning. Full responses from both systems for this example are shown in Appendix C (Figure C.6). Another example, where the baseline was the winner, is provided in Appendix C (Figure C.7).

Query ID: Q008 - Query Text: "How is transfer learning applied in computer vision tasks?"

Topic: Computer Vision, Coverage: Medium

Citations:

- Multi-Agent: 8 total (3 corpus + 5 LLM), 100% validated
- Baseline: 5 total

Judge scores are presented in Table 8.

Metric	Baseline	Multi-Agent
Overall Quality	3	4
Citation Quality	3	5
Factual Grounding	4	4
Winner	-	Multi-Agent

Table 8: Judge Scores for Query Q008

Judge Reasoning (verbatim):

"Response A clearly explains the core transfer-learning workflow (feature extractor vs fine-tuning) and supports it with canonical CV papers (e.g., Krizhevsky et al., 2012; Chen et al., 2020; He et al., 2020), but it largely stops at pre-2021 paradigms and omits newer transfer regimes (e.g., vision-language/zero-shot transfer). Response B covers the same fundamentals and adds recent, validated examples of multimodal and robotics transfer (e.g., D-CAT for cross-modal transfer (Daher et al., 2025) and self-supervised robot localization (Nava et al., 2024)), giving it broader, more current survey value; however, some higher-level claims about 'trust' and 'training-free adaptation' are less tightly tied to specific technical mechanisms than the core backbone/finetuning discussion."

¹ The complete evaluation run is available at: https://github.com/ElsonFilho/research-navigator/blob/main/docs/Screen_Evaluation_UI_100.pdf

7. Discussion

The multi-agent system's strongest advantage was observed in citation quality, achieving the largest improvement (+1.17 points, Cohen's $d = 1.43$). This pattern directly reflects the architectural design: corpus papers provide verifiable sources with full-text access, while triple-fallback validation (ArXiv → OpenAlex → Google Scholar) ensures citation accuracy. The 100% validation rate establishes citation credibility as the system's primary differentiator, essential for academic synthesis where unverified claims can undermine research credibility.

Counterintuitively, high coverage queries achieved the lowest win rate (77.8%), while medium (85.2%) and low coverage queries (84.2%) performed slightly better. This pattern demonstrates that multi-agent architecture delivers value even when corpus representation is limited, effectively balancing available corpus papers with validated LLM parametric knowledge to maintain performance. Multi-agent advantages generalized across diverse topic categories (75-100% win rates), with no topic showing baseline dominance. The consistency in corpus paper usage (3 papers across coverage levels) reflects effective retrieval filtering that identifies relevant papers even in sparse topic areas. The multi-source synthesis strategy, combining corpus depth, LLM breadth, and validated citations, proved most advantageous when queries required integrating diverse perspectives rather than simple factual retrieval.

This counterintuitive pattern challenges the assumption that RAG performance scales linearly with corpus coverage. Instead, architectural robustness through multi-source integration proved more important than corpus density alone. This finding has practical implications for RAG deployment: systems need not achieve comprehensive corpus coverage to deliver value if they effectively balance available corpus data with validated parametric knowledge. These findings demonstrate that architectural specialization yields measurable quality improvements specifically in academic literature synthesis.

Despite the multi-agent system's overall dominance, the baseline remained competitive in specific scenarios. The smallest performance gaps emerged for queries where corpus coverage was limited, and responses relied primarily on well-established parametric knowledge. For straightforward factual questions about fundamental AI concepts, the baseline's broad training data proved sufficient, reducing the marginal value of corpus retrieval and validation infrastructure.

As expected, the baseline's primary advantage was latency, operating approximately three times faster than the multi-agent system. For use cases such as quick factual lookups or early-stage exploration, this speed advantage can outweigh the benefits of retrieval and validation. However, the multi-agent system maintained a consistent win rate above the 60% threshold, with the lowest category-level performance still reaching 75%. This suggests that while increased computational cost and processing time represent meaningful trade-offs, they are often justified for tasks requiring comprehensive synthesis and verified citations.

The multi-agent system reflected deliberate scope decisions that introduced certain limitations. The corpus contained 296 papers from 16 institutions, a size chosen to enable local operation and efficient retrieval while maintaining coverage across multiple AI research areas. The temporal scope was restricted to papers published between 2020 and 2025, prioritizing recent developments over historical coverage. Approximately 60% of the corpus originated from Swiss institutions, while the remaining 40% provided international perspective. These scoping choices affected how the system handled queries on underrepresented topics or those requiring deeper historical context. In such cases, LLM-based synthesis played a more prominent role by leveraging parametric knowledge to bridge corpus gaps, allowing the system to still produce coherent and contextually grounded responses. While citation coverage may have been less comprehensive in these scenarios, synthesis capabilities helped mitigate both topical and temporal limitations.

The system was designed specifically for AI literature. Generalization to other research domains remained untested. While the architectural principles should transfer, domain-specific validation would be required before deployment in fields such as medicine, law, or social sciences.

The evaluation methodology also introduced some limitations. While the LLM-as-Judge approach enabled fast, scalable and systematic assessment, it is not equivalent to human expert evaluation. Future work should incorporate human evaluation to validate these automated results. The test set of 100 queries, though balanced across topics and complexity, cannot fully capture the diversity of real-world researcher needs. Queries were curated to ensure good coverage, but actual usage patterns may differ from this controlled setting. Moreover, the findings are specific to AI literature using current LLM capabilities. As both the research domain and underlying models evolve rapidly, the generalizability of results may change over time.

Practical implications for researchers conducting literature reviews: the multi-agent system offered clear value in scenarios requiring comprehensive synthesis with verified citations. It was particularly well suited for systematic literature reviews, meta-analyses, and research, where citation accuracy is critical. In contrast, the baseline's speed advantage made it more appropriate for preliminary topic exploration, quick factual lookups of well-established concepts, or time-constrained tasks where exhaustive synthesis is less essential. Overall, system selection reflected a trade-off between latency and output quality, with the multi-agent system justifying its approximately threefold higher processing time through substantial gains in citation accuracy and factual grounding.

The evaluation validated key architectural principles of the multi-agent system design. Multi-source citation validation proved essential for maintaining academic credibility, with OpenAlex's rate-free API emerging as a particularly valuable resource for comprehensive validation. Combining corpus-based retrieval with LLM parametric knowledge, rather than relying exclusively on either approach, was shown to improve robustness and overall system quality. These findings suggested that investment in validation and synthesis quality yielded greater returns than retrieval optimization alone, confirming that architectural complexity and increased latency were justified by measurable quality improvements in academic literature synthesis.

8. Conclusion

The exponential growth of academic literature has created a critical challenge for researchers: synthesizing knowledge across vast, rapidly expanding paper collections. This thesis addressed this challenge by developing and evaluating a multi-agent architecture that combines corpus retrieval, LLM parametric knowledge, and citation validation to provide comprehensive, grounded academic literature synthesis. Specifically, this work answered the research question: "**How does a multi-agent RAG architecture compare to a single LLM approach for academic literature synthesis in terms of response quality?**".

