Paper Circle: An Open-source Multi-agent Research Discovery and Analysis Framework

Anonymous ACL submission

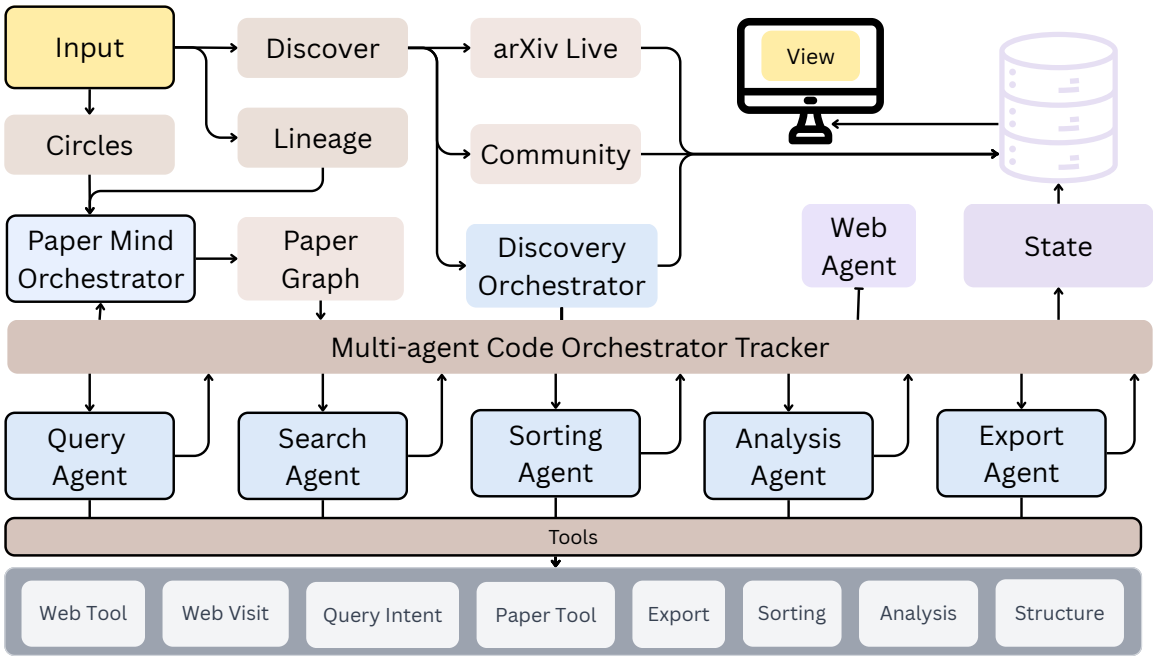

Figure 1: Overview of the Paper Circle pipeline. Given a user query, Paper Circle builds a paper set from multiple sources (e.g., paper graph, community, and arXiv live) via the Paper Mind for analysis and Discovery Orchestrators for search of the paper. A multi-agent layer (query, search, sorting, analysis, export) is coordinated by the Tracker, which maintains a shared state that is persisted to a backing store and displayed to the user through interface.

Abstract

The rapid growth of scientific literature has made it increasingly difficult for researchers to efficiently discover, evaluate, and synthesize relevant work. Recent advances in multi-agent large language models (LLMs) have demonstrated strong potential for understanding user intent and are being trained to utilize various tools. In this paper, we introduce Paper Circle, a multi-agent research discovery and analysis system designed to reduce the effort required to find, assess, organize, and understand academic literature. The system comprises two complementary pipelines: (1) a Discovery Pipeline that integrates offline and online retrieval from multiple sources, multi-criteria scoring, diversity-aware ranking, and structured outputs; and (2) an Analysis Pipeline that transforms individual papers into structured knowledge graphs with typed nodes (e.g., concepts, methods, experiments, and figures) and edges, enabling graph-aware question answering and coverage verification. Both pipelines are implemented within a coder LLM–based multi-agent orchestration framework and produce fully reproducible, synchronized outputs (JSON, CSV, BibTeX, Markdown, and HTML) at each agent step. This paper describes the system architecture, agent roles, retrieval and scoring methods, knowledge graph schema, and evaluation interfaces that together form the Paper Circle research workflow. We benchmark Paper Circle on both paper retrieval and paper review generation, reporting hit rate, MRR, and Recall@K. Results show consistent improvements with stronger agent models. We will

publicly release the website and source code.

1 Introduction

The pace of scientific publication has accelerated exponentially, creating a significant burden on researchers attempting to stay abreast of new developments (Reddy and Shojaee, 2025; Pramanick et al., 2023). Traditional search engines and recommendation systems often struggle to provide the depth and context required for rigorous literature reviews, leading to fragmented discovery workflows. Recently, the advent of Large Language Models (LLMs) has catalyzed a shift towards "AI Scientists", autonomous multi-agent systems (MAS) capable of generating hypotheses, conducting experiments, and even writing papers (Chen et al., 2025b; Naumov et al., 2025). While these systems demonstrate the potential of agentic workflows, there remains a critical gap between fully autonomous simulations and the practical, collaborative needs of human research communities.

Paper Circle addresses (as shown in the Figure 1) this gap by introducing a comprehensive *Multi-Agent Research Platform* that supports the entire lifecycle of literature engagement: from discovery and analysis to critique and synthesis. Unlike purely autonomous systems that aim to replace the researcher, Paper Circle is designed as a collaborative workbench that augments human intelligence through three integrated subsystems:

1. **Discovery Pipeline**: A multi-agent retrieval system that goes beyond simple keyword matching. It employs a multi-dimensional scoring framework to surface high-value research. Crucially, this pipeline is deterministic and produces structured artifacts (JSON, linear logs) at every step.

2. **Paper Mind Graph**: To facilitate deep understanding, Paper Circle constructs a dynamic Knowledge Graph from retrieved literature. This "Paper Mind" enables researchers to query the collective intelligence of a reading list, identifying latent connections between disparate works and supporting complex Question-Answering workflows that are grounded in specific citation sub-graphs.

3. **Review Agents**: This platform features a team of specialized review agents that generate detailed critiques and scores, consistently highlighting strengths and weaknesses to guide human reading priorities (Naumov et al., 2025).

By integrating these capabilities into a shared "Reading Circle" environment, Paper Circle transforms literature review from a solitary task into a community-driven, AI-augmented operation.

2 Related Work

2.1 Autonomous Scientific Discovery

The emerging field of AI-Scientists aim to automate the entire research lifecycle. Systems like DORA AI agent (Naumov et al., 2025) and EvoResearch (Gajjar, 2025) demonstrate end-to-end capabilities, from hypothesis generation to report writing. Similarly, O-Researcher (Li et al., 2025), MARS (Chen et al., 2025a), and AlphaResearch (Yu et al., 2025c) treat research as a multi-step optimization problem, often using reinforcement learning to refine discovery strategies. Specialized agents have also been proposed for causal discovery, such as CausalSteward (Wang et al., 2025) and other multi-agent frameworks (Le et al., 2025). While these systems push the boundaries of autonomy, Paper Circle prioritizes *curation and reproducibility* over full automation. Instead of replacing the researcher, Paper Circle acts as a force multiplier for human teams, ensuring that the discovery process remains transparent and verifiable.

2.2 MAS in Specialized Domains

MAS have shown remarkable success in specific scientific verticals. In chemistry and materials science, frameworks like ChemThinker (Ju et al., 2025), MOOSE-Chem (Yang et al., 2025), and ChemBOMAS (Han et al., 2025a) leverage LLMs to discover new molecules and optimize experiments (Kumbhar et al., 2025). In biology and healthcare, agents facilitate single-cell analysis (CellAgent (Xiao et al., 2024)), phenotype discovery (PhenoGraph (Niyakan and Qian, 2025)), and clinical data analysis (Spieser et al., 2025). Other applications range from drug discovery (Fehlis et al., 2025) and psychiatry diagnosis (Xiao et al., 2025) to financial forecasting, where systems like ASTRAFIN (Singh and Kumar, 2025) and other stock analysis agents (Chandrashekar et al., 2025; Wawer and Chudziak, 2025) predict market trends. Paper Circle complements these domain-specific tools by providing a *general-purpose* discovery pipeline that can be adapted to any discipline, serv-

ing as the foundational layer for literature review and knowledge management.

2.3 Community Simulation and Collaboration

A distinct line of research focuses on simulating or facilitating the social aspects of science. ResearchTown (Yu et al., 2025a,b) models the research community using agents to understand how ideas propagate. Other works explore collaborative dynamics through automated negotiation (NegoLog (Doğru et al., 2024), NEGOTIATOR (Keskin et al., 2024)) and cohesive dialogue generation (Chu et al., 2024). Frameworks like PiFlow (Pu et al., 2025), RED-EREF (Yuan and Xie, 2025), and blackboard systems (Salemi et al., 2025) propose mechanisms for agent collaboration in information discovery. Paper Circle distinguishes itself by moving beyond simulation; it provides a real-world platform for *human-AI collaboration*. It does not just model how researchers might interact, but actively facilitates those interactions through shared reading lists, discussion threads, and collaborative ranking.

3 Methodology

3.1 Background

Multi-Agent Systems (MAS) represent a paradigm where autonomous entities interact to solve complex problems distributedly. In the context of scientific discovery, MAS allows for the decomposition of intricate research tasks,such as literature search, reading, and reasoning,into manageable sub-routines handled by specialized agents (Wooldridge, 2002). Unlike monolithic LLM approaches, agentic workflows can maintain distinct personas (e.g., "The Skeptic", "The Creative") and leverage external tools, reducing hallucination and improving reasoning depth through inter-agent dialogue (Reddy and Shojaee, 2025).

