



AGENTIC RETRIEVAL-AUGMENTED GENERATION (RAG): AN ADAPTIVE FRAMEWORK FOR EVIDENCE-GROUNDED QUESTION ANSWERING

Divya Ughade¹

¹ Department of Computer Engineering Dattakala Group of Institutions, Faculty of Engineering Pune, Maharashtra, India, India

Corresponding author: divyaughade2006@gmail.com

Guide / Supervisor: Prof Dr. Bere S.S.

ARTICLE ID IJREK-2026-00071	ARTICLE TYPE Research Article	RESEARCH AREA Computer Science	PUBLICATION Volume 1, Issue Issue 1 (2026)
RECEIVED 15 October 2026	REVISED 15 October 2026	ACCEPTED 15 October 2026	PUBLISHED 15 October 2026

Abstract

Retrieval-Augmented Generation (RAG) connects a large language model (LLM) to external knowledge sources so that responses can be based on evidence beyond the model's parameters. Conventional RAG pipelines commonly follow a mostly fixed sequence: retrieve a small set of passages, place them in a prompt, and generate an answer. This approach can be inadequate for complex questions that require multiple searches, tool use, source comparison, or correction when the first retrieval is weak. This paper proposes a modular Agentic RAG framework in which specialized agents coordinate query planning, retrieval, tool selection, evidence grading, answer generation, and self-reflection. The system maintains a bounded feedback loop: if retrieved evidence is insufficient, the planner revises the query or selects an additional source; if evidence is adequate, a generator produces a response with citations and a confidence assessment. The paper synthesizes relevant work on RAG, reasoning-and-acting agents, self-reflection, corrective retrieval, and graph-based retrieval. It also presents an evaluation framework covering answer quality, retrieval and grounding quality, efficiency, and ablation tests. The included diagrams illustrate the proposed architecture, the contrast with single-pass RAG, and the evaluation workflow. Because no implementation or benchmark run is reported here, performance charts are explicitly conceptual and the evaluation plan is a proposal rather than a claim of measured improvement. The contribution is a practical, auditable design blueprint for building and evaluating Agentic RAG systems.

Keywords: Agentic RAG; retrieval-augmented generation; large language models; AI agents; query planning; tool use; self-reflection; grounding; evaluation

I. INTRODUCTION

Large language models have made natural-language interfaces useful for summarization, question answering, coding assistance, and knowledge work. Yet a model's internal parameters do not guarantee that a response reflects the latest or most relevant evidence. A model can produce a fluent answer that is outdated, unsupported, or incorrect. Retrieval-Augmented Generation was introduced as a way to combine parametric language-model knowledge with non-parametric external memory, allowing a model to condition generation on retrieved passages [1]. A conventional RAG workflow usually embeds a query, retrieves relevant chunks from a document collection, optionally reranks them, and supplies the selected context to a generator. This pipeline is relatively simple to operate and can provide traceable evidence. However, its decisions are often made in a fixed order. If the question is ambiguous, the corpus does not contain the answer, or the top retrieved passages are irrelevant, the system may still generate an answer from weak context. Complex questions may also require several sub-questions, multiple knowledge sources, or a tool such as a calculator, database query, or web search. Agentic RAG adds an orchestration layer that can plan steps, choose retrieval or tools, inspect evidence, and decide whether another attempt is warranted. It should not be interpreted as unlimited autonomy. A reliable design uses explicit tool permissions, bounded iteration, source tracking, and a stopping policy. Work on ReAct shows how reasoning and actions can be interleaved; Self-RAG explores adaptive retrieval and reflection; and Corrective RAG evaluates retrieved evidence and can trigger corrective actions [2]–[4]. Recent survey work describes Agentic RAG as a family of designs rather than one universally fixed architecture [5]. This paper proposes a modular architecture intended to balance adaptability with operational control. The objective is not to claim a new benchmark result, but to define a reproducible system design and evaluation protocol. The paper's main contributions are: (1) a role-based

Agentic RAG architecture; (2) an evidence-sufficiency loop with explicit stopping conditions; (3) a benchmark and logging framework for quality, grounding, latency, and cost; and (4) a discussion of safety, limitations, and deployment considerations.

