DOI: https://doi.org/10.15276/hait.09.2026.34
UDC 004.89

Evidence-grounded benchmarking of retrieval-augmented generation architectures for scientific claim verification

Oleksii I. Sheremet [1)]
ORCID: https://orcid.org/0000-0003-1298-3617; sheremet-oleksii@ukr.net. Scopus Author ID: 57170410800
Stefan Palis[2)]
ORCID: https://orcid.org/0000-0002-1941-2289; stefan.palis@tu-clausthal.de. Scopus Author ID: 24766914900
Oleksandr V. Sadovoi[3)]
ORCID: https://orcid.org/0000-0001-9739-3661; sadovoyav@ukr.net. Scopus Author ID: 57205432765
Sergey V. Podlesny[1)]
ORCID: https://orcid.org/0000-0001-8271-4004; spodlesny2318@gmail.com. Scopus Author ID: 56612000700
Kateryna S. Sheremet[1)]
ORCID: https://orcid.org/0000-0003-3783-5274; artks@ukr.net. Scopus Author ID: 57207768511
[1)] Donbas State Engineering Academy, 39, Mashinobudivnykiv Blvd. Kramatorsk, 84313, Ukraine
[2)] Clausthal University of Technology, 28, Leibnizstrasse. Clausthal-Zellerfeld, 38678, Germany
[3)] Dniprovsky State Technical University, 2, Dniprobudivska Str. Kamyanske, 51918, Ukraine

ABSTRACT

Relevance. Retrieval-augmented generation is usually evaluated with proxy metrics that ignore evidence grounding, and new architectures are rarely compared with strong baselines under equal conditions. **The purpose of the article** is to determine, under one protocol, whether agentic, graph-based, and corrective architectures improve evidence retrieval and verdict accuracy over lexical, dense, hybrid, and reranking baselines in scientific claim verification on abstracts, and at what cost. **Tasks:** formalise the protocol as an algorithm; implement eight pipelines on one sentence-level index of a public benchmark of biomedical abstracts with annotated evidence; evaluate them on a fixed subset with repeated runs and on the development split; quantify cost. **Methods:** shared embedding model, generation model, and output contract; ranking, evidence, and per-class verdict metrics with bootstrap intervals and paired tests corrected for multiple comparisons. **Scientific novelty:** an evidence-grounded protocol comparing seven pipeline families on one index and generator with statistical tests and cost. **Practical significance:** released code, identifiers, and prompts make the protocol reusable. **Results:** corrective and agentic pipelines ranked evidence best and cited annotated rationales most completely, but no difference from the dense baseline remained significant after correction; the lexical baseline was significantly worse; hybrid fusion and reranking did not help; verdict accuracy varied little with the retriever and was dominated by the generator's decision bias; the agentic pipeline needed two and a half times the latency and three times the calls of the baseline. **Conclusions:** gains are small, confined to sentence-level evidence ranking, and cost two to three times more calls.

Keywords: Retrieval-augmented generation; scientific claim verification; evidence-grounded evaluation; agentic retrieval; knowledge graph indexing; benchmarking

For citation: Sheremet O. I., Palis Stefan, Sadovoi O. V., Podlesny S. V., Sheremet K. S. "Evidence-grounded benchmarking of retrieval-augmented generation architectures for scientific claim verification". *Herald of Advanced Information Technology*. 2026; Vol.9 No.4: 539–566. DOI: https://doi.org/10.15276/hait.09.2026.34

INTRODUCTION

Large language models (LLMs) are increasingly used for knowledge-intensive tasks [1], but remain prone to hallucinations and outdated knowledge [2], [3]. Scientific and technical applications require answers grounded in verifiable primary sources [2]. Risk-based assessments of LLMs likewise identify ungrounded generation as a principal failure mode [4]. Retrieval-augmented generation (RAG) addresses these limitations by retrieving relevant passages from an external corpus and conditioning generation on this evidence [5], [6].

However, single-pass RAG can retrieve topically related but irrelevant passages, depend on query wording, and miss decisive evidence within a relevant document [7]. For scientific abstracts with sentence-level annotations, the challenge is often to rank the right sentences rather than find the right document [8].

Recent RAG architectures extend this basic pipeline with graph-based retrieval, retrieval-quality assessment, and iterative orchestration. GraphRAG uses corpus graphs and hierarchical community summaries for global questions [9]. LightRAG reduces graph-retrieval overhead through dual-level retrieval and efficient indexing [10]. Corrective RAG evaluates retrieved evidence and triggers corrective actions when it is insufficient [11]. Self-RAG learns when to retrieve and how to assess evidence support during generation [12]. Agentic

© Sheremet O., Palis Stefan., Sadovoi O., Podlesny S., Sheremet K., 2026

This is an open access article under the CC BY license (https://creativecommons.org/licenses/by/4.0/deed.uk)

RAG uses planning, query reformulation, and tools across multiple steps, trading higher latency and more model calls for potential accuracy gains [13], [14].

However, these approaches are usually evaluated under different experimental conditions and often against only a single dense baseline. It therefore remains unclear whether their added complexity provides meaningful gains over strong conventional retrieval methods. We compare retrieval quality, evidence recovery, verdict accuracy, and online cost under one shared index, generator, and output contract.

THE PURPOSE OF THE ARTICLE

The purpose of the article is to determine whether *agentic*, graph-based, and corrective RAG improve sentence-level evidence retrieval and verdict accuracy over sparse, dense, hybrid, and *reranking* baselines, and at what computational cost.

To achieve this purpose, the following tasks are solved:

– to formalise the evidence-grounded benchmarking protocol as an algorithm that fixes the corpus, the index, the generation model, and the output contract and varies only the retrieval and orchestration logic;

– to implement eight comparable pipelines on one sentence-level index: BM25, dense, hybrid, and reranking baselines, and corpus-restricted instantiations of Agentic RAG, GraphRAG, LightRAG, and Corrective RAG;

– to evaluate retrieval quality at the document and sentence levels (Precision@5, Recall@5, nDCG@5, MRR), evidence quality (precision, recall, and F1 of the cited sentences against gold rationales) and verdict quality (accuracy, per-class precision, recall and F1, macro-F1, balanced accuracy, confusion matrices) with bootstrap confidence intervals, paired significance tests, and repeated runs;

– to quantify per-query latency, model calls, and token usage separately from offline indexing cost, and to test robustness on the full *SciFact* development split, including claims with insufficient evidence.

The study focuses on local claim-verification queries over scientific abstracts. Global corpus-level questions and full-text documents are outside its scope. Conclusions apply to the implementations and configurations evaluated here.

The novelty of the work is a unified evidence-grounded benchmark. All pipelines use the same index and generator. This enables statistically tested comparisons of retrieval quality, evidence grounding, verdict accuracy, and computational cost under controlled conditions.

LITERATURE REVIEW

Foundations of Retrieval-Augmented Generation. The classical RAG pipeline combines a retriever that selects relevant passages from a corpus and a generator that produces an answer conditioned on the retrieved context [5]. This architecture was originally motivated by the need to ground generation in external evidence for knowledge-intensive tasks and to improve factuality relative to purely parametric generation [5]. In production practice, the retriever is typically implemented as dense vector search over embeddings (often with approximate nearest-neighbor indexing), followed by optional reranking; the generator is an LLM that consumes the query and retrieved passages within a limited context window. The overall structure of this classic retrieve-then-generate pipeline is shown in Fig. 1.

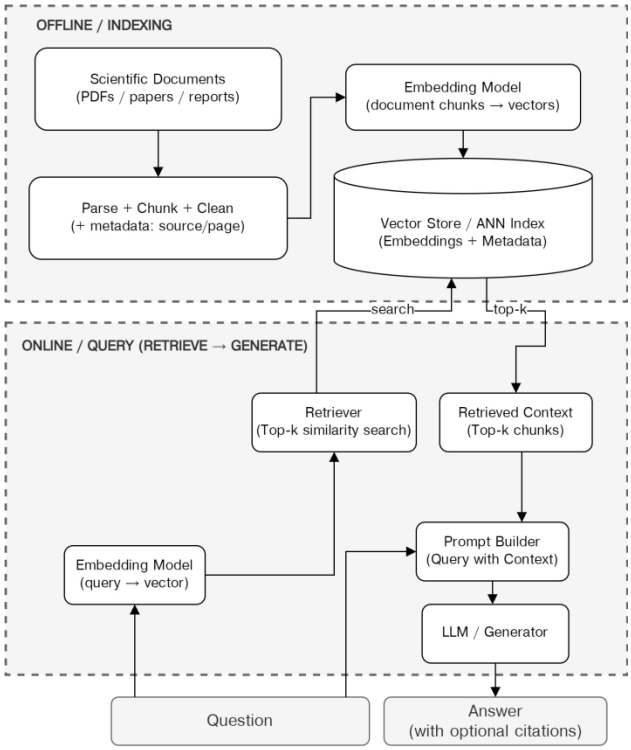


Fig. 1. **Classic RAG Workflow**
Source: **compiled by the authors**

Surveys emphasize that the RAG design space has expanded beyond the naive retrieve-then-generate template into “advanced” and “modular” forms [7]. Advanced RAG typically adds pre-retrieval methods (query rewriting/expansion, decomposition into sub-questions), retrieval-time

ISSN 2663-0176 (Print)
ISSN 2663-7731 (Online)

enhancements (hybrid lexical with dense search, reranking), and post-retrieval processing (filtering, compression, and contextual deduplication) to mitigate irrelevant retrieval and context-window constraints [7]. Modular RAG generalizes this idea by allowing interchangeable components and conditional routing (e.g., selective retrieval or specialized retrievers for entities vs. long-form explanations) [7].

A key enabler for modern RAG is the rapid progress in dense retrieval, embedding models, evaluation benchmarks, and vector database systems. Dense Passage Retrieval established the effectiveness of dual-encoder architectures for retrieving relevant passages through learned dense representations [15]. Subsequent surveys systematized dense retrieval methods based on pretrained language models, including representation learning, negative sampling, training objectives, and reranking strategies [16]. The BEIR benchmark further demonstrated the importance of heterogeneous, zero-shot evaluation across retrieval tasks and domains [17]. At the infrastructure level, vector database management systems provide specialized storage, indexing, and similarity-search mechanisms for high-dimensional embeddings [18]. These developments are particularly relevant for scientific corpora, where retrieval quality can be sensitive to chunk boundaries, mathematical notation, and discourse structure.

Reliability Challenges and Hallucination in Knowledge-Intensive QA. Despite RAG's grounding mechanism, hallucinations persist due to retrieval failures (missing key evidence), noisy or irrelevant context, and generator behaviour that extrapolates beyond the available evidence. Recent surveys propose more precise definitions and taxonomies of hallucination, including factual inconsistency, lack of faithfulness to the provided evidence and internally contradictory generation [2]. Empirical evaluation studies further note that hallucination rates and robustness vary substantially across models and setups [3], that comparative benchmarking of contemporary models remains necessary even for well-defined tasks [19], [20], and that retrieval quality is a major driver of downstream answer correctness in QA pipelines [3].

These findings motivate two broad lines of RAG research. The first focuses on improving retrieval (better queries, reranking, multi-step evidence gathering). The second focuses on controlling generation via evidence checking, critique, and corrective loops. Importantly, scientific-document QA intensifies these issues: correct answers may require combining evidence across multiple papers, resolving conflicting claims, or distinguishing "background discussion" from "main contribution," which can defeat single-chunk similarity search.

Knowledge Graphs and Graph-Augmented Reasoning for RAG. Knowledge graphs (KGs) explicitly represent entities and relations, providing a structured substrate for reasoning and multi-hop traversal. General surveys describe their representation, acquisition, and application principles [21], while more specialised reviews examine knowledge-graph reasoning across static, dynamic, and multimodal graph types [22] and the validation of automatically constructed graphs [23]. Semantic-network data models are also used to organise the components of complex technical systems for retrieval [24]. Recent roadmaps highlight the growing convergence of LLMs and KGs, arguing that hybrid systems can combine language-model capabilities with graph-based structure to improve grounding, interpretability, and multi-hop inference [25].

For document collections, a major question is how to build a graph representation that is useful for retrieval. One practical approach is to construct document knowledge graphs from extracted entities and relations, enabling retrieval to incorporate graph neighborhoods rather than only text similarity. Knowledge-graph prompting has been shown to support multi-document question answering by explicitly modeling relations among entities and evidence distributed across documents [26]. This idea is particularly relevant for scientific abstract corpora, where entities such as methods, datasets, and tasks recur across documents and their relationships are essential for global questions.

GraphRAG and Related "Local-to-Global" Retrieval. GraphRAG operationalizes a local-to-global pipeline in which corpus-level abstractions are precomputed offline and then used for global (corpus-wide) question answering at query time [9]. As illustrated in Fig. 2, the indexing stage begins with document parsing, chunking, and cleaning, after which an LLM-based information extraction step produces entities and relations that are assembled into a corpus entity graph.

The graph is then clustered into communities, and the system generates hierarchical community summaries ("summary-of-summaries") that are stored for retrieval [9]. During online inference, a query router first detects whether the user intent is global; for global questions, a global retriever selects the most relevant communities using summary-

and/or graph-derived signals, retrieves the corresponding community summaries (higher-level abstractions rather than raw chunks), and applies a map–reduce generation pattern: the LLM produces partial answers per community (map) and then synthesizes a single reconciled response (reduce) [9].