The baseline for our orchestration layer is the `smolagents` (Roucher et al., 2025) library. The pipeline uses a `CodeAgent (CoA)` as the central orchestrator, which can attend parallel agent calls and toll calls and multiple `ToolCallingAgent (ToCA)` instances, each attached to specific capabilities (e.g., arXiv retrival, PDF parsing). The baseline responsibilities include (i) tool invocation, (ii) multi-step planning via the orchestrator, and (iii) delegation to specialized agents. PaperCircle extends this foundation by adding structured outputs, offline search capabilities, and rigorous evaluation metrics. We preserve the baseline tool interface,

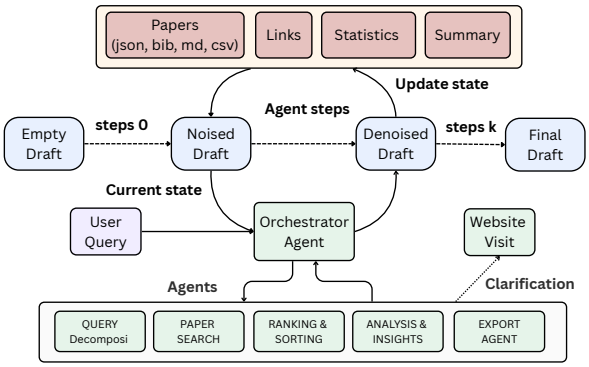

Figure 2: The main iterative diagram for the paper discovery framework. The system maintains an explicit, evolving discovery state (papers, links, statistics, and summaries) that is iteratively updated through agentic steps. Starting from an empty draft, the orchestrator agent alternates between noising and denoising operations over multiple steps, progressively refining the draft into a final result. When necessary, a web search agent is invoked for clarification or recent information.

where each tool receives explicit parameters and returns a formatted string response, allowing the orchestrator to chain steps while maintaining high readability and traceability.

3.2 System Architecture

Figure 1 illustrates the overall architecture of Paper Circle. The system consists of two complementary multi-agent pipelines: the *Discovery Pipeline* for finding relevant papers, and the *Analysis Pipeline* for deep understanding of individual papers.

3.3 Paper Discovery Agent Design

The main diagram of the discovery subsystem is shown in Figure 2, which is composed of multiple agents, each bound to a small, explicit tool interface. It is inspired by the TTD-DR (Han et al., 2025b) for iteratively updating the updated version at each agentic step. The core agents are:

Intent Classification Agent. Parses user text into search mode (offline, online, or both), conference filters, year range, and ranking preferences. Most importantly, it uses a web agent in the pipeline for any unclear queries or recent knowledge.

Paper Search Agent. Executes offline or online retrieval based on intent, merges results, performs deduplication, and updates state and outputs.

Sorting Agent. Reorders papers using recency, citations, similarity, novelty, BM25 (Chen and Wiseman, 2023), or combined weights; or applies a cross-encoder reranker (Wang et al., 2020).

Analysis Agent. Computes aggregate statistics

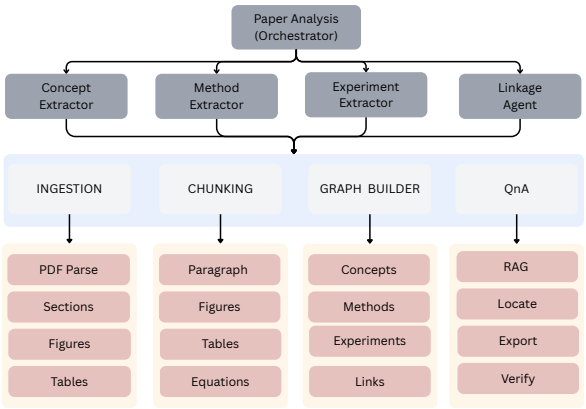

Figure 3: A paper analysis orchestrator agents for concepts, methods, experiments, and cross-entity linkages. The pipeline consists of four main stages: ingestion, which parses PDFs into structured elements (sections, figures, tables, equations); semantic chunking, which produces structure-aware text units; graph construction, which builds a typed knowledge graph of concepts, methods, experiments, and their relations with full traceability to source text; and a Q&A layer that enables graph-aware retrieval, verification, and export.

and insights, including source distribution, year trends, and top authors.

Export Agent. Produces synchronized exports and provides a consistent interface for downstreaming.

Web Search Agent. Provides auxiliary access to web search tools when online lookups are required.

3.4 Paper Analysis Agent

While the discovery pipeline addresses the challenge of finding relevant papers, researchers also need to understand and synthesize the content of individual papers deeply (Korat, 2025). Paper Circle addresses this with a complementary *Paper Analysis Agent* that transforms research papers into structured, queryable knowledge graphs with full traceability to the original text. The Paper Analysis Agent operates as a multi-stage pipeline with four specialized components as shown in the figure: (1) Ingestion Layer, (2) Graph Builder, (3) Q&A System, and Verification Layer.

PDF Ingestion and Chunking. The ingestion pipeline uses PyMuPDF for robust PDF parsing (Adhikari and Agarwal, 2024). The `PDFParser` class extracts: **Metadata**: Title, authors, abstract, arXiv ID, venue, and page count. **Sections**: Hierarchical section structure with parent-child relationships, identified via numbering patterns (e.g., "1.2 Background"). **Figures and Tables**: Caption text, page locations, and nearby context for linkage. **Equations**: Numbered equations with surrounding context.

Unlike token-based chunking, the `SemanticChunker` (Qu et al., 2025) creates chunks aligned with document structure. Paragraphs within sections are grouped up to a configurable limit (default 1500 characters), while figures, tables, and equations are preserved as distinct chunks with their captions and context.

Knowledge Graph Schema. The mind graph follows a typed schema with nodes (Zhang et al., 2025a) for papers, sections, concepts, methods, experiments, datasets, and visual elements (figures, tables, equations), and edges encoding structural and semantic relations (e.g., hierarchy, definition, proposal, usage, evaluation, illustration, dependency). All nodes and edges carry provenance metadata—including source chunk IDs, page numbers, verification status, confidence scores, and timestamps—ensuring full traceability to the original PDF.

3.5 Multi-Agent Extraction

The `GraphBuilder` (Zhu et al., 2024b) orchestrates four specialized `CoA`-based extractors. The *Concept Extractor* identifies and classifies key concepts by type and importance; the *Method Extractor* extracts algorithms and techniques from method sections; the *Experiment Extractor* recovers experimental setups, datasets, metrics, and results; and the *Linkage Agent* connects figures and tables to the concepts or methods they illustrate. Extraction proceeds in staged phases—concepts, methods, experiments, visual linkage, and inter-concept relations—each incrementally updating the shared `MindGraph`.

Graph-Aware Q&A. The Q&A module combines vector retrieval with graph traversal. An `EmbeddingStore` indexes text chunks and node descriptions, while the `GraphRetriever` retrieves top-k relevant nodes and chunks and expands context via 1-hop neighbors. The `PaperQA` agent generates answers grounded in retrieved text, graph relations, and linked figures or tables, and returns supporting evidence with confidence estimates. A `locate` function enables precise localization of concepts, figures, or tables by page and context.

Coverage Verification. To prevent silent omissions, a `CoverageChecker` evaluates figure, table, section, and equation coverage, producing an over-

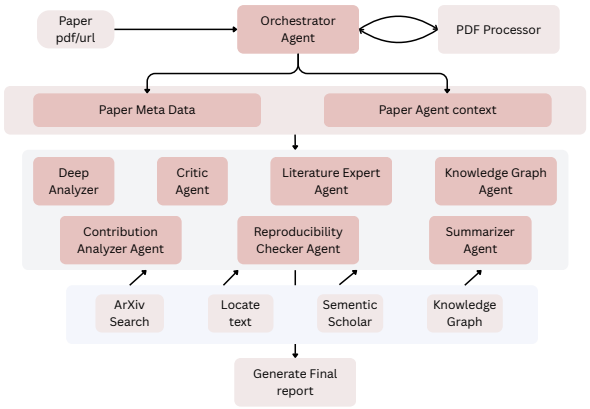

Figure 4: Multi-agent paper analysis and review architecture. Given a paper specified by a PDF or URL, an orchestrator agent coordinates PDF processing and maintains shared paper metadata and agent context. Specialized agents operate in parallel to perform deep technical analysis, contribution extraction, critical review, literature linking, reproducibility checking, summarization, and knowledge graph construction. External tools such as arXiv search, Semantic Scholar, and targeted text localization are invoked as needed. The orchestrator aggregates agent outputs into a unified, structured final report, enabling comprehensive, reviewer-style analysis with modular extensibility.

all coverage score and identifying unlinked or missing elements with actionable diagnostics. This provides a lightweight quality assurance step prior to downstream use.

3.6 Research Review Framework

In Sec. 3.4, we describe the paper analysis of agentic capabilities, which we further extend for automated peer-review-style assessment. Unlike AgentReview (Jin et al., 2024; D'Arcy et al., 2024), we follow the paper analysis perspective, which not only provides the review but also builds a strong graph between the concepts.

Architecture. The system is built upon a multi-agent orchestration framework (Figure 4) that coordinates the execution of seven specialized roles. Each agent is instantiated as a `ToCA` or `CoA` (Roucher et al., 2025).

Deep Analyzer. Focuses on the technical core of the paper. It breaks down the mathematical foundations, identifies specific methodology components, and extracts primary experimental findings.

Critic. Emulates a senior conference reviewer (e.g., NeurIPS, ICML). It provides a rigorous assessment of strengths and weaknesses, generates author-facing questions, and assigns scores for novelty, clarity, and significance.

Literature Expert. Interfaces with external academic databases including Semantic Scholar and arXiv. It maps the paper's position within the existing research landscape and verifies citation accuracy.

Contribution Analyzer. Separates explicit author claims from verified technical contributions, identifying potential overclaiming or missing baseline comparisons.

Reproducibility Checker: Quantifies the transparency of the research by assessing the availability of source code, hyperparameter specifications, dataset accessibility, and compute requirement disclosures.

Summarizer. Generates multi-fidelity summaries across different abstraction levels, ranging from concise executive summaries to deep technical precis.

Orchestration and Pipeline Execution The `Multi Agent Orchestrator` manages the lifecycle of these agents through a multi-stage pipeline. The system supports parallel execution using a `ThreadPoolExecutor`.

4 Experiments

4.1 Experimental setup

All the experiments are done with open-source model with 4×40 GB Nvidia GPUs. We used the Ollama[1] platform with the fastllm library (Gong et al., 2025).