II. BACKGROUND AND LITERATURE REVIEW

A. Retrieval-Augmented Generation Lewis et al. formalized a RAG approach that combines a sequence-to-sequence generator with a dense vector index used as non-parametric memory [1]. The key idea is that generation can be conditioned on passages retrieved for a query, which can improve specificity and provide a pathway for updating knowledge without retraining the entire language model. Practical RAG systems typically include document ingestion, text cleaning, chunking, embedding, indexing, retrieval, optional reranking, prompt construction, and answer generation. The effectiveness of RAG depends on more than the language model. Chunk boundaries, metadata, embedding quality, retrieval parameters, duplicate handling, source freshness, and context-window limits can all influence the evidence delivered to the generator. A generator cannot reliably cite facts that were not retrieved, and a large context is not automatically a relevant context. These observations motivate systems that evaluate retrieval quality before committing to an answer. B. Reasoning and Tool Use ReAct interleaves reasoning and actions so a language model can interact with external environments rather than only producing a final text sequence [2]. In an Agentic RAG system, the same broad pattern can be applied to search, SQL queries, calculators, APIs, or domain-specific tools. A planner can decompose a broad request into answerable sub-questions, while a router chooses an appropriate retrieval source or tool. Tool use must be constrained: arguments should be validated, permissions should be explicit, and irreversible actions should require human confirmation. C. Reflection and Corrective Retrieval Self-RAG proposes adaptive retrieval and reflection to assess retrieved passages and generated content, rather than always retrieving a fixed number of passages [3]. Corrective Retrieval-

Augmented Generation (CRAG) introduces a retrieval evaluator that assesses the quality of retrieved documents and can trigger alternative retrieval actions when results are weak [4]. These approaches support a general design principle: the system should test whether evidence is sufficient and relevant, not merely whether a search returned results. D. Graph-Based and Multi-Step Retrieval GraphRAG uses entities, relationships, and community summaries to support questions that require understanding themes across a large corpus, including questions that are not well served by retrieving a few isolated chunks [6]. Graph-oriented retrieval can complement vector search when the answer depends on relationships across documents. It also introduces additional indexing and maintenance costs. Agentic orchestration can choose between vector retrieval, keyword search, graph queries, or external tools based on the query’s needs. E. Research Gap and Design Motivation The literature indicates several complementary capabilities—retrieval, planning, tool use, reflection, and alternative search—but practical implementations must decide how to coordinate them, how to limit repeated attempts, and how to evaluate the trade-off between answer quality and cost. A highly autonomous workflow can improve adaptability while also increasing latency, token consumption, tool errors, and debugging difficulty. The proposed framework addresses this engineering gap by defining explicit roles, evidence checks, bounded retries, and a logging-oriented evaluation plan.

III. PROBLEM STATEMENT AND OBJECTIVES

The problem addressed is the brittleness of a single-pass retrieval-and-generation workflow when a user query requires multi-step reasoning or when the initial retrieved context is incomplete, contradictory, stale, or irrelevant. A system that generates from inadequate evidence risks unsupported claims. Conversely, repeatedly searching without a stopping rule can waste resources and still fail to answer the question. Objectives: Route each query to an appropriate retrieval source or tool based on its information need. Decompose complex questions into manageable sub-questions while preserving the original user intent. Assess retrieved evidence for relevance, coverage, recency, and source reliability. Use a bounded re-planning loop when evidence is insufficient, and stop when evidence is adequate or the retry budget is exhausted. Generate answers with source citations, uncertainty statements, and a calibrated confidence indicator. Evaluate answer quality, grounding, retrieval effectiveness, latency, token usage, and tool-call overhead against baselines.

IV. PROPOSED AGENTIC RAG SYSTEM

TABLE I	
Input	User query q, authorized source set S, tool registry T, retry limit R, time/token budget B.
Plan	Classify the query; create sub-questions if needed; select a first retrieval action.
Retrieve	Run the selected search or tool and collect candidates with provenance metadata.
Assess	Rerank and grade the evidence for relevance, coverage, reliability, recency, and conflicts.
Branch	If sufficient, proceed to generation. If insufficient and budget remains, revise the plan and retrieve again. If conflicting, compare sources or qualify the answer.
Generate	Draft an answer grounded in accepted evidence and attach citations to claims.
Reflect	Check coverage, citation support, contradictions, and unsupported claims.
Finalize	Return the verified draft, sources, and uncertainty note; otherwise return a limitation statement.