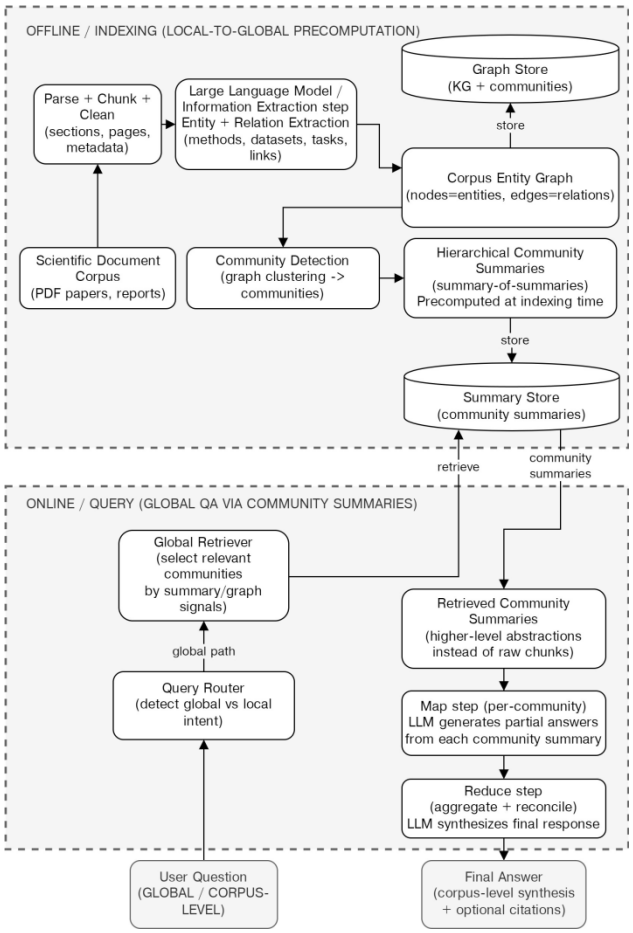


Fig. 2. **GraphRAG: Local-to-Global Workflow**
Source: **compiled by the authors**

The core rationale is that global questions are frequently underdetermined by any single passage, so retrieval should operate over community-level representations instead of relying only on nearest-neighbor chunk similarity [9]. Follow-up surveys and domain-specific evaluations indicate that graph-based retrieval can provide advantages in multi-document and multi-hop settings, although the magnitude of these gains depends on query type, corpus structure, and implementation details [27], [28].

A key limitation, also reflected by the substantial offline block in Fig. 2, is computational overhead: entity and relation extraction, community detection, and multi-level summarization introduce additional indexing-time costs compared with embedding-only pipelines. This makes GraphRAG particularly suitable for relatively stable corpora but potentially expensive for rapidly changing knowledge bases [9].

LightRAG and Efficiency-Driven Graph Retrieval. LightRAG is motivated directly by the cost barrier of GraphRAG. It proposes an efficient dual-level retrieval strategy that combines low-level, entity-oriented retrieval with a higher-level, relation-oriented graph representation, aiming to preserve much of the global-query benefit while reducing indexing and inference overhead [10]. It also emphasises more practical update behaviour (incremental integration of new information) relative to full reprocessing pipelines [10]. For scientific corpora that grow continuously, incremental indexing is a practical requirement, making efficiency-driven graph RAG approaches particularly relevant.

Corrective and Self-Reflective RAG. Corrective RAG addresses a simple but critical failure mode: even with strong embeddings, the retriever sometimes returns low-quality context and the generator proceeds anyway. CRAG introduces a retrieval evaluator that scores the retrieved evidence and triggers corrective actions (e.g., expanded retrieval or alternative search) when evidence quality is low, reducing hallucinations that originate from retrieval noise [11]. This line of work reframes RAG as a quality-controlled pipeline rather than a single-pass heuristic.

Self-RAG extends adaptivity further by training/learning signals that indicate when retrieval is needed, whether retrieved documents are relevant, and whether the generated answer is adequately supported by evidence [12]. Unlike always-retrieve pipelines, Self-RAG can reduce unnecessary retrieval for queries that are well covered by parametric knowledge while strengthening evidence grounding when retrieval is required [12]. In scientific QA, this is particularly useful because corpora contain both “facts” (e.g., dataset sizes) and “interpretations” (e.g., comparative claims), and the system must decide how much external evidence is necessary for a defensible answer.

Agentic RAG Architectures and Multi-Step Retrieval. Agentic RAG architectures treat retrieval and synthesis as a multi-step problem: the system may plan sub-questions, reformulate queries, run multiple retrieval calls, critique intermediate answers, and iterate until confidence is sufficient. Surveys on agentic AI characterize these systems as autonomous, goal-directed, and adaptive, often using patterns such as planning, reflection, tool use, and

memory [13], [14]. The expected benefit is higher accuracy on complex tasks; the cost is higher latency and compute due to iterative loops and additional model/tool calls.

For research-paper QA, agentic strategies are attractive because many queries are naturally multi-step (e.g., "Compare method A vs B across multiple papers and identify consistent findings"). However, rigorous evaluation remains difficult: improvements can depend on agent policies, stopping criteria, and tool configurations, which vary widely across implementations.

Summary. Across the above lines of work, a consistent theme emerges: newer RAG architectures trade higher complexity (graphs, evaluators, multi-step loops) for improved robustness on difficult queries. Yet many studies report results under incomparable conditions (different corpora, prompts, models, chunking, and indexing) and rarely against strong lexical, hybrid, or reranking baselines, making it hard to attribute gains to the architecture itself rather than to surrounding choices. Systematic RAG benchmarks have started to close this gap: MIRAGE compares retrievers and corpora for medical question answering and finds that the choice of corpus and retriever changes the outcome substantially [29]; the BERGEN library fixes retriever, reranker, generator, and prompt so that RAG variants can be compared in one configuration [30]; FlashRAG re-implements more than a dozen RAG methods in one toolkit and reports gains over the standard pipeline that vary by dataset rather than being uniform [31]; and a direct comparison of RAG and GraphRAG finds that graph indexing helps corpus-level and multi-hop questions but not detail-oriented, single-fact queries [32]. None of these evaluates evidence grounding at the sentence level or compares corrective, agentic, and graph-based orchestration on one index. Lexical retrieval is, moreover, a strong baseline on scientific abstracts: in BEIR, BM25 reaches nDCG@10 of 0.665 on SciFact and matches or outperforms the zero-shot dense retrievers evaluated there [17]. This motivates controlled benchmarking in which multiple architectures are tested under the same constraints on a corpus with human-annotated sentence-level evidence, such as SciFact [8], where the quality of evidence ranking can be measured directly.

TECHNICAL FOUNDATIONS OF INNOVATIVE RAG ARCHITECTURES

The architectural diversity observed across modern retrieval-augmented systems necessitates a rigorous technical examination of their underlying mechanisms. Each architecture embodies distinct design principles that manifest in specific algorithmic procedures, mathematical formulations, and computational trade-offs. This section provides formal specifications of the five *reference* architectures as described by their authors [9], [10], [11], [13], [14]; the *corpus-restricted instantiations* that were actually benchmarked, including the components that were disabled or simplified (web search, community summaries, tool libraries), are specified in the section Implementation Considerations.

A retrieval-augmented generation system operates over a retrieval corpus $D = \{d_1, d_2, \dots, d_n\}$, where each d_i denotes a retrievable unit, such as a document or a text chunk, and processes queries q from a query space Q. The fundamental objective can be formalized as finding an optimal response r^* that maximizes the conditional probability of generating a correct answer given both the query and the retrieved evidence:

$$r^* = \underset{r}{argmax}\, P\big(r \mid q, R(q,D)\big), \qquad (1)$$

where $R(q,D) \subseteq D$ denotes the retrieval function output, that is, the subset of retrieval units selected from corpus D for query q.

The retrieval function can be defined as selecting, among all candidate subsets containing no more than k elements, the subset with the highest cumulative relevance to the query:

$$R(q,D) = \underset{\substack{S \subseteq D \\ |S| \le k}}{argmax} \sum_{d_i \in S} \mathrm{rel}\,(q, d_i), \qquad (2)$$

where S is a candidate retrieval set, $\mathrm{rel}(q, d_i)$ denotes the relevance score assigned to retrieval unit d_i with respect to query q, and k is the maximum number of units that may be retrieved.

Traditional RAG implementations decompose this problem into sequential stages: document preprocessing, embedding computation, similarity-based retrieval, and conditioned generation. However, the innovative architectures examined in this study introduce additional computational stages that modify, extend, or iteratively refine this basic pipeline.

The baseline RAG architecture implements the canonical retrieve-then-generate paradigm through a deterministic pipeline. The technical specification encompasses four distinct processing phases.

Document preprocessing phase. The corpus undergoes chunking into fixed-size text segments with configurable overlap. Given a document d with token sequence $T = (t_1, t_2, \dots, t_m)$, the chunking

function is defined as

$$C(d) = \{c_i | a_i \leq m\}, \quad (3)$$

$$c_i = (t_{a_i}, \ldots, t_{\min(a_i+s-1,m)}), \quad (4)$$

$$a_i = 1 + (i-1)(s-o), \quad (5)$$

where s denotes the chunk size and o denotes the overlap, with $0 \leq o < s$. Thus, consecutive chunks are shifted by $s - o$ tokens, preserving contextual continuity across chunk boundaries.

Embedding Computation Phase. Each chunk c_i is transformed into a dense vector representation through an embedding function

$$E: C \to \mathbb{R}^d, \quad (6)$$

where C denotes the space of text chunks, d is the embedding dimensionality, and e_i is the vector representation of chunk c_i.

The embedding model produces vectors that capture semantic content:

$$e_i = E(c_i) \in \mathbb{R}^d. \quad (7)$$

The resulting embeddings are stored in a vector database with indexing structures optimized for approximate nearest neighbor (ANN) search, typically employing Hierarchical Navigable Small World (HNSW) graphs [33] or Inverted File (IVF) indices.

Retrieval Phase. At query time, the query q is embedded using the same embedding function, producing query vector $e_q = E(q)$.

Retrieval proceeds via cosine similarity computation:

$$sim(q, c_i) = (e_q \cdot e_i) / (||e_q|| \cdot ||e_i||). \quad (8)$$

The top-k chunks with highest similarity scores constitute the retrieved context:

$$R(q, D) = top\text{-}k\{c_i \mid sim(q, c_i)\}. \quad (9)$$

Generation Phase. Retrieved chunks are concatenated with the query in a prompt template and provided to the language model for response generation.

The generation probability follows:

$$P(r \mid q, R(q, D)) = \prod P(r_j \mid r_1, \ldots, r_{j-1}, q, R(q, D)), \quad (10)$$

where the product extends over all tokens in the response sequence.

The main components of the baseline RAG pipeline, together with their technical specifications and computational complexity characteristics, are summarized in Table 1.

Table 1. **Baseline RAG component specifications**

Component	Specification	Complexity
Chunking	Fixed-size with overlap	*O(n)* linear in corpus size
Embedding	Dense vector encoding	*O(n·d)* for *n* chunks
Indexing	HNSW or IVF structure	*O(n·log n)* construction
Retrieval	ANN search with cosine similarity	*O(log n)* per query
Generation	Autoregressive LLM	*O(L²)* for context length *L*

Source: **compiled by the authors**

GRAPHRAG

GraphRAG augments the retrieval pipeline with explicit knowledge graph construction and hierarchical community summarization. The architecture introduces substantial preprocessing overhead in exchange for enhanced capability on corpus-level queries [9].

Entity and Relation Extraction. The initial phase employs a language model to extract structured knowledge from document chunks.

For each chunk ci, the extraction function produces:

$$Extract(c_i) = \{(e_1, r, e_2) \mid e_1, e_2 \in Entities, r \in Relations\}, \quad (11)$$

where entities represent named entities, concepts, or significant terms, and relations capture semantic connections between entity pairs. The extraction prompt instructs the model to identify entities with their types (person, organization, concept, and event) and relations with descriptive labels.

Graph Construction. Extracted triples are consolidated into a unified knowledge graph $G = (V, E)$, where vertices V represent unique entities and edges E represent relations.

Entity resolution merges duplicate entities through string matching and semantic similarity:

$$merge(e_1, e_2) = True\ if\ sim_{semantic}(e_1, e_2) > \theta_{merge}. \quad (12)$$

The resulting graph captures cross-document relationships that are implicit in the original corpus but unavailable to vector-only retrieval.

Community Detection. GraphRAG applies the Leiden algorithm [34] to partition the entity graph into well-connected communities. To represent the effect of the resolution parameter, the modularity objective can be written as

ISSN 2663-0176 (Print)
ISSN 2663-7731 (Online)

$$Q = \frac{1}{2m}\sum_{i,j}\left(A_{ij} - \gamma\frac{k_i k_j}{2m}\right)\delta(g_i, g_j), \quad (13)$$

where A_{ij} is the corresponding element of the adjacency matrix, k_i and k_j are the degrees of vertices i and j, m is the total number of edges, γ is the resolution parameter, g_i and g_j denote the community assignments of vertices i and j, respectively, and $\delta(g_i, g_j)$ equals 1 when both vertices belong to the same community and 0 otherwise. Higher values of γ generally favor smaller and more numerous communities.

Hierarchical Summarization. For each community at each hierarchy level, a language model generates a textual summary capturing the key themes, entities, and relationships within that cluster.

The summarization function *S* produces:

$$S(Community_l, k) = \text{“Summary text describing community k at level l”}. \quad (14)$$

These summaries are stored alongside the graph structure, enabling retrieval at multiple granularity levels.

Dual-Mode Retrieval. GraphRAG supports two retrieval paradigms: local search and global search. Local Search operates when queries target specific entities or factual information.

The system identifies query-relevant entities through semantic matching, expands to their one-hop graph neighborhoods, and retrieves associated text chunks:

$$R_{local}(q) = \{\text{chunks associated with } N_1(\text{entities}(q))\}, \quad (15)$$

where N_1 denotes the one-hop neighborhood function.

Global Search activates for queries requiring corpus-wide synthesis. The system retrieves community summaries at an appropriate hierarchy level and applies map-reduce generation:

$$R_{global}(q) = \{S(Community_l, k) \mid relevance(q, S_l, k) > \theta\}. \quad (16)$$

The map phase generates partial answers from each relevant community summary; the reduce phase synthesizes these into a coherent final response [9].

Table 2 summarizes GraphRAG stages and their computational complexity. Here, *n* denotes the chunk count, T_{llm} the LLM inference time, and *e* the average number of entities per chunk. $|V|$ and $|E|$ are the vertex and edge counts, $|C|$ is the community count, and deg_{avg} is the average vertex degree.

LIGHTRAG

LightRAG addresses the computational overhead of full GraphRAG through architectural simplifications that preserve the core benefits of graph-based retrieval while substantially reducing indexing and query costs [10].

Streamlined Graph Construction. Rather than exhaustive community detection and multi-level summarization, LightRAG constructs a simpler entity-relation graph with efficient deduplication.

The graph construction employs incremental insertion:

$$G' = InsertIncremental(G, Extract(c_{new})), \quad (17)$$

where the incremental insertion function integrates new entities and relations without requiring full graph reconstruction. This enables online updates as new documents arrive, a critical capability for dynamic knowledge bases [10].