Database Curation. We curated a diverse corpus, as shown in Table 1 of research papers from leading CS and ML conferences, primarily sourced from OpenReview[2] and augmented with metadata and peer-review information.

Evaluation. Paper Circle provides built-in evaluation metrics. When a ground-truth paper title or identifier is provided, the system computes Mean Reciprocal Rank (MRR), Recall@K, Precision@K, and hit rates. These metrics are computed per step and stored in the JSON file for longitudinal tracking. For batch evaluation, a parallel benchmarking utility executes multiple queries concurrently and aggregates mean metrics and timing statistics. This

[1] https://ollama.com/
[2] https://openreview.net/

Conference	ICLR	NeurIPS	ICML	CVPR	IROS	ICRA	AAAI	ACL	ICCV	EMNLP	Other
Count	12	39	13	13	25	25	5	5	7	4	144

Table 1: The Database corpus across major conferences. The "Other" category includes venues such as AISTATS, RSS, SIGGRAPH, and WACV. **Count** indicates the number of the most recent conference venue included.

supports lightweight comparisons between search configurations (offline vs. online, BM25 (Chen and Wiseman, 2023) vs. semantic (all-MiniLM-L6-v2 (Wang et al., 2020)), with or without Qwen3-Reranker-0.6B (Zhang et al., 2025b)) without requiring external tooling.

Baseline Agent. This framework is developed using the Smolagent multi-agent tool, calling the (ToCA) agent and the code agent (CoA), with tools utilized being manually developed.

Architecture. We evaluate multiple retrieval baselines: `bm25`, `bm25+reranker` (BM25 (Chen and Wiseman, 2023)& cross-encoder (Zhang et al., 2025b)), `reranker` (Zhang et al., 2025b), `semantic` (Wang et al., 2020), and `hybrid` (BM25 combined with semantic retrieval). We also compare pipeline structures with different agent compositions: `full` includes all five agents (intent, search, sort, analysis, export), `minimal` uses only the search agent, `search_sort` uses search and sort, `search_analysis` uses search and analysis, and `no_intent` is a full pipeline with no intent.

4.2 Results

Natural Text-based retrieval. We evaluate our multi-agent paper retrieval system across multiple LLMs and retrieval baselines. We did two query type experiments, one a research assistant-based natural queries generated by running gpt-oss-20B models (called RAbench), and randomly sampling one paper record from the database, extracting a concise "topic" phrase from its title, keywords or abstract, then picking a natural-language template and optional prefix to turn that topic into a realistic search query. We also randomly chose a scope (conference/year/range/none) to add corresponding text to the query and to emit matching structured filters. This query we referred to as SemanticBench.

All experiments were conducted on a 50 query benchmark, measuring the success rate, the hit rate, the mean reciprocal rank (MRR), and the recall.

Model Comparison. Table 2 presents comprehensive evaluation results comparing agent-based models with retrieval baselines. The results reveal a clear performance hierarchy across methods and scales. Two agent models achieve the highest retrieval effectiveness with an 80% hit rate, qwen3-coder-30b-Q3KM (quantized) and `qwen3-coder:30b`—with qwen3-coder-30b-Q3KM also delivering the best ranking quality (MRR = 0.627) while requiring less memory for smolagent multi-step reasoning. These top-performing models are also the fastest, taking approx. 21–22 seconds per query, indicating no latency penalty for improved accuracy. The BM25 baseline remains highly competitive (78% HR), outperforming most agent-based approaches and highlighting the continued strength of lexical matching in academic retrieval. Finally, RA-Bench results show higher performance than SemanticBench, suggesting that LLM-perturbed queries may be easier for multi-agent retrieval, though this requires further investigation.

Paper analysis visualization. In the Figure 3, we provide various output visualizations, including concept built graph (A), concept definition chart (B), interactive Q&A with precise information (C), markdown analysis output (D), and finally flow chart connecting the concepts of blocks (E). All of this analysis togather provides the complete understanding of the paper.

Paper review analysis To evaluate our multi-agent review system, we conducted a study using the released ICLR 2024 reviews. We randomly selected 50 papers spanning diverse rating levels, and report the results in Figure 6. We observe that the code-oriented agent (`qwen3-coder-30B`) often struggles to sustain a coherent review workflow, whereas chat-style LLMs (e.g., `gpt-oss`) produce stronger and more consistent reviews. Overall, review quality improves with larger models, suggesting that capacity and instruction-following are particularly important for end-to-end reviewing.

4.3 Ablation Studies

We conduct comprehensive ablation studies to understand the contribution of different system com-

Table 2: Combined benchmark results for agent-based models and retrieval baselines. Best results are shown in **bold**. All the results are calculated using semantic benchmarks. Only the last (blue) is evaluated on 500 RAbench queries, which shows syntetically written query is easier to retrieve compared to the random template following.

Model/Method	Type	Success	Hit Rate	MRR	R@1	R@5	R@10	R@20	R@50	Time (s)	Steps
Qwen3C-30B-Inst-Q3_K_M	Agent	100%	**0.80**	**0.627**	**0.58**	**0.66**	**0.74**	**0.78**	**0.80**	22.2	1.42
qwen3-coder:30b (Team, 2025)	Agent	100%	**0.80**	0.518	0.46	0.52	0.72	0.76	**0.80**	**21.1**	1.34
BM25 (Chen and Wiseman, 2023)	Baseline	100%	0.78	0.541	0.48	0.60	0.66	0.78	0.78	–	–
microcoder-deepseekr1-14.8	Agent	52%	0.73	0.453	0.38	0.46	0.65	0.69	0.73	107.4	4.15
deepseek-coder-v3:16b (Zhu et al., 2024a)	Agent	100%	0.66	0.396	0.32	0.46	0.52	0.60	0.66	47.9	1.54
qwen2.5-coder:3b (Hui et al., 2024)	Agent	94%	0.60	0.366	0.28	0.45	0.53	0.55	0.57	210.4	1.51
qwen2.5-coder:14b (Hui et al., 2024)	Agent	82%	0.56	0.461	0.41	0.51	0.51	0.56	0.56	73.4	1.05
Semantic (Wang et al., 2020)	Baseline	100%	0.54	0.279	0.22	0.32	0.38	0.52	0.54	–	–
Simple (bag-of-words)	Baseline	100%	0.54	0.279	0.22	0.32	0.38	0.52	0.54	–	–
qwen2.5-coder:7b (Hui et al., 2024)	Agent	100%	0.54	0.311	0.26	0.36	0.40	0.52	0.54	59.3	0.84
Qwen3C-30B-Inst-Q3_K_M	Agent	100%	0.42	0.348	0.32	0.38	0.38	0.40	0.42	22.7	1.40
deepseek-coder:33b (Zhu et al., 2024a)	Agent	100%	0.12	0.087	0.08	0.08	0.12	0.12	0.12	180.4	0.14
qwen3vl-4b-orlex	Agent	12%	0.08	0.080	0.08	0.08	0.08	0.08	0.08	37.9	0.14
granite-code:34b (Mishra et al., 2024)	Agent	100%	0.02	0.010	0.00	0.02	0.02	0.02	0.02	111.3	0.04
Hybrid (BM25+sementic)	Baseline	100%	0.02	0.001	0.00	0.00	0.00	0.00	0.02	–	–
qwen2.5-coder:1.5b (Hui et al., 2024)	Agent	100%	0.00	0.000	0.00	0.00	0.00	0.00	0.00	63.7	0.00
microcoder-oss-20b	Agent	54%	0.00	0.000	0.00	0.00	0.00	0.00	0.00	47.6	0.00
Qwen3-Coder-30B-A3B-Inst-Q3_K_M	Agent	100%	0.98	0.882	0.83	0.93	0.95	0.96	0.97	21.53	1.36

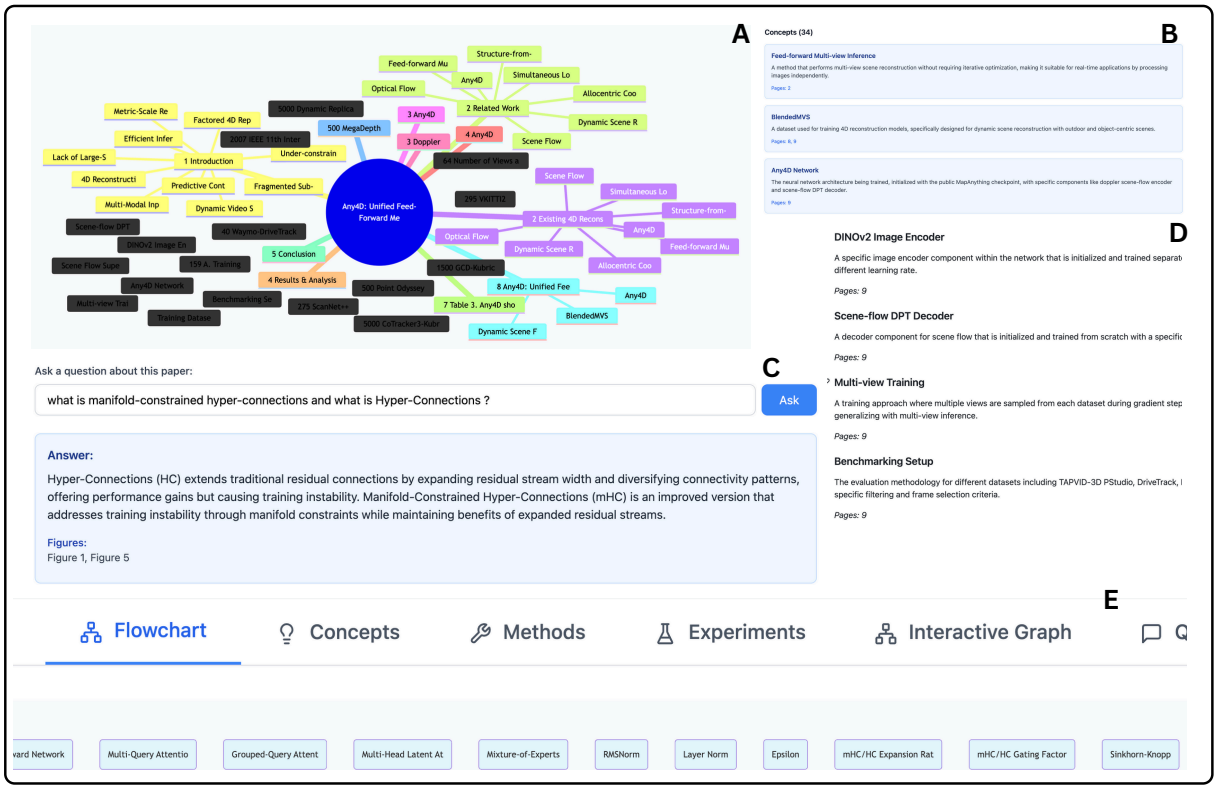

Figure 5: The main outputs of the analysis agent for a representative paper. (A) Interactive concept graph constructed from the paper, where nodes correspond to extracted concepts and edges denote semantic relationships. (B) Automatically generated concept explanations, each linked to the originating paper sections and pages. (C) Graph-aware question answering interface, providing answers grounded in extracted content along with supporting figures and references. (D) Structured Markdown exports summarizing all extracted concepts and methods for downstream use. (E) Flowchart view illustrating the high-level organization and relationships among concepts, methods, and experimental components of the paper.