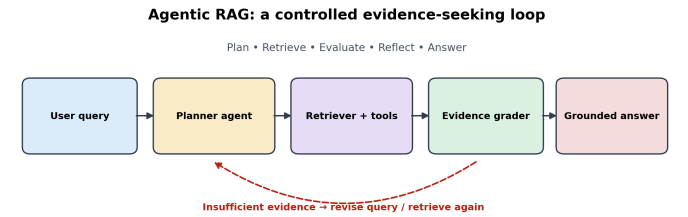


Fig. 1. Original conceptual illustration of the proposed Agentic RAG evidence-seeking loop. The feedback path is bounded by retry and

resource limits.

V. ARCHITECTURE AND WORKFLOW

The architecture follows a data path and a control path. The data path carries documents, chunks, embeddings, retrieved passages, and final citations. The control path carries the plan, tool choice, evidence assessment, retry decision, and stopping condition. Keeping these paths distinct helps with debugging and auditing: an answer failure can be traced to ingestion, retrieval, planning, grading, generation, or citation assembly.

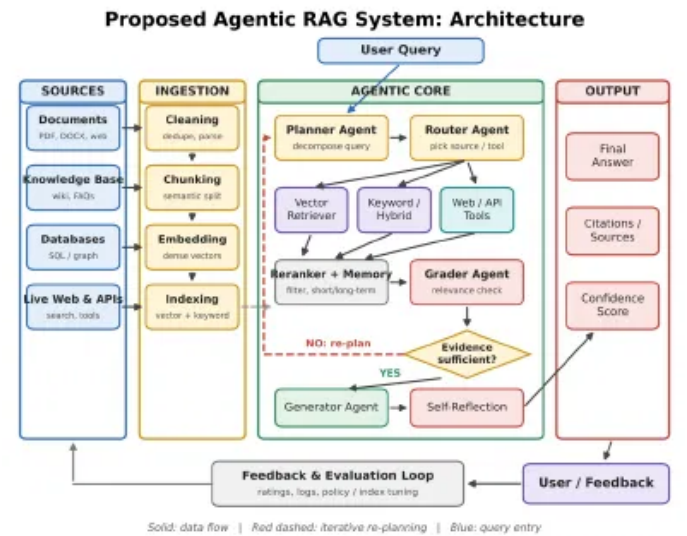


Fig. 2. User-provided proposed Agentic RAG architecture diagram (included from the supplied reference image).

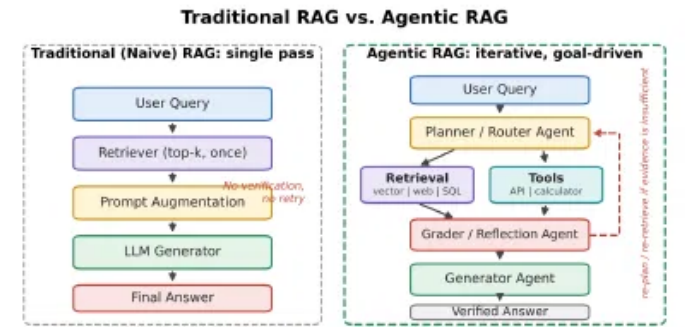


Fig. 3. User-provided comparison of traditional single-pass RAG and iterative, goal-driven Agentic RAG.

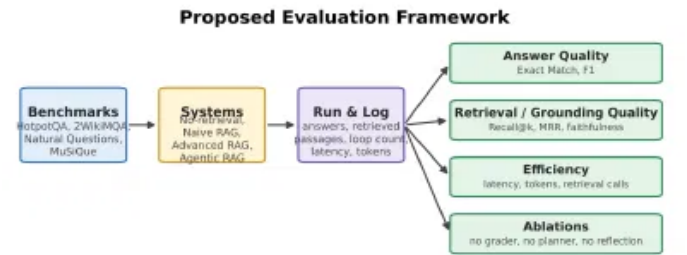


Fig. 4. User-provided evaluation framework diagram showing benchmarks, systems, run logs, and evaluation dimensions.

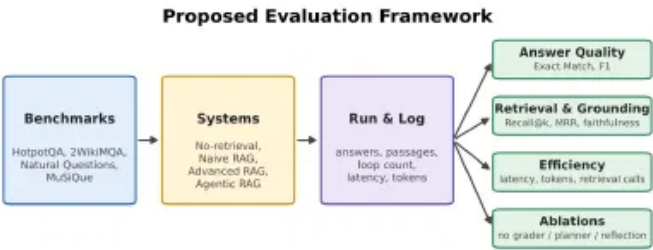


Fig. 5. Second user-provided version of the proposed evaluation framework; retained as supplied.