Table 2. **GraphRAG Processing Stages and Complexity**

Stage	Operation	Time Complexity	Space Complexity
Entity Extraction	LLM-based NER per chunk	$O(n \cdot T_{llm})$	$O(n \cdot e)$
Graph Construction	Triple consolidation	$O(\|E\| \cdot log\|V\|)$	$O(\|V\| + \|E\|)$
Entity Resolution	Pairwise similarity	$O(\|V\|^2)$ *worst case*	$O(\|V\|)$
Community Detection	Leiden algorithm	$O(\|E\| \cdot log\|V\|)$	$O(\|V\|)$
Summarization	LLM generation per community	$O(\|C\| \cdot T_{llm})$	$O(\|C\| \cdot L_{summary})$
Local Retrieval	Graph traversal + chunk lookup	$O(k \cdot deg_{avg})$	$O(k)$
Global Retrieval	Summary matching + map-reduce	$O(\|C\| \cdot T_{llm})$	$O(\|C\|)$

Source: **compiled by the authors**

Dual-Level Retrieval Paradigm. LightRAG operates through two complementary retrieval modes that may be invoked independently or combined.

Low-Level Retrieval focuses on specific entities and their associated relations, supporting fact-oriented queries.

The query is decomposed into low-level keywords, which are matched against entity representations in the knowledge graph:

$$R_{\text{low}}(q) = Retrieve_{entity}(K_{\text{low}}(q), G), \quad (18)$$

where $K_{\text{low}}(q)$ denotes the set of entity-specific keywords extracted from query q, G is the entity-relation graph, and $\text{Retrieve}_{\text{entity}}$ returns entity-centered graph context and associated text passages.

High-Level Retrieval addresses abstract, thematic, or corpus-level queries. High-level keywords extracted from the query are matched against relation representations and broader graph context:

$$R_{\text{high}}(q) = Retrieve_{relation}\big(K_{\text{high}}(q), G\big), \quad (19)$$

where $K_{\text{high}}(q)$ denotes the set of thematic keywords extracted from query q, and $\text{Retrieve}_{\text{relation}}$ returns relation-centered graph context and associated text passages.

When both retrieval modes are required, their outputs are combined as

$$R_{\text{hybrid}}(q) = Merge\left(R_{\text{low}}(q), R_{\text{high}}(q)\right), \quad (20)$$

where $Merge$ denotes a procedure that deduplicates and jointly ranks the contexts returned by the two retrieval modes before generation.

Retrieval Mode Selection. Query classification determines the appropriate retrieval mode through keyword analysis and query structure examination. Queries containing specific entity names or requesting factual details trigger low-level retrieval; queries involving comparison, trends, or thematic content activate high-level retrieval. A hybrid mode combines both retrieval levels for complex queries [10].

Computational Optimization. LightRAG achieves efficiency gains through several mechanisms:

– elimination of hierarchical community summarization, which removes the community-summary generation term $O(|\mathcal{K}| \cdot T_{\text{llm}})$ from the preprocessing cost, where $|\mathcal{K}|$ denotes the number of detected communities and T_{llm} denotes the average cost of one LLM-based community-summary generation call;

– incremental graph updates that integrate newly extracted entities and relations without requiring complete reconstruction of the existing graph;

– streamlined dual-level retrieval over entity- and relation-oriented graph representations, reducing dependence on computationally expensive corpus-wide community summarization.

These mechanisms reduce preprocessing and update overhead relative to full GraphRAG while preserving graph-based retrieval capabilities. The actual efficiency gains depend on corpus size, graph density, extraction quality, implementation details, and the number of model calls required during indexing and retrieval; Table 3 therefore summarizes these design differences qualitatively rather than as measured speedup factors.

Table 3. **Design Differences Between GraphRAG and LightRAG**

Aspect	GraphRAG Approach	LightRAG Approach	Qualitative Effect
Community detection	Leiden-based hierarchical community structure	Not used	Removes the community-detection stage from indexing
Hierarchical summarization	Multi-level LLM-generated community summaries	Not used; retrieval operates over entities and relations	Eliminates per-community LLM summarization calls
Retrieval structure	Local graph search and global community-summary search	Dual-level retrieval over entity- and relation-oriented representations	Preserves support for global queries without community summaries
Graph updates	Major corpus changes may require re-clustering and summary regeneration	Incremental insertion of new entities and relations	Enables incremental index maintenance

Source: **compiled by the authors**

ISSN 2663-0176 (Print)
ISSN 2663-7731 (Online)

CORRECTIVE RAG

Corrective RAG introduces an explicit quality evaluation mechanism that assesses retrieved documents before generation, enabling corrective actions when initial retrieval is inadequate [11]. The architecture adds minimal indexing overhead and is reported to improve generation faithfulness [11].

Retrieval Evaluation Mechanism. Following initial retrieval, each retrieved document undergoes relevance assessment through a lightweight evaluator.

The evaluator function E_{rel} produces a relevance score:

$$E_{rel}(q, d_i) \in [0, 1], \quad (21)$$

where scores near 1 indicate high relevance and scores near 0 indicate irrelevance. The evaluator can be implemented as a fine-tuned classification model or through LLM-based assessment with structured output.

Confidence Classification. Aggregate retrieval quality is classified into three categories based on the distribution of individual document scores:

$$\begin{gathered} \textit{S Correct:}\ \max(E_{rel}(q, d_i)) > \theta_{high} \\ \text{AND } \text{mean}(E_{rel}(q, d_i)) > \theta_{mean}, \\ \textit{Incorrect:}\ \max(E_{rel}(q, d_i)) < \theta_{low}, \\ \textit{Ambiguous:}\ \text{Otherwise}. \end{gathered} \quad (22)$$

The thresholds θ_{high}, θ_{low}, and θ_{mean} are tunable hyperparameters that control the sensitivity of the classification.

Corrective Action Protocol. Each classification triggers a distinct processing pathway. Table 4 summarizes the corresponding decision logic and actions, while the main pathways are outlined below:

– *Correct Classification* proceeds directly to generation using the retrieved documents as context, identical to baseline RAG behavior.

– *Incorrect Classification* triggers web search augmentation to supplement inadequate local retrieval.

The augmentation function performs:

$$R_{augmented}(q) = R(q, D) \cup WebSearch(q), \quad (23)$$

where *WebSearch* returns relevant documents from external search engines. This provides fallback capability when the local corpus lacks relevant information.

Ambiguous Classification triggers expanded retrieval with query reformulation. The system generates alternative query formulations and retrieves additional documents:

$$R_{expanded}(q) = R(q, D) \cup R(Reformulate(q), D), \quad (24)$$

where *Reformulate* produces semantically equivalent queries with different lexical choices, expanding retrieval coverage.

Knowledge Refinement. Prior to generation, retrieved content undergoes decomposition into atomic knowledge units and filtering based on relevance:

$$Knowledge_{refined} = Filter(Decompose(R_{final}(q)), q). \quad (25)$$

The decomposition function segments documents into discrete factual statements; the filter retains only statements with high query relevance. This refinement reduces noise in the generation context and improves answer faithfulness.

Table 4. **CRAG Decision Logic and Actions**

Retrieval Quality	Classification Criteria	Triggered Action	Expected Outcome
High Confidence	*max_score* > 0.7, *mean_score* > 0.5	Direct generation	Standard RAG behavior
Low Confidence	*max_score* < 0.3	Web search augmentation	External knowledge integration
Uncertain	Intermediate scores	Query reformulation with expanded retrieval	Improved coverage

Source: **compiled by the authors**

AGENTIC RAG

Agentic RAG embeds an autonomous decision-making agent into the retrieval pipeline, enabling dynamic adaptation through iterative refinement, multi-source integration, and self-critique [13], [14]. The architecture trades increased query-time computation for enhanced handling of complex queries.

Agent Architecture. The agentic system comprises a controller agent with access to a tool library $T = \{t_1, t_2, ..., t_m\}$. Tools encapsulate retrieval operations (vector search, graph query, web search), processing operations (summarization, entity extraction), and utility operations (relevance scoring, answer validation). The agent maintains a reasoning trace $S = (s_1, s_2, ..., s_n)$ documenting its decision sequence.

ReAct-Style Reasoning. The agent operates through interleaved reasoning and action steps following the ReAct paradigm [35].

At each iteration i, the agent observes the current state (query, retrieved context, partial answer) and produces:

$$(Thought_i, Action_i) = Agent(q, Context_i, S<i), \quad (26)$$

where $Thought_i$ represents the agent's reasoning about what information is needed and why, *and* $Action_i$ specifies a tool invocation with parameters. The action executes, producing an observation:

$$Observation_i = Execute(Action_i). \quad (27)$$

The observation is appended to the context for the next iteration. The agent continues iterating until one of several termination conditions is satisfied:

– Confidence threshold. The agent's self-assessed confidence in the accumulated evidence exceeds a threshold θ_{conf};

– Iteration limit. The maximum number of iterations I_{max} is reached;

– Explicit completion. The agent determines that sufficient information has been gathered.

The iteration limit prevents unbounded computation for intractable queries.

Tool Selection Strategy. The agent selects tools based on query analysis and accumulated context.

The selection function considers:

$$Tool_{selected} = arg\ max\ P(useful \mid q, Context, t) \quad for\ t \in T, \quad (28)$$

where usefulness probability is estimated through the agent's language model capabilities.

Common selection patterns include:

– *Initial retrieval*. Vector search for broad coverage;

– *Targeted follow-up*. Graph query for entity-specific information;

– *Knowledge gaps*. Web search for information absent from local corpus;

– *Verification*. Relevance scoring to validate retrieved content.

Multi-Source Integration. When multiple retrieval sources are invoked, the agent synthesizes results through explicit reasoning about source reliability, information overlap, and potential conflicts:

$$ContextIntegrated = Synthesize(R_{vector}(q), R_{graph}(q), R_{web}(q), AgentReasoning). \quad (29)$$

The synthesis process deduplicates redundant information, resolves contradictions through source priority or recency, and organizes content for effective generation.

COMPARISON OF RAG ARCHITECTURES

The five RAG architectures differ along several technical dimensions that influence their suitability for specific application scenarios.

The architectures employ fundamentally different retrieval paradigms (Table 5):

– **Single-pass dense retrieval (Baseline RAG):** one retrieval operation based on embedding similarity;

– **Graph-augmented retrieval (GraphRAG and LightRAG):** retrieval that uses explicit entity-relation structure in addition to textual representations;

– **Evaluated retrieval (CRAG):** retrieval followed by quality assessment and conditional corrective actions;

– **Iterative adaptive retrieval (Agentic RAG):** multiple retrieval and tool-use operations coordinated by agent reasoning.

Computational Cost Model. The online computational cost of a retrieval-augmented generation pipeline can be decomposed into retrieval, orchestration, and generation components:

$$C_{\text{total}}(q) = C_{\text{retrieval}}(q) + C_{\text{orchestration}}(q) + C_{\text{generation}}(q), \quad (30)$$

where $C_{\text{retrieval}}(q)$ denotes the cost of locating relevant evidence, $C_{\text{orchestration}}(q)$ denotes the cost of routing, evaluation, query reformulation, graph traversal, or tool coordination, and $C_{\text{generation}}(q)$ denotes the cost of producing the final response.

The individual terms are implementation-dependent and vary with the retrieval backend, index configuration, number of model or tool calls, and input and output sequence lengths.

For an iterative agentic pipeline, the total online computational cost can be represented more explicitly as

$$C_{\text{agentic}}(q) = \sum_{i=1}^{I(q)} \left(C^{(i)}_{\text{controller}}(q) + C^{(i)}_{\text{tool}}(q) \right) + C_{\text{generation}}(q), \quad (31)$$

where $I(q)$ is the number of agent iterations executed for query q, $C^{(i)}_{\text{controller}}(q)$ is the controller-model cost at iteration i, and $C^{(i)}_{\text{tool}}(q)$ is the cost of the tool invoked at that iteration.

ISSN 2663-0176 (Print)
ISSN 2663-7731 (Online)

Table 5. **Comprehensive Architectural Comparison**

Dimension	Baseline	GraphRAG	LightRAG	CRAG	Agentic RAG
Retrieval paradigm	Single-pass vector retrieval	Dual-mode graph retrieval	Dual-level graph retrieval	Evaluated and corrective retrieval	Iterative adaptive retrieval
Preprocessing overhead	Low	High: graph extraction, community detection, and hierarchical summarization	Moderate: graph extraction and indexing without hierarchical community summaries	Low	Low for the retrieval index; tool preparation is implementation-dependent
Query-time orchestration	Minimal	Local graph retrieval or global community-summary retrieval	Entity- and relation-oriented retrieval	Conditional evaluation, reformulation, or augmentation	Iterative controller and tool calls
Memory overhead	Embeddings and vector index	Graph, embeddings, and community summaries	Graph, embeddings, and retrieval indexes	Embeddings, vector index, and evaluator	Embeddings, tools, and agent working state
Update capability	Backend-dependent; incremental insertion may be supported	Major corpus changes may require costly graph and summary updates	Supports incremental graph updates	Backend-dependent	Depends on the update capabilities of the underlying tools
Multi-hop support	Limited	Strong for graph-supported queries	Moderate to strong	Limited to corrective retrieval	Strong
Hallucination control	Indirect through retrieved evidence	Indirect through structured evidence and summaries	Indirect through structured evidence	Explicit retrieval-quality evaluation	Explicit iterative verification and self-critique

Source: **compiled by the authors**

Scalability Characteristics. Scaling behavior differs substantially across architectures.

Corpus Scaling (increasing n). Baseline RAG and CRAG depend on the selected approximate-nearest-neighbor backend and its index parameters rather than on a universally guaranteed logarithmic query complexity. GraphRAG scaling depends on the size and density of the extracted graph, the number of detected communities, and the cost of community summarization. LightRAG reduces update overhead through incremental graph insertion, although entity-relation extraction, embedding computation, and index construction still grow with corpus size. Agentic RAG scaling depends on the retrieval tools invoked and the number of controller iterations.

Query Complexity Scaling. Baseline RAG may degrade as the semantic gap, multi-hop requirements, or need for corpus-level synthesis increases. Graph-based methods can provide more stable performance when explicit entity-relation structure is useful for the query. CRAG performance depends on the accuracy and calibration of its retrieval evaluator. Agentic RAG can adaptively increase the number of retrieval and verification steps for complex queries, trading latency and cost for potentially improved evidence coverage (Table 6).

EVIDENCE-GROUNDED BENCHMARKING PROTOCOL

The comparison method is formalised as Algorithm 1. It's essential property is that everything except the retrieval and orchestration logic is held fixed: the sentence-level chunking, the embedding model and both indices, the candidate pool size, the cut-off *k*, the generation model, the prompt, and the JSON output contract. Pipelines are executed in an interleaved order for every claim so that transient API conditions affect all systems alike, each end-to-end run is repeated *R* times, and all metrics are computed from the identical set of evaluation units, with invalid outputs counted as incorrect predictions rather than discarded.

