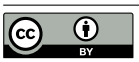
AgenticScholar: Agentic Data Management with Pipeline Orchestration for Scholarly Corpora

HAI LAN, School of Electrical Engineering and Computer Science, The University of Queensland, Australia
TINGTING WANG, School of Electrical Engineering and Computer Science, The University of Queensland, Australia
ZHIFENG BAO*, School of Electrical Engineering and Computer Science, The University of Queensland, Australia
GUOLIANG LI, Department of Computer Science and Technology, Tsinghua University, China
DAOMIN JI, School of Electrical Engineering and Computer Science, The University of Queensland, Australia
GE LEE, School of Electrical Engineering and Computer Science, The University of Queensland, Australia
FENG LUO, School of Electrical Engineering and Computer Science, The University of Queensland, Australia
ZI HUANG, School of Electrical Engineering and Computer Science, The University of Queensland, Australia
HAILANG QIU, School of Computer Science, Wuhan University, China
GANG HUA, School of Computer Science and Artificial Intelligence, Wuhan University of Technology, China

Managing the rapidly growing scholarly corpus poses significant challenges in representation, reasoning, and efficient analysis. An ideal system should unify structured knowledge management, agentic planning, and interpretable execution to support diverse scholarly queries – from retrieval to knowledge discovery and generation – at scale. Unfortunately, existing RAG and document analytics systems fail to achieve all query types simultaneously. To this end, we propose AgenticScholar, an agentic scholarly data management system that integrates a structure-aware knowledge representation layer, an LLM-centric hybrid query planning layer, and a unified execution layer with composable operators. AgenticScholar autonomously translates natural language queries into executable DAG plans, enabling end-to-end reasoning over multi-modal scholarly data. Extensive experiments demonstrate that AgenticScholar significantly outperforms existing systems in effectiveness, efficiency, and interpretability, offering a practical foundation for future research on agentic scholarly data management.

*Zhifeng Bao is the corresponding author.

Authors' Contact Information: Hai Lan, h.lan@uq.edu.au, School of Electrical Engineering and Computer Science, The University of Queensland, Brisbane, Queensland, Australia; Tingting Wang, tingting.wang@uq.edu.au, School of Electrical Engineering and Computer Science, The University of Queensland, Brisbane, Queensland, Australia; Zhifeng Bao, zhifeng.bao@uq.edu.au, School of Electrical Engineering and Computer Science, The University of Queensland, Brisbane, Queensland, Australia; Guoliang Li, liguoliang@tsinghua.edu.cn, Department of Computer Science and Technology, Tsinghua University, Beijing, China; Daomin Ji, d.ji@uq.edu.au, School of Electrical Engineering and Computer Science, The University of Queensland, Brisbane, Queensland, Australia; Ge Lee, ge.lee@uq.edu.au, School of Electrical Engineering and Computer Science, The University of Queensland, Brisbane, Queensland, Australia; Feng Luo, f.luo@uq.edu.au, School of Electrical Engineering and Computer Science, The University of Queensland, Brisbane, Queensland, Australia; Zi Huang, huang@itee.uq.edu.au, School of Electrical Engineering and Computer Science, The University of Queensland, Brisbane, Queensland, Australia; Hailang Qiu, helloqiu@whu.edu.cn, School of Computer Science, Wuhan University, Wuhan, Hubei, China; Gang Hua, huagang01@whut.edu.cn, School of Computer Science and Artificial Intelligence, Wuhan University of Technology, Wuhan, Hubei, China.



ACM 2836-6573/2026/6-ART131
https://doi.org/10.1145/3802008

CCS Concepts: • **Information systems → Data management systems**; • **Computing methodologies → Planning and scheduling**.

Additional Key Words and Phrases: Agentic Data Management, LLM-driven Query Planning

ACM Reference Format:
Hai Lan, Tingting Wang, Zhifeng Bao, Guoliang Li, Daomin Ji, Ge Lee, Feng Luo, Zi Huang, Hailang Qiu, and Gang Hua. 2026. AgenticScholar: Agentic Data Management with Pipeline Orchestration for Scholarly Corpora. *Proc. ACM Manag. Data* 4, 3 (SIGMOD), Article 131 (June 2026), 28 pages. https://doi.org/10.1145/3802008

1 Introduction

The exponential growth of scholarly corpora represents one of today's most complex data management challenges [34]. This vast corpus is far more than a collection of text files; it spans ***multiple modalities***, including but not limited to tables, figures, code snippets, and bibliographic metadata, forming an intricately interconnected and semi-structured ecosystem [44]. Managing and reasoning over this ecosystem is key to democratizing knowledge and promoting evidence-based research, yet it pushes the boundaries of traditional data management systems.

An Example Illustrating Scholarly Query Characteristics

In Figure 1, a researcher begins with an open question – "How has research on vector search been evolving, and what are promising yet unexplored problems since 2021?" – and the system must progressively clarify intents and compose a series of analytical steps to generate an evidence-grounded answer. We observe that typical scholarly queries often:
1. Start open-ended and evolve diverse intents. The researcher may begin with a general trend analysis on "vector search" and then diverge into different paths: stopping at the trend (**Intent 1**); expanding to performance comparison and research idea exploration across the full literature (**Intent 2**); or focusing on milestone papers for a targeted synthesis (**Intent 3**).
2. Require multi-step semantic reasoning. Addressing Q4 involves multiple analytical steps: first `extract` experimental results and limitations from individual papers, then `rank` the results and `summarize` shared limitations across papers, and finally `generate` a coherent response.
3. Depend on multi-modal evidence and structured synthesis. Resolving Q4 entails extracting quantitative results from tables (e.g., Table 5 of Starling) and figures (e.g., Figures 6 and 8 of SPANN, Figure 5 of Starling), aligning them across papers into comparative matrices, and grounding the synthesis in this integrated, multi-modal evidence.
4. Require context-aware knowledge generation. Addressing Q5 requires generating novel insights that extend beyond summarization. The system must interpret the problem context, analyze the methodological landscape, and infer which problem–method combinations remain underexplored to produce an informed and contextually grounded conclusion.

These query characteristics present a fundamental challenge for existing systems [42, 55, 66, 70], which will be discussed shortly in Section 2.2. While they may address isolated queries in Figure 1, they cannot autonomously orchestrate a complete analytical pipeline – from trend analysis (Q1) through multiple reasoning steps – to the ultimate stage of research idea exploration (Q5). This gap between capability and necessity motivates our introduction of AgenticScholar, **an agentic scholarly data management system** that autonomously interprets user intents, plans complex analytical pipelines, and executes them end-to-end within the rich semantic context of a scholarly ecosystem. We distill four key challenges for an agentic scholarly data management system:

Challenge 1: Semantic Representation and Contextualization of Scholarly Data. Scholarly documents contain multiple modalities that collectively convey a paper's semantics. The challenge is

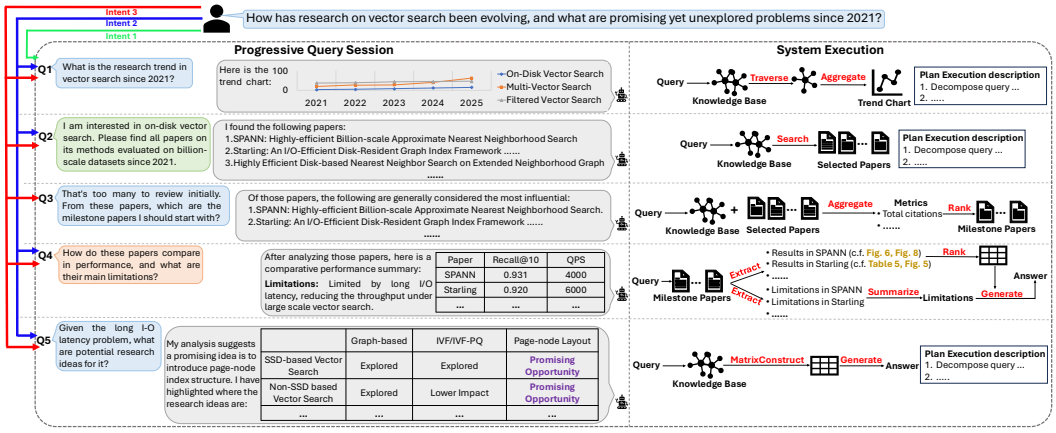


Fig. 1. The workflow of an agentic scholarly data management system, illustrating how it responds to diverse query intents.

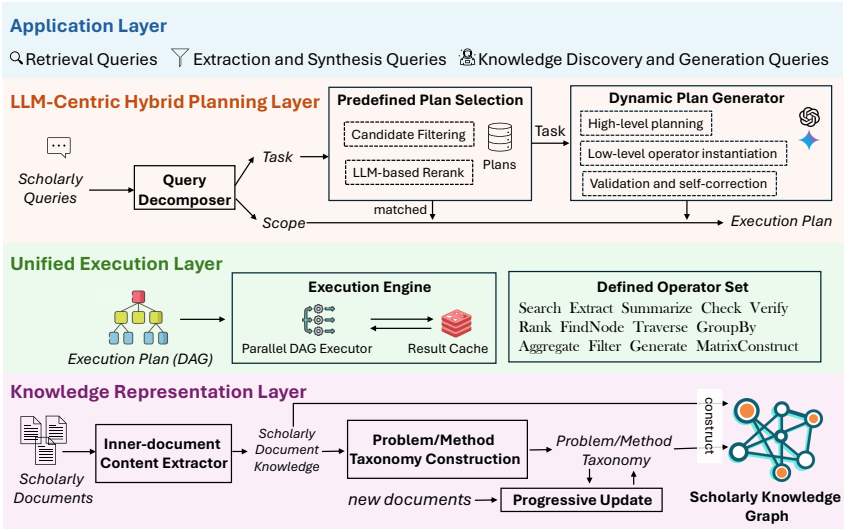


Fig. 2. System architecture of AgenticScholar.

not recognizing this diversity, but representing and indexing it in a way that preserves *both internal structure and semantic connections across components within a paper and those across related papers.* Queries like Q5 in Figure 1 demand reasoning at the topic level, integrating contextual evidence to perform comparative or synthetic analysis. Supporting them requires representations that maintain *contextual organization* beyond individual papers. Efficiently constructing and maintaining such structures, while enabling fine-grained retrieval and reasoning, remains an open data management problem. Existing systems [70, 75] typically flatten documents into unstructured text or dense embeddings, discarding structure and context essential for complex scholarly analysis.

Challenge 2: Planning Complex Analytical Pipeline. As shown in Figure 1, scholarly queries exhibit *diverse intents* that go far beyond the rigid patterns of SQL or keyword search. A central challenge is to translate high-level natural language into an accurate and efficient execution plan. The system must infer user intent, decompose it into *a coherent sequence of analytical steps*, and determine an efficient execution strategy amid the uncertainty of operating over multi-modal data. Traditional query optimizers – built for fixed relational schemas – lack the flexibility needed for such dynamic and context-aware semantic reasoning.

Challenge 3: Generality and Extensibility of the Execution Framework. The space of scholarly queries is vast and continually evolving. As shown in Figure 1, a researcher's exploration often begins with an open question and progresses through diverse queries. A system built on hard-coded, query-specific pipelines would be brittle and incapable of adapting to new analytical needs. The challenge, therefore, is to design a framework that *balances generality with extensibility*: (1) its core set of analytical steps must be sufficiently expressive and composable to support a broad spectrum of queries without requiring custom implementations for each new one; and (2) its architecture must allow new analytical steps and execution plans to be integrated seamlessly, enabling support for emerging query types without necessitating a full system redesign.

Challenge 4: Interpretability and Traceability. For research-oriented data management, correctness alone is not enough – researchers need to understand and trust how an answer was produced. The challenge lies in exposing interpretable reasoning chains that make internal decisions explicit, including applied analytical steps, intermediate results, and evidence propagation. This requires reasoning processes to be materialized and inspectable, akin to query execution plans in DBMSs. As illustrated in Figure 1, queries may include reasoning traces – for instance, Q1 shows its execution plan, while Q4 cites supporting tables and figures.

Our Contributions – A Holistic Architecture for Agentic Scholarly Data Management. To address the above challenges collectively, we introduce AgenticScholar, which embodies a holistic system architecture (Figure 2) to unify knowledge representation, LLM-centric hybrid planning, and unified execution to support diverse scholarly queries.

• **Knowledge Representation Layer** (Section 3). To address *Challenge 1*, we introduce a ***taxonomy-anchored knowledge graph*** that unifies fine-grained document structure with an explicit conceptual organization of the research landscape. Unlike existing academic knowledge graphs [57, 74, 76] that primarily organize papers via bibliographic relationships, our representation models papers together with their internal components (e.g., sections, tables, figures, and experimental metadata), enabling fine-grained evidence access during query execution. Crucially, the graph is anchored by ***hierarchical taxonomies*** of research problems and methods, which provide shared semantic abstractions and serve as the contextual backbone for topic-level analysis, such as research trend analysis and research idea exploration. To support this capability, we develop a novel algorithm to ***automatically construct*** these taxonomies from the corpus and ***progressively update and refine*** them as new research outputs emerge.