The comparative evaluation provided a clear answer to this question. The multi-agent architecture achieved an 83% win rate compared to the baseline's 8%, demonstrating significant advantages across all quality metrics. Statistical tests confirmed these improvements were highly significant, with the largest effect observed in Citation Quality (Cohen's $d = 1.43$). The multi-agent system's triple-fallback validation mechanism achieved 100% citation accuracy.

The multi-agent architecture's strength lay in overcoming the fundamental limitations of parametric-only approaches. By integrating access to recent research papers, and LLM parametric knowledge with external citation validation, the system addressed both the temporal constraints of training cutoffs and the hallucination risks of unverified outputs.

While the multi-agent system required approximately three times longer processing time than the baseline, this latency cost was justified by the substantial quality improvements, particularly for high-stakes academic synthesis tasks that require citation accuracy and factual grounding.

This research made three key contributions.

Methodologically, the study established a comparative evaluation framework using LLM-as-Judge methodology across diverse queries, validated through statistical significance tests and a reproducible evaluation protocol.

Empirically, the work quantified the value added by multi-agent architectures over pure parametric LLMs, demonstrating that specialized agent coordination delivers significant quality improvements. The evaluation measured concrete trade-offs between system complexity and response quality, establishing a performance baseline against which future multi-agent synthesis systems can be compared.

Practically, the research delivered a working system for academic literature synthesis with a reproducible architecture applicable to other research domains. The complete implementation, including the triple-fallback citation validation mechanism and evaluation framework, demonstrates the viability of multi-agent approaches for research assistance while providing a methodological foundation for future work.

Some directions emerge for extending this research.

Immediate development focuses on translating this research into a practical, user-facing system. An extended interface is under development and will enable researchers to upload custom paper collections for domain-specific literature reviews, with automated RAG setup requiring no technical expertise. This system will incorporate blind A/B comparison between baseline and multi-agent approaches, capturing user preferences to validate the architectural findings with human feedback. An interactive multi-query workflow will address the reality that comprehensive literature reviews require iterative refinement rather than single queries, while integrated reference access will allow users to check citations and read the papers directly. This evolution from a research experiment to deployable tool will provide both human evaluation of the comparative results and demonstrate practical viability for self-service research assistance.

Medium-term extensions could validate the architecture's generalizability through cross-domain testing in pharmaceutical research, legal case analysis, or financial literature where similar synthesis challenges exist. These domains would require different corpus collection approaches, combining the developed paper search and collection capabilities with a document uploading interface, and potentially integrating with domain-specific databases such as PubMed for medicine, legal case repositories, or financial data sources. Performance optimization through parallel validation, intelligent caching, and model selection could reduce latency while preserving quality, making the system practical for time-sensitive research workflows.

Long-term development could transform the system beyond academic research into a comprehensive knowledge synthesis platform applicable across research, industry, legal, and policy domains. Automated identification of emerging trends and knowledge gaps would provide continuous awareness of developments across fields, while collaborative annotation platforms would enable team-based synthesis and multimodal capabilities would support richer integration of visual and textual content. Advanced features such as hypothesis generation could guide exploration toward unexplored areas. As language models rapidly evolve, the modular architecture positions the system to incorporate improved capabilities, evolving from a synthesis tool into an active knowledge partner.

9. Summary

Academic researchers face increasing challenges synthesizing knowledge across rapidly expanding literature bases. This thesis develops and evaluates a multi-agent Retrieval-Augmented Generation architecture for academic literature synthesis, comparing it against a single large language model baseline to assess response quality improvements.

The multi-agent system integrates six specialized agents: a Coordinator, Query Decomposition Agent, Retrieval Agent (searching 296 curated papers), Parametric Agent (accessing LLM training knowledge), Citation Validator (implementing triple-fallback verification via ArXiv, OpenAlex, and Google Scholar), and Synthesis Agent. This architecture addresses fundamental limitations of parametric-only approaches by combining recent research papers with LLM training knowledge and external citation validation, overcoming both temporal constraints from training cutoffs and hallucination risks from unverified outputs. Evaluation across 100 diverse AI research queries employed LLM-as-judge methodology assessing overall quality, citation quality, and factual grounding.

Results demonstrated substantial advantages for the multi-agent architecture, achieving an 83% win rate versus the baseline's 8%. Statistical tests confirmed highly significant improvements across all quality metrics, with the largest effect in citation quality (Cohen's $d = 1.43$). These quality improvements came at threefold higher latency, representing an acceptable trade-off for high-stakes synthesis requiring citation accuracy and factual grounding.

In conclusion, this thesis demonstrates that multi-agent architectures combining specialized retrieval, parametric knowledge, and validated citations significantly outperform single LLM approaches for academic literature synthesis. The architectural complexity and increased processing time prove justified by substantial quality improvements, particularly for tasks requiring citation accuracy and factual grounding.

References

- Ammar, W., Groeneveld, D., Bhagavatula, C., Beltagy, I., Crawford, M., Downey, D., ... & Etzioni, O. (2018). Construction of the literature graph in Semantic Scholar. Proceedings of NAACL-HLT 2018, 84-91.
<https://aclanthology.org/N18-3011/>
- Anthropic. (2024). Claude: Large language models for AI assistance. <https://www.anthropic.com/clause>
- Bornmann, L., & Mutz, R. (2015). Growth rates of modern science: A bibliometric analysis based on the number of publications and cited references. Journal of the Association for Information Science and Technology, 66(11), 2215-2222. <https://doi.org/10.1002/asi.23329>
- Chang, Y., Wang, X., Wang, J., Wu, Y., Yang, L., Zhu, K., ... & Xie, X. (2024). A survey on evaluation of large language models. ACM Transactions on Intelligent Systems and Technology, 15(3), 1-45.
<https://dl.acm.org/doi/10.1145/3641289>
- Chen, S., Wang, T., Liu, W., Chen, T., Li, M., Wang, X., ... & Liu, Q. (2024). A survey on LLM-based multi-agent systems: Recent advances and new frontiers in application. arXiv preprint arXiv:2412.17481.
<https://arxiv.org/abs/2412.17481>
- Gao, Y., Xiong, Y., Gao, X., Jia, K., Pan, J., Bi, Y., ... & Wang, H. (2024). Retrieval-augmented generation for large language models: A survey. arXiv preprint arXiv:2312.10997. <https://arxiv.org/abs/2312.10997>
- Ji, Z., Lee, N., Frieske, R., Yu, T., Su, D., Xu, Y., ... & Fung, P. (2023). Survey of hallucination in natural language generation. ACM Computing Surveys, 55(12), 1-38. <https://dl.acm.org/doi/10.1145/3571730>
- L'ala, D., Feldman, S., Candra, R., & Caragea, C. (2023). PaperQA: Retrieval-augmented generative agent for scientific research. arXiv preprint arXiv:2312.07559. <https://arxiv.org/abs/2312.07559>
- Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., ... & Kiela, D. (2020). Retrieval-augmented generation for knowledge-intensive NLP tasks. Advances in Neural Information Processing Systems, 33, 9459-9474. <https://proceedings.neurips.cc/paper/2020/hash/6b493230205f780e1bc26945df7481e5-Abstract.html>
- Liu, Y., Wu, Y., Zhang, D., & Sun, L. (2025). Agentic AutoSurvey: Let LLMs survey LLMs. arXiv preprint arXiv:2509.18661.
<https://arxiv.org/abs/2509.18661>
- OpenAI. (2025). GPT-5.2 technical overview. <https://openai.com/>
- Park, J. S., O'Brien, J. C., Cai, C. J., Morris, M. R., Liang, P., & Bernstein, M. S. (2023). Generative agents: Interactive simulacra of human behavior. Proceedings of UIST 2023, 1-22.
<https://doi.org/10.1145/3586183.3606763>
- Wu, Q., Bansal, G., Zhang, J., Wu, Y., Li, B., Zhu, E., Jiang, L., Zhang, X., Zhang, S., Liu, J., Awadallah, A. H., White, R. W., Burger, D., & Wang, C. (2023). AutoGen: Enabling Next-Gen LLM Applications via Multi-Agent Conversation Framework. arXiv. <https://arxiv.org/abs/2308.08155>
- Yamauchi, Y., Yano, T., & Oyamada, M. (2025). An empirical study of LLM-as-a-judge: How design choices impact evaluation reliability. arXiv. <https://doi.org/10.48550/arXiv.2506.13639>
- Zhao, P., Zhang, H., Yu, Q., Wang, Z., Geng, Y., Fu, F., ... & Huang, X. (2024). Retrieval-augmented generation for AI-generated content: A survey. arXiv preprint arXiv:2402.19473. <https://arxiv.org/abs/2402.19473>
- Zheng, L., Chiang, W. L., Sheng, Y., Zhuang, S., Wu, Z., Zhuang, Y., Lin, Z., Li, Z., Li, D., Xing, E. P., Zhang, H., Gonzalez, J. E., & Stoica, I. (2023). Judging LLM-as-a-judge with MT-bench and Chatbot Arena. Advances in Neural Information Processing Systems, 36, 46595-46623.
https://proceedings.neurips.cc/paper_files/paper/2023/hash/91f18a85473732fe43f478ce2242e66d-Abstract-Datasets_and_Benchmarks.html