Table 6. **Qualitative Scalability and Resource Characteristics (schematic, implementation-dependent)**

Architecture	Corpus Scaling	Query Complexity	Memory Req.	Online Model / Tool Calls
Baseline RAG	Governed by the ANN backend and index configuration	May degrade on multi-hop and corpus-level queries	Low	Single generation call per query
GraphRAG	Governed by graph size, community count, and summarization cost	More stable when explicit graph structure matches the query	High	One or more calls; map–reduce over communities for global queries
LightRAG	Governed by the ANN backend and graph degree; supports incremental updates	Moderate degradation on complex queries	Medium	Small number of calls for keyword extraction and generation
CRAG	Governed by the ANN backend plus evaluator invocation	Depends on evaluator calibration	Low-Medium	Evaluator call plus conditional corrective re-retrieval
Agentic RAG	Governed by the ANN backend, multiplied by agent iterations	Adapts through additional retrieval iterations	Low	Multiple controller and tool calls, bounded by the iteration limit

Source: **compiled by the authors**

Algorithm 1. Evidence-grounded benchmarking of retrieval-augmented pipelines
Input: abstract corpus A; claim set Q with gold labels y_q and gold rationale sets G_q; set of pipelines Π; cut-off k; candidate pool size k_{pool}; number of repeats R.
Output: metric estimates with 95 % confidence intervals and paired tests for every $\pi \in \Pi$.
1: split every abstract $a \in A$ into sentences; $C \leftarrow$ {chunks a::s} with stable identifiers
2: $E(c) \leftarrow$ embed(c) for all $c \in C$; build the dense index I_d and the BM25 index I_s over C
3: **for** graph-based π: extract entities and relations from every abstract, build $G = (V, E)$, detect communities ▷ offline cost, recorded separately
4: **for** $r \leftarrow 1$ **to** R **do**
5: **for each** $q \in Q$ in fixed order **do**
6: **for each** $\pi \in \Pi$ (interleaved) **do**
7: reset the model-call counter; $t_0 \leftarrow$ now
8: $L \leftarrow \pi$.retrieve(q, k_{pool}) ▷ chunk ranking obtained with I_d, I_s, G, and orchestration calls
9: $o \leftarrow$ Generate(q, L[1..k]) ▷ shared prompt and JSON contract
10: $t_1 \leftarrow$ now; log latency $t_1 - t_0$, model calls, tokens, branch and iteration diagnostics
11: **if** o is not valid JSON with an admissible label **then** $\hat{y} \leftarrow$ invalid, $\hat{C} \leftarrow \emptyset$ **else** $(\hat{y}, \hat{C}) \leftarrow o$
12: $D \leftarrow$ documents of L in order of first appearance
13: compute P@k, R@k, nDCG@k, MRR for D against gold documents and for L against G_q; evidence precision, recall, and F1 of $\hat{C}$ against G_q; verdict correctness [$\hat{y} = y_q$]
14: aggregate over Q and R; percentile-bootstrap CIs (B = 10 000); Wilcoxon and McNemar tests against the dense baseline with Holm correction; SD of the system means across repeats
15: **return** tables of estimates, intervals, tests, and per-query cost

Steps 1-3 constitute the offline stage, whose cost is reported separately from per-query latency; steps 7-13 are executed for every (pipeline, claim, repeat) triple and produce one record with the ranking, the parsed output, the timing, and the diagnostics; step 14 turns the records into interval estimates and paired tests. The retrieval function of step 8 is the pipeline-specific instantiation of (2), and the generation step 9 realises (1) with the retrieved evidence as the conditioning set. Fig. 3 summarises the resulting framework: the corpus, the claims, and the shared retrieval resources are common to all pipelines, only the retrieval and orchestration block differs, and the shared generation step and the evaluation of retrieval, evidence, verdict, and cost are again identical.

IMPLEMENTATION CONSIDERATIONS

We benchmark *corpus-restricted* implementations of the reference architectures on SciFact [8]. All pipelines operate on the same

ISSN 2663-0176 (Print)
ISSN 2663-7731 (Online)

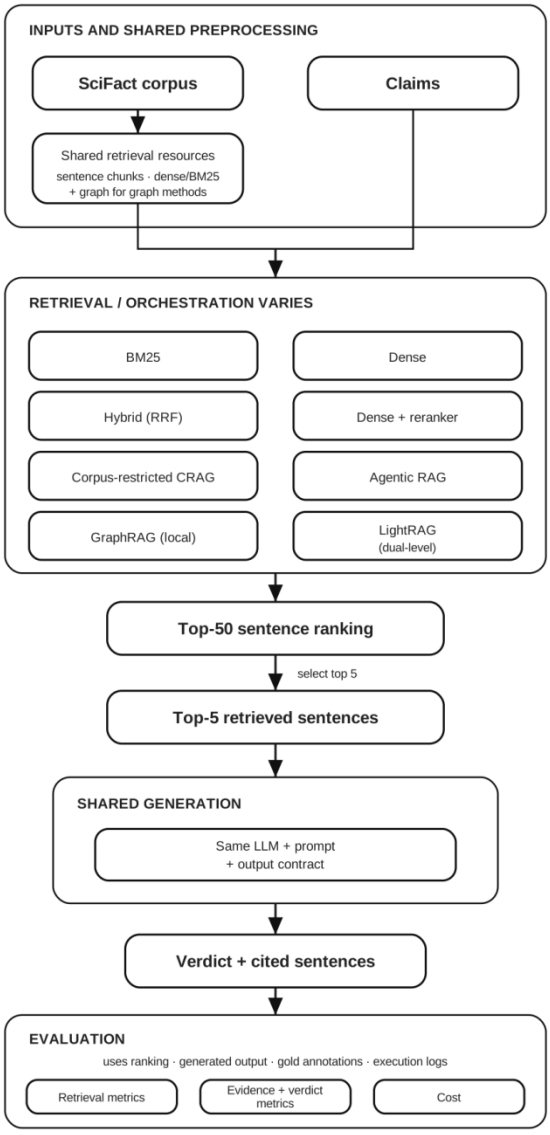


Fig. 3. **Unified evidence-grounded benchmarking framework**
Source: **compiled by the authors**

sentence-level index and use the same generator,prompt, and output contract. External web search is disabled. Self-RAG [12] is excluded because its reflection-token training is incompatible with the shared API-served generator. The following descriptions identify simplified or disabled components, so comparisons apply to these implementations rather than the full reference methods. Code, benchmark identifiers, prompts, and raw outputs are available at https://github.com/neocode/scifact-rag-benchmark.

Unified embedding backbone and indices. A single embedding model and a single sentence-level index are used by all pipelines, so that observed differences reflect retrieval and orchestration logic rather than indexing heterogeneity. The implementation employs E5-base-v2, a 768-dimensional encoder from the E5 family [36] that uses the instruction prefixes “query:” and “passage:” for asymmetric retrieval. Passage embeddings are L2-normalised and computed once; dense retrieval is *exact* cosine-similarity search over all chunks (no approximate-nearest-neighbour index), which removes index approximation as a confounding factor. A lexical index over the same chunks is built with Okapi BM25 [37] (k_1 = 1.5, b = 0.75). Each pipeline returns a ranking of k_{pool} = 50 chunks, from which the first k = 5 chunks form the generation context and document-level rankings are derived by first appearance.

Chunking strategy and evidence alignment. Because SciFact provides sentence-level rationale annotations [8], we use individual sentences as retrieval chunks. This directly aligns retrieved units with gold evidence, at the cost of shorter chunks with less context. The main benchmark contains 4,440 chunks from 500 abstracts, and the full corpus contains 45,952 chunks. Abstract titles are stored as chunk metadata.

Baseline retrievers. We compare the advanced pipelines against four conventional baselines without LLM-based orchestration: (i) *BM25* lexical retrieval, (ii) *Dense* retrieval using E5 cosine similarity, (iii) *Hybrid* retrieval using reciprocal-rank fusion [38] of the BM25 and dense top-50 lists,

$$score_{\text{RRF}}(c) = \Sigma_l \, 1 / (\kappa + rank_l(c)), \qquad (32)$$

where the sum runs over the fused lists l and κ = 60; (iv) *Dense + reranker* – the dense top-20 re-ordered by the cross-encoder BAAI/bge-reranker-base [39], with the remaining dense candidates appended. All four baselines use exactly one model call per query (generation).

Graph construction for GraphRAG and LightRAG. Both graph-based pipelines share one entity–relation graph built offline. For every abstract a single LLM call extracts typed entities and relations together with the identifiers of the sentences that express each relation (prompt in the repository). Entities are resolved by normalised-name matching, and every entity and relation keeps its document and sentence provenance, which links graph nodes to the indexed chunks. For the main corpus the graph contains 6,472 entities and 6,391 edges (4 of 500 extraction outputs could not be parsed and were skipped); for the full corpus, 53,444 entities and 65,544 edges (29 of 5,183 failures). Communities are detected with the Louvain algorithm at resolution 1.0 (networkx

implementation; leidenalg was not installed in the reported runs, so the communities are Louvain partitions and the Leiden refinement [34] remains an option of the code) – 1,212 and 7,647 communities respectively – and are reported as graph statistics only, because the GraphRAG instantiation implements the *local* search path of [9]: query entities are identified by name occurrence in the claim and by E5 similarity between the claim and entity names (top-8), expanded to their one-hop neighbourhood, and the provenance sentences of the resulting entities form the candidate set, which is re-ranked by dense similarity to the claim; when fewer than five candidates are found the pipeline falls back to dense retrieval (the fallback rate is reported). The *global* path – community summaries with map–reduce generation – is not exercised by claim-level queries and is disabled. The LightRAG instantiation implements dual-level retrieval [10]: one LLM call extracts low-level (entity) and high-level (theme or relation) keywords from the claim; low-level keywords are matched to entity names and high-level keywords to relation descriptions by E5 similarity (top-8 each), and the union of the provenance sentences is re-ranked densely. Neither pipeline uses hierarchical summaries or web access.

Corpus-restricted Corrective RAG. The evaluator of (21) is implemented as one LLM call that scores each of the five retrieved sentences in [0, 1] with respect to the claim. Classification follows (22) and Table 4 with the thresholds $\theta_{\text{high}} = 0.7$, $\theta_{\text{mean}} = 0.5$ and $\theta_{\text{low}} = 0.3$, adapted from the reference design (the original method uses only an upper and a lower threshold on the evaluator score; the mean-score condition is introduced here), fixed a priori and not tuned on the benchmark claims. The *Correct* branch proceeds to generation with the original ranking. The *Ambiguous* branch issues one LLM query-reformulation call, retrieves densely with the reformulated query and fuses both rankings by reciprocal-rank fusion, realising (24). Because web search is disabled, the *Incorrect* branch of (23) is replaced by a corpus-restricted substitute: reformulation followed by hybrid (dense and BM25) retrieval fused with the original ranking. Knowledge refinement (25) is realised by demoting evaluated sentences scored below 0.3 to the end of the ranking. The pipeline therefore makes two or three model calls per query, and the frequency of each branch is reported. This variant should be read as a *modified, corpus-restricted CRAG*, not as the full method of [11].

Agentic RAG controller. The agentic pipeline follows the ReAct pattern (26)-(27) with a single tool, corpus search. After an initial dense retrieval with the claim, an LLM controller inspects the evidence gathered so far and returns either a new search query (a sub-question, a synonym or expanded abbreviation, or a focus on a specific entity) or a *finish* action; each search retrieves 20 candidates and all rankings are fused by reciprocal-rank fusion. The iteration budget is $I_{\max} = 3$ controller decisions, chosen to accommodate the two to three retrieval cycles that multi-hop claims typically require while bounding worst-case latency; the number of executed iterations, the number of additional searches and the termination reason are logged for every claim. No web search, graph tool, or external knowledge source is available to the agent.

Constrained generation format. All pipelines use one prompt and one output contract: a JSON object with the fields *label*, *answer*, and *evidence*, where *label* ∈ {SUPPORTS, REFUTES} in the main experiment and additionally NOT_ENOUGH_INFO in the development-split experiment, and *evidence* lists the identifiers of the cited sentences, restricted to the five sentences in the context. The generation model is Gemini 2.5 Flash (API model name gemini-2.5-flash) [40], called at temperature 0 in JSON-response mode with the model's internal "thinking" disabled (thinking budget 0), so that every pipeline uses an identical generator without hidden reasoning tokens in latency or cost; the same model and settings serve all orchestration calls (retrieval evaluation, query reformulation, agent control, keyword and graph extraction). Outputs are parsed strictly; an output that is not a JSON object with an admissible label is counted as an incorrect prediction with no cited evidence, so every pipeline is scored on the identical set of evaluation units, and the JSON validity rate is reported as a separate reliability indicator.

Retrieval depth and evaluation scope. Retrieval is evaluated at $k = 5$: the five chunks passed to the generator are the units scored at the sentence level, and the documents they belong to, in order of first appearance within the 50-chunk ranking, are scored at the document level. This cut-off reflects a realistic context budget (five sentences amount to roughly 100–250 tokens) and matches the SciVer convention of citing at most a few rationale sentences per abstract. The dual-granularity protocol reveals effects that are invisible at the document level: a system may retrieve the correct abstract consistently yet fail to prioritise the annotated rationale sentences within it.

ISSN 2663-0176 (Print)
ISSN 2663-7731 (Online)

Reproducibility. The benchmark subset is generated deterministically (seed 42) and the exact claim, abstract, and chunk identifiers are exported with the results; all hyper-parameters (Table 7) were fixed a priori, and no parameter was tuned on the evaluation claims, so no calibration split overlaps the test claims. The main experiment consists of three independent end-to-end runs with the systems interleaved per claim; the development-split experiment was run once. The generation model version, library versions, and hardware are recorded automatically (Python 3.12, PyTorch 2.10, sentence-transformers 5.2, transformers 5.0, google-genai SDK; 24-core Linux workstation with an NVIDIA GeForce RTX 3090). Offline graph construction required 500 extraction calls (273 k prompt and 535 k output tokens, 38 min) for the main corpus and 5,183 calls (2.72 M prompt and 5.46 M output tokens, 6.5 h) for the full corpus; these costs are reported separately from per-query latency.