• **LLM-Centric Hybrid Query Planning Layer** (Section 4). To tackle *Challenge 2*, this layer places an ***LLM at the center of query planning*** to interpret diverse user intents. Unlike traditional query optimizers in DBMSs, the planner integrates ***LLM-driven semantic reasoning*** to interpret ambiguous natural language queries, decompose analytical intents, and reason about operator composition beyond relational algebra. To reduce planning cost while ensuring reliable performance, the planner adopts a ***hybrid planning framework*** that combines reusable predefined plans for common scholarly queries with dynamic plan generation for queries outside this library. ***Explicit plan validation and self-correction*** are further incorporated to ensure plan executability.

• **Unified Execution Layer** (Section 5). To address *Challenge 3*, we introduce the Unified Execution Layer, which serves as the operational backbone of AgenticScholar. First, we analyze typical scholarly queries in Figure 3 and design a set of ***general, reusable operators*** that can be composed to support a wide range of analytical goals (see Section 5.1). Second, we design a ***parallel execution engine*** with ***result caching*** to accelerate query processing by reusing (intermediate) results across queries (see Section 5.2). The architecture remains extensible, allowing new operators to be seamlessly integrated as emerging query types arise. To ensure ***interpretability and traceability*** (Challenge 4), the execution engine materializes the full query plan and data lineage for each

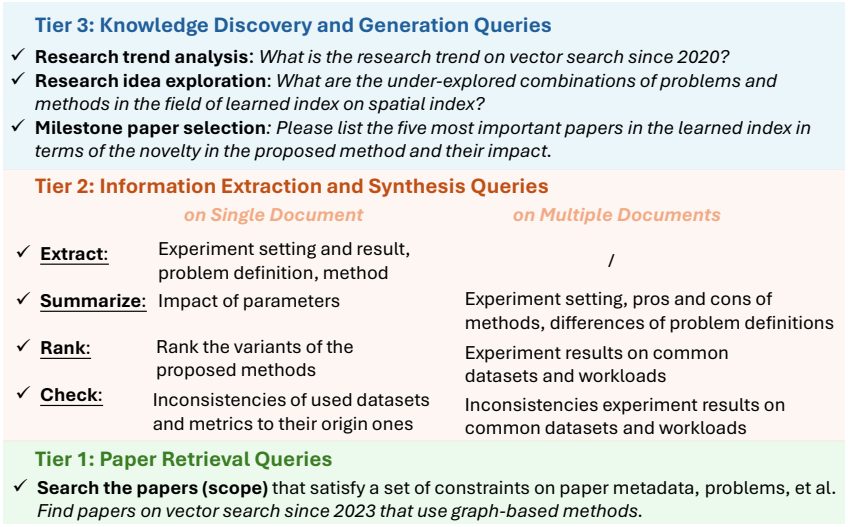


Fig. 3. Representative scholarly queries across three tiers.

result, and exposes them to users for inspection. Moreover, users can control the level of detail in the generated outputs through internal operator parameters – for example, requesting a concise summary, a detailed explanation, or a fully evidence-backed reasoning trace.

• **System Implementation and Evaluation** (Section 6). Together, these layers form a unified framework that addresses the four challenges outlined above and supports diverse, open-ended scholarly queries. We implement AgenticScholar and conduct comprehensive experiments to evaluate its capabilities across retrieval, information extraction and synthesis, and knowledge discovery and generation queries. Furthermore, we present an end-to-end case study simulating realistic, open-ended scholarly exploration and perform ablation studies to quantify the impact of key architectural choices.

2 Typical Queries and Related Work

2.1 Typical Scholarly Queries

Figure 3 shows representative scholarly queries in AgenticScholar.

Tier 1: Paper Retrieval Queries. This foundational tier focuses on locating papers matching specific criteria, serving as the basis for subsequent analytical tasks.

Tier 2: Information Extraction and Synthesis Queries. This tier involves fine-grained content analysis, covering both *single-document extraction* to extract details of a paper and *multi-document synthesis* to synthesize knowledge across papers.

Tier 3: Knowledge Discovery and Generation Queries. This advanced tier extends beyond retrieval or summarization to generate novel insights and actionable guidance.

2.2 Related Systems

Several systems support scholarly queries to a certain extent, as shown in Table 1. We categorize these systems into four classes according to their design focus and depth of reasoning.

Scholarly-Centric Systems. This class of systems [10, 21] supports large-scale paper retrieval and literature summarization. Among them, Elicit [21] covers all tiers of scholarly queries but offers only limited query refinement – restricted to improving existing summaries – and often incurs longer response times. Other systems are designed for a single query tier. For example,

Table 1. Comparisons on related systems (● supported; ◐ partially supported; × not supported).

System	Tier 1 Queries	Tier 2 Queries		Tier 3 Queries		
	Retrieval	Single doc.	Multiple doc.	Trend Analysis	Idea Exploration	Milestone Selection
Elicit [21]	●	●	●	●	●	●
Ai2 Asta[1] [10]	●	×	×	×	×	×
ChatPDF [12]	×	●	●	×	×	×
RAG [25]	×	●	●	×	×	×
Gemini Deep Research [59]	◐	●	●	●	●	●
Tongyi Deep Research [60]	◐	●	●	●	●	●
AgenticData [70]	×	●	●	×	×	×
AgenticScholar	●	●	●	●	●	●

[1]Ai2 Asta claims to support literature summarization but currently fails on complex Tier-2 and Tier-3 queries based on our test; hence marked as not supported.

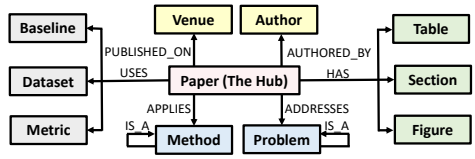


Fig. 4. The schema of the knowledge graph.

ChatPDF [12] allows users to upload papers for information extraction and synthesis, whereas PASA [29] targets paper retrieval by leveraging textual and citation-based signals to locate studies.

LLM-based Tools. Retrieval-Augmented Generation (RAG) [20, 25, 28, 32, 38, 46, 78] enhances LLMs by grounding their outputs in retrieved sources. This approach improves reliability in single-document extraction and multi-document synthesis, providing a scalable alternative to direct LLM APIs [5–8, 26], which are constrained by token limits and high computational costs. However, most existing RAG implementations retrieve only a limited number of content chunks, resulting in poor source coverage and weak contextual integration. Consequently, they are less effective for open-ended queries that require progressive exploration and deep reasoning across corpora.

Agentic Deep Research Systems. Representative applications, such as Gemini Deep Research [59] and Tongyi Deep Research [60], primarily target Tier-3 queries. Unlike standard LLM APIs, they autonomously plan multi-step reasoning – decomposing complex questions, extracting real-time information, and synthesizing structured, citation-grounded reports. This enables higher-level capabilities like identifying research trends, milestone papers, and promising ideas. However, each presents trade-offs: Gemini Deep Research produces more comprehensive results but with longer response times, reducing interactivity, whereas Tongyi Deep Research may respond faster but is less effective for nuanced higher-tier queries.

Semi-Structured Document Analytics Systems. Recent systems [11, 42, 43, 55, 62, 66, 70, 72, 75] extend LLM-based reasoning to document analytics but mainly support Tier-2 queries. They struggle with Tier-3 queries, which are open-ended and involve evolving user intents, due to their reliance on rigid queryable forms such as manually written programs [43, 55, 66] or SQL-like queries [42, 72]. While systems such as AgenticData [70] and Unify [75] introduce natural language interfaces, their data models remain insufficient for multi-modal and context-aware representation, and their limited operator spaces restrict their ability to perform complex semantic reasoning, such as research idea exploration.

3 Knowledge Representation Layer: From Multi-Modal Structure to Hierarchical Research Taxonomies

Supporting the diverse queries shown in Figure 3, especially Tier-3 queries that require reasoning over research topics, demands a storage representation that goes beyond document-centric storage. Such a representation must both expose the fine-grained, multi-modal structure of individual papers and anchor them within a shared, explicit conceptual organization of the research landscape.

Thus, the representation of a scholarly corpus must satisfy three key requirements: (1) provide fine-grained access to multi-modal content such as sections, tables, and figures; (2) maintain explicit links among related elements – including datasets, metrics, baselines, and citations – to capture experimental and bibliographic relationships; and (3) organize papers in structured hierarchies of problems and methods, enabling reasoning over research topics rather than isolated documents.

Taxonomy-Anchored Knowledge Graph. To this end, AgenticScholar introduces a *Knowledge Representation Layer* that transforms the scholarly corpus into a structured knowledge graph (KG). While prior academic KGs such as MAG [76], OpenAlex [57], and AMiner [74] primarily model bibliographic relationships among papers, authors, venues, and citations, such document-centric representations are insufficient for supporting topic-level and compositional reasoning. As a result, our KG is anchored by *Taxonomy Nodes* – specifically `Problem` and `Method` nodes. These nodes define shared abstractions of research problems and algorithmic strategies, serving as first-class semantic primitives for reasoning. For instance, when addressing Q5 in Figure 1, the system can directly infer unexplored problem–method combinations by traversing the problem and method taxonomies within the vector search domain, rather than aggregating evidence across individual papers, thereby enabling more efficient and scalable reasoning.

As illustrated in Figure 4, our KG is structured *around these taxonomy nodes* to capture the multi-modal structure of scholarly documents. Each `Paper` node acts as a central hub that links to Content Nodes (including `Section`, `Figure`, and `Table`), Experimental Context Nodes (including `Dataset`, `Metric`, and `Baseline`), and Bibliographic Nodes (including `Author` and `Venue`). This joint modeling of taxonomy-level abstractions and document-level evidence enables AgenticScholar to support complex Tier-3 queries that require topic-level reasoning grounded in document content.

As the KG instantiation follows engineering practices, we provide implementation details in Appendix B.1 in [22]; the following section presents our novel algorithm for taxonomy construction.

3.1 Reference-enhanced Taxonomy Construction

Constructing high-quality taxonomies for a scholarly corpus remains a significant challenge. While various approaches exist, they face several limitations: (1) Corpus-driven methods [14, 30, 33, 65, 67, 83, 87] build hierarchies directly from papers. While effective at capturing domain-specific nuances, they often suffer from data sparsity, yielding fragmented or overly narrow structures [45]. (2) LLM-driven methods [13, 33, 71, 82] leverage an LLM's internal knowledge to ensure broader conceptual coverage. However, they frequently fail to capture emerging topics that appear in the corpus but are absent from the model's pre-training data. (3) Crucially, both paradigms typically lack mechanisms for progressive refinement and incorporating newly papers may require reconstructing the entire taxonomy from scratch – a prohibitive cost for rapidly evolving research fields.

Our Novel Solution. To overcome these limitations, we propose a *Reference-enhanced Taxonomy Construction Framework*. The framework synergizes an LLM-derived scaffold with corpus-grounded evidence, ensuring that the resulting taxonomy is structurally comprehensive and faithful to the target corpus. Crucially, it supports *progressive refinement*, allowing the taxonomy to evolve dynamically as new papers arrive, without full reconstruction. As outlined in Algorithm 1, the taxonomy construction proceeds in four stages:

Algorithm 1 TaxonomyConstruction

Input: scholarly corpus D, optional new papers D_{new}, refinement factor α, match threshold τ_{match};
Output: taxonomy T;

```
    // Stage 1: Structured Information Extraction
 1: A ← {};                                          ▷ map: paper ↦ problem/method aspects
 2: for p ∈ D do A[p] ← ExtractAspect(p);
    // Stage 2: Cross-paper Standardization
 3: C ← CrossPaperStandardization(A);
    // Stage 3: Reference Taxonomy Generation and Alignment
 4: L ← InferTopicLabel(D), T_ref ← GenerateReferenceTaxonomy(L);
 5: T ← CopyTree(T_ref);
 6: for (C, D_C) ∈ C do
 7:     T.u ← TopDownLocateParent(T.root, C);
 8:     T.v ← FindMatchingChild(T.u, C, τ_match);
 9:     if T.v == NULL then T.v ← AddChild(T.u, C);
10:     Assign papers D_C to T.v;
    // Stage 4: Progressive Update and Branch Refinement
11: if D_new ≠ ∅ then
12:     A_new ← {};
13:     for p ∈ D_new do
14:         A ← ExtractAspect(p), C ← MapToConceptsOrCreate(A);
15:         T.u ← TopDownLocateParent(T.root, C);
16:         T.v ← FindMatchingChild(T.u, C, τ_match)
17:         if T.v == NULL then T.v ← AddChild(T.u, C);
18:         Assign paper p to T.v;
19:         if NewPaperCount(T.v) ≥ α then RefineBranch(T.v)
20: return T
```

• *Structured Information Extraction*: Extract key aspects that characterize paper's problem or method.
• *Cross-Paper Standardization*: Synthesize extracted aspects across papers into standardized concepts by identifying recurring concepts and unifying synonyms.
• *Reference Taxonomy Alignment*: Align the extracted aspects of each paper and the standardized concepts with an LLM-generated reference taxonomy, constructed using topic labels from the corpus, to produce a taxonomy that integrates corpus-grounded evidence with domain knowledge.
• *Progressive Update and Branch Refinement*: Incorporating newly arriving papers by progressively refining the taxonomy structure through three operations: assigning them to existing nodes, creating new nodes, or introducing new branches.