VI. ALGORITHM AND CONTROL LOGIC

The following high-level procedure defines the control loop. It is intentionally model-agnostic and can be implemented using an orchestration framework, a custom state machine, or a workflow engine. Input User query *q*, authorized source set *S*, tool registry *T*, retry limit *R*, time/token budget *B*. Plan Classify the query; create sub-questions if needed; select a first retrieval action. Retrieve Run the selected search or tool and collect candidates with provenance metadata. Assess Rerank and grade the evidence for relevance, coverage, reliability, recency, and conflicts. Branch If sufficient, proceed to generation. If insufficient and budget remains, revise the plan and retrieve again. If conflicting, compare sources or qualify the answer. Generate Draft an answer grounded in accepted evidence and attach citations to claims. Reflect Check coverage, citation support, contradictions, and unsupported claims. Finalize Return the verified draft, sources, and uncertainty note; otherwise return a limitation statement. A core invariant is that a generated factual claim should either be supported by identifiable evidence or explicitly marked as uncertain. If the source corpus lacks the answer, the system should not treat fluent generation as evidence. In high-stakes applications, the final response should be reviewed by a qualified person and tools that change external state should require authorization.

VII. PROPOSED EVALUATION FRAMEWORK

Evaluation should compare Agentic RAG with at least two baselines: (i) a no-retrieval LLM baseline, where appropriate; and (ii) a conventional single-pass RAG pipeline using the same underlying language model and source corpus. A fair comparison holds the generator, corpus, prompt constraints, and answer format constant where possible. Agentic RAG should be evaluated on the same questions and with repeated runs when model or retrieval behavior is stochastic. A. Benchmark and Test-Set Design The test set should include fact lookup, multi-hop questions, comparison questions, questions with conflicting sources, questions whose answer is absent from the corpus, and queries requiring deterministic tools. Each example should have a reference answer or a documented grading rubric, supporting source passages, and a difficulty label. The dataset should be separated into development and held-out test portions to avoid tuning on the evaluation answers. B. Answer Quality Answer quality can be measured using exact match and token-level F1 for short factoid questions, alongside rubric-based human assessment for multi-sentence answers. Human assessment should rate correctness, completeness, relevance, clarity, and appropriate uncertainty. Automated LLM-as-judge scores can assist at scale but should not be the sole measure because the judge may share biases with the generator. The paper does not report values for these metrics because an experiment has not been conducted. C. Retrieval and Grounding Quality Retrieval should be assessed using Recall@k, Mean Reciprocal Rank (MRR), and nDCG where relevance labels are available. Grounding can be measured as the proportion of factual claims supported by cited passages, citation precision, citation coverage, and the rate of unsupported claims. For multi-hop questions, the evaluation should check whether all required evidence components were retrieved, not only whether one relevant passage appeared in the context. D. Efficiency and Reliability Efficiency measures include end-to-end latency, time to first token, number of retrieval calls, number of tool calls, input/output tokens, and estimated cost per answered query. Reliability measures include tool failure rate, retry rate, percentage of queries that hit the budget limit, and abstention quality when the corpus is insufficient. Results should be reported as averages and distributions, not only a single best run. E. Ablation Study Ablations can isolate the contribution of each component: remove the planner, disable query rewriting, remove reranking, disable the evidence grader,

remove self-reflection, disable web/API tools, or set the retry limit to zero. A component should be considered useful only if its quality gain justifies its latency and cost. A full experiment should report the same test examples for all ablations and document model versions, prompts, retrieval settings, corpus snapshots, and date of execution.

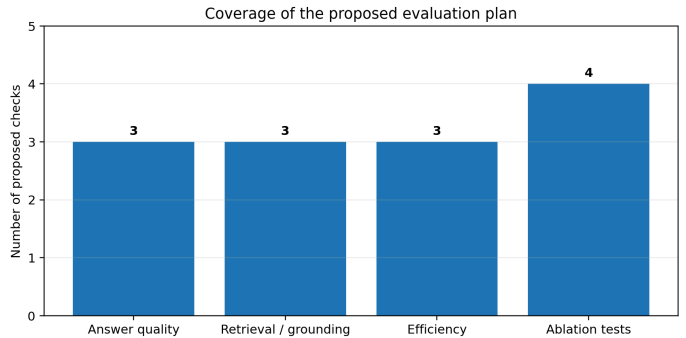


Fig. 7. Number of proposed checks in each evaluation dimension. This chart counts items in the evaluation plan; it is not a performance result.