EXPERIMENTAL EVALUATION

Datasets and benchmark construction. SciFact provides scientific claims paired with gold evidence sentences from biomedical abstracts [8]. Two evaluation sets are used. The *main benchmark* is a fixed subset of the training split: 120 claims with a SUPPORT or CONTRADICT label and at least one annotated rationale were sampled with seed 42 (81 SUPPORTS and 39 REFUTES; 8 claims have evidence in more than one abstract; 2.09 gold sentences per claim on average). The corpus comprises the 125 evidence abstracts of these claims and 375 hard-negative abstracts selected by keyword overlap with the claims, 500 abstracts and 4,440 sentence chunks in total. Because every claim in this set has evidence in the corpus, it isolates evidence ranking quality but cannot measure behaviour in the absence of evidence. The *development-split benchmark* therefore uses the standard open SciFact

Table 7. **Implementation summary (all values fixed a priori)**

Parameter	Value	Rationale
Embedding model	intfloat/e5-base-v2 (768-d, “query:”/“passage:” prefixes)	Shared by all pipelines [36]
Dense search	Exact cosine similarity over L2-normalised embeddings	No ANN approximation
Lexical index	Okapi BM25, $k_1 = 1.5$, $b = 0.75$	Sparse baseline and hybrid component [37]
Hybrid fusion	Reciprocal-rank fusion, $\kappa = 60$, top-50 lists	Rank-based, score-scale free [38]
Reranker	BAAI/bge-reranker-base over dense top-20	Cross-encoder baseline [39]
Chunking	One sentence per chunk; title kept as metadata	Alignment with SciFact rationales
Cut-off / pool	$k = 5$ context chunks; $k_{pool} = 50$	Rank-sensitive evaluation, document ranking
Graph extraction	One LLM call per abstract; typed entities, relations with sentence ids	Shared by GraphRAG and LightRAG
Community detection	Louvain (networkx), resolution 1.0; leidenalg not installed	Statistics only; global path not exercised
Graph query matching	Top-8 entities / relations by E5 similarity; 1-hop expansion	GraphRAG local / LightRAG dual-level
CRAG thresholds	$\theta_{high} = 0.7$, $\theta_{mean} = 0.5$, $\theta_{low} = 0.3$; refinement < 0.3	Adapted from [11]; web search disabled
Agent budget	$I_{max} = 3$ controller decisions; 20 candidates per search	Bounded latency and cost
Generation model	Gemini 2.5 Flash (gemini-2.5-flash), temperature 0, JSON mode, thinking budget 0	Identical for all pipelines and orchestration calls [40]
Output format	JSON {label, answer, evidence}; strict parsing	Deterministic scoring; validity reported
Repeats	3 independent end-to-end runs (main), 1 run (development split); systems interleaved	Stability and paired tests
Seed	42 (subset sampling, hard negatives)	Reproducible identifiers

Source: **compiled by the authors**

development split without modification – 300 claims (124 SUPPORTS, 64 REFUTES, 112 NOT ENOUGH INFO) – over the full corpus of 5,183 abstracts (45,952 chunks). The two claim sets come from disjoint official splits (zero shared identifiers), and no parameter was tuned on either of them.

Evaluation metrics. Let L_q be the chunk ranking passed to the generator, D_q the induced document ranking, G_q the gold rationale set (the union of annotated sentences over all evidence abstracts of the claim) and $\hat{C}_q$ the set of sentence identifiers cited in the JSON output. Retrieval is evaluated at both granularities with Precision@5, Recall@5, nDCG@5 [41] and MRR under binary relevance:

$$\text{nDCG@}k = \text{DCG@}k / \text{IDCG@}k, \quad (33)$$

$$\text{DCG@}k = \Sigma_{i=1..k}\, rel_i / \log_2(i + 1), \quad (34)$$

$$\text{MRR} = 1 / rank_{first}, \quad (35)$$

where rel_i is 1 if the item at rank i is relevant, IDCG@k is the DCG of the ideal ranking of the $|G|$ gold items, and $rank_{first}$ is the rank of the first relevant item (MRR = 0 if none is retrieved).

Evidence quality is measured on the *cited* sentences, which is what a user of the system would actually see:

$$P_{ev} = |\hat{C}_q \cap G_q| / |\hat{C}_q|, \quad R_{ev} = |\hat{C}_q \cap G_q| / |G_q|, \quad (36)$$

$$F1_{ev} = 2\, P_{ev}\, R_{ev} / (P_{ev} + R_{ev}), \quad (37)$$

with $P_{ev} = 0$ when nothing is cited; in addition, the share of claims for which at least one complete annotated rationale set of SciFact is cited (full-rationale recovery) is reported; since Gq is the union of all such sets, this share can exceed evidence recall. Following the SciFact and FEVER convention, a joint verdict-and-rationale score counts a claim as solved only when the verdict is correct and at least one complete rationale set is cited (for claims without evidence in the corpus, a correct abstention); it is the metric that penalises a right verdict reached for the wrong reason. Verdict quality is measured by accuracy, per-class precision, recall and F1, macro-F1 and balanced accuracy, together with confusion matrices; outputs that fail the JSON contract are counted as incorrect with no cited evidence, and the JSON validity rate is reported separately. For the development split the verdict task is three-class, and retrieval and evidence metrics are computed on the 188 claims that have annotated evidence.

Statistical analysis. Every metric is first averaged over the repeats for each claim and then over claims. Uncertainty is quantified by 95 % percentile-bootstrap confidence intervals over claims (10 000 resamples). Each pipeline is compared with the dense baseline by paired analysis of per-claim differences: a paired-bootstrap interval of the mean difference and the Wilcoxon signed-rank test for continuous metrics, and the exact McNemar test for verdict correctness on per-claim majority verdicts across repeats (one observation per claim, so that identical repeats are not counted several times); p-values are Holm-adjusted across the seven comparisons within each metric [42]. Generation stability is reported as the standard deviation of the system-level means across the three repeats.

Efficiency measurement. Latency is the wall-clock time from the arrival of the claim to the parsed JSON output and includes retrieval, all orchestration calls, and generation; it is decomposed into retrieval-and-orchestration time and generation time. Pipelines run sequentially in one process without response caching of online calls, the first three claims of every run are treated as warm-up and excluded, and mean, median, standard deviation, and 95th percentile are reported together with the number of model calls (by stage) and the prompt and output tokens per query. Embeddings and reranking were computed on the GPU of the workstation described above; the language model was accessed through the public Gemini API, so absolute latencies include network and service time and are comparable only within a run. Offline costs (graph extraction, index construction) are reported separately.

Results: retrieval quality. Table 8 reports the retrieval metrics of the eight pipelines on the main benchmark. Document-level retrieval is close to saturation for every pipeline built on the E5 index: Doc nDCG@5 lies between 0.906 (GraphRAG) and 0.943 (CRAG), Doc Recall@5 between 0.937 and 0.983, and none of the paired differences to the dense baseline (0.933) is significant. Only BM25 is clearly worse (Doc nDCG@5 = 0.812, Holm-adjusted p = 0.002), which shows that lexical matching alone is insufficient for claims that paraphrase the evidence. At the sentence level the spread is wider. The corpus-restricted CRAG (0.670) and the agentic pipeline (0.661) obtain the highest Sent nDCG@5, followed by the dense baseline and LightRAG (both 0.641), GraphRAG (0.611), the reranker (0.579), hybrid fusion (0.569) and BM25 (0.459). The two strongest conventional alternatives to dense retrieval therefore *reduce* sentence-level quality: fusing BM25 into the ranking imports its weaker ordering (Δ Sent nDCG@5 = −0.073 [−0.120, −0.026], Holm-adjusted p = 0.032), and the

ISSN 2663-0176 (Print)
ISSN 2663-7731 (Online)

general-domain cross-encoder re-orders single biomedical sentences less accurately than the E5 similarity it replaces (Δ = −0.062 [−0.116, −0.008]). LightRAG reproduces the dense ranking almost exactly, because its entity- and relation-matched candidate set (on average 197 of 4,440 chunks) nearly always contains the dense top-5, whereas the smaller neighbourhood of GraphRAG (124 chunks) occasionally excludes a gold sentence (Δ = −0.030 [−0.062, −0.002]). Neither graph pipeline ever fell back to plain dense retrieval.

Table 8. **Retrieval quality on the main benchmark (120 claims, mean over 3 runs; 95 % bootstrap confidence intervals for nDCG@5 and MRR)**

System	Document Precision@5	Document Recall@5	Document nDCG@5	Document MRR	Sentence Precision@5	Sentence Recall@5	Sentence nDCG@5	Sentence MRR
BM25	0.197	0.893	0.812 [0.751, 0.870]	0.786 [0.721, 0.849]	0.185	0.530	0.459 [0.391, 0.529]	0.535 [0.462, 0.609]
Dense (baseline)	0.218	0.981	0.933 [0.898, 0.963]	0.919 [0.878, 0.956]	0.277	0.735	0.641 [0.579, 0.701]	0.681 [0.615, 0.745]
Hybrid (RRF)	0.217	0.983	0.917 [0.880, 0.951]	0.896 [0.851, 0.938]	0.247	0.678	0.569 [0.506, 0.631]	0.606 [0.541, 0.674]
Dense + reranker	0.213	0.972	0.908 [0.868, 0.942]	0.889 [0.843, 0.930]	0.230	0.654	0.579 [0.515, 0.643]	0.656 [0.587, 0.722]
CRAG (corpus-restricted)	0.218	0.978	0.943 [0.910, 0.971]	0.935 [0.898, 0.967]	0.284	0.752	0.670 [0.609, 0.727]	0.714 [0.647, 0.775]
Agentic RAG	0.218	0.981	0.933 [0.899, 0.963]	0.920 [0.880, 0.956]	0.289	0.775	0.661 [0.602, 0.716]	0.681 [0.616, 0.744]
GraphRAG (local)	0.208	0.937	0.906 [0.858, 0.948]	0.901 [0.849, 0.946]	0.265	0.688	0.611 [0.545, 0.676]	0.654 [0.583, 0.723]
LightRAG (dual-level)	0.217	0.973	0.933 [0.895, 0.965]	0.922 [0.879, 0.960]	0.277	0.731	0.641 [0.578, 0.702]	0.680 [0.612, 0.745]

Source: **compiled by the authors**

Results: evidence and verdict quality. Table 9 summarises the evidence metrics computed on the cited sentences and the verdict metrics, and Table 10 gives the confusion matrices and the per-class precision and recall pooled over the three runs. The agentic pipeline cites annotated rationales most completely (evidence recall 0.704, full-rationale recovery 0.836), followed by CRAG (0.685) and the dense baseline (0.666); the advantage of the agentic pipeline over the baseline is +0.039 [−0.005, +0.085] (Holm-adjusted p = 0.353). The joint verdict-and-rationale score, which requires both a correct verdict and a complete cited rationale, ranges from 0.592 (BM25) to 0.775 (Agentic RAG) against 0.742 for the dense baseline, i.e. the ordering of evidence recall rather than of verdict accuracy. Evidence precision is nearly flat (0.598-0.647) for all E5-based pipelines, i.e. the generator cites about one non-annotated sentence for every two annotated ones regardless of the retriever; part of this is annotation incompleteness rather than error. Verdict accuracy ranges from 0.758 (BM25) to 0.892 (dense with reranker); CRAG (0.886) and the agentic pipeline (0.875) are above the dense baseline (0.858), GraphRAG (0.833) and LightRAG (0.850) slightly below it. The confusion matrices reveal a systematic asymmetry of the generator under the two-class contract: every REFUTES claim was classified correctly by every pipeline (REFUTES recall 1.000), and all errors are SUPPORTS claims labelled REFUTES (SUPPORTS recall 0.790 for the dense baseline, 0.840 with reranking, 0.642 for BM25). When the retrieved sentences do not confirm a supported claim explicitly, the model prefers REFUTES to abstention, which it cannot express in this contract; balanced accuracy and macro-F1

Table 9. **Evidence and verdict quality on the main benchmark (120 claims, 3 runs; 95 % confidence interval for evidence recall; verdict metrics pooled over runs)**

System	Evidence precision	Evidence recall	Evidence F1	Full rationale recovered	Joint verdict-and-rationale score	Verdict accuracy	Macro-F1	Balanced accuracy	F1 SUPPORTS	F1 REFUTES	Valid JSON share	Accuracy SD over 3 runs
BM25	0.525	0.499 [0.424, 0.575]	0.471	0.650	0.592	0.758	0.755	0.821	0.782	0.729	1.000	0.000
Dense (baseline)	0.629	0.666 [0.597, 0.733]	0.607	0.800	0.742	0.858	0.852	0.895	0.883	0.821	1.000	0.000
Hybrid (RRF)	0.618	0.616 [0.543, 0.687]	0.579	0.772	0.717	0.858	0.852	0.895	0.883	0.821	1.000	0.000
Dense + reranker	0.598	0.600 [0.528, 0.673]	0.559	0.750	0.675	0.892	0.885	0.920	0.913	0.857	1.000	0.000
CRAG (corpus-restricted)	0.644	0.685 [0.616, 0.750]	0.623	0.819	0.756	0.886	0.879	0.916	0.908	0.851	1.000	0.004
Agentic RAG	0.647	0.704 [0.639, 0.768]	0.635	0.836	0.775	0.875	0.868	0.907	0.898	0.839	1.000	0.000
GraphRAG (local)	0.615	0.635 [0.563, 0.707]	0.586	0.764	0.700	0.833	0.828	0.877	0.859	0.796	1.000	0.000
LightRAG (dual-level)	0.631	0.656 [0.585, 0.726]	0.606	0.783	0.733	0.850	0.844	0.889	0.875	0.813	1.000	0.000

Source: **compiled by the authors**

Table 10. **Confusion matrices and per-class precision and recall on the main benchmark (rows of the matrix: gold label; pooled over 3 runs, 360 decisions per system: 243 SUPPORTS and 117 REFUTES)**

System	Gold SUPPORTS → SUPPORTS	Gold SUPPORTS → REFUTES	Gold REFUTES → REFUTES	Gold REFUTES → SUPPORTS	Invalid outputs	Precision SUPPORTS	Recall SUPPORTS	Precision REFUTES	Recall REFUTES
BM25	156	87	117	0	0	1.000	0.642	0.574	1.000
Dense (baseline)	192	51	117	0	0	1.000	0.790	0.696	1.000
Hybrid (RRF)	192	51	117	0	0	1.000	0.790	0.696	1.000
Dense + reranker	204	39	117	0	0	1.000	0.840	0.750	1.000
CRAG (corpus-restricted)	202	41	117	0	0	1.000	0.831	0.741	1.000
Agentic RAG	198	45	117	0	0	1.000	0.815	0.722	1.000
GraphRAG (local)	183	60	117	0	0	1.000	0.753	0.661	1.000
LightRAG (dual-level)	189	54	117	0	0	1.000	0.778	0.684	1.000

Source: **compiled by the authors**

therefore track the SUPPORTS recall. The reranker helps verdicts although it ranks evidence worse, plausibly because it promotes sentences that state the measured outcome rather than the annotated rationale. All outputs were valid JSON, and the three repeats at temperature 0 produced identical verdicts for every pipeline except CRAG, where one claim changed in one run (accuracy SD = 0.004); the uncertainty of every estimate is thus dominated by claim sampling, not by generation noise.