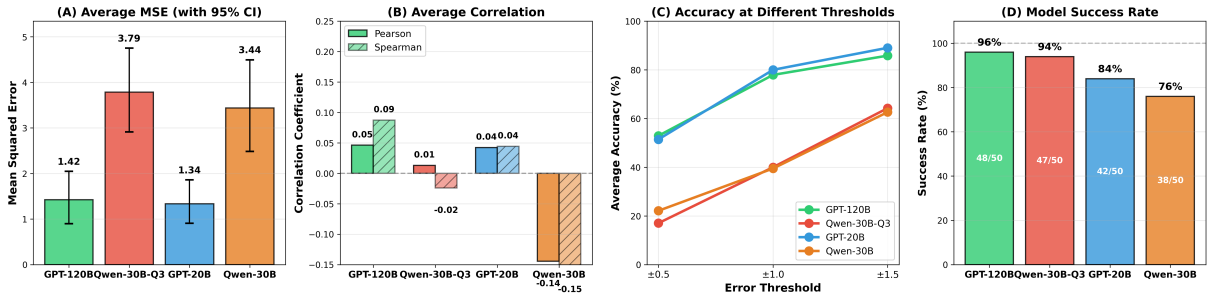

Figure 6: Paper review results analysis. This study was conducted on 50 randomly selected ICLR 2024 reviews.

ponents, including retrieval baselines, query configuration, and pipeline structures. **Full Query utilization**

To assess the full capability of our system, we conducted an extended evaluation using the `qwen3-coder-30b` model across 500 queries under various configurations. Results are presented in Table 3.

Table 3: Extended benchmark results for the Qooba agent (`qwen3-coder-30b`) across different configurations.

Configuration	Queries	Hit Rate	MRR	R@1	R@5	Time (s)
Default (Full Agent)	500	**0.9818**	**0.8824**	**0.8381**	**0.9312**	**21.54**
With Filters & Offline	50	0.9600	0.8485	0.7800	0.9000	22.76
Offline Only	50	0.9200	0.6476	0.5600	0.7400	41.45
No Mentions	50	0.6400	0.4316	0.3600	0.5200	38.35
Online/Offline Mix	50	0.6200	0.4595	0.4200	0.5000	38.50

Observations. The "With Filters & Offline" configuration performs better, suggesting that explicit context (conference/year filters) combined with local database access is highly effective. Notably, the "No Mentions" and "Online/Offline Mix" configurations show significant performance degradation (62–64% hit rate), indicating that specific paper references and structured retrieval chains are critical for accuracy. Overall, configurations exhibit similar latency, indicating stable scaling of the multi-agent pipeline across query settings as well.

4.4 Retrieval Baseline Ablations

Table 4: Ablation study results comparing retrieval baselines and pipeline structures using `qwen3-coder-30b`. Full represents the full pipeline structure, minimal represents

Configuration	Baseline	Structure	Hit Rate	MRR	R@1	R@5	Time (s)
BM25 Full	bm25	full	0.9600	0.8629	0.8000	0.9200	33.75
BM25 Search Sort	bm25	search_sort	0.9600	0.8620	0.8000	0.9200	33.95
BM25 No Intent	bm25	no_intent	0.9600	0.8554	0.8000	0.9200	**31.47**
BM25 Search Analysis	bm25	search_analysis	0.9600	0.8437	0.7800	0.9200	32.81
BM25 Minimal	bm25	minimal	0.9600	0.8420	0.7800	0.9200	33.34
Hybrid Full	hybrid	full	0.9600	0.8620	0.8000	0.9200	31.65
BM25 + Reranker	bm25+reranker	full	0.9600	**0.8692**	0.8000	**0.9400**	935.07
Semantic Full	semantic	full	0.9400	0.7097	0.6200	0.8800	31.28

Retrieval Baseline Impact. BM25-based methods consistently outperform pure semantic retrieval. The semantic baseline shows a significant drop in R@1 (0.62) compared to BM25-based methods (0.80), suggesting that lexical matching remains crucial for precise paper retrieval. The hybrid approach performs on par with BM25, indicating that combining lexical and semantic signals does not provide additional benefits in this setting.

Reranking Trade-offs. The BM25 + Reranker configuration achieves the highest MRR (0.8692) and R@5 (0.9400), but at a substantial computational cost, approximately 28× slower than other methods. This presents a clear accuracy-efficiency trade-off that practitioners must consider based on their deployment requirements.

Pipeline Complexity. Reducing pipeline complexity (Minimal, Search Analysis configurations) leads to slight drops in MRR and R@1 while maintaining high overall hit rates (96%). Interestingly, removing intent analysis ("No Intent" configuration) results in a faster pipeline with competitive performance, suggesting that intent classification may be redundant for well-structured queries.

5 Conclusion

Paper Circle shows how multi-agent workflows can streamline research literature management. Its discovery pipeline unifies heterogeneous search sources and multi-criteria scoring into a reproducible tool, using a simple agent–tool interface with shared state, deterministic ranking, and synchronized multi-format outputs. Its analysis pipeline converts papers into structured knowledge graphs that enable graph-aware QA, coverage checks, and human-in-the-loop verification. Future work will focus on the optimization of the unification of the pipeline.

6 Limitations

Our review agent shows weak alignment with human judgments: across models, the correlation with human reviewer scores remains low ($r < 0.25$), and several metrics can even exhibit negative correlations, indicating that the system may rank papers in the opposite order of human preference. As a result, even the best-performing configurations do not reliably distinguish strong from weak submissions, and the system should not be used as a trusted mechanism for comparing or ranking papers. Based on our analysis, we found that this review process gets the benefit of a large model, so this problem can be overcome by large open/closed source models.

References

Narayan S Adhikari and Shradha Agarwal. 2024. A comparative study of pdf parsing tools across diverse document categories. *arXiv preprint arXiv:2410.09871*.

Prof. Chandrashekar, M. Akram, Mohin Khan, Piyush Kumar, and Pratap Mandal. 2025. A survey on stock investment risk analysis using crewai multi- agent system. *International Research Journal of Modernization in Engineering Technology and Science*.

Guoxin Chen, Zile Qiao, Wenqing Wang, Donglei Yu, Xuanzhong Chen, Hao Sun, Minpeng Liao, Kai Fan, Yong Jiang, Wayne Xin Zhao, and 1 others. 2025a. Mars: Optimizing dual-system deep research via multi-agent reinforcement learning. *arXiv preprint arXiv:2510.04935*.

Renqi Chen, Haoyang Su, SHIXIANG TANG, Zhenfei Yin, Qi Wu, Hui Li, Ye Sun, Wanli Ouyang, Philip Torr, and Nanqing Dong. 2025b. Ai-driven automation can become the foundation of next-era science of science research. *NIPS 2025*.

Xiaoyin Chen and Sam Wiseman. 2023. Bm25 query augmentation learned end-to-end. *arXiv preprint arXiv:2305.14087*.

KuanChao Chu, Yi-Pei Chen, and Hideki Nakayama. 2024. Cohesive conversations: Enhancing authenticity in multi-agent simulated dialogues. *COLM 2024*.

Mike D'Arcy, Tom Hope, Larry Birnbaum, and Doug Downey. 2024. Marg: Multi-agent review generation for scientific papers. *arXiv preprint arXiv:2401.04259*.

Mamata Das, PJA Alphonse, and 1 others. 2023. A comparative study on tf-idf feature weighting method and its analysis using unstructured dataset. *arXiv preprint arXiv:2308.04037*.

Anıl Doğru, Mehmet Onur Keskin, Catholijn M. Jonker, Tim Baarslag, and Reyhan Aydoğan. 2024. Negolog: An integrated python-based automated negotiation framework with enhanced assessment components. *IJCAI 2024*.

Yao Fehlis, Charles Crain, Aidan Jensen, Michael Watson, James Juhasz, Paul Mandel, Betty Liu, Shawn Mahon, Daren Wilson, and Nick Lynch-Jonely. 2025. Accelerating drug discovery through agentic ai: A multi-agent approach to laboratory automation in the dmta cycle. *arXiv.org*.

Prof.Anjali Gajjar. 2025. Evoresearch: A multi-agent ai framework for automated paper analysis. *International Journal of Innovative Research in Advanced Engineering*.

Ruihao Gong, Shihao Bai, Siyu Wu, Yunqian Fan, Zaijun Wang, Xiuhong Li, Hailong Yang, and Xianglong Liu. 2025. Past-future scheduler for llm serving under sla guarantees. In *Proceedings of the 30th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2*, pages 798–813.

Dong Han, Zhehong Ai, Pengxiang Cai, Shanya Lu, Jianpeng Chen, Zihao Ye, Shuzhou Sun, Ben Gao, Lingli Ge, Weida Wang, and 1 others. 2025a. Chembomas: Accelerated bo in chemistry with llm-enhanced multi-agent system. *arXiv preprint arXiv:2509.08736*.