Declaration of Authorship

With my signature, I hereby declare that this thesis is entirely my own work and has been completed without the involvement of any third parties. All sources and references used have been properly cited and acknowledged. I further consent to the use of plagiarism detection software to verify the originality of my work. I acknowledge that, in the event of confirmed plagiarism, the thesis will be deemed as failed.

Zurich, 10 February 2026



Elson Felix Mendes Filho

Appendix

Appendix A: Corpus Collection and Management Interface

Research Navigator - Data Collection

Institutions

- Swiss Institutions
 - ✓ University of Zurich
 - ✓ ETH Zurich
 - ✓ EPFL
 - ✓ IDSIA
 - ✓ USI
 - ✓ University of Bern
 - ✓ University of Basel
 - ✓ University of Geneva
 - ✓ ZHAW
- International Institutions
 - ✓ University of São Paulo
 - ✓ MIT
 - ✓ Stanford University
 - ✓ University of Toronto
 - ✓ University of Oxford
 - ✓ University of Cambridge
 - ✓ Max Planck Institute

Categories

- arXiv Categories
 - ✓ cs.AI - Artificial Intelligence
 - ✓ cs.LG - Machine Learning
 - ✓ cs.CV - Computer Vision
 - ✓ cs.CL - Natural Language Processing
 - ✓ cs.RO - Robotics
 - ✓ cs.NE - Neural Computing
 - ✓ stat.ML - Machine Learning (Statistics)
 - ✓ cs.HC - Human-Computer Interaction
 - ✓ cs.IR - Information Retrieval

Settings

From Year (to Present) 2020 Enable Citation Enrichment [Semantic Scholar]

2015 2025

Collect Papers

Collection Summary

Collection Summary: 336 Papers Collected

Papers by Institution

Institution	Paper Count
EPFL	0
Stanford University	~45
University of Cambridge	~15
University of Zurich	~10
IDSIA	~5
USI	~5
University of Geneva	~5
Max Planck Institute	~5
University of Bern	~5

Top Categories

Category	Paper Count
cs.HC	~45
cs.CV	~25
stat.ML	~25
cs.RO	~45
cs.CL	~55
cs.CV	~85
cs.AI	~125
cs.LG	~175

Papers Review

Review and select papers:

- ✓ Uncheck papers you don't want
- ☰ Click arXiv to read abstract
- 🖨 Click PDF to download paper

#	Title	Institution	Year	Cites	Categories	Links
1	CRDE 2.5 - ZeResFDG: Frequency-Decoupled, Rescaled and Zero-Projected Guidance for SD/SDXL Latent Diffusion Models	ETH Zurich	2025	0	cs.CV	View Download
2	Under pressure: learning-based analog gauge reading in the wild	ETH Zurich	2024	0	cs.CV, cs.LG, cs.RO	View Download
3	Accessible Robot Control in Mixed Reality	ETH Zurich	2023	5	cs.RO, cs.CV	View Download
335	GRIP: Generating Interaction Poses Using Spatial Cues and Latent Consistency	Max Planck Institute	2023	0	cs.CV	View Download
336	The Sensorium competition on predicting large-scale mouse primary visual cortex activity	Max Planck Institute	2022	0	q-bio.NC, cs.RI, cs.LG	View Download

Select All Deselect All Selected: 336/336 papers

Step 2: Confirm Selection

Confirm My Selection

✓ Selection Confirmed!

Selected: 300 out of 336 papers

Step 2/2: Extracting PDFs...

✓ PDF Extraction Complete!

✓ Complete Ready for ChromaDB ingestion!

Download Files

[Download JSON](#) [Export as CSV](#)

papers_final_20251223_123143.json 669.9 KB · papers_export_20251223_123143.csv 458.5 KB

papers_with_fulltext_20251223_123143.json [30.33 MB]

Use via API · Built with Gradio · Settings

Figure A.1.: Corpus Collection and Management User Interface (paper_collector_ui.py).

Appendix B: RAG Setup and Processing Interface

The screenshot shows the 'Research Navigator - RAG Setup' application interface. At the top, there's a header with the title and a sub-instruction: "Set up your Retrieval-Augmented Generation system for academic literature analysis." Below the header, there's a "Process:" section with four steps:

1. Select Dataset - Choose your papers JSON file
2. Review Settings - Check configuration
3. Generate Embeddings - Process papers (chunk + embed + store)
4. View Statistics - Check results

On the left, there are two main sections: "Configuration" and "Step 1: Select Dataset".

Configuration:

- Current Settings:**
 - Embedding Model: text-embedding-3-small
 - Chunk Size: 800 tokens
 - Chunk Overlap: 100 tokens
 - Collection: research_papers
 - Storage: C:\Users\prico\OneDrive\Documents\ELSON\Estudios\CAS_GenAI_ResearchNavigator\research-navigator\data\chromadb
- To change settings, edit `src/rag/config.py`.