Statistical comparison with the dense baseline. Table 11 reports the paired differences, and Fig. 4 shows the two metrics on which the pipelines differ most, with their confidence intervals, on both benchmarks. After Holm correction, the only differences that remain significant on the main benchmark are the deficits of BM25 on every metric (e.g. Δ Sent nDCG@5 = −0.182 [−0.239, −0.125], p < 0.001; Δ accuracy = −0.100, 1 / 13 discordant claims, McNemar p = 0.002, Holm-adjusted p = 0.013) and the loss of sentence-level ranking quality caused by hybrid fusion (p = 0.032). The gains of CRAG and the agentic pipeline in sentence ranking and evidence recall, and the verdict gains of the reranker and CRAG (7 / 3 and 5 / 2 discordant claims, McNemar p = 0.344 and 0.453), have confidence intervals that include zero and are far from significance, and must be read as numerical observations. The win/loss counts show why: with a strong dense retriever the orchestrating pipelines change the top-5 for only a minority of claims (CRAG improves Sent nDCG@5 on 20 claims and degrades it on 9; the agentic pipeline on 21 and 15), leaving the remaining rankings identical.

Table 11. **Paired comparison with the dense baseline on the main benchmark (per-claim differences; Wilcoxon signed-rank and exact McNemar tests, Holm-adjusted)**

System	Δ Sentence nDCG@5 [95 % CI]	Holm-adjusted *p*	Δ Evidence recall [95 % CI]	Holm-adjusted *p*	Δ Verdict accuracy (system-only / baseline-only correct claims)	Holm-adjusted McNemar *p*
BM25	−0.182 [−0.239, −0.125]	< 0.001	−0.167 [−0.230, −0.106]	< 0.001	−0.100 (1 / 13)	0.013
Hybrid (RRF)	−0.073 [−0.120, −0.026]	0.032	−0.050 [−0.112, +0.013]	0.327	+0.000 (3 / 3)	1.000
Dense + reranker	−0.062 [−0.116, −0.008]	0.149	−0.066 [−0.127, −0.005]	0.145	+0.033 (7 / 3)	1.000
CRAG (corpus-restricted)	+0.029 [−0.009, +0.067]	0.353	+0.019 [−0.021, +0.059]	0.824	+0.028 (5 / 2)	1.000
Agentic RAG	+0.020 [−0.005, +0.046]	0.353	+0.039 [−0.005, +0.085]	0.353	+0.017 (3 / 1)	1.000
GraphRAG (local)	−0.030 [−0.062, −0.002]	0.353	−0.031 [−0.079, +0.014]	0.824	−0.025 (2 / 5)	1.000
LightRAG (dual-level)	+0.000 [−0.017, +0.016]	0.650	−0.010 [−0.039, +0.015]	0.859	−0.008 (1 / 2)	1.000

Source: **compiled by the authors**

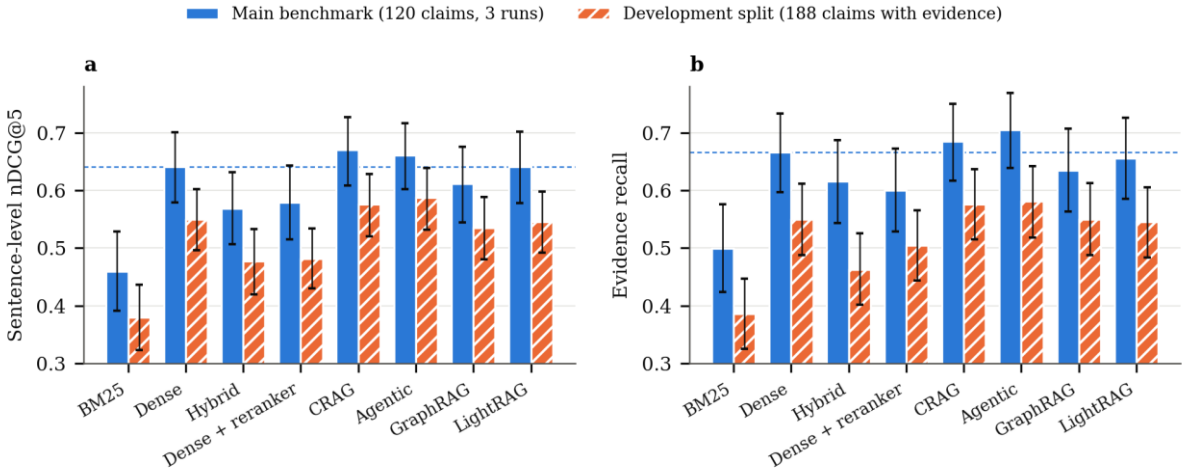


Fig. 4. **Evidence-grounded quality of the eight compared pipelines on the two benchmarks:** a – **Sentence-level nDCG@5; b – evidence recall of the eight pipelines on the main benchmark and on the development split; whiskers are 95 % bootstrap confidence intervals; the dashed line marks the dense baseline on the main benchmark**

Source: **compiled by the authors**

Online efficiency and pipeline diagnostics Table 12 reports the latency distribution, the number of model calls and the token usage per query, and Fig. 5 places every pipeline on the quality–cost plane. The four baselines and GraphRAG answer a claim in 0.88-0.95 s (median), of which more than 94 % is the single generation call; retrieval itself costs 10 ms for exact dense search, 52 ms with reranking, and 22 ms for the graph neighbourhood. Every additional orchestration call adds roughly one generation latency: LightRAG needs 2 calls and 1.72 s, CRAG 2.57 calls and 2.38 s, and the agentic pipeline 2.95 calls and 2.44 s (mean 2.96 s, 95th percentile 4.6 s), i.e. 2.6 times the median latency of the dense baseline.

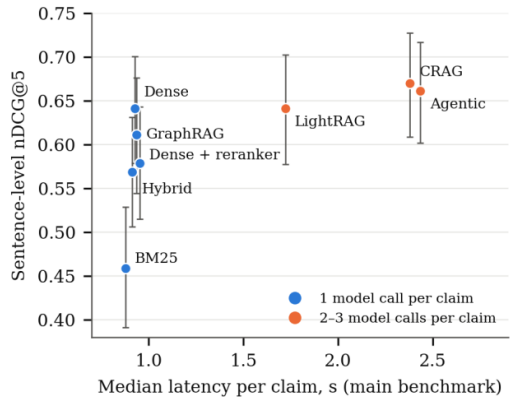


Fig. 5. **Quality-cost trade-off on the main benchmark: sentence-level nDCG@5 (with 95 % confidence intervals) against median end-to-end latency per claim**

Source: **compiled by the authors**

Prompt tokens grow accordingly (405 per query for the baseline, 838 for CRAG, 1351 for the agentic pipeline); at the list prices in effect during the runs (0.30 and 2.50 USD per million prompt and output tokens) this corresponds to an indicative cost of about 0.038, 0.087, and 0.121 US cents per claim respectively. The corrective evaluator classified the initial retrieval as Correct for 42.5 % of the decisions, Ambiguous for 53.9 % and Incorrect for 3.6 % (mean maximum evaluator score 0.92), so the corpus-restricted substitute for web search was rarely triggered. The agentic controller issued no additional search for 51.7 % of the claims, exhausted the budget of three decisions for 24.2 % and made 1.93 decisions (1.17 extra searches) on average. Offline graph construction for the 500 abstracts took 38 min and 535 k output tokens (about 1.42 USD), i.e. roughly 3770 dense-baseline queries; because the graph pipelines add no online calls (GraphRAG) or one call (LightRAG), this cost is amortised only if the graph is reused across many queries.

Robustness on the standard development split. Table 13 reports the development-split experiment over the full corpus, in which 112 of the 300 claims have no evidence in the corpus and the verdict task is three-class. With a corpus ten times larger, document retrieval is no longer saturated (Doc nDCG@5 0.643-0.868), and the ordering of the main benchmark is preserved and sharpened: CRAG and the agentic pipeline lead at both levels (Sent nDCG@5 0.575 and 0.587 versus 0.550 for the dense baseline; the agentic difference is +0.037 [+0.002, +0.072], unadjusted p = 0.024, Holm-adjusted p = 0.094), LightRAG tracks the baseline, GraphRAG is slightly below it, and hybrid fusion (0.477) and reranking (0.482) are now significantly worse than dense retrieval at both levels (Holm-adjusted $p \leq 0.010$). Three-class accuracy ranges from 0.670 (BM25) to 0.767 (agentic; macro-F1 0.761), and no verdict difference to the dense baseline is significant. Table 14 gives the per-class precision, recall, and F1. The joint verdict-and-rationale score lies between 0.600 (BM25) and 0.703 (CRAG; dense baseline 0.690). The NOT ENOUGH INFO class exposes a trade-off between abstention and coverage: the pipelines abstain correctly on 81–89 % of the claims without evidence but issue an unsupported verdict on the remaining 11-19 %, and they abstain wrongly on 21 % (CRAG, agentic) to 43 % (BM25) of the claims that do have evidence. Better retrieval thus reduces false abstentions (accuracy on evidenced claims 0.734 for the agentic pipeline and CRAG against 0.702 for the baseline and 0.537 for BM25) at the price of a few more unsupported verdicts. The harder corpus also changes the behaviour of the orchestrating pipelines: the corrective evaluator judged the initial retrieval *Incorrect* for 19.3 % of the claims (against 3.6 % on the main benchmark), the agentic controller exhausted its budget for 57 % of the claims (2.06 extra searches on average, median latency 3.73 s), and three CRAG reformulation calls degenerated into runaway outputs of about 66 k tokens and four minutes each, which inflate the mean CRAG latency to 4.9 s while the median stays at 2.47 s – a practical argument for output-length caps on orchestration calls. Graph extraction for the full corpus consumed 5.46 M output tokens and 6.5 h. For context, the strongest supervised SciFact systems, which are trained on the task and read whole abstracts, such as MultiVerS [43], report sentence-level selection-and-label F1 around 0.65-0.70 on this split; the pipelines here are zero-shot, cite at most five sentences and are not trained on SciFact, so the comparison is indicative only.

ISSN 2663-0176 (Print)
ISSN 2663-7731 (Online)

Table 12. **Online efficiency on the main benchmark (per query; warm-up excluded; sequential execution; 351 measurements per system)**

System	Mean latency, ms	Median latency, ms	Latency SD, ms	Latency p95, ms	Retrieval and orchest-ration, ms	Genera-tion, ms	Model calls per claim	Prompt tokens per claim	Output tokens per claim
BM25	897	878	149	1120	7	890	1.00	379	92
Dense (baseline)	951	926	239	1143	10	941	1.00	405	102
Hybrid (RRF)	929	914	181	1158	18	911	1.00	397	98
Dense + reranker	974	954	179	1212	52	922	1.00	396	99
CRAG (corpus-restricted)	2318	2378	490	2868	1385	933	2.57	838	246
Agentic RAG	2956	2435	1180	4614	2016	940	2.95	1351	323
GraphRAG (local)	950	937	119	1167	22	928	1.00	406	103
LightRAG (dual-level)	1744	1721	242	2061	822	922	2.00	513	171

Source: **compiled by the authors**

Table 13. **Development-split results (300 claims, full corpus of 5,183 abstracts, three-class verdict, one run; retrieval and evidence metrics on the 188 claims with evidence; 95 % confidence intervals in brackets)**

System	Document nDCG@5	Sentence nDCG@5	Sen-tence MRR	Evidence recall	Ver-dict accura cy	Joint verdict-and-ratio-nale score	Macro -F1	F1 NOT ENOU GH INFO	Valid JSON share	Median latency, ms
BM25	0.643 [0.583, 0.701]	0.379 [0.323, 0.436]	0.450	0.386 [0.326, 0.447]	0.670	0.600	0.655	0.685	1.000	921
Dense (baseline)	0.835 [0.792, 0.875]	0.550 [0.496, 0.602]	0.606	0.550 [0.488, 0.612]	0.757	0.690	0.744	0.757	1.000	853
Hybrid (RRF)	0.775 [0.725, 0.824]	0.477 [0.420, 0.533]	0.556	0.463 [0.402, 0.525]	0.713	0.637	0.700	0.705	1.000	954
Dense + reranker	0.731 [0.679, 0.781]	0.482 [0.430, 0.534]	0.545	0.505 [0.443, 0.566]	0.733	0.653	0.711	0.732	1.000	906
CRAG (corpus-restricted)	0.868 [0.827, 0.906]	0.575 [0.521, 0.629]	0.649	0.576 [0.515, 0.636]	0.763	0.703	0.754	0.752	0.997	2470
Agentic RAG	0.866 [0.825, 0.903]	0.587 [0.532, 0.639]	0.646	0.581 [0.519, 0.642]	0.767	0.697	0.761	0.757	1.000	3728
GraphRAG (local)	0.801 [0.751, 0.847]	0.535 [0.480, 0.588]	0.579	0.549 [0.488, 0.612]	0.740	0.683	0.735	0.738	0.997	921
LightRAG (dual-level)	0.846 [0.802, 0.884]	0.546 [0.491, 0.598]	0.606	0.545 [0.483, 0.605]	0.750	0.697	0.740	0.744	1.000	1741

Source: **compiled by the authors**

Discussion. Under controlled conditions, the next-generation architectures show only small numerical gains over the strong dense baseline. Corrective retrieval and agentic query reformulation improve sentence-level evidence ranking and recall by 2 to 4 points on both benchmarks. However, these gains do not remain significant after correction for multiple comparisons. Document-level retrieval is largely saturated on the main benchmark, but not on the full-corpus development split. Verdict accuracy varies little with the retriever. Generator decision bias may explain this pattern, but a single-generator experiment cannot establish that explanation. Lexical fusion and cross-encoder reranking reduce sentence-level ranking quality with the off-the-shelf components tested here.