In the following, we describe each stage, referencing Algorithm 1 and the illustrative example in Figure 5. We use the construction of the problem taxonomy as a representative case, as the construction processes for problem and method taxonomies are analogous. Implementation details are provided in Appendix B.2 in [22].

Stage 1: Structured Information Extraction. The first stage extracts essential aspects from each paper in the scholarly corpus to facilitate alignment (Line 2 in Algorithm 1). We adopt a context-aware extraction strategy that combines *Shared Content* (e.g., Abstract, Introduction, and Related Work), which orients the LLM to the paper's overall scope, with *Targeted Content* (e.g., the Problem Definition for the problem taxonomy), enabling precise extraction of problem- or method-specific information. The extracted aspects differ by taxonomy type:
• For problem taxonomy, the aspects consist of formal specifications of the *input* and *output*. As illustrated in Figure 5, this is achieved by decomposing each problem into concrete data constraints (e.g., "Input: a query vector and an attribute filter" in Paper 1) and objectives (e.g., "Output: top-k candidates" in Paper 1). Such formalization enables fine-grained distinctions between closely related problems, e.g., differentiating KNN retrieval (Paper 2) from range search (Paper 3).

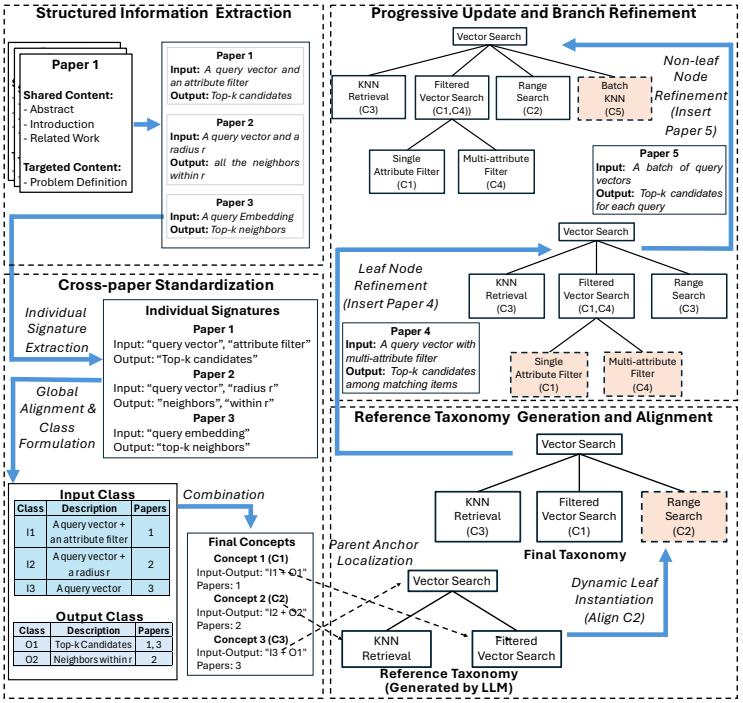


Fig. 5. Reference-enhanced taxonomy construction.

• For method taxonomy, the aspects include *key techniques* (e.g., core components, ordered workflows), and, crucially, the *strengths and weaknesses*. While technical descriptions capture how a method operates, strengths and weaknesses explain why it was designed, enabling methods to be grouped not only by mechanics but also by shared optimization goals (e.g., trading accuracy for speed). This reveals implicit methodological families that are often missed by technical descriptions.

Stage 2: Cross-Paper Standardization. While the previous stage yields isolated aspects, this stage synthesizes them into coherent, corpus-level concepts (Line 3 in Algorithm 1). We employ a three-step process to standardize extracted aspects – unifying diverse phrasings into canonical classes, synthesizing these classes into composite concepts, and assigning each paper to the concept, which serves as corpus-ground evidence for the next stage.

Step 1: Individual Signature Extraction. We extract key noun phrases using an LLM, which serve as semantic signatures for the target aspect. As shown in Figure 5, when the input is the target aspect, Paper 1 yields signatures "query vector" and "attribute filter". These signatures capture essential semantics required for alignment.

Step 2: Global Alignment and Class Formation. We prompt an LLM to identify semantically equivalent signatures and group them under a canonical class. As shown in Figure 5, the distinct signatures "top-k candidates" (Paper 1) and "top-k neighbors" (Paper 3) are identified as synonyms and consolidated into Output Class O_1.

Step 3: Assignment and Concept Formulation. Canonical classes derived from aspects are combined to form *concepts*, which represent abstractions of problem families (e.g., pairing Input Class I_1 with Output Class O_1), and each paper is mapped to the most appropriate concept. As shown in Figure 5, Paper 1 is assigned to Concept C_1 (representing $I_1 + O_1$), while Paper 2 is assigned to Concept C_2. Papers that do not match existing concepts form new subgroups that are recursively re-processed through *Step 1* and *Step 2*, ensuring comprehensive coverage.

Stage 3: Reference Taxonomy Generation and Alignment. After standardizing corpus-grounded concepts, this stage integrates an LLM-derived reference taxonomy with these concepts to construct the final taxonomy, which forms the core of our framework. This integration are in three steps.

Step 1. Reference Taxonomy Generation. We first instruct an LLM to infer a topic label from the corpus and generate a skeletal reference taxonomy (Line 4 in Algorithm 1). In Figure 5, the LLM identifies topic "Vector Search" and generates child nodes "KNN Retrieval" and "Filtered Vector Search" based on general domain knowledge.

Step 2. Node Localization. We then map each standardized concept to the reference taxonomy through two sub-steps.
(1) *Parent Anchor Localization:* We traverse the reference taxonomy in a top-down manner to identify the most specific parent node that subsumes the concept (Line 7 in Algorithm 1). In Figure 5, for Concept C_2, the system identifies the root node "Vector Search" as the parent anchor, since C_2 does not semantically fit under the existing "KNN Retrieval" or "Filtered Vector Search" nodes.
(2) *Dynamic Leaf Instantiation:* If the concept does not match any existing child of the parent anchor, we instantiate a new node (Lines 8-9 in Algorithm 1). In Figure 5, because a specific "Range Search" node is absent from the reference taxonomy, the system dynamically creates a new node for C_2, prompts the LLM to generate a canonical name "Range Search", and attaches it to the taxonomy.

Step 3. Paper Assignment. Finally, all papers associated with the concept are linked to the corresponding taxonomy node (Line 10 in Algorithm 1). For example, Paper 2 is assigned to "Range Search" node, while Paper 1 is assigned to "Filtered Vector Search", ensuring that every document is reachable through the taxonomy hierarchy.

Stage 4: Progressive Update and Branch Refinement. To accommodate the dynamic nature of research, our framework supports progressive update, allowing the taxonomy to evolve as papers arrive *without requiring reconstruction from scratch* (Lines 11-20 in Algorithm 1). Specifically, new papers are first processed by extracting aspects following Stage 1 and mapping them to existing concepts or creating new concepts (Line 14 in Algorithm 1). They are then assigned to the most specific existing taxonomy node via Step 2 and Step 3 of Stage 3 (Lines 15-18 in Algorithm 1). To prevent taxonomy nodes from becoming overly broad due to concept drift, we further employ a branch refinement mechanism.

Step 1. Refinement Trigger. As shown in Line 19 of Algorithm 1, when the number of new papers linked to a node exceeds a predefined threshold α, the node is flagged for refinement.

Step 2. Refinement Execution. The refinement strategy adapts based on the node's position in the taxonomy and leverages corpus-grounded clustering to discover fine-grained structures that are often absent from the LLM's pre-training data.
Case 1: Leaf Node Refinement. This case captures the emergence of granular subtopics. As illustrated in Figure 5, the leaf node "Filtered Vector Search" accumulates new papers, including Paper 4, which introduces multi-attribute filtering. The system performs semantic clustering on these papers and splits the node into a more detailed subtree consisting of "Single Attribute Filter" (C_1) and "Multi-attribute Filter" (C_4).
Case 2: Non-leaf Node Refinement. This case handles papers that fit a high-level category but do not align with any child nodes. In Figure 5, Paper 5 maps to the root "Vector Search" but fits none of its children. The refinement process identifies this distinct cluster and instantiates a new sibling branch, "Batch KNN" (C_5), extending the taxonomy breadth without disrupting existing branches.

4 LLM-Centric Hybrid Planning Layer: From Natural-Language Queries to Executable Plans

The Query Planning layer translates high-level natural-language queries into executable Directed Acyclic Graphs (DAGs) of operators. While this objective shares the high-level spirit of classical database query optimization [37, 64], our system operates in a fundamentally different setting.

Unique Challenges. Traditional query optimizers operate over algebraic SQL queries with well-defined schemas and formally specified operator semantics. In contrast, our planner must reason over open-ended natural language scholarly queries, which introduces challenges absent in DBMS planning: (1) *Lack of algebraic semantics:* Natural language queries do not admit a fixed algebraic representation, rendering rule-based transformations and cost-based enumeration ineffective. (2) *Open-ended analytical intents:* Scholarly queries are often open-ended and may evolve into diverse analytical intents (Figure 1), leading to a vast plan space that extends well beyond relational algebra. (3) *Verification complexity:* Unlike deterministic SQL compilation, generative planning requires explicit mechanisms to validate plan correctness and executability before execution.

Our Planner. To address these challenges, we propose an *LLM-centric hybrid query planner* that places LLMs at the core of semantic reasoning while carefully constraining their role to ensure efficiency and reliability. Our planner is built on three key principles:

- *LLM-driven Semantic Reasoning.* We leverage LLMs to interpret ambiguous natural language queries, decompose analytical intents, and reason about operator composition.
- *Hybrid Planning for Efficiency and Generality.* To reduce planning cost while ensuring reliable performance, we combine a library of predefined plans for common scholarly queries with dynamic plan generation for queries outside this library.
- *Robustness through Validation and Self-correction.* We introduce explicit plan validation and an iterative self-correction loop to mitigate the non-determinism and errors of LLM-generated plans.

Figure 6 illustrates the end-to-end process by which a natural language query is translated into a validated execution plan; the details are presented in the following subsections.

4.1 LLM-Driven Query Decomposition

To bridge the gap between ambiguous natural language and modular execution, our system first decomposes the analytical intent into two components: a `Scope` and a `Task`. The `Scope` specifies the entities or conceptual boundary of analysis, while the `Task` describes the analytical step to be performed over that scope.

This decomposition is achieved by prompting an LLM with a structured instruction that explicitly separates "what to analyze" from "how to analyze it". As shown in Figure 6, given the query "Find papers on vector search since 2023 that use graph-based methods and build a table comparing their indexing speed and memory usage", the LLM produces: (1) `Scope` – "papers on vector search since 2023 that use graph-based methods"; and (2) `Task` – "build a table comparing their indexing speed and memory usage".

This decomposition yields a clearer and more modular representation of analytical intents, reducing ambiguity and enabling subsequent planning stages to operate more reliably and efficiently.

4.2 Predefined Plan Selection

For common and well-understood scholarly queries (Figure 3), invoking a fully generative planner is inefficient. To achieve low latency and high reliability for such queries, our system first attempts to match the query's `Task` component to a plan in a *Predefined Plan Library* (Appendix C in [22]) . This library plays a role analogous to cached query plans in traditional DBMSs [2, 19, 48]; however,

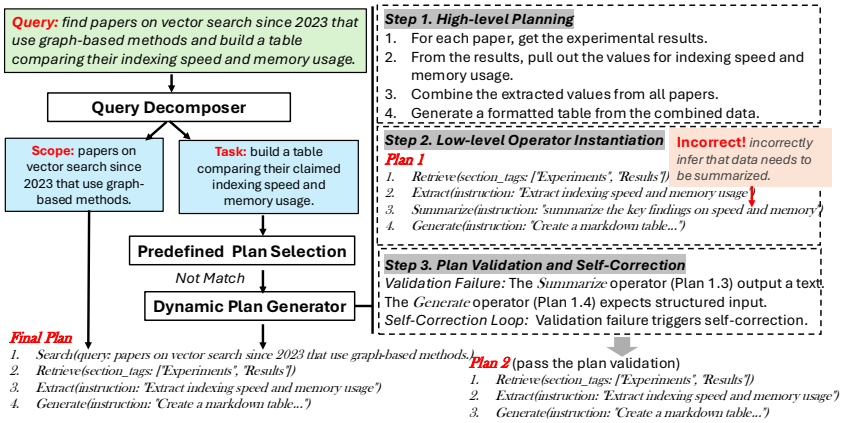


Fig. 6. From query to validated execution plan.

despite this resemblance, our design differs in two fundamental ways to accommodate the semantic nature of natural-language queries:

• **Natural-language-driven Parameters.** In our setting, plan "parameters" are not SQL-like constants (e.g., `WHERE id = ?`) but semantic intents expressed in natural language (e.g., "Extract indexing speed and memory usage"). Consequently, plan selection requires semantic understanding rather than syntactic matching.