Step 1: Select Dataset:

Select Papers JSON File: (Most recent file is selected by default)

File: `papers_with_fulltext_20251223_123143.json`

Dataset Information:

File: `papers_with_fulltext_20251223_123143.json` Total Papers: 300 Papers with Full Text: 296 (98.7%)

Sample Paper:

- Title: CADE 2.5 - ZeResFDG: Frequency-Decoupled, Rescaled and Zero-Projected Guidance for...
- Authors: Denis Rychkovskiy
- Institution: ETH Zurich
- Date: 2025-10-14 19:57:58+0000

Estimated Processing:

- Chunks: ~4,440 (assuming 15 chunks/paper)
- Tokens: ~2,220,000 (assuming 500 tokens/chunk)
- Cost: ~\$0.0444 (with text-embedding-3-small)
- Time: ~25.2 minutes

Ready to process!

Step 2: Generate Embeddings:

Generate Embeddings & Store in Vector DB

Processing Status:

- Loaded 300 papers
- Created 12109 chunks
- Estimated cost: \$0.1861
- Estimated time: ~68.6 minutes
- Generated 12109 embeddings in 5644.6s
- Avg: 0.47s per embedding
- Stored 12109 chunks in ChromaDB
- Processing complete!

Vector Store Statistics:

Collection: research_papers Total Chunks: 12,109 Embedding Dimensions: 1536 Distance Metric: cosine Storage Location: C:\Users\prico\OneDrive\Documents\ELSON\Estudios\CAS_GenAI_ResearchNavigator\research-navigator\data\chromadb

Processing Summary:

- Papers processed: 300
- Chunks created: 12,109
- Embeddings generated: 12,109
- Successfully stored: 12,109
- Total time: 5644.6s
- Actual cost: \$0.1861

RAG system is ready!

You can now query your papers using similarity search.

Figure B.1: Text Processing and Vector Storage Configuration User Interface (`rag_setup_ui.py`).

Appendix C: Evaluation Interface

Evaluation User Interface (*evaluation_ui.py*)