Rujun Han, Yanfei Chen, Zoey CuiZhu, Lesly Miculicich, Guan Sun, Yuanjun Bi, Weiming Wen, Hui Wan, Chunfeng Wen, Solène Maître, and 1 others. 2025b. Deep researcher with test-time diffusion. *arXiv preprint arXiv:2507.16075*.

Binyuan Hui, Jian Yang, Zeyu Cui, Jiaxi Yang, Dayiheng Liu, Lei Zhang, Tianyu Liu, Jiajun Zhang, Bowen Yu, Kai Dang, and 1 others. 2024. Qwen2. 5-coder technical report. *arXiv preprint arXiv:2409.12186*.

Yiqiao Jin, Qinlin Zhao, Yiyang Wang, Hao Chen, Kaijie Zhu, Yijia Xiao, and Jindong Wang. 2024. Agentreview: Exploring peer review dynamics with llm agents. *arXiv preprint arXiv:2406.12708*.

Jiaxin Ju, YIZHEN ZHENG, Huan Yee Koh, Can Wang, and Shirui Pan. 2025. Chemthinker: Thinking like a chemist with multi-agent llms for deep molecular insights. *ICLR 2025*.

Mehmet Onur Keskin, Berk Buzcu, Berkecan Koçyiğit, Umut Çakan, Anıl Doğru, and Reyhan Aydoğan. 2024. Negotiator: A comprehensive framework for human-agent negotiation integrating preferences, interaction, and emotion. *IJCAI 2024*.

Arpan Shaileshbhai Korat. 2025. Synergistic minds: A collaborative multi-agent framework for integrated ai tool development using diverse large language models. *World Journal of Advanced Research and Reviews*.

Shrinidhi Kumbhar, Venkatesh Mishra, Kevin Coutinho, Divij Handa, Ashif Iquebal, and Chitta Baral. 2025. Hypothesis generation for materials discovery and design using goal-driven and constraint-guided llm agents. *NAACL 2025*.

Hao Duong Le, Xin Xia, and Chen Zhang. 2025. Multi-agent causal discovery using large language models. *ICLR 2025*.

Weizhen Li, Jianbo Lin, Zhuosong Jiang, Jingyi Cao, Xinpeng Liu, Jiayu Zhang, Zhenqiang Huang, Qianben Chen, Weichen Sun, Qiexiang Wang, and 1 others. 2025. Chain-of-agents: End-to-end agent foundation models via multi-agent distillation and agentic rl. *arXiv preprint arXiv:2508.13167*.

Mayank Mishra, Matt Stallone, Gaoyuan Zhang, Yikang Shen, Aditya Prasad, Adriana Meza Soria, Michele Merler, Parameswaran Selvam, Saptha Surendran, Shivdeep Singh, and 1 others. 2024. Granite code models: A family of open foundation models for code intelligence. *arXiv preprint arXiv:2405.04324*.

Vladimir Naumov, Diana Zagirova, Sha Lin, Yupeng Xie, Wenhao Gou, Anatoly Urban, Nina Tikhonova, Khadija M. Alawi, Mike Durymanov, and Fedor Galkin. 2025. Dora ai scientist: Multi-agent virtual research team for scientific exploration discovery and automated report generation. *bioRxiv*.

Seyednami Niyakan and Xiaoning Qian. 2025. Phenograph: A multi-agent framework for phenotype-driven discovery in spatial transcriptomics data augmented with knowledge graphs. *bioRxiv*.

Aniket Pramanick, Yufang Hou, Saif M. Mohammad, and Iryna Gurevych. 2023. A diachronic analysis of paradigm shifts in nlp research: When, how, and why? *EMNLP 2023*.

Yingming Pu, Tao Lin, and Hongyu Chen. 2025. Piflow: Principle-aware scientific discovery with multi-agent collaboration. *arXiv preprint arXiv:2505.15047*.

Renyi Qu, Ruixuan Tu, and Forrest Bao. 2025. Is semantic chunking worth the computational cost? In *Findings of the Association for Computational Linguistics: NAACL 2025*, pages 2155–2177.

Chandan K Reddy and Parshin Shojaee. 2025. Towards scientific discovery with generative ai: Progress, opportunities, and challenges. *AAAI 2025*.

Aymeric Roucher, Albert Villanova del Moral, Thomas Wolf, Leandro von Werra, and Erik Kaunismäki. 2025. 'smolagents': a smol library to build great agentic systems. `https://github.com/huggingface/smolagents`.

Alireza Salemi, Mihir Parmar, Palash Goyal, Yiwen Song, Jinsung Yoon, Hamed Zamani, Hamid Palangi, and Tomas Pfister. 2025. Llm-based multi-agent blackboard system for information discovery in data science. *arXiv preprint arXiv:2510.01285*.

Er. Jagpreet Singh and Prasant Kumar. 2025. Astrafin:-ai financial agent. *INTERNATIONAL JOURNAL OF SCIENTIFIC RESEARCH IN ENGINEERING AND MANAGEMENT*.

Jackson Spieser, Ali Balapour, Jarek Meller, Krushna Patra, and Behrouz Shamsaei. 2025. Multi-agent ai systems for biological and clinical data analysis. *Preprints.org*.

Qwen Team. 2025. Qwen3 technical report. *Preprint*, arXiv:2505.09388.

Wenhui Wang, Furu Wei, Li Dong, Hangbo Bao, Nan Yang, and Ming Zhou. 2020. Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers. *Advances in neural information processing systems*, 33:5776–5788.

Xinyue Wang, Kun Zhou, Wenyi Wu, Har Simrat Singh, Fang Nan, Songyao Jin, Aryan Philip, Saloni Patnaik, Hou Zhu, Shivam Singh, and 1 others. 2025. Causal-copilot: An autonomous causal analysis agent. *arXiv preprint arXiv:2504.13263*.

Michał Wawer and Jarosław A. Chudziak. 2025. Integrating traditional technical analysis with ai: A multi-agent llm-based approach to stock market forecasting. *International Conference on Agents and Artificial Intelligence*.

Michael Wooldridge. 2002. *An introduction to multiagent systems*. John Wiley & Sons.

Mengxi Xiao, Ben Liu, He Li, Jimin Huang, Qianqian Xie, Xiaofen Zong, Mang Ye, and Min Peng. 2025. Moodangels: A retrieval-augmented multi-agent framework for psychiatry diagnosis. *NIPS 2025*.

Yihang Xiao, Jinyi Liu, Yan Zheng, Xiaohan Xie, Jianye Hao, Mingzhi Li, Ruitao Wang, Fei Ni, Yuxiao Li, Jintian Luo, and 1 others. 2024. Cellagent: An llm-driven multi-agent framework for automated single-cell data analysis. *arXiv preprint arXiv:2407.09811*.

Zonglin Yang, Wanhao Liu, Ben Gao, Tong Xie, Yuqiang Li, Wanli Ouyang, Soujanya Poria, Erik Cambria, and Dongzhan Zhou. 2025. Moose-chem: Large language models for rediscovering unseen chemistry scientific hypotheses. *ICLR 2025*.

Haofei Yu, Zirui Cheng, Zhaochen Hong, Kunlun Zhu, Jinwei Yao, Tao Feng, and Jiaxuan You. 2025a. Research town: Simulator of research community. *ICLR 2025*.

Haofei Yu, Zhaochen Hong, Zirui Cheng, Kunlun Zhu, Keyang Xuan, Jinwei Yao, Tao Feng, and Jiaxuan You. 2025b. Researchtown: Simulator of human research community. *ICML 2025*.

Zhaojian Yu, Kaiyue Feng, Yilun Zhao, Shilin He, Xiao-Ping Zhang, and Arman Cohan. 2025c. Alpharesearch: Accelerating new algorithm discovery with language models. *arXiv preprint arXiv:2511.08522*.

Yurun Yuan and Tengyang Xie. 2025. Reinforce llm reasoning through multi-agent reflection. *arXiv preprint arXiv:2506.08379*.

Bohui Zhang, Yuan He, Lydia Pintscher, Albert Meroño Peñuela, and Elena Simperl. 2025a. Schema generation for large knowledge graphs using large language models. *arXiv preprint arXiv:2506.04512*.

Yanzhao Zhang, Mingxin Li, Dingkun Long, Xin Zhang, Huan Lin, Baosong Yang, Pengjun Xie, An Yang, Dayiheng Liu, Junyang Lin, Fei Huang, and Jingren Zhou. 2025b. Qwen3 embedding: Advancing text embedding and reranking through foundation models. *arXiv preprint arXiv:2506.05176*.

Qihao Zhu, Daya Guo, Zhihong Shao, Dejian Yang, Peiyi Wang, Runxin Xu, Y Wu, Yukun Li, Huazuo Gao, Shirong Ma, and 1 others. 2024a. Deepseek-coder-v2: Breaking the barrier of closed-source models in code intelligence. *arXiv preprint arXiv:2406.11931*.

Yuqi Zhu, Xiaohan Wang, Jing Chen, Shuofei Qiao, Yixin Ou, Yunzhi Yao, Shumin Deng, Huajun Chen, and Ningyu Zhang. 2024b. Llms for knowledge graph construction and reasoning: Recent capabilities and future opportunities. *World Wide Web*, 27(5):58.

A Paper Review Results

We evaluate how well large language models can predict human paper-review scores on ICLR submissions. From the ICLR 2024 dataset, we randomly sampled 50 papers to cover a broad range of human-assigned ratings and evaluated four tool-enabled LLMs: `gpt-oss:120b`, `gpt-oss:20b`, `qwen3-coder-30b`, and a quantized `qwen3-coder-30b` variant. For each paper, the model produces numerical scores for standard review dimensions (overall rating, soundness, presentation, and contribution), which we compare against the corresponding human scores.

Metrics. We report regression error (MSE, MAE, RMSE), rank/linear association (Pearson, Spearman), and thresholded accuracy (percentage of predictions within ± 0.5, ± 1.0, and ± 1.5 of the human score). We also report the mean and standard deviation of signed errors to characterize systematic bias. Due to occasional missing fields or filtering during preprocessing, the number of evaluated papers N can differ slightly across models.