Table 14. **Per-class verdict quality on the development split (300 claims, three-class verdict, one run; 124 SUPPORTS, 64 REFUTES, and 112 NOT ENOUGH INFO gold labels)**

System	Precision SUPPORTS	Recall SUPPORTS	F1 SUPPORTS	Precision REFUTES	Recall REFUTES	F1 REFUTES	Precision NOT ENOUGH INFO	Recall NOT ENOUGH INFO	F1 NOT ENOUGH INFO
BM25	0.886	0.565	0.690	0.756	0.484	0.590	0.556	0.893	0.685
Dense (baseline)	0.875	0.734	0.798	0.719	0.641	0.678	0.683	0.848	0.757
Hybrid (RRF)	0.876	0.685	0.769	0.706	0.562	0.626	0.612	0.830	0.705
Dense + reranker	0.910	0.734	0.812	0.636	0.547	0.588	0.648	0.839	0.732
CRAG (corpus-restricted)	0.864	0.766	0.812	0.729	0.672	0.699	0.700	0.812	0.752
Agentic RAG	0.868	0.742	0.800	0.730	0.719	0.724	0.702	0.821	0.757
GraphRAG (local)	0.845	0.702	0.767	0.750	0.656	0.700	0.664	0.830	0.738
LightRAG (dual-level)	0.858	0.734	0.791	0.732	0.641	0.683	0.674	0.830	0.744

Source: **compiled by the authors**

Added components therefore need task-specific validation against dense retrieval. The graph pipelines provide no quality advantage for these local queries over short abstracts, despite substantial offline extraction cost. It should be mentioned that these findings apply only to the tested configurations: CRAG has no web augmentation, the graph pipelines omit community summaries, and the agent has one tool and a three-decision budget. Global questions, full-text documents, or other generators may therefore yield different results.

CONCLUSIONS

This study presented a controlled, statistically tested benchmarking comparison of eight retrieval-augmented pipelines for scientific claim verification on abstract corpora – sparse, dense, hybrid, and reranking baselines and corpus-restricted instantiations of Agentic RAG, GraphRAG, LightRAG, and Corrective RAG – on a fixed SciFact subset and on the standard open development split. By fixing the sentence-level index, the embedding model, the generation model, the prompt, and the output contract, and by repeating every run three times with interleaved execution, the design attributes the observed differences to the retrieval and orchestration logic of the specific implementations compared. The following conclusions are drawn.

1. Document-level retrieval on abstract corpora is saturated by the shared dense index: on the main benchmark Doc nDCG@5 lies between 0.906 and 0.943 for all E5-based pipelines and no paired difference to the dense baseline (0.933) is significant; only BM25 (0.812) is significantly worse. Coarse document retrieval therefore does not discriminate among the architectures, and evaluation must be carried out at the sentence level.

2. Corrective re-retrieval and agentic query reformulation are the only mechanisms that show a consistent numerical improvement in sentence-level evidence ranking and evidence recall over dense retrieval, and they do so consistently on both benchmarks (Sent nDCG@5 0.670 and 0.661 versus 0.641; evidence recall 0.704 versus 0.666 on the main benchmark; Δ Sent nDCG@5 = +0.037 [+0.002, +0.072] on the development split), but the effects are small – two to four points – and do not survive Holm correction; they must be reported as numerical observations with intervals, not as established superiority.

3. Strong conventional baselines are not automatically better: reciprocal-rank fusion with BM25 and cross-encoder reranking with BAAI/bge-reranker-base degrade sentence-level ranking relative to E5 dense retrieval on both benchmarks (main: Sent nDCG@5 0.569 and 0.579 versus 0.641; development split: 0.477 and 0.482 versus 0.550,

Holm-adjusted $p \leq 0.010$), whereas LightRAG reproduces the dense ranking and GraphRAG's local neighbourhood slightly restricts it. In this task the embedding model, not the orchestration, determines retrieval quality, and every added component must be validated against the plain dense ranking.

4. Verdict accuracy (0.758-0.892 with two classes, 0.670-0.767 with three) varied little with the retriever and was dominated by the generator's decision bias, which a single-generator design can expose but not disentangle: under the two-class contract every pipeline recognises all REFUTES claims and errs only by refuting supported ones, and under the three-class contract the pipelines abstain wrongly on 21-43 % of evidenced claims while issuing unsupported verdicts for 11-19 % of claims without evidence. Better retrieval shifts this trade-off towards fewer false abstentions. On the main benchmark, BM25 is the only pipeline showing a statistically significant deficit in verdict accuracy relative to the dense baseline after correction; no other pipeline differs significantly. Three repeats at temperature 0 were identical except for one CRAG claim in one run, so the uncertainty of all estimates stems from claim sampling.

5. The price of the small gains is two to three times the online cost: the dense baseline answers a claim in a median of 0.93 s with one model call and 405 prompt tokens, CRAG in 2.38 s with 2.57 calls, and the agentic pipeline in 2.44 s (95th percentile 4.6 s) with 2.95 calls and 1351 prompt tokens; the corrective evaluator triggered re-retrieval for 57 % of the decisions, and the agent used its full budget for 24 % of the claims. Graph pipelines add little online cost but require an offline extraction pass (38 min for 500 abstracts, 6.5 h for 5,183) whose cost exceeds thousands of baseline queries; runaway orchestration outputs observed on the full corpus show that output-length caps are a necessary deployment safeguard.

6. The evidence-grounded protocol of Algorithm 1 – dual-granularity ranking metrics, evidence metrics on the cited sentences, per-class verdict metrics, confidence intervals and paired tests, and per-query cost – proved more discriminating than document-level metrics alone and is recommended for future benchmarks of retrieval-augmented systems on scientific abstract corpora.

These findings have several limitations. They concern corpus-restricted implementations and one generator, not the full reference architectures. Sentence-level chunking differs from common token-window approaches. Incomplete rationale annotations also count plausible but unannotated evidence as false positives. Keyword-selected hard negatives disadvantage lexical retrieval on the main benchmark, so those results should be read alongside the full-corpus development split. SciFact may occur in the generator's pre-training data. Both corrective and agentic pipelines use query reformulation and rank fusion. An untested minimal ablation, dense retrieval with one reformulation call and rank fusion, is needed to isolate the evaluator's or controller's contribution. Statistical power is limited. With 120 claims, minimum detectable paired differences at $\alpha = 0.05$ and power 0.8 are about 0.04 for sentence-level nDCG@5 and 0.06 for evidence recall. The latter is about 0.05 for the 188 evidenced development claims. Observed gains of 0.02–0.04 lie near these thresholds, so non-significance does not establish equivalence. Latencies include API network and service time and are comparable only within a run. Conclusions should not be generalised from local claims over abstracts to global questions or full-text documents.

Future work should cover full-text documents and global questions that use GraphRAG community summaries. It should compare generators, add learned sparse retrievers and fine-tuned rerankers, and test adaptive routing to the cheapest adequate pipeline.

Conflict of interests: Authors declare that they do not have conflict of interests, in particular financial, personal, copyright or any of a different nature, which could have influenced the research, as well as the results published in this article

Financing: Research was carried out without financial support

Accessibility data: The benchmark code; the identifiers of the claims, abstracts, and sentence chunks; all prompts; the raw per-claim outputs; and the analysis tables that underlie every figure and table in this article are openly available in the repository https://github.com/neocode/scifact-rag-benchmark (MIT licence). The SciFact dataset is distributed by the Allen Institute for AI under its original license

Using artificial intelligence tools: Authors confirm that they did not use instrumental AI tools for writing this work

REFERENCES

1. Hagos, D. H., Battle, R. & Rawat, D. B. "Recent advances in generative ai and large language models: Current Status, challenges, and perspectives". *IEEE Transactions on Artificial Intelligence*. 2024; 5 (12): 5873–5893, https://www.scopus.com/pages/publications/85201754568?origin=resultslist.
DOI: https://doi.org/10.1109/TAI.2024.3444742.

2. Huangm, L., et al. "A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions". *ACM Transactions on Information Systems*. 2025; 43 (2): 1–55, https://www.scopus.com/pages/publications/85219749251?origin=resultslist.
DOI: https://doi.org/10.1145/3703155.

3. Hu, R., Zhong, J., Ding, M., Ma, Z. & Chen, M. "Evaluation of hallucination and robustness for large language models". In: *Proc. IEEE 23rd Int. Conf. Software Quality, Reliability, and Security Companion (QRS-C)*. 2023. p. 374–382, https://www.scopus.com/pages/publications/85186748853?origin=resultslist. DOI: https://doi.org/10.1109/QRS-C60940.2023.00089.

4. Neretin, O. S. &. Kharchenko, V. S. "IMECA method of risk-based assessment and ensuring cybersecurity of large language models". *Herald of Advanced Information Technology*. 2026; 9 (1): 60–70, https://www.scopus.com/pages/publications/105031947081?origin=resultslist.
DOI: https://doi.org/10.15276/hait.09.2026.05.

5. Lewis, P., et al. "Retrieval-augmented generation for knowledge-intensive NLP tasks". In *Advances in Neural Information Processing Systems (NeurIPS)*. 2020; 33: 9459–9474, https://www.scopus.com/pages/publications/85108449607?origin=resultslist. DOI: https://doi.org/10.48550/arXiv.2005.11401.

6. Tran, T. V., Nguyen, T. K. T. & Pham, T. N. L. "Context-aware educational chatbots via retrieval-augmented generation". *Herald of Advanced Information Technology*. 2026; 9 (3): 289–306, https://www.scopus.com/pages/publications/105044438253?origin=resultslist.
DOI: https://doi.org/10.15276/hait.09.2026.19.

7. Fan, W., Ding, Y., Ning, L., Wang, S., Li, H., Yin, D., Chua, T.-S. & Li, Q. "A survey on RAG meeting LLMs: Towards retrieval-augmented large language models". In: *Proc. 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '24)*. 2024. p. 6491–6501, https://www.scopus.com/pages/publications/85199415229?origin=resultslist.
DOI: https://doi.org/10.1145/3637528.3671470.

8. Waddenm D., et al. "Fact or fiction: Verifying scientific claims". In *Proc. 2020 Conf. Empirical Methods in Natural Language Processing (EMNLP)*. 2020. p. 7534–7550, https://www.scopus.com/pages/publications/85118423252?origin=resultslist. DOI: https://doi.org/10.18653/v1/2020.emnlp-main.609.

9. Edgem, D., et al. "From local to global: A graph RAG approach to query-focused summarization". *arXiv*. 2024. DOI: https://doi.org/10.48550/arXiv.2404.16130.

10. Guo, Z., Xia, L., Yu, Y., Ao, T. & Huang, C. "LightRAG: Simple and fast retrieval-augmented generation". *arXiv*. 2024. DOI: https://doi.org/10.48550/arXiv.2410.05779.

11.Yan, S.-Q., Gu, J.-C., Zhu, Y. & Ling, Z.-H. "Corrective retrieval augmented generation". *arXiv*. 2024. DOI: https://doi.org/10.48550/arXiv.2401.15884.

12. Asai, A., Wu, Z., Wang, Y., Sil, A. & Hajishirzi, H. "Self-RAG: Learning to retrieve, generate, and critique through self-reflection". In: *Proc. 12th Int. Conf. Learning Representations (ICLR)*. 2024. DOI: https://doi.org/10.48550/arXiv.2310.11511.

13.Acharya, D. B., Kuppan, K. & Divya, B. "Agentic AI: Autonomous intelligence for complex goals – A comprehensive survey". *IEEE Access*. 2025; 13: 18912–18936, https://www.scopus.com/pages/publications/85216361734?origin=resultslist. DOI: https://doi.org/10.1109/ACCESS.2025.3532853.

14.Pati, A. K. "Agentic AI: A comprehensive survey of technologies, applications, and societal implications". *IEEE Access*. 2025; 13: 151824–151837, https://www.scopus.com/pages/publications/105010010744?origin=resultslist. DOI: https://doi.org/10.1109/ACCESS.2025.3585609.

15. Karpukhin, V., et al. "Dense passage retrieval for open-domain question answering". In: *Proc. 2020 Conf. Empirical Methods in Natural Language Processing (EMNLP)*. 2020. p. 6769–6781, https://www.scopus.com/pages/publications/85118435873?origin=resultslist.
DOI: https://doi.org/10.18653/v1/2020.emnlp-main.550.

16. Zhao, W. X., Liu, J., Ren, R. & Wen, J.-R. “Dense text retrieval based on pretrained language models: A survey”. *ACM Transactions on Information Systems*. 2024; 42 (4):. 89, https://www.scopus.com/pages/publications/85186365769?origin=resultslist.
DOI: https://doi.org/10.1145/3637870.

17. Thakur, N., Reimers, N., Rücklé, A., Srivastava, A. & Gurevych, I. “BEIR: A heterogeneous benchmark for zero-shot evaluation of information retrieval models”. In: *Proc. NeurIPS Datasets and Benchmarks Track*. 2021, https://www.scopus.com/pages/publications/105000299897?origin=resultslist. DOI: https://doi.org/10.48550/arXiv.2104.08663.

18. Pan, J. J., Wang, J. & Li, G. “Survey of vector database management systems”. *The VLDB Journal*. 2024; 33 (5): 1591–1615, https://www.scopus.com/pages/publications/85198748615?origin=resultslist. DOI: https://doi.org/10.1007/s00778-024-00864-x.

19. Speranskyy, V. O., Maksymenko H. E. & Gromova, A. “Research on the quality of automated program code generation using neural networks”. *Applied Aspects of Information Technology*. 2026; 9 (3): 296–309. DOI: https://doi.org/10.15276/aait.09.2026.20.

20. Gorokhovatskyi, O. “Perplexity-based AI-generated text classification in ukrainian using small language models”. *Advanced Information Systems*. 2026; 10 (3): 5–12, https://www.scopus.com/pages/publications/105044453147?origin=resultslist. DOI: https://doi.org/10.20998/2522-9052.2026.3.01.

21. Ji, S., Pan, S., Cambri,a E., Marttinen P. & Yu, P. S. “A survey on knowledge graphs: Representation, acquisition, and applications”. *IEEE Transactions on Neural Networks and Learning Systems*. 2022; 33 (2): 494–514, https://www.scopus.com/pages/publications/85105114155?origin=resultslist. DOI: https://doi.org/10.1109/TNNLS.2021.3070843.

22. Liang, K., et al. “A survey of knowledge graph reasoning on graph types: Static, dynamic, and multi-modal”. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 2024; 46 (12): 9456–9478, https://www.scopus.com/pages/publications/85197053678?origin=resultslist.
DOI: https://doi.org/10.1109/TPAMI.2024.3417451.

23. Osiichuk, I. V., Zagorodna, N. V., Skorenkyy, Y. L., Karpinski, M. P. & Tokkuliyeva, A. K. “The taxonomy of knowledge graph validation approaches”. *Applied Aspects of Information Technology*. 2026; 9 (3): 326–339. DOI: https://doi.org/10.15276/aait.09.2026.22.