• **Reliability-oriented plan selection.** As the plan library is extensible and may grow large over time, directly prompting an LLM to select from all available plans is unreliable: long prompts with many candidates often lead to misselection. To ensure robust plan reuse, we adopt a two-stage selection strategy:

(1) Candidate Filtering: A semantic search is first performed to compare the task description against the descriptions of all plans in the library, retrieving a small candidate set of plans.

(2) LLM-based Reranking and Validation: The candidate plans are then evaluated by a reasoning LLM, which assesses their alignment with the analytical intent and assigns a confidence score.

A plan is selected only if its confidence score exceeds a high threshold (e.g., $> 90\%$); otherwise, the query is treated as ad-hoc and passed to the dynamic plan generator. For example, the task *"Extract indexing speed and memory usage"* directly matches a predefined extraction plan specialized for experiment sections.

This design enables our system to retain the low latency benefits of plan reuse while preserving the semantic flexibility and robustness required for natural language scholarly queries.

4.3 Dynamic Plan Generation with Self-correction

For queries that fall outside the predefined library, our system invokes the *Dynamic Plan Generator*. To manage the combinatorial explosion of the plan space and reduce hallucinations, we employ a decomposed generation process: generating a high-level logical plan via few-shot demonstrations, followed by low-level operator instantiation. Crucially, we introduce a **self-correction loop** to ensure robustness.

Step 1. High-Level Planning with Demonstrations. To narrow the search space, our system first generates a high-level logical plan as an abstract sequence of steps. To guide the LLM, we retrieve representative `(query, plan)` demonstrations from an extensible library via semantic search and inject them into the LLM context.

Step 2. Low-Level Operator Instantiation. Our system then proceeds to the low-level instantiation stage. It iterates through each abstract step of the logical plan and makes a series of focused,

Table 2. Summary of Operator Library.

Category	Operator	Description
Knowledge Access	`Search`	Locate documents from the knowledge graph
Knowledge Navigation	`FindNode`	Locate specific entry-point nodes within taxonomies
	`Traverse`	Traverse the knowledge graph from a set of starting nodes
	`Retrieve`	Fetch the raw content of specific sections from a document
Knowledge Generation	`MatrixConstruct`	Create a structured matrix summarizing problems and methods
	`Generate`	Generate new content based on inputs and specific instructions
Semantic Content Processing	`Extract`	Extract specific, structured information from raw text content
	`Summarize`	Summarise textual information into a concise, coherent overview
	`Check`	Compare information across multiple peer documents
	`Verify`	Verify a specific claim against a designated source of evidence
	`Rank`	Order a set of entities based on a specific, measurable criterion
Relational-Style Data Manipulation	`GroupBy`	Partition a list of entities based on a specified attribute
	`Aggregate`	Compute summary statistics over a set of entities
	`Filter`	Return a subset of entities that satisfy the condition

less complex calls to the LLM. For each step, the LLM's task is to select the single most appropriate operator from the operator library and fully parameterize it to accomplish that specific sub-task. This enables our system to ask the LLM to solve a much simpler problem at each step rather than a complex problem of generating an entire DAG at once, significantly increasing the success rate.

Step 3. Robustness through Plan Validation and Self-Correction. Generated plans are never trusted implicitly. Before execution, every plan is passed to a *Plan Validator* that performs fast, deterministic checks for DAG validity, operator correctness. If a plan fails validation, or if a syntactically valid plan fails at runtime, our system triggers a *self-correction loop*. It is re-invoked with an augmented context that includes the failed plan and the specific error message. The prompt is shifted from "planning" to "debugging", instructing the LLM to analyze the error and propose a corrected plan. As shown in Figure 6, the `Summarize` operator is ineffective because the subsequent `Generate` operator requires structured numeric evidence to construct the table. However, `Summarize` produces only high-level textual descriptions, which likely omit the detailed quantitative information needed for table generation.

Final Plan Composition. The final stage assembles the complete execution plan by composing the plan for the `Scope` component with the plan for the `Task` component. Our system chains these two parts together, ensuring the output of the initial operator (e.g., `Search` or `FindNode` from the scope analysis) is correctly fed as the input to the subsequent task plan (either the selected predefined plan or the dynamically generated one). The result is an DAG passed to the execution engine.

5 Unified Execution Layer: Operator Orchestration and Execution

As established in Section 2.1, scholarly queries span a wide spectrum from simple document retrieval to advanced knowledge discovery and generation. A system designed to support this diversity, along with future analytical needs, requires a *unified and extensible* execution framework rather than a collection of hard-coded, query-specific pipelines. To achieve this, we first analyze the typical query patterns and our underlying knowledge representation to design a library of composable operators (Section 5.1). Following established principles from traditional database systems [39, 49, 56], these operators serve as the building blocks for execution plans, which are structured as Directed Acyclic Graphs (DAGs). We build an execution engine that orchestrates these DAG-based plans to deliver accurate and interpretable results (Section 5.2).

5.1 Operator Library

The operator library in AgenticScholar provides building blocks for execution plans and is summarized in Table 2. We introduce novel operators tightly coupled with the taxonomy-anchored knowledge graph, enabling topic-level reasoning to support Tier-3 queries beyond retrieval and summarization. Representative operators are described below, with the full details in Appendix D in [22].

`Search`. This operator serves as the main entry point for retrieving relevant scholarly documents from the knowledge graph. It supports natural language queries (e.g., Q2 in Figure 1) that combine precise constraints with abstract intent via a hybrid retrieval strategy balancing lexical precision and semantic recall. Given a query, `Search` uses an LLM to decompose it into structured metadata constraints (e.g., title, authors, venue, year) and semantic intents over research aspects (e.g., topics, problems, methods, datasets). Metadata constraints are handled using BM25-based lexical retrieval [61], while semantic intents are matched via dense retrieval over BGE embeddings [80] indexed by FAISS [18]. Retrieved candidates are then refined by an LLM-based reranker that jointly reasons over the query, paper metadata, and KG-derived aspect summaries to produce aligned results for downstream operators.

`FindNode`. This operator identifies an entry-point taxonomy node for topic-level reasoning. It supports direct key-based lookup when an identifier is provided, or semantic matching over precomputed node description embeddings to select the best-matching concept.

`Traverse`. This operator enables controlled multi-hop navigation over the knowledge graph from given starting nodes. It uses a typed `traversal_path` to constrain relationship and entity types at each hop, preventing unintended expansion. At runtime, `Traverse` compiles the path into a multi-hop graph query (e.g., Cypher [23]) and returns an entity list for downstream operators such as `Aggregate` (e.g., Q1 in Figure 1) and `Rank` (e.g., Q3 in Figure 1).

`MatrixConstruct`. This operator supports research idea exploration (e.g., Q5 in Figure 1) by synthesizing a structured view of a research topic to reveal promising directions. Given a topic, `MatrixConstruct` uses problem and method taxonomies to identify representative problems and methods, and organizes them into a problem-method matrix. Each cell in the matrix aggregates supporting evidence (e.g., representative papers and summaries). The matrix can be inspected directly or passed to `Generate` for evidence-grounded synthesis, enabling identification of sparse or missing problem–method combinations.

Remark on Extensibility and Generality. The operator library is designed to be both general and extensible: *Generality* is achieved through two mechanisms: (1) a concise set of well-chosen operators and (2) flexible instruction parameters that allow each operator to adapt to a wide range of tasks without altering the system's core logic. *Extensibility* arises from the modular architecture, which enables new operators to be added seamlessly as analytical intents evolve. Together, these properties ensure that the system can scale and adapt in tandem with the changing research landscape.

5.2 Unified Query Execution Engine

The Unified Query Execution Engine realizes DAG-based query plans produced by the planner introduced in Section 4. It instantiates operators, resolves their dependencies, and manages intermediate and final results during execution. Its design emphasizes scalable parallel execution, efficient result reuse, and end-to-end traceability to support reliable and interpretable query execution. Algorithm 2 sketches the end-to-end execution workflow.

5.2.1 Execution Model. There are three key steps:

Algorithm 2 UnifiedQueryExecution

Input: query plan P;
Output: final result R;
Step 1: Scope Execution and Plan Unfolding
1: $S \leftarrow$ ExecuteScopeOperators(P);
2: $G \leftarrow$ UnfoldPlan(P, S); ▷ Instantiate operators based on execution_mode
Step 2: Dependency Resolution
3: **for** operator $o \in G$ **do**
4: $dep\_count[o] \leftarrow$ number of unresolved dependencies of o;
5: **if** $dep\_count[o] = 0$ **then** enqueue o into Q;
Step 3: Operator Dispatch and Parallel Execution
6: **while** Q is not empty **do**
7: $o \leftarrow$ dequeue from Q ▷ May execute in parallel
8: $inputs \leftarrow$ FetchInputs(o, *Buffer*, *Cache*, *Storage*);
9: **if** CacheHit(o, *inputs*) **then** $output \leftarrow$ LoadFromCache(o, *inputs*);
10: **else** $output \leftarrow$ Execute(o, *inputs*), StoreInCache(o, *inputs*, *output*);
11: $Buffer[o] \leftarrow output$;
12: RecordTrace(o, *inputs*, *output*);
13: **for** downstream operator d of o **do**
14: $dep\_count[d] \leftarrow dep\_count[d] - 1$;
15: **if** $dep\_count[d] = 0$ **then** enqueue d into Q;
16: **return** outputs produced by terminal operators in G;

Step 1. Scope Execution and Plan Unfolding. The engine instantiates the query plan in two phases. Instead of materializing the entire DAG upfront, it first executes the operators in the query's `Scope` (e.g., `Search`; Line 1 in Algorithm 2) to obtain an initial result set (e.g., N papers). It then enters a plan unfolding stage by inspecting each operator's execution_mode (Line 2 in Algorithm 2). Operators marked as `instance` (e.g., `Extract`) are expanded into N parallel instances, one per item, while `group` operators (e.g., `Summarize`) are instantiated once to aggregate across all results. This process transforms the compact plan into a data-aware execution graph that explicitly exposes parallelism.
Step 2. Dependency Resolution. Given the unfolded plan, the engine performs a topological analysis to determine execution order, enqueuing operators with no unresolved dependencies as ready for execution (Lines 3–5 in Algorithm 2).
Step 3. Operator Dispatch and Parallelism. Operators are dispatched once all required inputs are available. Upon completion, the engine propagates outputs, updates dependency counters of downstream operators, and enqueues operators whose dependencies are satisfied (Lines 6–15 in Algorithm 2). Multiple operators may become ready concurrently, enabling parallel execution while preserving dependency correctness.

5.2.2 Result Management, Reuse, and Traceability. During execution, AgenticScholar manages intermediate and final results through transient buffering, persistent caching, and execution traceability, enabling efficient reuse of computation while ensuring transparent and interpretable query execution.
Transient Buffering. Intermediate results are stored in a transient in-memory buffer keyed by operator IDs (Line 11 in Algorithm 2). Upon completion, each operator materializes its output into the buffer, from which downstream operators retrieve inputs by referencing upstream IDs. Operators without dependencies read directly from the storage layer (Line 8 in Algorithm 2), enabling efficient in-run reuse without redundant computation.
Persistent Caching. Beyond operator-level buffering, the engine maintains an execution cache to persist and reuse results across executions. Unlike transient buffers, the cache stores outputs of

Table 3. Effectiveness on Tier-1 queries.

Methods	Factual Queries		Analytical Queries	
	R-Precision	MAP	R-Precision	MAP
BM25	0.49	0.52	0.42	0.45
GTR	0.44	0.47	0.36	0.39
E5	0.51	0.54	0.46	0.48
Instructor	0.52	0.53	0.45	0.46
EmbeddingGemma	0.49	0.49	0.42	0.41
ColBERT	0.58	0.60	0.45	0.42
Qwen3-Embedding	0.61	0.65	0.50	0.49
AgenticScholar	**0.73**	**0.74**	**0.69**	**0.70**

plans and subplans for reuse (Line 10 in Algorithm 2). When identical or overlapping plans are reissued, cached results are retrieved instead of recomputed (Lines 8–9 in Algorithm 2). Cached entries are persisted to disk, enabling durability across process lifetimes and accelerating iterative and incremental execution.