Figure C.1: Test query selection and validation

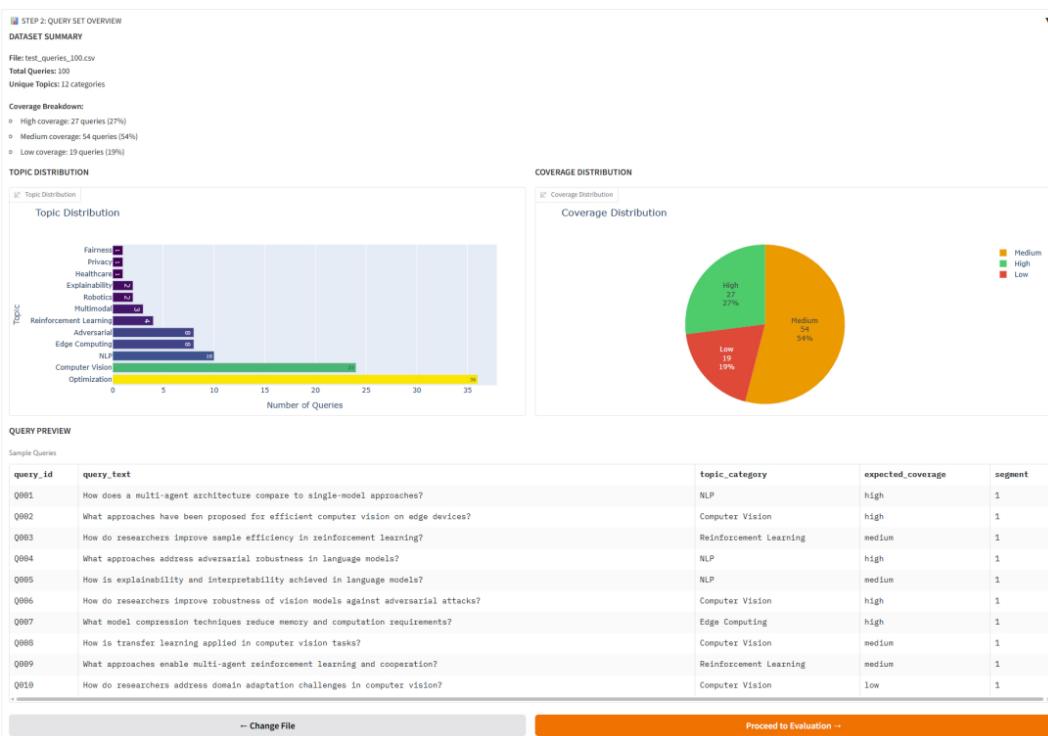


Figure C.2: Query set summary and preview



Figure C.3: Automated evaluation execution progress

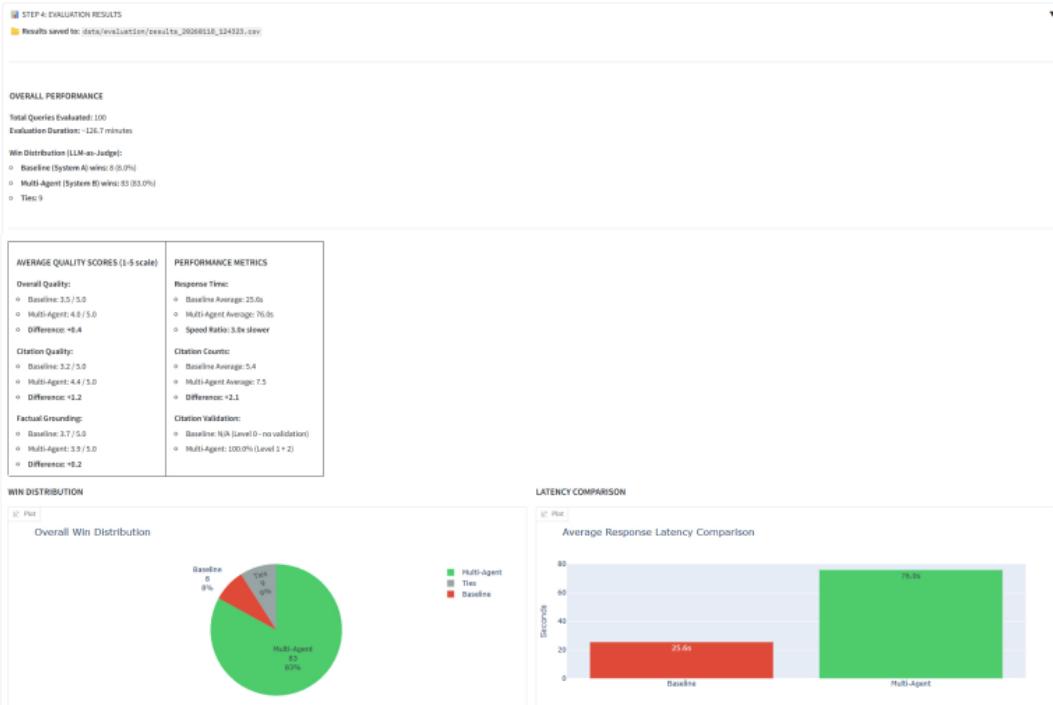


Figure C.4: Evaluation results - Win distribution and quality metrics

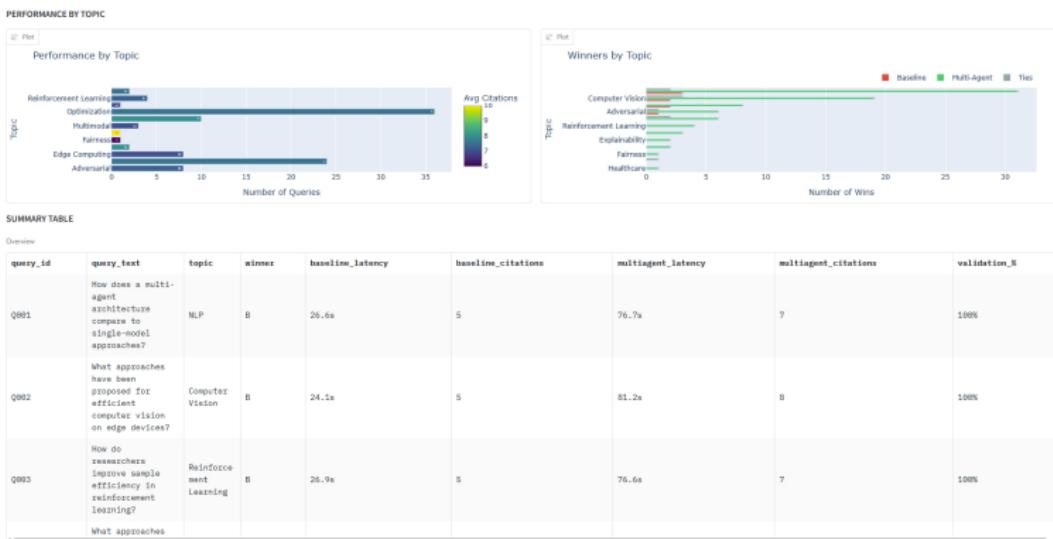


Figure C.5: Evaluation results - Performance by topic and coverage

Q008: How is transfer learning applied in computer vision tasks?