Key findings. Across categories, `gpt-oss:120b` achieves the best overall accuracy on *rating* and *contribution* (e.g., rating MAE $= 1.68$; contribution MAE $= 0.62$), while `gpt-oss:20b` is competitive and often stronger on more technical subscores such as *soundness* and *presentation*. Despite moderate absolute errors on several dimensions, correlations with human scores remain weak across models (generally $|r| < 0.25$), suggesting that models struggle to preserve the relative ranking of papers even when their average deviation is limited. Code-specialized models (Qwen3-Coder) remain viable baselines, but show larger errors on overall rating and contribution in this setting.

B System Overview

Paper Circle is a full-stack platform with a web frontend and a Python backend as shown in the Figure 7. The frontend (React, TypeScript, Vite, TailwindCSS) provides discovery, reading circles, and discussion features. The backend exposes discovery APIs via FastAPI and implements the multi-agent pipelines used by the system. Supabase (PostgreSQL + Auth) provides storage for users, communities, papers, and sessions.

The discovery backend includes two major pipelines: (i) a refactored research discovery pipeline focused on deterministic retrieval, scoring, and diversity, and (ii) a multi-agent research pipeline that produces structured step-by-step outputs with offline search support. Both pipelines are accessible through API endpoints and are integrated into the Paper Circle user interface for interactive discovery workflows.

B.1 System Architecture

Figure 8 illustrates the overall architecture of Paper Circle. The system consists of two complementary multi-agent pipelines: the *Discovery Pipeline* for finding relevant papers, and the *Analysis Pipeline* for deep understanding of individual papers.

The discovery pipeline, as shown in the Figure 8 is composed of six agents: intent classification, paper search, sorting, analysis, export, and web search. The intent classifier parses natural-language queries into structured constraints (search mode, conferences, year range, max results, and ranking preferences). The paper search agent is the primary retrieval worker; it updates the global state and writes outputs after every search step. The sorting and analysis agents operate on the shared paper list to refine ranking and derive insights. The export agent centralizes output access for downstream workflows, while the web search agent supplements the pipeline with external lookup tools when required. All agents are coordinated by the

Model	Category	MSE	MAE	RMSE	Pearson	Spearman	Acc. ± 0.5	Acc. ± 1.0	Acc. ± 1.5	Mean Err.	Std Err.	N
oss-120B	RATING	4.6934	1.6844	2.1664	-0.0407	0.0571	25.00%	43.75%	58.33%	0.2177	2.1555	48
oss-120B	SOUNDNESS	0.7316	0.6351	0.8554	-0.0054	0.0474	58.33%	85.42%	87.50%	-0.0816	0.8515	48
oss-120B	PRESENTATION	0.6564	0.6038	0.8102	0.0701	0.1259	60.42%	83.33%	91.67%	-0.0920	0.8049	48
oss-120B	CONTRIBUTION	0.6349	0.6240	0.7968	0.0717	0.0734	56.25%	85.42%	91.67%	0.0087	0.7967	48
oss-20	RATING	4.7607	1.7647	2.1819	0.0989	0.1869	21.43%	40.48%	52.38%	1.5980	1.4856	42
oss-20	SOUNDNESS	0.4241	0.5190	0.6512	-0.0106	-0.0226	59.52%	92.86%	97.62%	0.3294	0.5618	42
oss-20	PRESENTATION	0.4271	0.5171	0.6535	-0.1270	-0.1299	64.29%	90.48%	97.62%	0.3512	0.5511	42
oss-20	CONTRIBUTION	0.6482	0.6702	0.8051	0.2221	0.1757	50.00%	83.33%	97.62%	0.6250	0.5075	42
qwen30B-code_qk_3	RATING	11.8533	2.9879	3.4429	-0.2233	-0.2837	8.51%	17.02%	29.79%	2.9085	1.8422	47
qwen30B-code_qk_3	SOUNDNESS	1.6941	1.1730	1.3016	0.0113	-0.0096	17.02%	46.81%	72.34%	1.1454	0.6182	47
qwen30B-code_qk_3	PRESENTATION	1.4257	1.0191	1.1940	0.0378	0.0271	27.66%	59.57%	78.72%	0.9787	0.6840	47
qwen30B-code_qk_3	CONTRIBUTION	2.2921	1.3865	1.5140	0.0196	0.0224	12.77%	34.04%	65.96%	1.3865	0.6080	47
Qwen 30B	RATING	10.2331	2.7930	3.1989	-0.1820	-0.2216	7.89%	13.16%	26.32%	2.6930	1.7266	38
Qwen 30B	SOUNDNESS	1.7172	1.2096	1.3104	-0.1157	-0.1057	13.16%	39.47%	73.68%	1.1491	0.6298	38
Qwen 30B	PRESENTATION	0.9526	0.7180	0.9760	-0.1319	-0.1495	55.26%	73.68%	81.58%	0.6522	0.7261	38
Qwen 30B	CONTRIBUTION	2.5212	1.4746	1.5878	-0.2119	-0.2160	13.16%	26.32%	55.26%	1.4640	0.6146	38

Table 5: **Paper review score prediction on ICLR 2024.** We compare four LLMs on predicting human review scores across rating, soundness, presentation, and contribution. We report error metrics (MSE/MAE/RMSE), correlation (Pearson/Spearman), and thresholded accuracy (within ± 0.5, ± 1.0, ± 1.5 of the human score). N denotes the number of papers evaluated for each model after preprocessing.

`CodeAgent`, which enforces a minimal-step policy for efficiency and uses the intent classifier to decide offline versus online search.

The analysis pipeline operates on individual papers, transforming PDF documents into structured knowledge graphs. It employs four specialized extraction agents (concept, method, experiment, and linkage) that process paper content in phases, building a typed graph with full traceability to source locations. The resulting graph supports question answering, coverage verification, and multi-format export.

B.2 State Management and Outputs

State is maintained in `PipelineState`. Each step increments a counter, logs action metadata, and regenerates synchronized artifacts. The outputs include: (i) `papers.json` with full paper metadata and computed scores, (ii) `links.json` with structured links and PDFs/DOIs, (iii) `stats.json` with aggregate statistics and a leaderboard, (iv) `summary.json` with generated insights and key findings, (v) `retrieval_metrics.json` when evaluation is enabled, and (vi) human-readable exports (CSV, BibTeX, Markdown) plus a live HTML dashboard. This approach ensures that each agent step is reproducible and auditable.

B.3 Retrieval

The pipeline supports both offline and online retrieval. Offline search loads papers from a local JSON corpus and optionally filters by conference and year. It ranks results using BM25 by default, with optional semantic similarity (sentence transformers) or hybrid scoring when available. An optional cross-encoder reranker can refine the top results; when enabled, it reranks a first-stage candidate set. Online search aggregates results from arXiv, Semantic Scholar, OpenAlex, and DBLP via their public APIs. A query intent classifier detects search mode, conference constraints, year ranges, and ranking preferences, and routes the query to the appropriate retrieval pathway. Deduplication is applied across sources by normalizing titles.

B.4 Ranking and Scoring

After retrieval, papers are scored along multiple axes: recency, similarity to the query (TF–IDF (Das et al., 2023) when available), novelty based on title token frequency, and normalized BM25 scores (Chen and Wiseman, 2023). The

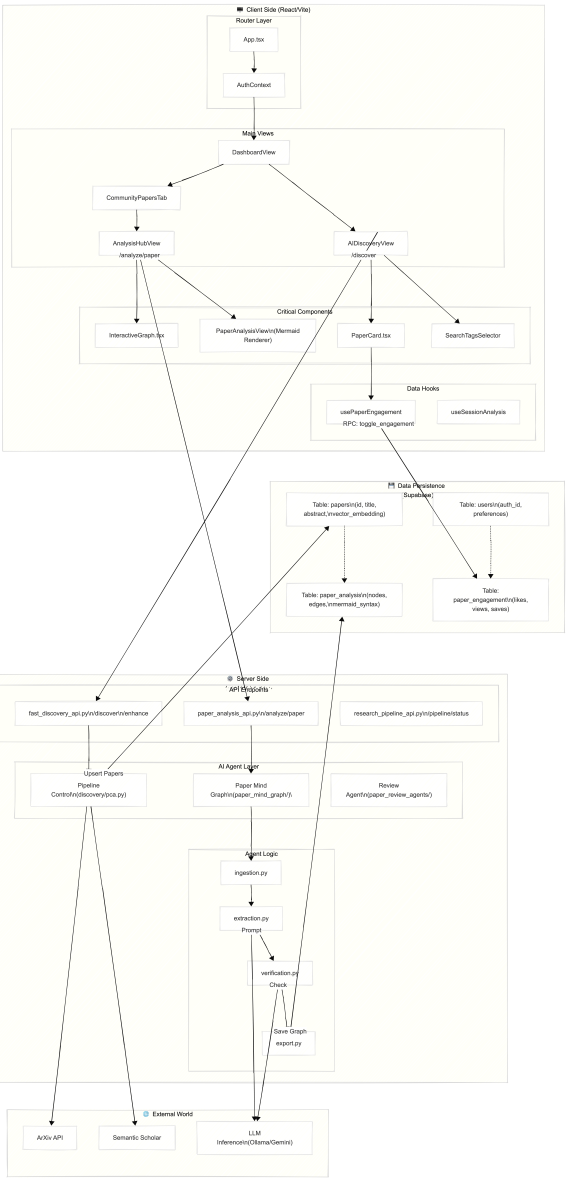

Figure 7: Frontend of the website from client side.

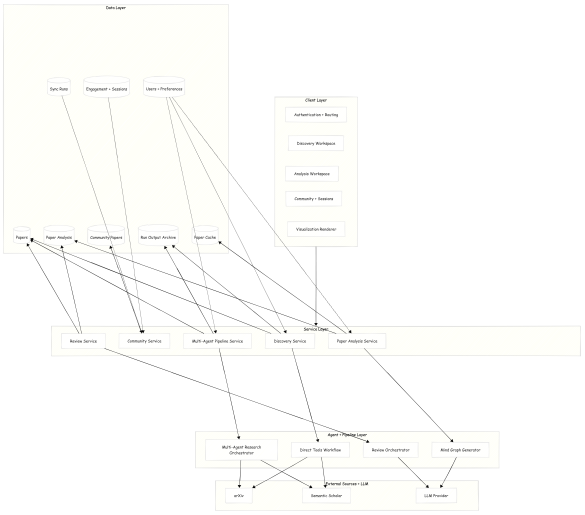

Figure 8: Discovery pipeline front-end view of the website, which takes an input query and iteratively refines the state and updates the database.

system supports sorting by any single criterion or by a weighted combined score. Relevance scores are computed as a weighted mixture of similarity, recency, citation count, and BM25. Final ranks are assigned after sorting, and the updated ordering is reflected in all exported artifacts. When reranker-based sorting is requested, a cross-encoder replaces the default scoring with direct relevance scores.