VIII. DESIGN-LEVEL COMPARISON

The following comparison summarizes expected design characteristics, not benchmark findings. Scores are illustrative judgments on a 1–5 scale intended to make trade-offs visible; they must not be interpreted as measured system performance. The scale should be replaced with empirical results after implementation and evaluation.

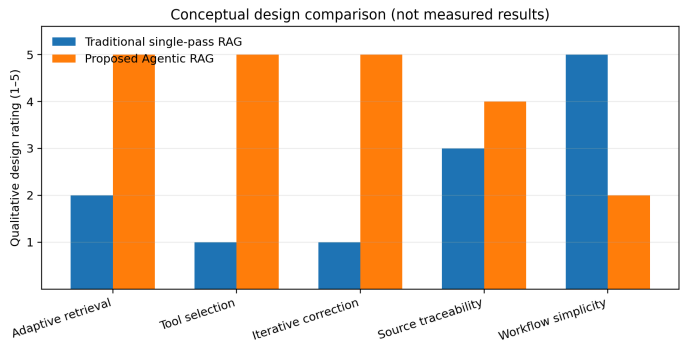


Fig. 8. Conceptual comparison of design capabilities. Ratings are illustrative and are not derived from experimental measurements.

TABLE III		
Dimension	Traditional single-pass RAG	Proposed Agentic RAG
Failure handling	Weak first retrieval may flow into generation	Can retry, qualify, or abstain within limits

IX. SECURITY, PRIVACY, AND GOVERNANCE

Agentic RAG introduces risks beyond ordinary text generation because it can invoke tools and move information between systems. Retrieved documents may contain prompt-injection text designed to manipulate an agent. Tool outputs and documents must therefore be treated as untrusted data; they must not override system instructions, access controls, or tool schemas. The orchestration layer should use allow-listed tools, least-privilege credentials, input validation, output validation, and explicit authorization for consequential actions. Sensitive data should be minimized before indexing and should have retention and deletion policies. Retrieval must enforce document-level permissions for every query, not merely when the index is created. Logs should capture enough information for debugging and audits while avoiding unnecessary storage of personal or confidential content. For external web retrieval, the system should record source URL and retrieval time and distinguish authoritative sources from unverified pages. Confidence is especially important. A self-reported confidence score from an LLM can be poorly calibrated. A safer interface can expose evidence coverage, source agreement, freshness, and whether

the system reached a retry limit. For regulated or high-impact decisions, Agentic RAG should support human review and should not be presented as an autonomous substitute for professional judgment.

X. APPLICATIONS

Potential applications include enterprise knowledge assistants that answer questions over policy documents; research assistants that gather and compare cited sources; customer-support systems that search product manuals and case records; software engineering assistants that retrieve documentation and run approved tests; and analytics assistants that combine natural-language questions with SQL or deterministic calculations. The appropriate architecture depends on source quality, the cost of a wrong answer, privacy constraints, and the availability of domain-specific evaluation data.

XI. LIMITATIONS

This paper proposes an architecture and evaluation protocol; it does not present an implemented system, user study, benchmark scores, statistical significance tests, or a production deployment. Therefore, no claim is made that the proposed framework outperforms a baseline in measured accuracy or latency. The comparison chart is conceptual and the evaluation chart counts proposed checks only. Further work must implement the design, publish the configuration and dataset protocol, conduct reproducible experiments, and report both improvements and regressions. Agentic workflows also face practical limitations: more calls can increase cost and latency; planners can decompose a query incorrectly; evidence graders can miss subtle contradictions; tool APIs can fail; sources may be stale or biased; and retries may reinforce an initial mistaken assumption. The architecture reduces some failure modes through explicit checks but cannot eliminate hallucinations or guarantee correctness.