Responses

<p>Response A (Baseline - GPT-3.2) Citation: 5 ▼ View response</p> <p>Transfer learning in computer vision refers to reusing representations learned from large-scale source datasets (typically ImageNet) to improve performance and sample efficiency on a target task with limited labeled data. Dominant research directions can be organized around (i) supervised pretraining and fine-tuning, (ii) self-supervised pretraining, and (iii) architectural choices that affect transferability, each with distinct trade-offs in compute, label dependence, and robustness across tasks.</p> <p>A foundational paradigm is supervised pretraining followed by task-specific adaptation. Krizhevsky et al. (2012), “ImageNet Classification with Deep Convolutional Neural Networks” demonstrated that deep convolutional networks trained on large labeled datasets learn hierarchical features that generalize beyond the source task. In practice, transfer is applied by initializing a target model with pretrained weights and then either (a) using the network as a fixed feature extractor (freezing most layers and training a shallow classifier head) or (b) fine-tuning some or all layers on the target dataset. The former reduces overfitting and compute when target data are scarce, while the latter typically yields higher accuracy when moderate target data exist, at the risk of catastrophic forgetting and sensitivity to hyperparameters (learning rate, layer-wise freezing, and regularization).</p> <p>Architectural innovations have been motivated partly by improving the quality and portability of learned features. He et al. (2016), “Deep Residual Learning for Image Recognition” introduced residual connections that enable substantially deeper networks, improving representational capacity and optimization stability. In transfer settings, residual networks become a standard backbone because their pretrained features tend to be strong and stable across diverse downstream tasks (classification, detection, segmentation). The trade-off is increased depth and compute relative to earlier CNNs, though residual design mitigates optimization difficulties that would otherwise hinder both pretraining and fine-tuning.</p> <p>A second major direction replaces labeled pretraining with self-supervised representation learning, which learns transferable features from unlabeled images via pretext objectives. Chen et al. (2020), “A Simple Framework for Contrastive Learning of Visual Representations” (SimCLR) showed that contrastive objectives with strong data augmentation can produce representations that transfer competitively to downstream tasks, often narrowing the gap with supervised pretraining. Transfer is applied similarly—either linear evaluation (training a linear classifier on frozen features) to measure representation quality, or full fine-tuning for maximal downstream performance. SimCLR’s findings emphasize that augmentation choice and large batch sizes (or equivalent mechanisms) are critical, creating a compute-performance trade-off where transfer can require substantial pretraining, but it reduces reliance on labeled source data and can improve robustness to domain shifts.</p> <p>Relatively, He et al. (2020), “Momentum Contrast for Unsupervised Visual Representation Learning” (MoCo) addressed compute constraints by using a momentum-updated encoder and a memory queue to maintain many negative examples without extremely large batches. This design improved practicality and stability of contrastive pretraining and yielded strong transfer to detection and segmentation when fine-tuned. Compared with SimCLR, MoCo highlights a system-level trade-off: memory queues and momentum encoders add algorithmic complexity but can reduce hardware requirements while preserving transfer quality.</p> <p>A third direction focuses on scaling and architectural efficiency to improve transfer under realistic resource constraints. Tan and Le (2019), “EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks” proposed compound scaling to jointly scale depth, width, and resolution, achieving better accuracy-efficiency trade-offs. In transfer learning pipelines, EfficientNet backbones pretrained on large datasets can deliver strong downstream accuracy with fewer parameters and FLOPs, benefiting deployment and enabling fine-tuning on limited hardware. The trade-off is that optimal scaling and resolution choices can be task-dependent; higher input resolution can improve fine-grained recognition but increases compute and may require careful regularization during fine-tuning.</p> <p>Across these strands, transfer learning is operationalized through consistent mechanisms—initialization from pretrained weights, selective freezing, and fine-tuning—while research differs in how the pretrained representation is obtained (supervised vs. self-supervised) and how architectures balance capacity and efficiency. Supervised pretraining remains straightforward and strong when labeled source data are available, whereas self-supervised methods reduce label dependence and can improve generality but may require heavier pretraining compute or more complex training dynamics. Architectural advances such as residual connections and compound scaling further modulate the accuracy-efficiency frontier, shaping how effectively representations transfer across computer vision tasks.</p> <p>References:</p> <ul style="list-style-type: none"> Chen, T., Kornblith, S., Norouzi, M., & Hinton, G. (2020). A simple framework for contrastive learning of visual representations. ICML 2020. He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. CVPR 2016. He, K., Fan, H., Wu, Y., Xie, S., & Girshick, R. (2020). Momentum contrast for unsupervised visual representation learning. CVPR 2020. Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet classification with deep convolutional neural networks. NeurIPS 2012. Tan, H., & Le, Q. (2019). EfficientNet: Rethinking model scaling for convolutional neural networks. ICML 2019. 	<p>Response B (Multi-Agent RAG) Citation: 8 (3 corpus, 5 LLM) ▼ View response</p> <p>Transfer learning in computer vision is most commonly operationalized as reusing representations learned on a source problem—often large-scale recognition or multimodal alignment—and adapting them to a target task with different labels, outputs, or sensing conditions. In classical pipelines, this reuse is implemented by initializing a “backbone” feature extractor with pretrained weights and then training task-specific heads (or fine-tuning the full network) on the downstream dataset. The enduring appeal of this approach is that early and mid-level visual features learned from broad data distributions tend to be generic, improving sample efficiency and optimization stability relative to training from scratch. This paradigm was popularized by large-scale supervised pretraining on ImageNet (Krizhevsky et al., 2012) and became standard practice in downstream structured prediction tasks such as detection and segmentation, where pretrained backbones are paired with specialized heads and losses (Girshick, 2015; Long et al., 2015).</p> <p>Recent work extends transfer learning beyond supervised pretraining toward self-supervised and multimodal regimes, motivated by the cost of labels and the desire for more robust, generalizable representations. Contrastive self-supervised learning methods learn transferable features by enforcing instance discrimination under strong augmentations, demonstrating that representations can transfer competitively to detection and segmentation without labeled source data (He et al., 2020; Chen et al., 2020). In practice, these methods are applied similarly to supervised transfer—pretrain a backbone, then either “linear probe” (train only a linear head) or fine-tune end-to-end—but they shift the key design choices toward prevent task formulation, augmentation policy, and compute/memory trade-offs. A recurring theme across both supervised and self-supervised transfer is the tension between preserving generality (freezing layers or using small learning rates) and adapting to domain-specific cues (full fine-tuning), especially under domain shifts.</p> <p>Multimodal and cross-modal transfer learning further broadens what is transferred: not only visual features, but also alignment mechanisms and semantic grounding learned from paired modalities. Doher et al. (2020) exemplify this direction by proposing a decoupled cross-attention transfer mechanism that enables knowledge transfer from a source modality (A) to a target modality (B) for unimodal inference. Rather than merely sharing a backbone, their approach introduces a cross-attention loss that aligns query and value embeddings across modalities, effectively distilling relational structure learned with richer sensing into a model that operates with a single modality at test time (Doher et al., 2020). This formulation highlights an important evolution in transfer learning: transferring intermediate interaction patterns (e.g., attention correspondences) can be as critical as transferring static feature extractors, particularly when the target setting lacks modalities available during training.</p> <p>Vision-language models (VLMs) provide another prominent transfer route, enabling zero-shot or few-shot adaptation by leveraging language supervision and large-scale multimodal pretraining. However, the promise of training-free transfer raises questions about where such predictions should be trusted under distribution shift or ambiguous prompts. Dong et al. (2020) address this credibility dimension by studying trust in VLM predictions and proposing training-free adaptation strategies for zero-/few-shot settings, situating transfer learning not only as a performance tool but also as a calibration and robustness problem when models are deployed outside their pretraining distribution (Dong et al., 2020). This emphasis complements earlier transfer narratives focused on accuracy: as transfer increasingly occurs without task-specific training, uncertainty estimation and failure detection become critical to safe application.</p> <p>Transfer learning is also increasingly intertwined with robotics and embodied vision, where labeled data are scarce and the operational domain differs from web-scale imagery. Nava et al. (2020) demonstrate self-supervised learning for visual robot localization using LED state prediction as a pretext task, illustrating a pragmatic form of in-domain transfer: a model learns representations from an auxiliary objective that is easy to supervise <i>in situ</i>, and these representations support downstream localization without requiring extensive manual annotation (Nava et al., 2020). Compared with generic contrastive pretraining, such task-aligned self-supervision can better match the target environment’s statistics and sensing artifacts, mitigating domain mismatch that often limits transfer from natural-image corpora.</p> <p>Across these strands, the literature converges on a few practical regularities. First, transfer is rarely a single technique; it is a bundle of choices about what to reuse (backbone weights, attention alignments, multimodal correspondences), what to replace (heads, decoders), and how strongly to adapt (linear probing versus full fine-tuning). Second, the best transfer strategy depends on the target regime: limited labels favor stronger priors and constrained adaptation, while richer labels justify deeper fine-tuning (Krizhevsky et al., 2012; Chen et al., 2020). Third, as transfer expands to multimodal and training-free settings, reliability and modality gaps become first-order concerns, motivating methods that explicitly align representations across modalities or interrogate prediction trustworthiness (Doher et al., 2020; Dong et al., 2020). The implication is that future progress will likely come from transfer mechanisms that are not only label efficient but also explicitly robust to domain and modality shifts, especially in safety-critical deployments such as robotics and sensor-driven perception (Nava et al., 2020).</p> <p>References:</p> <ul style="list-style-type: none"> Chen, T., Kornblith, S., Norouzi, M., & Hinton, G. (2020). A simple framework for contrastive learning of visual representations. ICML 2020. [LLM] Doher, L., Wang, Z., & Mieles, M. (2020). D-CAT: Decoupled Cross-Attention Transfer between Sensor Modalities for Unimodal Inference. [CORPUS] Dong, H., Liu, M., Liang, J., Chatzil., E., & Fink, O. (2020). To Trust Or Not To Trust Your Vision-Language Model’s Prediction. [CORPUS] Girshick, R. (2015). Fast R-CNN. ICCV 2015. [LLM] He, K., Fan, H., Wu, Y., Xie, S., & Girshick, R. (2020). Momentum contrast for unsupervised visual representation learning. CVPR 2020. [LLM] Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet classification with deep convolutional neural networks. NeurIPS 2012. [LLM] Long, J., Shelhamer, E., & Darrell, T. (2015). Fully convolutional networks for semantic segmentation. CVPR 2015. [LLM] Nava, M., Carlotto, H., Crupi, L., Pelosi, D., & Giusti, A. (2020). Self-Supervised Learning of Visual Robot Localization Using LED State Prediction as a Pretext Task. [CORPUS]
--	---

Judge Evaluation

Winner: System B (Multi-Agent RAG)

<p>Judge Scores (1-5 scale)</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th>Metric</th><th>System A</th><th>System B</th><th>Δ</th></tr> </thead> <tbody> <tr> <td>Overall Quality</td><td>3</td><td>4</td><td>+1</td></tr> <tr> <td>Citation Quality</td><td>3</td><td>5</td><td>+2</td></tr> <tr> <td>Factual Grounding</td><td>4</td><td>4</td><td>0</td></tr> </tbody> </table>	Metric	System A	System B	Δ	Overall Quality	3	4	+1	Citation Quality	3	5	+2	Factual Grounding	4	4	0	<p>Judge Reasoning</p> <p>Response A clearly explains the core transfer-learning workflow (feature extractor vs fine-tuning) and supports it with canonical CV papers (e.g., Krizhevsky et al., 2012; Chen et al., 2020; He et al., 2020), but it largely stops at pre-2021 paradigms and omits newer transfer regimes (e.g., vision-language/zero-shot transfer). Response B covers the same fundamentals and adds recent, validated examples of multimodal and robotics transfer (e.g., D-CAT for cross-modal transfer (Doher et al., 2020) and self-supervised robot localization (Nava et al., 2020)), giving it broader, more current survey value; however, some higher-level claims about “trust” and “training-free adaptation” are less tightly tied to specific technical mechanisms than the core backbone/finetuning discussion.</p>
Metric	System A	System B	Δ														
Overall Quality	3	4	+1														
Citation Quality	3	5	+2														
Factual Grounding	4	4	0														

Figure C.6: Detailed individual query evaluation example 1 (Query Q008).