24. Pustovoitov, P., Kompaniiets, V., Liu, Y., Sokol, G. & Awad-Alla, M. “Data model of components of complex technical systems based on semantic networks”. *Advanced Information Systems*. 2025; 9 (4): 17–22, https://www.scopus.com/pages/publications/105018817830?origin=resultslist.
DOI: https://doi.org/10.20998/2522-9052.2025.4.03.

25. Pan, S., Luo, L., Wang, Y., Chen, C., Wang, J. & Wu, X. “Unifying large language models and knowledge graphs: A roadmap”. *IEEE Transactions on Knowledge and Data Engineering*. 2024; 36 (7): 3580–3599, https://www.scopus.com/pages/publications/85182350576?origin=resultslist.
DOI: https://doi.org/10.1109/TKDE.2024.3352100.

26. Wang, Y., Lipka, N., Rossi, R. A., Siu, A., Zhang, R. & Derr, T. “Knowledge graph prompting for multi-document question answering”. In: *Proc. 38th AAAI Conf. Artificial Intelligence*. 2024; 38 (17): 19206–19214, https://www.scopus.com/pages/publications/85188263953?origin=resultslist.
DOI: https://doi.org/10.1609/aaai.v38i17.29889.

27. Peng, B., Zhu, Y., Liu, Y., Bo, X., Shi, H., Hong, C., Zhang, Y. & Tang, S. “Graph retrieval-augmented generation: A survey”. *ACM Transactions on Information Systems*. 2025; 44 (2): 1–52, https://www.scopus.com/pages/publications/105030815593?origin=resultslist.
DOI: https://doi.org/10.1145/3777378.

28. Knollmeyer, S., Caymazer, O. & Grossmann, D. “Document GraphRAG: Knowledge graph enhanced retrieval augmented generation for document question answering within the manufacturing domain”. *Electronics*. 2025; 14 (11): 2102, https://www.scopus.com/pages/publications/105007684025?origin=resultslist. DOI: https://doi.org/10.3390/electronics14112102.

29. Xiong, G., Jin, Q., Lu, Z. & Zhang, A. “Benchmarking retrieval-augmented generation for medicine”. In: *Findings of the Association for Computational Linguistics: ACL* 2024. Bangkok, Thailand. 2024. p. 6233–6251. DOI: https://doi.org/10.18653/v1/2024.findings-acl.372.

30. Rau, D., Déjean, H., Chirkova, N., Formal, T., Wang, S., Clinchant, S. & Nikoulina, V. “BERGEN: A benchmarking library for retrieval-augmented generation”. In: *Findings of the Association for Computational Linguistics: EMNLP* 2024. Miami, FL, USA. 2024. p. 7640–7663. DOI: https://doi.org/10.18653/v1/2024.findings-emnlp.449.

31. Jin, J., Zhu, Y., Dong, G., et al. “FlashRAG: A modular toolkit for efficient retrieval-augmented generation research”. In: *Companion Proceedings of the ACM Web Conference 2025 (WWW Companion ’25)*. Sydney, Australia. 2025. DOI: https://doi.org/10.1145/3701716.3715313.

32.Han, H., Ma, L., Wang, Y., et al. “RAG vs. GraphRAG: A systematic evaluation and key insights”. *arXiv*. 2025. DOI: https://doi.org/10.48550/arXiv.2502.11371.

33.Malkov, Y. A. & Yashunin, D. A. “Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs”. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 2020; 42 (4): 824–836, https://www.scopus.com/pages/publications/85081585869?origin=resultslist. DOI: https://doi.org/10.1109/TPAMI.2018.2889473.

34.Traag, V. A., Waltman, L. & van Eck, N. J. “From Louvain to Leiden: Guaranteeing well-connected communities”. *Scientific Reports*. 2019; 9: 5233, https://www.scopus.com/pages/publications/85063527139?origin=resultslist. DOI: https://doi.org/10.1038/s41598-019-41695-z.

35.Yao, S., Zhao, J., Yu, D., Du, N., Shafran, I., Narasimhan, K. & Cao, Y. “ReAct: Synergizing reasoning and acting in language models”. In: *Proc. 11th Int. Conf. Learning Representations (ICLR)*. 2023, https://www.scopus.com/pages/publications/85199863002?origin=resultslist.
DOI: https://doi.org/10.48550/arXiv.2210.03629.

36.Wang, L., et al. “Text embeddings by weakly-supervised contrastive pre-training”. *arXiv*. 2022. DOI: https://doi.org/10.48550/arXiv.2212.03533.

37.Robertson, S. & Zaragoza, H. “The probabilistic relevance framework: BM25 and Beyond”. *Foundations and Trends in Information Retrieval*. 2009; 3 (4): 333–389, https://www.scopus.com/pages/publications/74549177601?origin=resultslist. DOI: https://doi.org/10.1561/1500000019.

38.Cormack, G. V., Clarke, C. L. A. & Buettcher, S. “Reciprocal rank fusion outperforms condorcet and individual rank learning methods”. In: *Proc. 32nd Int. ACM SIGIR Conf. Research and Development in Information Retrieval*. 2009. p. 758–759, https://www.scopus.com/pages/publications/72449144635?origin=resultslist. DOI: https://doi.org/10.1145/1571941.1572114.

39.Xiao, S., Liu, Z., Zhang, P., Muennighoff, N., Lian, D. & Nie, J.-Y. “C-Pack: Packed resources for general chinese embeddings”. In: *Proc. 47th Int. ACM SIGIR Conf. Research and Development in Information Retrieval*. 2024. p. 641–649, https://www.scopus.com/pages/publications/85200606856?origin=resultslist. DOI: https://doi.org/10.1145/3626772.3657878.

40.“Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities”. *arXiv*. 2025. DOI: https://doi.org/10.48550/arXiv.2507.06261.

41. Järvelin, K. & Kekäläinen, J. “Cumulated gain-based evaluation of IR techniques”. *ACM Transactions on Information Systems*. 2002; 20 (4): 422–446, https://www.scopus.com/pages/publications/1842637192?origin=resultslist. DOI: https://doi.org/10.1145/582415.582418.

42. Smucker, M. D., Allan, J. & Carterette, B. “a comparison of statistical significance tests for information retrieval evaluation”. In: *Proc. 16th ACM Conf. Information and Knowledge Management (CIKM)*. 2007. p. 623–632. https://www.scopus.com/pages/publications/63449088172?origin=resultslist. DOI: https://doi.org/10.1145/1321440.1321528.

43. Wadden, D., Lo, K., Wang, L. L., Cohan, A., Beltagy, I. & Hajishirzi, H. “MultiVerS: Improving scientific claim verification with weak supervision and full-document context”. In: *Findings of the Association for Computational Linguistics: NAACL* 2022. 2022. p. 61–76, https://www.scopus.com/pages/publications/85137317573?origin=resultslist.
DOI: https://doi.org/10.18653/v1/2022.findings-naacl.6.

Received 27.08.2026
Received after revision 22.10.2026
Accepted 30.10.2026

DOI: https://doi.org/10.15276/hait.09.2026.34
УДК 004.89

Порівняльне оцінювання архітектур генерації з доповненням пошуком для верифікації наукових тверджень на основі доказів

Шеремет Олексій Іванович[1)]
ORCID: https://orcid.org/0000-0003-1298-3617; sheremet-oleksii@ukr.net. Scopus Author ID: 57170410800
Штефан Паліс[2)]
ORCID: https://orcid.org/0000-0002-1941-2289; stefan.palis@tu-clausthal.de. Scopus Author ID: 24766914900
Садовой Олександр Валентинович[3)]
ORCID: https://orcid.org/0000-0001-9739-3661; sadovoyav@ukr.net. Scopus Author ID: 57205432765
Подлєсний Сергій Володимирович[1)]
ORCID: https://orcid.org/0000-0001-8271-4004; spodlesny2318@gmail.com. Scopus Author ID: 56612000700
Шеремет Катерина Сергіївна[1)]
ORCID: https://orcid.org/0000-0003-3783-5274; artks@ukr.net. Scopus Author ID: 57207768511
[1)] Донбаська державна машинобудівна академія, бул. Машинобудівників, 39. Краматорськ, 84313, Україна
[2)] Технічний університет Клаусталя, вул. Лейбніца, 28. Клаусталь-Целлерфельд, 38678, Німеччина
[3)] Дніпровський державний технічний університет, вул. Дніпробудівська, 2. Кам'янське, 51918, Україна

АНОТАЦІЯ

Актуальність. Генерацію з доповненням пошуком зазвичай оцінюють непрямими метриками без перевірки доказовості вердиктів, а нові архітектури рідко порівнюють із сильними базовими за однакових умов. **Метою статті** є визначити за єдиним контрольованим протоколом, чи покращують агентні, графові та коригувальні архітектури пошук доказів і точність вердикту порівняно з лексичним, щільним, гібридним і переранжувальним базовими підходами у верифікації наукових тверджень на корпусах анотацій та якою є ціна цього. **Завдання:** формалізувати протокол як алгоритм; реалізувати вісім конвеєрів на одному індексі речень відкритого еталонного набору біомедичних анотацій із розміченими доказовими реченнями; оцінити їх на фіксованій підмножині з повторними запусками та на стандартному розбитті для розроблення; оцінити витрати. **Методи:** одна модель векторних представлень, генеративна модель і формат відповіді для всіх конвеєрів; метрики ранжування, доказів і вердиктів за класами з бутстреп-інтервалами та парними тестами з поправкою на множинні порівняння. **Наукова новизна:** протокол оцінювання на основі доказів, що порівнює сім сімейств конвеєрів на одному індексі й генераторі зі статистичними тестами та витратами. **Практична значимість:** опубліковані код, ідентифікатори та промпти роблять протокол повторно використовним. **Результати:** коригувальний і агентний конвеєри найкраще ранжували доказові речення і найповніше цитували анотовані обґрунтування, однак жодна відмінність від щільного базового не залишилася значущою після поправки; лексичний базовий підхід був значуще гіршим, а гібридне злиття та переранжування не покращили щільний пошук; точність вердиктів мало залежала від пошуку і визначалася переважно упередженням генератора; агентний конвеєр потребував у два з половиною рази більшої затримки і втричі більшої кількості викликів моделі, ніж щільний базовий. **Висновки:** виграш невеликий, зосереджений у ранжуванні доказових речень і коштує дво-триразового збільшення кількості викликів моделі.

Ключові слова: генерація з доповненням пошуком; верифікація наукових тверджень; оцінювання на основі доказів; агентний пошук; індексування на основі графів знань; порівняльне оцінювання

ABOUT THE AUTHORS



Oleksii I. Sheremet - Doctor of Engineering Sciences, Professor, Head of the Department of Electromechanical Systems of Automation and Electric Drive. Donbas State Engineering Academy, 39, Mashinobudivnykiv Blvd. Kramatorsk, Ukraine ORCID: https://orcid.org/0000-0003-1298-3617; sheremet-oleksii@ukr.net. Scopus ID: 57170410800
***Research field*:** Machine learning and artificial intelligence in general technical problems and electromechanics; predicative analytics based on artificial intelligence technology

Шеремет Олексій Іванович - доктор технічних наук, професор, завідувач кафедри Електромеханічних систем автоматизації Донбаської державної машинобудівної академії, бул. Машинобудівників, 39. Краматорськ, Україна
Галузь досліджень: Машинне навчання та штучний інтелект у загальнотехнічних задачах і електромеханіці; предиктивна аналітика на основі технологій штучного інтелекту



Stefan Palis - Dr.-Ing. habil., Professor of Intelligent Automation Systems, Institute of Electrical Information Technology, Clausthal University of Technology, Leibnizstrasse, 28. Clausthal-Zellerfeld, Germany. ORCID: https://orcid.org/0000-0002-1941-2289; stefan.palis@tu-clausthal.de. Scopus Author ID: 24766914900
***Research field*:** Intelligent automation and control; data-driven modeling and control; machine learning for dynamical systems; nonlinear and distributed-parameter systems

Штефан Паліс - Dr.-Ing. habil., професор інтелектуальних систем автоматизації, Інститут електротехнічних інформаційних технологій, Технічний університет Клаусталя, вул. Лейбніца, 28. Клаусталь-Целлерфельд, Німеччина
Галузь досліджень: Інтелектуальна автоматизація та керування; моделювання та керування на основі даних; машинне навчання для динамічних систем; нелінійні системи та системи з розподіленими параметрами



Oleksandr V. Sadovoi - Doctor of Engineering Sciences, Professor, Department of Electrical Engineering and Electromechanics. Dniprovsky State Technical University, 2, Dniprobudivska, Str. Kamyanske, Ukraine
ORCID: https://orcid.org/0000-0001-9739-3661; sadovoyav@ukr.net. Scopus Author ID: 57205432765
***Research field*:** Optimal control of electromechanical systems

Садовой Олександр Валентинович - доктор технічних наук, професор кафедри Електротехніки та електромеханіки Дніпровського державного технічного університету, вул. Дніпробудівська, 2. Кам'янське, Україна
Галузь досліджень: Оптимальне керування електромеханічними системами



Sergey V. Podlesny - Candidate of Engineering Sciences, Associate Professor, Dean of Faculty of Machine Automation and Information Technology. Donbas State Engineering Academy, 39, Mashinobudivnykiv Blvd. Kramatorsk, Ukraine
ORCID: https://orcid.org/0000-0001-8271-4004; spodlesny2318@gmail.com. Scopus Author ID: 56612000700
***Research field*:** Machine automation and information technologies, with a focus on applied mechanics

Подлєсний Сергій Володимирович - кандидат технічних наук, доцент, декан факультету Автоматизації машинобудування та інформаційних технологій Донбаської державної машинобудівної академії, бул. Машинобудівників, 39. Краматорськ, Україна
Галузь досліджень: Автоматизація машинобудування та інформаційні технології з акцентом на прикладну механіку



Kateryna S. Sheremet - Laboratory Assistant, Department of Intelligent Decision Support Systems. Donbas State Engineering Academy, 39, Mashinobudivnykiv Blvd. Kramatorsk, Ukraine
ORCID: https://orcid.org/0000-0003-3783-5274, artks@ukr.net. Scopus Author ID: 57207768511
***Research field*:** Machine learning; decision support systems

Шеремет Катерина Сергіївна - лаборант кафедри Інтелектуальних систем прийняття рішень Донбаської державної машинобудівної академії, бул. Машинобудівників, 39. Краматорськ, Україна
Галузь досліджень: Машинне навчання; системи підтримки прийняття рішень