Execution Traceability. To support auditing and interpretation, all runtime artifacts are keyed by execution node IDs. The engine records per-node provenance and metadata, including dependencies, status transitions, and runtime statistics (Line 12 in Algorithm 2), enabling reconstruction and inspection of the annotated execution DAG that produced each result.

6 Experimental Study

We implement AgenticScholar [22] and conduct a comprehensive experimental study to evaluate its effectiveness, efficiency, and end-to-end usability. Our evaluation is grounded in a diverse corpus designed to assess the system's versatility across various scholarly domains, covering broad domains, specialized sub-fields, and emerging areas. A complete list of topics is in Appendix F in [22].

Our evaluation begins by benchmarking the performance of AgenticScholar's core capabilities: retrieval (Section 6.1), information extraction and synthesis (Section 6.2), and knowledge discovery and generation (Section 6.3). To demonstrate its practical usability, we then present an end-to-end case study that simulates a realistic open-ended scholarly exploration workflow (Section 6.4). Finally, we conduct a series of ablation studies (Section 6.5) to validate the measurable benefits of our key architectural decisions.

6.1 Evaluation of Retrieval

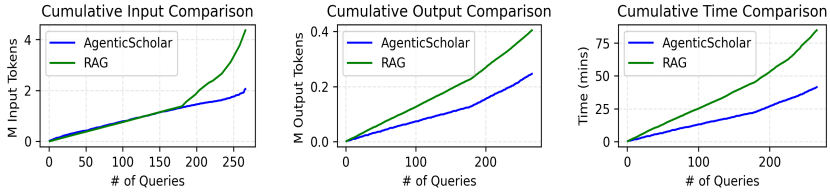
We evaluate the effectiveness of AgenticScholar in paper retrieval (Tier 1 in Figure 3) using 237 research papers spanning 10 topics.

Query Generation. We construct 220 queries of two types: *factual queries*, which target explicit paper metadata (e.g., title, authors, venue) and research aspects (e.g., topics, problems, methods, datasets); and *analytical queries*, which require reasoning within or across topics. Each query is derived from paper metadata and aspect summaries and is paired with 1–10 manually verified ground-truth papers. Additional details are provided in Appendix H.1 in [22].

Baselines. We compare AgenticScholar with the classical lexical baseline *BM25* [61], strong dense retrievers – *GTR* [51], *E5* [77], *Instructor* [69], *EmbeddingGemma* [73], and *Qwen3-Embedding* [84] – and the late-interaction retriever *ColBERT* [63]. AgenticScholar and ColBERT have ∼0.1B parameters, while others range from ∼0.3B to ∼1.5B. Most models use 768-d embeddings, except E5 (1024), and ColBERT uses 128-d token embeddings. All methods are reranked with GPT-4.1 on the top-k candidates for fair comparison [1]. We omit *LLM-Direct* retrieval due to context-length limitations.

Table 4. Effectiveness on Tier-2 queries.

Method	NDCG@1	NDCG@3	NDCG@5	NDCG@7
AgenticScholar	0.606	0.629	0.644	0.655
RAG	0.411	0.482	0.491	0.494



(a) Input Tokens (b) Output Tokens (c) Response Time

Fig. 7. Cost and efficiency on Tier-2 queries.

Evaluation Metrics. We use R-Precision [9], which is precision at R (number of relevant papers), and Mean Average Precision (MAP) [17], which averages precision across relevant ranks. We omit Recall@k and Precision@k since fixed cutoffs are incomparable when queries have different numbers of relevant papers.

Results. Table 3 shows an overall result (see Appendix H.2 in [22] for results by topic). AgenticScholar consistently outperforms all baselines. This confirms its effectiveness, particularly in performing cross-modal synthesis that unifies precise lexical matching with the semantic generalization of dense encoders. For factual queries, this synthesis enables AgenticScholar to capture both explicit facts and semantically related expressions often missed by single-modal methods. For analytical queries, LLM-driven query decomposition and reranking facilitate nuanced reasoning over abstract research aspect, going beyond simple text similarity. These strengths demonstrate that AgenticScholar broadens retrieval coverage while enhancing contextual and analytical fidelity.

6.2 Information Extraction and Synthesis

We evaluate our system's ability to extract relevant information and synthesize it into coherent answers (Tier 2 in Figure 3).

Query Generation. We consider four topics, each containing 10 papers, to construct two test sets.
• *LLM-generated test set.* Using three topics, we employ an LLM to generate evaluation queries. Based on our plan library (Appendix C in [22]) , we generate 480 queries covering 16 single-document plans and 72 queries for six multi-document plans with group sizes ranging from 3 to 10 papers. In addition, we generate 12 queries with group sizes ranging from 3 to 10 papers that require dynamic plan generation.
• *Human-annotated test set.* We additionally construct a human-annotated test set consisting of 197 queries over two topics, derived from predefined single-document and multi-document plans that involve the `Rank` operator (see Table 12 in the appendix in [22]), enabling quantitative evaluation. Specifically, we manually examine the experimental results reported in each paper to determine rankings of methods or datasets. For single-document rank plans, methods or datasets are ranked according to commonly reported evaluation metrics within each paper. For multi-document rank plans, we group 3–10 papers and rank the evaluated methods based on shared datasets and metrics within each group.

Baselines. We compare AgenticScholar with an *RAG-based* method implemented using LangChain [35]. The method retrieves relevant document chunks and synthesizes answers using `GPT-4.1 nano`. We

Table 5. Results on Tier-3 queries. Elicit is the baseline; other entries report relative improvements (green, ↑) or degradations (red, ↓). *Efficiency* shows percentage runtime change (positive = faster).

Metric	Elicit	Gemini DR	Tongyi DR	Open DR	SmolAgent	AgenticScholar$_{noR}$	AgenticScholar$_{noC}$	**AgenticScholar**
Research Trend Analysis — Problem-oriented								
Correctness	5.84	+21.2%	-10.3%	+25.0%	+26.7%	+32.7%	+30.7%	+34.8%
Relevance	5.83	+45.5%	+41.2%	+56.1%	+47.5%	+52.1%	+53.5%	+54.7%
Diversity	5.42	+26.0%	+17.5%	+24.0%	+11.8%	+27.9%	+26.8%	+29.3%
Specificity	5.23	+54.7%	+25.4%	+53.7%	+23.1%	+59.1%	+62.9%	+61.8%
Efficiency	1,276 s	+40.2%	+68.1%	+71.6%	+18.7%	+68.9%	+70.5%	+69.6%
Research Trend Analysis — Method-oriented								
Correctness	6.02	+6.5%	-6.8%	+27.1%	+24.6%	+26.7%	+29.7%	+28.9%
Relevance	7.40	+13.9%	+2.7%	+21.6%	+6.1%	+18.4%	+15.4%	+20.1%
Diversity	4.83	+41.6%	+36.2%	+49.1%	+24.2%	+50.3%	+49.1%	+51.6%
Specificity	5.83	+21.3%	+16.6%	+37.9%	+17.5%	+41.2%	+36.0%	+41.9%
Efficiency	1,432 s	+52.9%	+69.7%	+73.8%	-47.2%	+75.1%	+71.8%	+74.2%
Research Idea Exploration								
Novelty	6.38	+17.1%	-21.6%	+16.0%	+12.7%	+22.7%	+26.0%	+25.1%
Feasibility	7.18	-5.0%	-1.3%	+2.4%	-2.5%	+6.5%	+6.5%	+8.4%
Impact	7.82	+0.3%	-10.9%	+1.0%	+5.5%	+3.2%	+3.2%	+5.8%
Efficiency	1,723 s	+66.7%	+77.3%	+80.7%	+91.9%	+75.3%	+77.4%	+78.0%
Milestone Paper Selection								
Impact	6.67	+19.9%	0.0%	+7.9%	-21.3%	+12.4%	+19.9%	+19.9%
Coverage	5.67	+20.5%	0.0%	+12.9%	-41.6%	+12.4%	+11.6%	+11.6%
Coherence	6.33	+23.7%	+5.4%	+16.9%	-40.8%	+10.5%	+15.7%	+13.1%
Efficiency	213 s	-209.9%	-36.6%	-48.4%	-333.8%	+99.6%	+99.6%	+99.6%

set the chunk size to 1000 to capture broader scholarly context and retrieve 8 chunks per document for each query.

Evaluation Metrics. We evaluate effectiveness on the human-annotated test set using NDCG@k [31], a widely used ranking metric. Cost and efficiency are measured on the LLM-generated test set in terms of input/output tokens and response time.

Results. We observe the following:

Effectiveness. Table 4 reports NDCG@k for k ranging from 1 to 7. AgenticScholar consistently outperforms RAG across all k values, indicating that it accurately extracts relevant information from papers and effectively executes the Rank operator to place correct results at the top of the ranked list.

Cost and Efficiency. Figure 7 reports cumulative input tokens, output tokens, and response time for the *learned index* topic using the LLM-generated test set. Results on other topics show similar trends and are provided in Appendix I in [22]. Across all metrics, AgenticScholar consistently outperforms the standard RAG framework, using fewer input and output tokens and achieving lower cumulative latency as query volume increases.

6.3 Knowledge Discovery and Generation

We evaluate how effective AgenticScholar supports knowledge discovery and generation queries (Tier-3 in Figure 3), i.e., research trend analysis, idea exploration, and milestone paper selection. The implementation details are provided in Appendix E in [22].

Query Generation. We use five research topics spanning broad domains, specialized sub-fields, and emerging areas (Appendix F in [22]). For each topic, we manually design Tier-3 queries of three types – *research trend analysis*, *research idea exploration*, and *milestone paper selection* – following

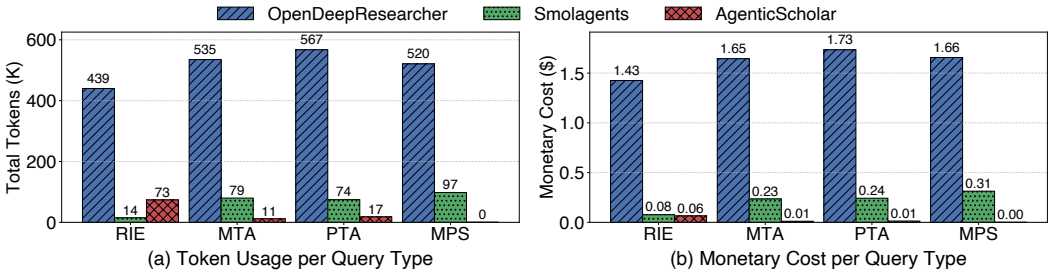


Fig. 8. The comparison of token and monetary cost on Tier-3 Queries (RIE: Research Idea Exploration; MTA: Method-oriented Trend Analysis; PTA: Problem-oriented Trend Analysis; MPS: Milestone Paper Selection).

a unified template that specifies the topic, query type, and output format. Query examples and prompt templates are provided in Appendix J in [22].

Baselines. We compare AgenticScholar against three closed-source systems – Elicit [21], Tongyi Deep Research [60], and Gemini Deep Research [59] – and two open-source systems, SmolAgent [68] and Open Deep Researcher [54]. We also evaluate two variants of AgenticScholar: AgenticScholar$_{noC}$, which adopts LLM-generated reference taxonomies, and AgenticScholar$_{noR}$, which derives taxonomies solely from the corpus. For fairness, all open-source baselines and AgenticScholar use GPT-5 as the backbone LLM.

Evaluation Metrics. We use the *LLM-as-a-judge protocol* [27, 40, 58], with GPT-5 serving as an impartial reviewer to score each method on specific dimensions (1–10 scale).

- **Research Trend Analysis:** (1) *Correctness* – factual accuracy, logical coherence, and temporal consistency; (2) *Relevance* – topical alignment with the target domain; (3) *Diversity* – coverage of distinct, non-overlapping trends; and (4) *Specificity* – concreteness and technical precision.
- **Research Idea Exploration:** (1) *Feasibility* – technical soundness and practical achievability; (2) *Novelty* – originality relative to existing research directions; and (3) *Impact* – potential significance for future studies.
- **Milestone Paper Selection:** (1) *Impact* – influence and contribution to the research community; (2) *Coverage* – representativeness across key subtopics; and (3) *Coherence* – consistency and thematic alignment within the selected set.

Additionally, for each query type, we report *efficiency*, measured as average inference time per query, as well as *token* and *monetary* costs for AgenticScholar and open-source baselines.

Results. We observe the following:

Effectiveness. As shown in Table 5, AgenticScholar achieves the best overall performance across nearly all dimensions.