Q025: How do transformer architectures enable efficient attention mechanisms?

Responses
<p>Response A (Baseline - GPT-5.2) Citations: 6</p> <p>Transformer architectures achieve efficient attention by combining a highly parallelizable formulation of sequence modeling with a family of algorithmic and architectural modifications that reduce the computational and memory burdens of attention. The dominant research directions can be organized around (i) the baseline scaled dot-product attention that enables parallel computation, (ii) structural sparsification to avoid all-pairs interactions, and (iii) low-rank or kernel-based reformulations that change the asymptotic cost of attention.</p> <p>The foundational mechanism is the scaled dot-product attention introduced in Vaswani et al. (2017). "Attention Is All You Need," where attention weights are computed via a softmax over query-key dot products and used to aggregate value vectors. This design enables efficient training relative to recurrent models by eliminating sequential dependencies: all token-to-token interactions within a layer are computed with batched matrix multiplications on GPUs/TPLS. Multi-head attention further improves representational efficiency by allowing multiple subspaces of interactions at fixed per-head dimensionality, trading additional constant-factor compute for improved modeling capacity and parallelism. However, the baseline formulation has quadratic time and memory complexity in sequence length due to the dense attention matrix, motivating a large body of work on efficiency.</p> <p>A prominent direction is sparse attention, which reduces computation by restricting each token's receptive field. Child et al. (2019). "Generating Long Sequences with Sparse Transformers" introduced factored sparse patterns (e.g., striped and local blocks) that guarantee connectivity across positions while avoiding full all-pairs attention. This yields sub-quadratic complexity and makes long-context generation feasible, but it imposes an inductive bias: performance depends on whether the chosen sparsity pattern matches task structure. Beltagy et al. (2020). "Longformer: The Long Document Transformer" extended this idea with a combination of sliding-window local attention and a small number of global tokens that attend broadly. The global tokens provide a mechanism for document-level aggregation while keeping most attention local, offering a favorable trade-off for long-document NLP. The limitation is that global attention must be allocated carefully (often task-specific), and purely local windows can miss long-range dependencies unless mediated through multiple layers or global tokens.</p> <p>A second major direction replaces softmax attention with linear-time approximations. Katharopoulos et al. (2020). "Transformers are RNNs: Fast Autoregressive Transformers with Linear Attention" reformulated attention by using feature maps that make the attention computation associative, enabling computation as a sequence of prefix sums rather than explicit attention matrices. This reduces complexity from $O(n^2)$ to $O(n)$ in sequence length for autoregressive settings, substantially improving scalability. The trade-off is approximation: changing the attention kernel can alter optimization dynamics and may reduce accuracy on tasks where exact softmax attention is beneficial.</p> <p>Kernel-based random feature approximations represent another influential approach. Choromanski et al. (2021). "Rethinking Attention with Performers" approximated softmax attention using FAVOR+ (Fast Attention Via Positive Orthogonal Random features), yielding linear-time attention with unbiased (or low-bias) estimators under certain conditions. Performers preserve the ability to model global interactions without hand-designed sparsity patterns, and they can be efficient for long sequences. The key trade-off is variance/approximation error controlled by the number of random features: higher fidelity requires more features, increasing compute and memory.</p> <p>Low-rank projection methods offer a related but distinct strategy. Wang et al. (2020). "Linformer: Self-Attention with Linear Complexity" proposed projecting keys and values to a lower dimension along the sequence length axis, motivated by an empirical low-rank structure in attention maps. This reduces complexity to linear in sequence length while keeping a softmax-like mechanism. The approach can be effective when attention matrices are indeed low-rank, but it introduces a bottleneck that may degrade performance on tasks requiring rich, high-rank interactions; additionally, projection choices (learned vs. fixed) affect stability and generalization.</p> <p>Across these lines, efficient attention mechanisms hinge on exploiting structure—parallel matrix operations in the baseline transformer, sparsity patterns that reduce pairwise interactions, and algebraic approximations (kernelization or low-rank projections) that avoid materializing the full attention matrix. The central trade-offs are between expressiveness and scalability, and between task-agnostic global modeling (often favored by kernel/low-rank methods) versus task-aligned inductive bias (often favored by sparse/local designs). Collectively, these methods extend transformers to long-context regimes while preserving the core advantage established by Vaswani et al. (2017): attention as a parallelizable, content-addressable computation.</p> <p>References:</p> <p>Beltagy, I., Peters, M. E., & Cohan, A. (2020). Longformer: The Long-Document Transformer. arXiv.</p> <p>Child, R., Gray, S., Radford, A., & Sutskever, I. (2019). Generating Long Sequences with Sparse Transformers. arXiv.</p> <p>Choromanski, K., Likhoshesterov, V., Dohan, D., Song, X., Gane, A., Sarlos, T., Hawkins, P., Davis, J. Q., Mohiuddin, A., Kaiser, L., Belanger, D., Colwell, L., & Weller, A. (2021). Rethinking Attention with Performers. ICLR 2021.</p> <p>Katharopoulos, A., Vyas, A., Pappas, N., & Fleuret, F. (2020). Transformers are RNNs: Fast Autoregressive Transformers with Linear Attention. ICML 2020.</p> <p>Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2017). Attention Is All You Need. NeurIPS 2017.</p> <p>Wang, S., Li, B. Z., Khabsa, M., Fang, H., & Ma, H. (2020). Linformer: Self-Attention with Linear Complexity. arXiv.</p>
<p>Response B (Multi-Agent RAG) Citations: 7 (1 corpus, 6 LLM)</p> <p>Transformer architectures enable efficient attention mechanisms by reframing sequence modeling around parallelizable matrix operations and by motivating a family of approximations that mitigate the quadratic cost of dense self-attention. Historically, the modern "Transformer" label is associated with the architecture popularized by Vaswani et al. (2017), though Schmidhuber (2025) emphasizes that key ingredients—attention terminology and earlier forms of linearized self-attention—have deeper antecedents that were not always cited in later work. This historical perspective matters because many contemporary "efficient attention" methods can be read as rediscoveries or refinements of earlier ideas about avoiding explicit all-pairs interactions.</p> <p>In the canonical Transformer, scaled dot-product attention computes interactions between all query-key pairs, producing an $(n \times n)$ attention matrix for a length-n sequence. While this yields strong capacity for modeling long-range dependencies, it incurs $O(n^2)$ time and memory. The architectural shift away from recurrence is nonetheless a major efficiency gain in practice: by removing sequential dependencies in computation, Transformers allow attention and feed-forward sublayers to be implemented as large batched linear algebra kernels that exploit GPU/TPU parallelism (Vaswani et al., 2017). Multi-head attention further supports efficient computation because multiple attention "heads" can be realized through parallel projections and batched matrix multiplications, improving expressivity without sacrificing hardware-friendly structure (Vaswani et al., 2017). Thus, even before approximation methods, Transformers are "efficient" relative to RNNs in wall-clock training throughput due to parallelization, despite their unfavorable asymptotic scaling in context length.</p> <p>The quadratic bottleneck becomes dominant as contexts grow, motivating methods that reduce the number of computed interactions or avoid the full attention matrix. One prominent strategy is structured sparsity, which preserves the softmax attention form but restricts each token's receptive field. Child et al. (2019) propose sparse patterns (e.g., local windows combined with striped connections) that reduce complexity by limiting attention to a subset of positions. This approach is efficient because it replaces dense matrix multiplications with sparse or block-sparse operations, and it encodes inductive biases—especially locality—that often match natural language and other sequential signals. Beltagy et al. (2020) extend this idea with a hybrid design in which most tokens use sliding-window attention while a small set of designated "global" tokens attend broadly. The resulting computation is near-linear in sequence length for the local component, while still permitting long-range information flow through global positions (Beltagy et al., 2020). Across these sparse approaches, a recurring trade-off is that efficiency is gained by constraining connectivity: performance depends on whether the imposed sparsity aligns with task-relevant dependency structure (Child et al., 2019; Beltagy et al., 2020).</p> <p>A second strategy replaces softmax attention with kernelizable or linearized forms that permit associative reordering of computation. Katharopoulos et al. (2020) show that attention weights can be expressed via feature maps $\langle \phi(b) \phi(Q) \phi(b) \rangle$ ("top V(b g)), avoiding explicit $(n \times n)$ construction and yielding linear complexity in (n) (up to feature dimension). This line of work connects directly to Schmidhuber's (2025) observation that "linearized self-attention" ideas appeared earlier than the 2017 Transformer framing, underscoring that efficient attention is partly a matter of algebraic reformulation. Choromanski et al. (2021) further develop this direction by approximating softmax attention with random feature methods (FAVOR+), aiming to retain the benefits of softmax while achieving linear scaling. These methods offer global receptive fields without explicit sparsity, but they introduce approximation error and may require careful tuning of feature dimensionality to balance accuracy and efficiency (Katharopoulos et al., 2020; Choromanski et al., 2021).</p> <p>A third approach exploits low-rank structure in attention maps. Wang et al. (2020) propose projecting keys and values along the sequence dimension to a smaller rank, reducing complexity to linear in (n) for fixed projection size. Compared with random-feature approximations, low-rank projection is deterministic and often straightforward to integrate into standard Transformer stacks, but it can impose a representational bottleneck when attention patterns are not well-approximated by low-rank factors (Wang et al., 2020). In practice, these families—sparse, kernelized/linear, and low-rank—represent different hypotheses about where redundancy lies: in unnecessary token-to-token interactions (sparsity), in the softmax normalization itself (linearization), or in the intrinsic dimensionality of attention maps (low-rank).</p> <p>Overall, transformer architectures enable efficient attention through two complementary mechanisms: (i) a baseline design that is highly parallelizable and thus computationally efficient on modern accelerators (Vaswani et al., 2017), and (ii) a rich set of principled modifications that reduce the quadratic dependence on context length by restricting connectivity or reparameterizing attention computation (Child et al., 2019; Beltagy et al., 2020; Katharopoulos et al., 2020; Wang et al., 2020; Choromanski et al., 2021). The central implication is that "efficient attention" is not a single technique but a design space trading exactness and flexibility against scalability; the most effective choice depends on whether a task benefits more from global interactions, locality-biased structure, or compressible attention geometry, and on how these assumptions align with the data and deployment constraints.</p> <p>References</p> <p>Beltagy, I., Peters, M. E., & Cohan, A. (2020). Longformer: The Long-Document Transformer. [LLM]</p> <p>Child, R., Gray, S., Radford, A., & Sutskever, I. (2019). Generating Long Sequences with Sparse Transformers. [LLM]</p> <p>Choromanski, K., Likhoshesterov, V., Dohan, D., Song, X., Gane, A., Sarlos, T., Hawkins, P., Davis, J. Q., Mohiuddin, A., Kaiser, L., Belanger, D., Colwell, L., & Weller, A. (2021). Rethinking Attention with Performers. [LLM]</p> <p>Katharopoulos, A., Vyas, A., Pappas, N., & Fleuret, F. (2020). Transformers are RNNs: Fast Autoregressive Transformers with Linear Attention. [LLM]</p> <p>Schmidhuber, J. (2025). Who invented deep residual learning? [CORPUS]</p> <p>Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2017). Attention is all you need. NeurIPS 2017. [LLM]</p> <p>Wang, S., Li, B. Z., Khabsa, M., Fang, H., & Ma, H. (2020). Linformer: Self-Attention with Linear Complexity. [LLM]</p>