XII. CONCLUSION AND FUTURE SCOPE

This paper presented a modular Agentic RAG framework that extends a conventional retrieval-and-generation pipeline with planning, routing, tool use, evidence grading, generation, reflection, and bounded re-planning. Its main design principle is to make retrieval adaptive while retaining traceability and control. An explicit sufficiency gate and stopping policy help distinguish a successful evidence search from a search that merely returned documents. Source citations and structured logs provide the basis for auditing and systematic evaluation. Future work should implement the architecture and evaluate it on public and domain-specific benchmarks; compare vector, keyword, and graph retrieval; test alternative evidence graders; measure the effect of each agent through ablation; and study cost-quality trade-offs across retry budgets. Additional research should investigate calibrated confidence, prompt-injection resistance, privacy-preserving retrieval, multilingual sources, and human-in-the-loop review. Empirical conclusions should be added only after controlled runs with documented methods and reproducible

TABLE II

Dimension	Traditional single-pass RAG	Proposed Agentic RAG
Control flow	Mostly fixed retrieval → generation	Planner selects actions and can re-plan
Retrieval	Usually one retrieval stage	Can perform targeted, iterative retrieval
Tool use	Often external or manually wired	Router selects approved tools as needed
Evidence check	May rely on relevance scores or prompt instructions	Explicit sufficiency and conflict assessment
Latency / cost	Usually lower and more predictable	Potentially higher due to additional calls
Auditability	Depends on logging and citation design	Designed for step-level logs and provenance

REFERENCES

[1] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W.-t. Yih, T. Rocktäschel, S. Riedel, and D. Kiela, “Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks,” *Advances in Neural Information Processing Systems*, vol. 33, 2020.

[2] S. Yao, J. Zhao, D. Yu, N. Du, I. Shafraan, K. R. Narasimhan, and Y. Cao, “ReAct: Synergizing Reasoning and Acting in Language Models,” *International Conference on Learning Representations (ICLR)*, 2023.

[3] A. Asai, Z. Wu, Y. Wang, A. Sil, and H. Hajishirzi, “Self-RAG: Learning to Retrieve, Generate, and Critique through Self-Reflection,” *International Conference on Learning Representations (ICLR)*, 2024.

[4] S.-Q. Yan, J.-C. Gu, Y. Zhu, and Z.-H. Ling, “Corrective Retrieval Augmented Generation,” *arXiv:2401.15884*, 2024.

[5] A. Singh, A. Ehtesham, S. Kumar, T. T. Khoei, and A. V. Vasilakos, “Agentic Retrieval-Augmented Generation: A Survey on Agentic RAG,” *arXiv:2501.09136*, revised 2026.

[6] D. Edge et al., “From Local to Global: A Graph RAG Approach to Query-Focused Summarization,” *arXiv:2404.16130*, 2024.

[7] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W.-t. Yih, T. Rocktäschel, S. Riedel, and D. Kiela, “Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks,” *arXiv:2005.11401*, 2020.

[8] J. Johnson, M. Douze, and H. Jégou, “Billion-Scale Similarity Search with GPUs,” *IEEE Transactions on Big Data*, vol. 7, no. 3, pp. 535–547, 2021.

[9] V. Karpukhin et al., “Dense Passage Retrieval for Open-Domain Question Answering,” *Proceedings of EMNLP*, pp. 6769–6781, 2020.

[10] N. Reimers and I. Gurevych, “Sentence-BERT: Sentence Embeddings using Siamese BERT- Networks,” *Proceedings of EMNLP-IJCNLP*, pp. 3982–3992, 2019.

[11] R. Nogueira and K. Cho, “Passage Re-ranking with BERT,” *arXiv:1901.04085*, 2019.

[12] K. Guu, K. Lee, Z. Tung, P. Pasupat, and M. Chang, “REALM: Retrieval-Augmented Language Model Pre-Training,” *Proceedings of ICML, PMLR* 119, pp. 3929–3938, 2020.

[13] O. Khattab and M. Zaharia, “ColBERT: Efficient and Effective Passage Search via Contextualized Late Interaction over BERT,” *Proceedings of SIGIR*, pp. 39–48, 2020.

[14] X. Li, “A Review of Prominent Paradigms for LLM-Based Agents: Tool Use, Planning (Including RAG), and Feedback Learning,” *Proceedings of the 31st International Conference on Computational Linguistics (COLING)*, 2025.

[15] S. Borgeaud et al., “Improving Language Models by Retrieving from Trillions of Tokens,” *Proceedings of ICML, PMLR* 162, pp. 2206–2240, 2022.

[16] OpenAI, “GPT-4 Technical Report,” *arXiv:2303.08774*, 2023.

Publication archival note: Official scholarly contribution published under CC BY 4.0 International License. Permanent record on <https://www.ijrek.org> | Article ID: IJREK-2026-00071