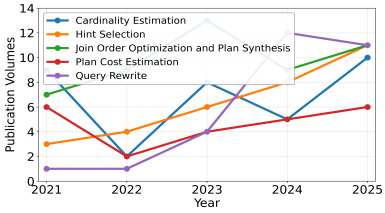
• *Research trend analysis.* AgenticScholar and its variants perform strongly across all evaluation dimensions. High *correctness* and *relevance* stem from its knowledge-grounded reasoning pipeline, which decomposes topics into structured subtopics and retrieves evidence via semantically anchored nodes (e.g., `Problem`, `Method`). They also achieve superior *specificity* and *diversity*, reflecting the robustness of its taxonomy in modeling hierarchical problem–method relations. Notably, Open Deep Researcher attains comparable – and occasionally higher – *relevance*, which we attribute to its strong online search capability for dynamically retrieving up-to-date sources.

• *Research idea exploration.* Consistent with trend analysis, the AgenticScholar family produces research ideas that are *novel*, *feasible*, and *impactful*, driven by its structured discovery of underexplored problem–method intersections. This grounding enables forward-looking ideas while avoiding the speculative outputs common in baselines. Notably, AgenticScholar$_{noR}$ achieves slightly higher *Novelty*, suggesting that removing reference-guided constraints reduces anchoring bias and

Table 6. Breakdown of token cost on Tier-3 queries.

Query type	Planning[1]	Retrieval	Reasoning	Total Tokens
Research Idea Exploration	410	0	73,245	73,519
Method Trend Analysis	410	0	11,345	11,649
Problem Trend Analysis	410	0	17,463	17,935
Milestone Paper Selection	410	0	0	372

[1] As these queries directly match plans in our predefined plan library, their token cost in the planning stage remains fixed.



(a) Research Trend

Problem Summary

Paper ID	Description
LearnedRewrite	**Input:** A SQL query, a set of rewrite rules, and a cost estimator. **Output:** An optimized rewritten SQL query with minimal execution cost.
...	...

Method Summary

LearnedRewrite	Description	Comments
Policy Tree Search	Models all possible query rewrite orders as a tree structure, where nodes represent rewritten queries. Uses to explore promising rewrite.	• Effectively explores a large search space of rewrite orders. • Relies on accurate benefit estimation, which may be challenging
...	...	...

(b) Problem and Method Summaries for Milestone Papers

1. Main Idea
Enable optimizers to incrementally adapt rewrite strategies based on live query execution feedback.

2. Innovation
Creates a self-improving query optimizer that adapts to real-world usage patterns and data drift.

3. Key Challenge
Handling noisy feedback and ensuring semantic equivalence during online updates.

Score
90

Research Problem	Reinforcement Learning	Continue/Online Learning	Tree Search w/ Learned Guidance	...
Query Rewrite	Highly Promising	Highly Promising	Explored (*LearnedRewrite*)	...
Hint Selection	Explored (*Bao*)	Moderately Promising	Highly Promising	...

(c) Explored Research Ideas

Fig. 9. AgenticScholar's results for case study on learning-based query optimization.

encourages broader hypothesis generation. However, this increased exploration comes at the cost of lower *Impact* and *Feasibility*.

• *Milestone paper selection.* AgenticScholar and its variants remain highly competitive, achieving strong *Impact* scores. Their relatively lower *Coverage* and *Coherence* compared to Gemini Deep Research and Open Deep Researcher. This may stem from the latter's web search capabilities, which enable broader and more up-to-date evidence retrieval.

Efficiency. AgenticScholar consistently delivers strong efficiency gains across Tier-3 tasks, with substantial speedups on *Milestone Paper Selection*. The only exception is *Research Idea Exploration*, where it is slightly slower than Open Deep Researcher, likely due to evaluating a larger space of problem–method combinations for verification and scoring. Overall, these results indicate that structure-guided planning and targeted retrieval reduce unnecessary browsing and stabilize tool-call overhead, with the largest benefits emerging in focused selection tasks.

Token and Monetary Costs. Figure 8 compares token and monetary costs against open-source baselines, with all systems using a unified GPT-5 backbone. Owing to its internal knowledge layer, AgenticScholar minimizes token consumption across most query types, except research idea exploration. This efficiency stems from its lightweight LLM-based planner and reliance on internal knowledge for search and reasoning, which substantially reduces token overhead in the planning and retrieval stage. For instance, the average token usage for Open Deep Researcher and SmolAgent during planning and retrieval is 237,057 and 12,056, respectively, whereas AgenticScholar consumes

only 355 tokens on average. Table 6 further provides a stage-level breakdown, AgenticScholar consistently maintains minimal token consumption in planning and retrieval

6.4 Case Study on Learning-based Query Optimization

To illustrate the end-to-end capabilities of AgenticScholar in a realistic research setting, we present a case study simulating a typical researcher's workflow. The task is to explore the rapidly evolving area of *learning-based query optimization* and identify underexplored research directions – an objective that challenges existing systems due to multi-step semantic reasoning over large scholarly corpora. The case study proceeds as a four-step progressive query session, demonstrating how AgenticScholar enables iterative intent refinement and reasoning toward novel research ideas.

Step 1: Understanding the Research Landscape (Tier 3). The researcher begins with an open-ended question to gain a high-level overview of the field and issues the query: *Q1: "What are the overall research trends in applying learning-based methods to query optimization since 2021?"* As shown in Figure 9(a), AgenticScholar generates a trend chart depicting publication volumes over time across key sub-topics. This goes beyond simple keyword search – AgenticScholar's *Knowledge Representation Layer* has already organized the corpus into a hierarchical taxonomy of research problems in query optimization. Its *Execution Layer* then applies `Traverse` and `Aggregate` operators to quantify and visualize the evolution of these research focus areas, allowing users to understand how interest shifts across sub-topics.

Step 2: Identifying Foundational Studies Across Key Sub-fields (Tier 3). The trend analysis from Step 1 provides a crucial insight: a clear upward trend in *query rewrite* and *hint selection*, indicating that the community is actively exploring these techniques to build more robust query plans under estimation errors. To further understand this development, the researcher decides to investigate foundational studies across these two areas and issue the query *"Can you identify three milestone papers for 'query rewrite' and 'hint selection'?"* AgenticScholar returns the three most influential papers for each of the two requested topics: *'learned query rewriting'*: SIA [85], LLM-R^2 [41], and LearnedRewrite [86]; *'optimizer hint selection'*: Bao [47], Steering [50], and COOOL [81]. This highlights AgenticScholar's ability to conduct fine-grained, taxonomy-guided knowledge discovery that identifies representative contributions within specific sub-fields.

Step 3: Summarizing Core Problems and Methods (Tier 2). The researcher seeks a concise overview of each group of milestone papers to understand the central problems and methodological paradigms. The researcher issues the query: *"Could you summarize the core problems and methods for each of these three groups of papers?"* As shown in Figure 9(b), AgenticScholar produces structured summaries for each paper (e.g., LearnedRewrite), distilling both the problem formulation and proposed method. This step demonstrates how AgenticScholar's `Extract` and `Summarize` operators transform unstructured scholarly text into interpretable, structured insights – laying the foundation for higher-level reasoning.

Step 4: Identifying Novel Research Opportunities (Tier 3). With a clear understanding of the milestone papers' problems and methods, the researcher is now ready to identify new research directions and issues the query: *"Okay, what would be some promising research ideas or directions to explore in query rewriting and hint selection?"*. As shown in Figure 9(c), AgenticScholar traverses its knowledge graph to generate problem–method combinations representing underexplored. It then ranks these ideas to provide concise descriptions of the most promising ideas for future research.

6.5 Validating Architectural Design

6.5.1 The Effectiveness of Reference-enhanced Taxonomy Construction. To assess the quality of taxonomies produced by our reference-enhanced construction method, we follow [87] and build a

Table 7. Level-wise taxonomy evaluation.

Level	Methods	Node quality (soft) Pre.	Rec.	F1.	Edge quality (soft) Pre.	Rec.	F1.
1	TaxoAdapt	0.632	0.561	0.594	0.498	0.421	0.456
	CHTaxo	0.661	0.598	0.628	0.523	0.462	0.491
	AgenticScholar$_{noR}$	0.694	0.640	0.677	0.562	0.504	0.531
	AgenticScholar$_{noC}$	0.688	0.676	0.682	**0.586**	0.512	0.547
	AgenticScholar	**0.718**	**0.684**	**0.703**	0.574	**0.538**	**0.555**
2	TaxoAdapt	0.598	0.532	0.563	0.458	0.392	0.422
	CHTaxo	0.620	0.558	0.587	0.487	0.424	0.453
	AgenticScholar$_{noR}$	0.673	0.587	0.628	0.515	0.456	0.484
	AgenticScholar$_{noC}$	0.649	0.632	0.640	0.541	0.466	0.501
	AgenticScholar	**0.684**	**0.642**	**0.657**	**0.549**	**0.487**	**0.507**
3	TaxoAdapt	0.543	0.472	0.505	0.402	0.327	0.361
	CHTaxo	0.568	0.495	0.529	0.428	0.345	0.382
	AgenticScholar$_{noR}$	**0.621**	0.533	0.574	0.452	0.396	0.422
	AgenticScholar$_{noC}$	0.593	0.574	0.583	**0.477**	0.406	0.439
	AgenticScholar	0.608	**0.582**	**0.595**	0.466	**0.432**	**0.448**

Table 8. Effectiveness of Predefined Plan Selection.

Variant	Input Tokens (K)	Output Tokens (K)	Planning Time (s)
Enable	0.41	0.013	0.96
Disable	4.08	0.77	15.17

benchmark using gold-standard taxonomies manually curated from seven authoritative surveys published in ACM Computing Surveys[1] (see Appendix K.1.1 in [22] for details). As baselines, we include two state-of-the-art taxonomy construction methods, *TaxoAdapt* [33] and *CHTaxo* [87], along with two variants of our approach, AgenticScholar$_{noR}$ and AgenticScholar$_{noC}$. We evaluate taxonomy quality using *soft* Precision, Recall, and F1 [24], which account for semantic similarity between concept labels and partial structural alignment, and further assess structural quality via an LLM-as-judge protocol (See Appendix K.1.2 in [22] for details). As shown in Table 7, AgenticScholar consistently achieves the best overall performance across soft Precision, Recall, and F1 at nearly all taxonomy levels. These results indicate improvements in both concept identification (node quality) and hierarchical organization (edge quality), showing that anchoring taxonomies in corpus evidence while leveraging LLM knowledge yields more coherent and fine-grained structures.

6.5.2 *The Effectiveness of Hybrid Planning.*

Efficiency of Predefined Plan Selection We generate 50 Tier-2 queries which can hit our predefined plans pool on the *vector search* topic and compare two variants: enable predefined plan selection (Enable) and disable it (Disable). Table 8 shows the average *number of input/output tokens* and *planning latency* per query of each variant. We can observe: with predefined plan selection enabled, AgenticScholar reduces both input/output tokens and planning time by more than 90%, indicating that Predefined Plan Selection can effectively reduce token overhead and planning latency.

Robustness through Self-Correction We designed 16 complex Tier-2 queries on the *vector search* topic that genuinely reflect our daily research needs. We record the frequency and cumulative

[1]https://dl.acm.org/journal/csur

Table 9. Robustness through Self-Correction.

	0	1	2	3
Frequency	10	4	1	1
Cum. Pct.	62.5%	87.5%	93.75%	100%

Table 10. Impact of Performance Optimizations.

Variant	Input Tokens (K)	Output Tokens (K)	Time (s)
Enable All	7.7	0.9	9.3
W/O Cache	23.0	2.8	13.7
W/O Cache and Para. Proc.	22.8	2.8	23.3

percentage of different repair attempts in Table 9. We observe that in most cases ($\geq$ 60%), AgenticScholar is already able to generate perfect plans, and with a reasonable small self-correction overhead ($\leq$ 3 times).

6.5.3 The Impact of Execution Performance Optimizations. We generated 266 Tier-2 queries on the *learned indexes* topic and compare three variants: disable cache (W/O Cache), disable cache and parallel processing (W/O Cache and Para. Proc.), and enable all of them (Enable All). Table 10 shows the average *number of input/output tokens* and *latency* per query of each variant. We can observe: with caching and parallel processing enabled, AgenticScholar uses about one-third of the input tokens and achieves the lowest latency (9.3 s). Disabling caching increases token usage and latency, while removing both optimizations roughly doubles the runtime.