B.5 Analysis and Monitoring

The pipeline computes aggregate statistics such as source distribution, year distribution, top authors and venues, keyword frequency, and citation summaries. These analytics populate structured summaries and are visualized in an auto-refreshing HTML dashboard. Each agent action is logged with timestamps and paper counts, enabling reproducibility and step-level auditing of the pipeline. The pipeline also maintains a step log that captures the agent name, action, results preview, and parameters used.

C Retrieval Pipeline

Paper Circle supports both offline and online retrieval to balance coverage, speed, and reproducibility. The choice between retrieval modes is controlled by the intent classification agent, which parses user queries to determine the optimal search strategy.

C.1 Offline Retrieval

The `OfflinePaperSearchEngine` enables fast (See the Figure 9, reproducible search over a local

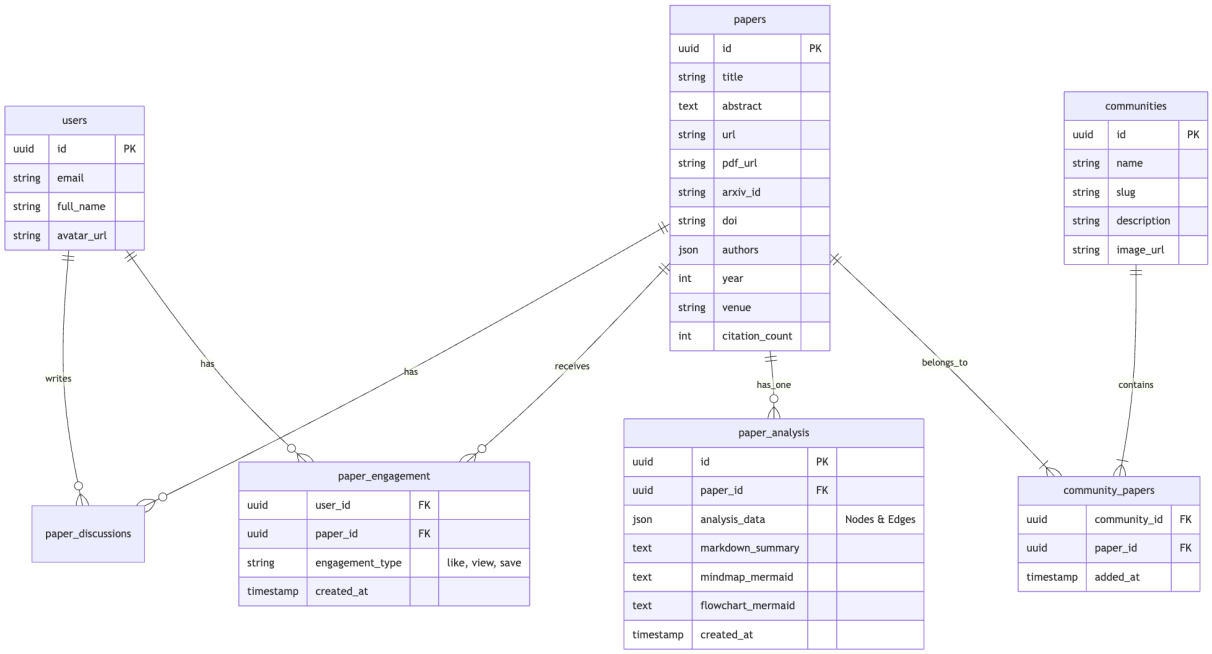

Figure 9: Paper analysis and database management for fast inference.

database of academic papers stored as JSON files. Each database file contains structured paper metadata including title, authors, abstract, venue, year, track, keywords, and DOI.

The offline search process:

1. **Database Loading**: Papers are loaded from the specified database path with optional filtering by conference (e.g., ICLR, NeurIPS, ACL) and year range.

2. **Text Preparation**: For each paper, searchable text is constructed by concatenating the title, abstract, and keywords.

3. **BM25 Indexing**: When available, papers are indexed using the Okapi BM25 algorithm via the `rank_bm25` library. The index uses tokenized documents for sparse retrieval.

4. **Query Execution**: User queries are tokenized and scored against the BM25 index, returning a ranked list of candidates.

An optional cross-encoder reranker can refine the top-k results from the first-stage retrieval. When enabled via the `AdvancedReranker` module, the system uses a transformer-based reranker (e.g., Qwen3-Reranker) to compute more precise relevance scores between the query and candidate documents.

C.2 Online Retrieval

For broader or more current searches, Paper Circle aggregates results from multiple academic APIs:

- **arXiv**: Queries the arXiv API for preprints, extracting title, authors, abstract, categories, and PDF links.
- **Semantic Scholar**: Retrieves papers with citation counts, abstracts, and venue information via the Semantic Scholar Academic Graph API.
- **OpenAlex**: Accesses the OpenAlex catalog for open-access metadata and citation networks.
- **DBLP**: Searches the DBLP computer science bibliography for venue-specific results.

Each source is queried in parallel using a thread pool executor for efficiency. Results are normalized into the common `Paper` data structure before merging.

C.3 Deduplication

After retrieval, the pipeline performs two-stage deduplication to eliminate redundant entries:

1. **DOI-based deduplication**: Papers with matching DOIs are deduplicated, preferring entries with richer metadata (e.g., abstracts, PDF URLs).

2. **Title-based deduplication**: Titles are normalized by removing punctuation and converting to lowercase. Duplicate titles are merged, again preferring metadata-complete entries.

The deduplication step is critical when aggregating results from multiple sources, as the same paper often appears in arXiv, Semantic Scholar, and OpenAlex with varying metadata quality.

C.4 Query Expansion

The query generation agent converts natural-language user input into a structured search specification containing:

- **Core keywords**: Primary search terms extracted from the query.
- **Required constraints**: Mandatory terms that must appear in results.
- **Related terms**: Synonyms or related concepts to expand recall.
- **Negative keywords**: Terms to exclude from results.
- **Plausible paper titles**: Hypothesized titles for targeted retrieval.

This structured specification enables consistent query construction across heterogeneous data sources while capturing user intent more precisely than raw keyword matching.

D Scoring and Ranking

Paper Circle employs a multi-criteria scoring framework designed for research discovery rather than general information retrieval. Each paper receives scores along multiple dimensions, which are combined using mode-specific weights to produce a final ranking.

D.1 Scoring Dimensions

The system computes the following scores for each retrieved paper:

Similarity Score Relevance to the user query is computed using TF–IDF (Das et al., 2023) vectorization and cosine similarity. The query and paper text (concatenated title and abstract) are transformed into TF–IDF vectors using scikit-learn's `TfidfVectorizer`. The similarity score is the cosine of the angle between these vectors:

$$\text{similarity}(q, p) = \frac{\vec{v}_q \cdot \vec{v}_p}{\|\vec{v}_q\| \cdot \|\vec{v}_p\|} \tag{1}$$

where $\vec{v}_q$ and $\vec{v}_p$ are the TF–IDF vectors for the query and paper, respectively.

Recency Score Papers are scored by publication year, with more recent papers receiving higher scores. The recency score is normalized relative to the current year:

$$\text{recency}(p) = \frac{\text{year}(p) - \text{year}_{\min}}{\text{year}_{\max} - \text{year}_{\min}} \tag{2}$$

where $\text{year}_{\min}$ and $\text{year}_{\max}$ are the minimum and maximum years in the corpus.

Novelty Score Novelty measures how different a paper is from the corpus centroid, computed as the TF–IDF distance from the average document vector. Papers with unusual terminology or unique topic combinations receive higher novelty scores, surfacing potentially overlooked works.

BM25 Score When the `rank_bm25` library is available, the Okapi BM25 algorithm provides an alternative relevance measure that accounts for term frequency saturation and document length normalization. BM25 scores are normalized to the $[0, 1]$ range for comparability with other dimensions.

Citation Count When available from the source API (primarily Semantic Scholar and OpenAlex), citation counts provide a proxy for impact. Citation-based ranking is optional and disabled by default to avoid recency bias against new papers.

D.2 Combined Score Computation

The final combined score is a weighted sum of individual dimensions:

$$\text{combined}(p) = w_s \cdot \text{similarity} + w_r \cdot \text{recency} + w_n \cdot \text{novelty} + w_b \cdot \text{bm25} \tag{3}$$

The weights (w_s, w_r, w_n, w_b) are determined by the search mode:

- **Stable mode**: Prioritizes relevance and authority. Weights: $w_s = 0.5$, $w_r = 0.2$, $w_n = 0.1$, $w_b = 0.2$.
- **Discovery mode**: Prioritizes novelty to surface non-obvious results. Weights: $w_s = 0.3$, $w_r = 0.1$, $w_n = 0.4$, $w_b = 0.2$.

- **Balanced mode**: Equal emphasis across dimensions. Weights: $w_s = 0.3$, $w_r = 0.2$, $w_n = 0.2$, $w_b = 0.3$.

Users can override these weights at query time via API parameters, enabling custom relevance trade-offs for specific research contexts.

D.3 Sorting Stage

After scoring, the sorting agent reorders papers according to user preferences. Supported sort criteria include:

- `recency`: Most recent papers first.
- `citations`: Highest-cited papers first.
- `similarity`: Most relevant papers first.
- `novelty`: Most unusual papers first.
- `bm25`: Best BM25 matches first.
- `combined`: Weighted combined score (default).