Judge Evaluation

Winner: System A (Baseline - GPT-5.2)

Judge Scores (1-5 scale)	Judge Reasoning		
Metric	System A	System B	Δ
Overall Quality	4	4	0
Citation Quality	4	3	-1
Factual Grounding	4	3	-1

Figure C.7: Detailed individual query evaluation example 2 (Query Q025).

Appendix D: Query Set Summary

The evaluation employed a comprehensive test set of 100 queries (`test_queries_100.csv`)² systematically distributed across research topics, corpus coverage levels, and evaluation segments to enable robust comparative analysis between baseline and multi-agent systems.

Query Set Characteristics

The test set incorporated three design principles:

Topic Diversity. Queries span 12 research areas, emphasizing Optimization (36%), Computer Vision (24%), and NLP (10%), with smaller categories (Adversarial, Reinforcement Learning, Multimodal, Robotics, Explainability, Healthcare, Privacy, Fairness) ensuring broad domain coverage.

Coverage Balance. Queries were distributed as high (27%), medium (54%), and low (19%) corpus coverage, testing retrieval quality, hybrid performance, and parametric knowledge reliance respectively.

Segment Organization. Equal distribution across four segments enabled progress tracking during evaluation.

ID	Topic	Coverage	Query Text
Q051	Adversarial	High	How do adversarial training methods improve model robustness?
Q053	Adversarial	Medium	How do researchers detect adversarial examples?
Q025	Computer Vision	High	How do vision transformers compare to convolutional neural networks?
Q008	Computer Vision	Medium	How is transfer learning applied in computer vision tasks?
Q027	Computer Vision	Low	How is few-shot learning applied to computer vision tasks?
Q007	Edge Computing	High	What model compression techniques reduce memory and computation requirements?
Q023	Edge Computing	Medium	How is knowledge distillation used to compress neural networks?
Q014	Multimodal	Medium	What approaches improve multimodal learning combining vision and language?
Q001	NLP	High	How does a multi-agent architecture compare to single-model approaches?
Q005	NLP	Medium	How is explainability and interpretability achieved in language models?
Q021	Optimization	High	How do researchers optimize gradient descent for faster convergence?
Q029	Optimization	Medium	How do researchers address vanishing gradients in deep networks?
Q058	Optimization	Low	What ensemble methods improve model performance and robustness?
Q003	Reinforcement Learning	Medium	How do researchers improve sample efficiency in reinforcement learning?
Q019	Robotics	Medium	How is reinforcement learning applied to robotic manipulation tasks?

Table D9: Representative Query Sample

² Complete query set available at: https://github.com/ElsonFilho/research-navigator/tree/main/data/test_queries