6.5.4 Impact of Backbone LLMs on End-to-End Performance. We study how different backbone LLMs affect the end-to-end performance of AgenticScholar. We instantiate AgenticScholar with eight representative backbones from different providers, including GPT-5 and GPT-5 mini [52, 53], Claude Sonnet 4.5 and Claude Opus 4.5 [3, 4], Gemini 3 Flash Preview and Gemini 3 Pro Preview [15, 16], and Grok-4.1 Fast variants without and with explicit reasoning (NR/R) [79]. For fair comparison, we fix the agentic pipeline, prompts, and all non-LLM components, varying only the backbone. Table 11 reports results for research idea exploration (see Appendix K.3 in [22] for research trend analysis). Following [36], we categorize backbone LLMs into Better, Average, and Worse based on relative performance (top, middle, and bottom 33%). We observe that: (1) backbone choice has limited impact on effectiveness but substantially influences efficiency and cost; (2) GPT-5 achieves the strongest effectiveness, followed by Gemini-3.0-Pro; (3) Grok-4.1 Fast variants deliver the best efficiency and lowest cost; and (4) among the evaluated backbones, Gemini-3.0-Flash offers the best accuracy–efficiency trade-off.

7 Lessons Learned and Future Work

Knowledge Graph Construction and Maintenance. Our experience shows that the quality of structured knowledge is pivotal to the success of AgenticScholar. While the automated pipeline for constructing knowledge graphs and taxonomies is effective, it remains prone to occasional errors in entity extraction, relation detection, and semantic labeling that can propagate through downstream components. Heavy reliance on large models also incurs substantial computational and financial cost. Future efforts can be made to incorporate human-in-the-loop validation and cost-efficient hybrid pipelines that combine lightweight NLP components with compact fine-tuned models.

Multi-Modal Data Extraction. Figures in scholarly documents are essential for high-quality scholarly synthesis, yet quantitative extraction remains a challenge. Recovering accurate numerical values from complex plots, such as spider charts or line charts, remains unreliable, even with SOTA multi-modal LLMs. This insight motivates the integration of specialized vision–language

Table 11. The impact of backbone LLMs for research idea exploration. The colors express relative values (worse than average, average, and better than average).

Metric	Token cost	Monetary cost	Efficiency	Novelty	Feasibility	Impact
GPT-5	73,245	$0.640	379.1s	8.22	7.92	8.42
GPT-5 Mini	66,531	$0.116	338.1s	7.60	7.36	7.66
Claude 4.5 Sonnet	71,017	$0.957	468.9s	7.98	7.66	8.02
Claude 4.5 Opus	77,001	$1.737	673.5s	8.12	7.78	8.18
Gemini-3.0 Flash	60,240	$0.160	305.2s	7.72	7.44	7.78
Gemini-3.0 Pro	74,094	$0.793	433.7s	8.36	7.84	8.34
Grok-4.1 Fast (NR)	54,429	$0.026	147.4s	7.28	7.14	7.34
Grok-4.1 Fast (R)	93,794	$0.046	233.6s	7.38	7.22	7.44
Average	71,294	$0.559	372.4s	7.83	7.55	7.90
Standard deviation	±16.6%	±107.2%	±43.0%	±5.1%	±4.0%	±5.2%

models and structured figure-parsing frameworks to improve numerical fidelity and support richer evidence-grounded analysis.

Model Selection and Monetary Cost in Complex Analysis. A key lesson is that scholarly queries often require reasoning across multiple papers and synthesizing experimental evidence. This creates an important trade-off among model efficiency, reliability, and cost-effectiveness. Lightweight models such as `GPT-4.1-nano` offer faster inference and lower monetary cost, making them attractive for cost-sensitive large-scale use, but they are also more prone to hallucinations than stronger models such as `GPT-5`, `Gemini 2.5 Pro`. Future work could therefore explore adaptive model selection strategies that dynamically choose models based on task complexity and the desired cost-quality balance. One promising direction is to decompose complex analysis into smaller subtasks that can be handled by lightweight, more cost-effective models, while reserving more powerful and expensive models for reasoning-intensive stages. Such a design could improve overall cost-effectiveness while maintaining a balance among accuracy, speed, and monetary cost.

Domain-Aware Planning and Reasoning. Finally, our study underscores the value of domain sensitivity in agentic reasoning. While AgenticScholar already generalizes across disciplines, domain-agnostic planning can miss field-specific reasoning conventions – such as performance-driven evaluation in computer science, proof-centric validation in physics, or evidence hierarchies in medicine. Future efforts will explore domain-aware planning that integrates domain-specific reasoning and analytical workflows, enabling results that are both precise and contextually aligned with expert reasoning practices.

8 Conclusion

In this paper, we have developed AgenticScholar, an agentic data management system for scholarly corpora. It unifies knowledge representation, hybrid query planning, and operator-based execution to support diverse retrieval, synthesis, and discovery tasks. To the best of our knowledge and evaluation, AgenticScholar is the first system to execute agentic reasoning over multi-modal scholarly data through DAG-based plans, offering a practical foundation for future research in agentic data management. While AgenticScholar demonstrates strong performance on several scholarly domains, our current evaluation focuses primarily on research papers from a limited set of fields in computer science. Supporting broader disciplines may require domain-aware planning and reasoning mechanisms to better capture field-specific structures and analytical conventions.

Acknowledgments

This project is supported in part by ARC FT240100832 and DP240101211.

References

[1] Anirudh Ajith, Mengzhou Xia, Alexis Chevalier, Tanya Goyal, Danqi Chen, and Tianyu Gao. 2024. Litsearch: A retrieval benchmark for scientific literature search. *arXiv preprint arXiv:2407.18940* (2024).

[2] Günes Aluç, David DeHaan, and Ivan T. Bowman. 2012. Parametric Plan Caching Using Density-Based Clustering. In *ICDE*. 402–413.

[3] Anthropic. 2025. Introducing Claude Sonnet 4.5. https://www.anthropic.com/news/claude-sonnet-4-5. Accessed: 2026-01-02.

[4] Anthropic. 2026. Claude Opus 4.5. https://www.anthropic.com/claude/opus. Accessed: 2026-01-02.

[5] Llama API. 2025. https://www.llama.com/products/llama-api/.

[6] Mistral API. 2025. https://docs.mistral.ai/capabilities/document/.

[7] OpenAI API. 2025. https://platform.openai.com/docs/models.

[8] Qwen API. 2025. https://qwen.ai/apiplatform.

[9] Javed A Aslam and Emine Yilmaz. 2005. A geometric interpretation and analysis of r-precision. In *Proceedings of the 14th ACM international conference on Information and knowledge management*. 664–671.

[10] Ai2 Asta. 2025. https://asta.allen.ai/chat

[11] Chengliang Chai, Jiajun Li, Yuhao Deng, Yuanhao Zhong, Ye Yuan, Guoren Wang, and Lei Cao. 2025. Doctopus: Budget-aware Structural Table Extraction from Unstructured Documents. *Proc. VLDB Endow.* 18, 11 (2025), 3695–3707.

[12] ChatPDF. 2025. https://www.chatpdf.com/

[13] Boqi Chen, Fandi Yi, and Dániel Varró. 2023. Prompting or Fine-tuning? A Comparative Study of Large Language Models for Taxonomy Construction. *2023 ACM/IEEE International Conference on Model Driven Engineering Languages and Systems Companion (MODELS-C)* (2023), 588–596.

[14] Jindong Chen, Ao Wang, Jiangjie Chen, Yanghua Xiao, Zhendong Chu, Jingping Liu, Jiaqing Liang, and Wei Wang. 2019. CN-Probase: A Data-Driven Approach for Large-Scale Chinese Taxonomy Construction. *2019 IEEE 35th International Conference on Data Engineering (ICDE)* (2019), 1706–1709.

[15] Google Cloud. 2025. Gemini 3 Flash (Preview) on Vertex AI. https://docs.cloud.google.com/vertex-ai/generative-ai/docs/models/gemini/3-flash. Accessed: 2026-01-02.

[16] Google Cloud. 2025. Gemini 3 Pro Preview (Vertex AI Model Garden). https://console.cloud.google.com/vertex-ai/publishers/google/model-garden/gemini-3-pro-preview. Accessed: 2026-01-02.

[17] Gordon V Cormack and Thomas R Lynam. 2006. Statistical precision of information retrieval evaluation. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*. 533–540.

[18] Matthijs Douze, Alexandr Guzhva, Chengqi Deng, Jeff Johnson, Gergely Szilvasy, Pierre-Emmanuel Mazaré, Maria Lomeli, Lucas Hosseini, and Hervé Jégou. 2024. The Faiss library. (2024). arXiv:2401.08281 [cs.LG]

[19] Kayhan Dursun, Carsten Binnig, Ugur Çetintemel, and Tim Kraska. 2017. Revisiting Reuse in Main Memory Database Systems. In *SIGMOD Conference*. ACM, 1275–1289.

[20] Darren Edge, Ha Trinh, Newman Cheng, Joshua Bradley, Alex Chao, Apurva Mody, Steven Truitt, Dasha Metropolitansky, Robert Osazuwa Ness, and Jonathan Larson. 2024. From local to global: A graph rag approach to query-focused summarization. *arXiv preprint arXiv:2404.16130* (2024).

[21] Elicit. 2025. https://elicit.com/

[22] Supplementary Materials for AgenticScholar. 2026. https://github.com/DataAutonomyLab/agenticscholar.

[23] Nadime Francis, Alastair Green, Paolo Guagliardo, Leonid Libkin, Tobias Lindaaker, Victor Marsault, Stefan Plantikow, Mats Rydberg, Petra Selmer, and Andrés Taylor. 2018. Cypher: An Evolving Query Language for Property Graphs. In *SIGMOD*. 1433–1445.

[24] Pasi Fränti and Radu Mariescu-Istodor. 2023. Soft precision and recall. *Pattern Recognition Letters* 167 (2023), 115–121.

[25] Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, Qianyu Guo, Meng Wang, and Haofen Wang. 2023. Retrieval-Augmented Generation for Large Language Models: A Survey. *CoRR* abs/2312.10997 (2023). https://doi.org/10.48550/arXiv.2312.10997

[26] Gemini. 2025. https://ai.google.dev/gemini-api/docs.

[27] Jiawei Gu, Xuhui Jiang, Zhichao Shi, Hexiang Tan, Xuehao Zhai, Chengjin Xu, Wei Li, Yinghan Shen, Shengjie Ma, Honghao Liu, et al. 2024. A survey on llm-as-a-judge. *arXiv preprint arXiv:2411.15594* (2024).

[28] Zirui Guo, Lianghao Xia, Yanhua Yu, Tu Ao, and Chao Huang. 2024. LightRAG: Simple and Fast Retrieval-Augmented Generation. *CoRR* abs/2410.05779 (2024).

[29] Yichen He, Guanhua Huang, Peiyuan Feng, Yuan Lin, Yuchen Zhang, Hang Li, and Weinan E. 2025. PaSa: An LLM Agent for Comprehensive Academic Paper Search. In *ACL (1)*. Association for Computational Linguistics, 11663–11679.

[30] Jiaxin Huang, Yiqing Xie, Yu Meng, Yunyi Zhang, and Jiawei Han. 2020. CoRel: Seed-Guided Topical Taxonomy Construction by Concept Learning and Relation Transferring. *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (2020), 1928–1936.