D.4 Cross-Encoder Reranking

For high-precision use cases, the pipeline supports optional cross-encoder reranking. When enabled, a transformer-based reranker (configured via `RerankerConfig`) processes query-document pairs through a cross-attention model to compute more accurate relevance scores than first-stage retrieval alone. The `MultiStageRetriever` first retrieves a larger candidate set (e.g., top-200) using BM25, then reranks to produce the final top-k results. This two-stage approach balances efficiency with ranking quality.

E Diversity and Postprocessing

Relevance-based ranking alone can produce homogeneous results, with multiple papers covering similar topics or methods. Paper Circle addresses this through diversity-aware postprocessing that ensures the top results span a broader range of perspectives.

E.1 Maximal Marginal Relevance

To improve topical coverage, Paper Circle applies Maximal Marginal Relevance (MMR) to the candidate list after initial scoring. MMR iteratively selects papers that maximize a combination of relevance to the query and dissimilarity to already-selected papers:

$$\text{MMR} = \arg\max_{p \in R \setminus S} \left[\lambda \cdot \text{sim}(p, q) - (1 - \lambda) \cdot \max_{s \in S} \text{sim}(p, s) \right] \tag{4}$$

where R is the candidate set, S is the set of already-selected papers, q is the query, and λ controls the relevance–diversity trade-off.

The diversity parameter λ is mode-dependent:

- **Stable mode**: $\lambda = 0.8$ (relevance-focused).
- **Discovery mode**: $\lambda = 0.5$ (diversity-focused).
- **Balanced mode**: $\lambda = 0.65$.

Similarity between papers is computed using TF–IDF cosine similarity over concatenated title and abstract text. This ensures that top results cover distinct subtopics rather than repeating variations of the same idea.

E.2 Secondary Views

The pipeline constructs specialized views over the ranked list to serve different discovery goals:

Hidden Gems Papers with high novelty scores but moderate relevance scores are surfaced as "hidden gems." These are papers that may not rank highly on traditional relevance metrics but offer unique perspectives or cover underexplored topics. The hidden gems view is computed by sorting papers by novelty score and filtering for those below rank 20 in the combined ranking.

Canonical Papers Papers with high citation counts or appearing in top-tier venues are flagged as "canonical" works. This view helps users identify foundational papers in a research area, complementing the recency-focused main ranking.

Source Distribution The postprocessing stage also reports the distribution of papers across sources (arXiv, Semantic Scholar, etc.), enabling users to assess coverage and identify potential gaps in the retrieval.

E.3 Statistics and Analytics

After ranking, the analysis agent computes aggregate statistics stored in `stats.json`:

- **Year distribution**: Paper counts by publication year.
- **Source distribution**: Paper counts by retrieval source.

- **Top authors**: Authors appearing most frequently in results.
- **Top venues**: Conferences and journals with highest representation.
- **Keyword frequency**: Most common terms in paper titles.
- **Citation statistics**: Total, average, median, min, and max citation counts.
- **Score statistics**: Average similarity, novelty, recency, and BM25 scores.

These analytics are visualized in an auto-refreshing HTML dashboard that updates every 10 seconds during pipeline execution, providing real-time visibility into the discovery process.

E.4 Insight Generation

The pipeline automatically generates human-readable insights from the collected data:

- **Publication trends**: Identifies the year with the most publications.
- **Primary source**: Reports which API contributed the most results.
- **Prolific authors**: Highlights researchers with multiple papers in the collection.
- **Citation leaders**: Identifies the most-cited paper.
- **Hot topics**: Lists the most frequent keywords.
- **Open access availability**: Reports the percentage of papers with direct PDF links.

These insights are stored in `summary.json` and displayed on the dashboard, helping users quickly understand the landscape of retrieved literature.

F Outputs and Interfaces

The pipeline maintains synchronized structured outputs after every agent step. The primary artifacts include:

- `papers.json`: Full paper metadata and scores.
- `links.json`: Structured links and PDF/DOI entries.
- `stats.json`: Aggregate statistics and leaderboards.
- `summary.json`: Insights and key findings.
- `retrieval_metrics.json`: Step-level evaluation metrics.

Additional exports include CSV, BibTeX, Markdown, and an auto-refreshing HTML dashboard. These outputs allow the same discovery session to be used for curation, citation management, and reporting.

The system exposes REST APIs via FastAPI. The discovery endpoint accepts a query and mode, returns structured search specifications, and provides the full ranked list with scores. Mode weights can be queried or overridden at runtime, enabling customized relevance/authority/novelty trade-offs.

G Evaluation

We evaluate Paper Circle along three axes: (i) retrieval effectiveness under different configurations, (ii) stability and reproducibility of rankings across steps, and (iii) the utility of diversity-aware postprocessing for surfacing non-redundant results. Paper Circle provides built-in evaluation metrics but does not enforce a fixed benchmark dataset. When a ground-truth paper title or identifier is provided, the system computes Mean Reciprocal Rank (MRR), Recall@K, Precision@K, and hit rates. These metrics are computed per step and stored in JSON file for longitudinal tracking.

As a minimal illustrative scenario, consider a known target paper in the local corpus: the pipeline is run once using offline retrieval and once using online sources. The resulting MRR and Recall@K values allow direct comparison of configuration impact, while repeated runs confirm stable rankings when deterministic scoring is enabled. Although lightweight, this framing aligns evaluation with discovery goals rather than task-specific QA benchmarks.

For batch evaluation, a parallel benchmarking utility executes multiple queries concurrently and aggregates mean metrics and timing statistics. This supports lightweight comparisons between search configurations (offline vs. online, BM25 vs. semantic, with or without reranking) without requiring external tooling.

Knowledge Graph Schema. The mind graph follows a typed schema with nodes for papers, sections, concepts, methods, experiments, datasets,

and visual elements (figures, tables, equations), and edges encoding structural and semantic relations such as hierarchy, definition, proposal, usage, evaluation, illustration, and dependency. Each node and edge is annotated with provenance metadata, including source chunk IDs, page numbers, verification status, confidence scores, and timestamps, providing full traceability from any graph element back to the original PDF.

G.1 Multi-Agent Extraction

The `GraphBuilder` orchestrates four specialized extraction agents, each implemented as a `CodeAgent` with domain-specific instructions:

Concept Extractor Identifies key concepts from text chunks, classifying each by type (definition, technique, theory, phenomenon) and importance (core, supporting, background). The agent outputs structured JSON with concept names, descriptions, and classifications.

Method Extractor Focuses on sections containing method-related keywords ("method", "approach", "architecture", "algorithm"). For each method, it extracts the name, description, category (proposed, baseline, component), and key steps.

Experiment Extractor Processes experiment sections to extract experimental setups, datasets used, evaluation metrics, and key results. It also identifies dataset nodes for cross-referencing.

Linkage Agent Connects figures and tables to the concepts and methods they illustrate. Given a figure caption, nearby text, and a list of existing concepts, the agent determines which concepts the figure relates to and the type of relationship (illustrates, summarizes, compares, demonstrates).

The extraction proceeds in five phases: (1) concept extraction from body chunks, (2) method extraction from method sections, (3) experiment and dataset extraction, (4) figure and table linkage, and (5) inter-concept relationship discovery. Each phase updates the shared `MindGraph` data structure.

G.2 Graph-Aware Q&A

The Q&A system combines vector-based retrieval with graph traversal. The `EmbeddingStore` indexes both text chunks and node descriptions using sentence-transformers (with a simple bag-of-words fallback when unavailable). Given a question, the `GraphRetriever`:

1. Retrieves the top-k most similar chunks and nodes.
2. Expands context by including 1-hop graph neighbors.
3. Returns chunks, nodes, and connecting edges.

The `PaperQA` agent constructs a prompt with the retrieved context, including text chunks with their section sources, relevant concept descriptions, and graph relationships. The response includes the answer, supporting sections, relevant figures and tables, and a confidence estimate.

A `locate` function allows users to find where specific items are discussed in the paper by searching across nodes, figures, tables, and text chunks, returning page numbers and context snippets.

G.3 Coverage Verification

To ensure nothing is silently dropped during extraction, the `CoverageChecker` produces a detailed coverage report:

- **Figure coverage**: How many figures are linked to concepts or methods.
- **Table coverage**: How many tables are linked to results or experiments.
- **Section coverage**: How many sections have extracted concepts.
- **Equation coverage**: How many equations are linked to concepts they define.

The report includes an overall coverage score (0–100%), lists of unlinked items with suggestions, and critical issues (e.g., "No figures are linked to concepts/methods"). This enables quality assurance before downstream use.

G.4 Human Verification Workflow

The `VerificationManager` supports human-in-the-loop review:

- `verify_node`: Mark a node as human-verified.
- `edit_node`: Modify node title or description.
- `add_edge`: Create new relationships.
- `remove_edge`: Delete incorrect relationships.
- `flag_for_review`: Flag nodes for review with a reason.

Each action is logged with timestamps, maintaining a complete edit history. Nodes carry a `verification_status` field (auto-generated, human-verified, human-edited, or flagged) that propagates through exports.

G.5 Export Formats

The system exports to multiple formats for different use cases:

- **JSON**: Full graph data including nodes, edges, chunks, and metadata.
- **Markdown**: Structured reading notes with section outlines.
- **Mermaid**: Mind maps and flowcharts for visualization.
- **HTML**: Interactive D3.js-based graph visualization.

All exports preserve traceability metadata, enabling users to navigate from any extracted element back to the original source.

H Implementation and Deployment

The backend is implemented in Python with FastAPI for service endpoints and relies on standard scientific libraries for retrieval and scoring (scikit-learn, NumPy, pandas). The multi-agent pipeline is defined in textttbackend/agents/discovery/pca.py, while the refactored deterministic pipeline is implemented in textttbackend/core/paperfinder.py. Both pipelines expose functionality through API servers, including a fast discovery variant designed for low-latency responses.

The frontend is built with React and TypeScript and integrates discovery results through the API. Supabase provides authentication and persistent data storage for user profiles, communities, sessions, and paper metadata. Containerization support is provided via a Dockerfile, and deployment configurations are included for common platforms (Railway, Render, and Vercel). Environment variables control API URLs and database credentials, enabling local development or hosted deployment without code changes.