[31] Kalervo Järvelin and Jaana Kekäläinen. 2002. Cumulated gain-based evaluation of IR techniques. *ACM Trans. Inf. Syst.* 20, 4 (2002), 422–446.
[32] Zhengbao Jiang, Frank F. Xu, Luyu Gao, Zhiqing Sun, Qian Liu, Jane Dwivedi-Yu, Yiming Yang, Jamie Callan, and Graham Neubig. 2023. Active Retrieval Augmented Generation. In *EMNLP*. 7969–7992.
[33] Priyanka Kargupta, Nan Zhang, Yunyi Zhang, Rui Zhang, Prasenjit Mitra, and Jiawei Han. 2025. TaxoAdapt: Aligning LLM-Based Multidimensional Taxonomy Construction to Evolving Research Corpora. *arXiv preprint arXiv:2506.10737* (2025).
[34] Uri Katz, Mosh Levy, and Yoav Goldberg. 2024. Knowledge Navigator: LLM-guided Browsing Framework for Exploratory Search in Scientific Literature. In *Findings of EMNLP*. 8838–8855.
[35] LangChain. 2025. https://www.langchain.com/.
[36] Jiale Lao, Andreas Zimmerer, Olga Ovcharenko, Tianji Cong, Matthew Russo, Gerardo Vitagliano, Michael Cochez, Fatma Özcan, Gautam Gupta, Thibaud Hottelier, H. V. Jagadish, Kris Kissel, Sebastian Schelter, Andreas Kipf, and Immanuel Trummer. 2025. SemBench: A Benchmark for Semantic Query Processing Engines. *CoRR* abs/2511.01716 (2025). https://doi.org/10.48550/arXiv.2511.01716
[37] Viktor Leis, Andrey Gubichev, Atanas Mirchev, Peter A. Boncz, Alfons Kemper, and Thomas Neumann. 2015. How Good Are Query Optimizers, Really? *Proc. VLDB Endow.* 9, 3 (2015), 204–215.
[38] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems* 33 (2020), 9459–9474.
[39] Guoliang Li, Xuanhe Zhou, Ji Sun, Xiang Yu, Yue Han, Lianyuan Jin, Wenbo Li, Tianqing Wang, and Shifu Li. 2021. openGauss: An Autonomous Database System. *Proc. VLDB Endow.* 14, 12 (2021), 3028–3041.
[40] Haitao Li, Qian Dong, Junjie Chen, Huixue Su, Yujia Zhou, Qingyao Ai, Ziyi Ye, and Yiqun Liu. 2024. Llms-as-judges: a comprehensive survey on llm-based evaluation methods. *arXiv preprint arXiv:2412.05579* (2024).
[41] Zhaodonghui Li, Haitao Yuan, Huiming Wang, Gao Cong, and Lidong Bing. 2024. LLM-R2: A Large Language Model Enhanced Rule-based Rewrite System for Boosting Query Efficiency. *Proc. VLDB Endow.* 18, 1 (2024), 53–65.
[42] Yiming Lin, Madelon Hulsebos, Ruiying Ma, Shreya Shankar, Sepanta Zeighami, Aditya G. Parameswaran, and Eugene Wu. 2024. Towards Accurate and Efficient Document Analytics with Large Language Models. *CoRR* abs/2405.04674 (2024).
[43] Chunwei Liu, Matthew Russo, Michael Cafarella, Lei Cao, Peter Baile Chen, Zui Chen, Michael Franklin, Tim Kraska, Samuel Madden, Rana Shahout, et al. 2025. Palimpzest: Optimizing ai-powered analytics with declarative query processing. In *CIDR*. 2.
[44] Kyle Lo, Lucy Lu Wang, Mark Neumann, Rodney Kinney, and Daniel S. Weld. 2020. S2ORC: The Semantic Scholar Open Research Corpus. In *ACL*. 4969–4983.
[45] Yuyin Lu, Hegang Chen, Pengbo Mao, Yanghui Rao, Haoran Xie, Fu Lee Wang, and Qing Li. 2024. Self-supervised Topic Taxonomy Discovery in the Box Embedding Space. *Trans. Assoc. Comput. Linguistics* 12 (2024), 1401–1416.
[46] Haoran Luo, Guanting Chen, Yandan Zheng, Xiaobao Wu, Yikai Guo, Qika Lin, Yu Feng, Zemin Kuang, Meina Song, Yifan Zhu, et al. 2025. HyperGraphRAG: Retrieval-Augmented Generation with Hypergraph-Structured Knowledge Representation. *arXiv preprint arXiv:2503.21322* (2025).
[47] Ryan Marcus, Parimarjan Negi, Hongzi Mao, Nesime Tatbul, Mohammad Alizadeh, and Tim Kraska. 2021. Bao: Making Learned Query Optimization Practical. In *SIGMOD 2021*. ACM, 1275–1288.
[48] Songsong Mo, Yile Chen, Hao Wang, Gao Cong, and Zhifeng Bao. 2023. Lemo: A Cache-Enhanced Learned Optimizer for Concurrent Queries. *Proc. ACM Manag. Data* 1, 4 (2023), 247:1–247:26.
[49] MySQL. 2025. https://www.mysql.com/.
[50] Parimarjan Negi, Matteo Interlandi, Ryan Marcus, Mohammad Alizadeh, Tim Kraska, Marc T. Friedman, and Alekh Jindal. 2021. Steering Query Optimizers: A Practical Take on Big Data Workloads. In *SIGMOD 2021*. ACM, 2557–2569.
[51] Jianmo Ni, Chen Qu, Jing Lu, Zhuyun Dai, Gustavo Hernandez Abrego, Ji Ma, Vincent Y Zhao, Yi Luan, Keith B Hall, Ming-Wei Chang, et al. 2021. Large dual encoders are generalizable retrievers. *arXiv preprint arXiv:2112.07899* (2021).
[52] OpenAI. 2026. GPT-5 mini Model (OpenAI API Documentation). https://platform.openai.com/docs/models/gpt-5-mini. Accessed: 2026-01-02.
[53] OpenAI. 2026. GPT-5 Model (OpenAI API Documentation). https://platform.openai.com/docs/models/gpt-5. Accessed: 2026-01-02.
[54] OpenDeepResearcher. 2025. https://github.com/mshumer/OpenDeepResearcher.
[55] Liana Patel, Siddharth Jha, Melissa Z. Pan, Harshit Gupta, Parth Asawa, Carlos Guestrin, and Matei Zaharia. 2025. Semantic Operators and Their Optimization: Towards AI-Based Data Analytics with Accuracy Guarantees. *Proc. VLDB Endow.* 18, 11 (2025), 4171–4184.
[56] PostgreSQL. 2025. https://www.postgresql.org/.

[57] Jason Priem, Heather A. Piwowar, and Richard Orr. 2022. OpenAlex: A fully-open index of scholarly works, authors, venues, institutions, and concepts. *CoRR* abs/2205.01833 (2022). https://doi.org/10.48550/arXiv.2205.01833
[58] Vyas Raina, Adian Liusie, and Mark Gales. 2024. Is LLM-as-a-Judge Robust? Investigating Universal Adversarial Attacks on Zero-shot LLM Assessment. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*. 7499–7517.
[59] Gemini Deep Research. 2025. https://gemini.google/overview/deep-research/.
[60] Tongyi Deep Research. 2025. https://github.com/Alibaba-NLP/DeepResearch.
[61] Stephen Robertson, Hugo Zaragoza, et al. 2009. The probabilistic relevance framework: BM25 and beyond. *Foundations and Trends® in Information Retrieval* 3, 4 (2009), 333–389.
[62] Matthew Russo, Sivaprasad Sudhir, Gerardo Vitagliano, Chunwei Liu, Tim Kraska, Samuel Madden, and Michael J. Cafarella. 2025. Abacus: A Cost-Based Optimizer for Semantic Operator Systems. *CoRR* abs/2505.14661 (2025).
[63] Keshav Santhanam, Omar Khattab, Jon Saad-Falcon, Christopher Potts, and Matei Zaharia. 2022. ColBERTv2: Effective and Efficient Retrieval via Lightweight Late Interaction. In *NAACL-HLT*. Association for Computational Linguistics, 3715–3734.
[64] Patricia G. Selinger, Morton M. Astrahan, Donald D. Chamberlin, Raymond A. Lorie, and Thomas G. Price. 1979. Access Path Selection in a Relational Database Management System. In *SIGMOD*. 23–34.
[65] Chao Shang, Sarthak Dash, Md. Faisal Mahbub Chowdhury, Nandana Mihindukulasooriya, and A. Gliozzo. 2020. Taxonomy Construction of Unseen Domains via Graph-based Cross-Domain Knowledge Transfer. In *Annual Meeting of the Association for Computational Linguistics*. 2701–2709.
[66] Shreya Shankar, Tristan Chambers, Tarak Shah, Aditya G. Parameswaran, and Eugene Wu. 2025. DocETL: Agentic Query Rewriting and Evaluation for Complex Document Processing. *Proc. VLDB Endow.* 18, 9 (2025), 3035–3048.
[67] Jiaming Shen, Zeqiu Wu, Dongming Lei, Chao Zhang, Xiang Ren, Michelle T. Vanni, Brian M. Sadler, and Jiawei Han. 2018. HiExpan: Task-Guided Taxonomy Construction by Hierarchical Tree Expansion. *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (2018), 2180–2189.
[68] Smolagents. 2025. https://smolagents.org/.
[69] Hongjin Su, Weijia Shi, Jungo Kasai, Yizhong Wang, Yushi Hu, Mari Ostendorf, Wen-tau Yih, Noah A Smith, Luke Zettlemoyer, and Tao Yu. 2022. One embedder, any task: Instruction-finetuned text embeddings. *arXiv preprint arXiv:2212.09741* (2022).
[70] Ji Sun, Guoliang Li, Peiyao Zhou, Yihui Ma, Jingzhe Xu, and Yuan Li. 2025. AgenticData: An Agentic Data Analytics System for Heterogeneous Data. *CoRR* abs/2508.05002 (2025).
[71] Yushi Sun, Hao Xin, Kai Sun, Yifan Ethan Xu, Xiao Yang, Xin Luna Dong, Nan Tang, and Lei Chen. 2024. Are Large Language Models a Good Replacement of Taxonomies? *arXiv preprint arXiv:2406.11131* (2024).
[72] Zhaoze Sun, Chengliang Chai, Qiyan Deng, Kaisen Jin, Xinyu Guo, Han Han, Ye Yuan, Guoren Wang, and Lei Cao. 2025. QUEST: Query Optimization in Unstructured Document Analysis. *Proc. VLDB Endow.* 18, 11 (2025), 4560–4573.
[73] Henrique Schechter Vera, Sahil Dua, Biao Zhang, Daniel Salz, Ryan Mullins, Sindhu Raghuram Panyam, Sara Smoot, Iftekhar Naim, Joe Zou, Feiyang Chen, et al. 2025. Embeddinggemma: Powerful and lightweight text representations. *arXiv preprint arXiv:2509.20354* (2025).
[74] Huaiyu Wan, Yutao Zhang, Jing Zhang, and Jie Tang. 2019. AMiner: Search and Mining of Academic Social Networks. *Data Intell.* 1, 1 (2019), 58–76.
[75] Jiayi Wang and Jianhua Feng. 2025. Unify: An Unstructured Data Analytics System. In *ICDE*. 4662–4674.
[76] Kuansan Wang, Zhihong Shen, Chiyuan Huang, Chieh-Han Wu, Yuxiao Dong, and Anshul Kanakia. 2020. Microsoft Academic Graph: When experts are not enough. *Quant. Sci. Stud.* 1, 1 (2020), 396–413.
[77] Liang Wang, Nan Yang, Xiaolong Huang, Binxing Jiao, Linjun Yang, Daxin Jiang, Rangan Majumder, and Furu Wei. 2022. Text embeddings by weakly-supervised contrastive pre-training. *arXiv preprint arXiv:2212.03533* (2022).
[78] Yujing Wang, Hainan Zhang, Liang Pang, Binghui Guo, Hongwei Zheng, and Zhiming Zheng. 2025. MaFeRw: Query Rewriting with Multi-Aspect Feedbacks for Retrieval-Augmented Large Language Models. In *AAAI*. 25434–25442.
[79] xAI. 2025. Grok 4.1 Fast and Agent Tools API. https://x.ai/news/grok-4-1-fast. Accessed: 2026-01-02.
[80] Shitao Xiao, Zheng Liu, Peitian Zhang, and Niklas Muennighoff. 2023. C-Pack: Packaged Resources To Advance General Chinese Embedding. *CoRR* abs/2309.07597 (2023).
[81] Xianghong Xu, Zhibing Zhao, Tieying Zhang, Rong Kang, Luming Sun, and Jianjun Chen. 2023. COOOL: A Learning-To-Rank Approach for SQL Hint Recommendations. In *VLDB 2023 (CEUR Workshop Proceedings, Vol. 3462)*.
[82] Qingkai Zeng, Yuyang Bai, Zhaoxuan Tan, Shangbin Feng, Zhenwen Liang, Zhihan Zhang, and Meng Jiang. 2024. Chain-of-Layer: Iteratively Prompting Large Language Models for Taxonomy Induction from Limited Examples. *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management* (2024), 3093–3102.
[83] Chao Zhang, Fangbo Tao, Xiusi Chen, Jiaming Shen, Meng Jiang, Brian M. Sadler, Michelle T. Vanni, and Jiawei Han. 2018. TaxoGen: Unsupervised Topic Taxonomy Construction by Adaptive Term Embedding and Clustering. *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (2018).

[84] 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. 2025. Qwen3 Embedding: Advancing Text Embedding and Reranking Through Foundation Models. *CoRR* abs/2506.05176 (2025).

[85] Qi Zhou, Joy Arulraj, Shamkant B. Navathe, William Harris, and Jinpeng Wu. 2021. SIA: Optimizing Queries using Learned Predicates. In *SIGMOD 2021*, Guoliang Li, Zhanhuai Li, Stratos Idreos, and Divesh Srivastava (Eds.). ACM, 2169–2181.

[86] Xuanhe Zhou, Guoliang Li, Chengliang Chai, and Jianhua Feng. 2021. A Learned Query Rewrite System using Monte Carlo Tree Search. *Proc. VLDB Endow.* 15, 1 (2021), 46–58.

[87] Kun Zhu, Lizi Liao, Yuxuan Gu, Lei Huang, Xiaocheng Feng, and Bing Qin. 2025. Context-Aware Hierarchical Taxonomy Generation for Scientific Papers via LLM-Guided Multi-Aspect Clustering. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*. 15627–15645.

Received October 2025; revised January 2026; accepted February 2026