Proceedings of the 4th International Conference on Machine Learning and Automation
DOI: 10.54254/2755-2721/2026.39116

Paradigm Evolution and Engineering Selection of Retrieval-Augmented Generation for Large Language Models

Kaijie Zhuang

School of Big Data and Artificial Intelligence, Guangdong University of Finance & Economics, Foshan, China
z2685248755@outlook.com

Abstract. Large language models (LLMs) suffer from two persistent problems in their parametric knowledge: update lag and factual deviation, which hinder their deployment in professional domains. Retrieval-augmented generation (RAG) connects external corpora at inference time, expanding the knowledge boundary without modifying model parameters, and has become the mainstream solution for enterprise knowledge bases, intelligent question answering, and document understanding. This paper divides RAG technology into four stages along its design focus: naive retrieval, precise retrieval, adaptive routing, and agentic reasoning, and examines the principal contradictions and new costs of each stage. On this basis, a hierarchical evolution framework is proposed that cross-compares six levels —retrieval, knowledge, fusion, generation, conflict arbitration, and engineering evaluation —against the four stages, identifying gaps in existing research, especially the unresolved arbitration of multi-source contradictory documents across all stages. The paper also discusses applicable boundaries: when knowledge is already well mastered by the model, when the context window suffices to cover the full corpus, or when the knowledge base is small and sparsely updated, adding a retrieval link only increases latency and noise. Finally, selection guidance is provided along three dimensions: accuracy, computational cost, and robustness, serving as a reference for engineering design and future research.

Keywords: Retrieval-Augmented Generation, Large language models, Agents, Hallucination, Knowledge retrieval

1. Introduction

Large language models (LLMs) have demonstrated strong language capabilities in tasks such as general dialogue, text summarization, and code generation, yet the update lag of parametric knowledge and factual deviation persist: when faced with time-sensitive information and professional knowledge, models still produce answers that contradict facts.

Retrieval-augmented generation provides another knowledge supply path. Lewis et al. connected the non-parametric retrieval module to the sequence-to-sequence model in 2020, and verified the role of non-parametric memory in knowledge-intensive tasks [1]. Dense vector retrieval, cross-encoder re-ranking, and hypothetical document embeddings have been proposed respectively [2-4]. The retrieval link thus evolved from a fixed pipeline to an adaptive system with routing judgment

© 2026 The Authors. This is an open access article distributed under the terms of the Creative Commons Attribution License 4.0 (https://creativecommons.org/licenses/by/4.0/).

DOI: 10.54254/2755-2721/2026.39116

and self-reflection capabilities [5, 6], and accelerated integration with the ReAct-style reasoning-acting agent paradigm [7, 8]. Compared with training knowledge into parameters, non-parametric retrieval saves knowledge in an external database, updates can be completed by adding or deleting documents, and the answers can be traced, so it is widely used in knowledge base applications.

Existing reviews have focused on the classification of RAG systems: Gao et al. summarized them as naive, advanced, and modular paradigms, focusing on sorting out components [9]; Singh et al. specifically reviewed Agentic RAG, focusing on the classification of agent architectures [8]. The difference between this paper and the above work is that agentic retrieval since the end of 2023 is listed as an independent paradigm, and the RAG hierarchical evolution framework is proposed, trying to respond to two problems that have not been fully discussed: what logic does the design focus migrate between stages, and what price is paid for each step of evolution; In engineering practice, different scenarios should match the retrieval architecture of what complexity.

2. Background and preliminary knowledge

2.1. Basic theoretical support

The large language model is based on the transformer architecture [10], and the language rules are written into the parameters after self-supervised pre-training. The problem is that the cost of pre-training is large, and the parameters are solidified after training, which cannot keep up with rapidly changing new facts; If the knowledge is updated by fine-tuning, it is prone to catastrophic forgetting, and each update requires re-training, so the flexibility is poor. Retrieval-augmented generation is proposed to avoid this dilemma.

2.2. Definition of core concepts

The retriever measures the relevance between queries and documents, and filters relevant fragments from large-scale candidate sets. BM25 and DPR represent two paradigms of sparse lexical matching and dense semantic coding respectively [2, 9]. The retrieval quality directly frames the upper limit of the generation quality. The generator is a large language model, which generates answers under the joint condition of query and retrieval context. It needs to weigh the fidelity of evidence and the usefulness of answers. Knowledge base is the carrier of external information, and its data quality, segmentation granularity and update frequency will affect the final output [1, 9].

3. Four stages of technological evolution

As shown in Figure 1, RAG technology has migrated along its design focus through four stages: naive retrieval directly splices vector recall, focusing on supplementing external knowledge; precise retrieval uses dense encoding, re-ranking, and hybrid retrieval to suppress recall noise; adaptive routing dynamically selects paths according to query complexity; and agentic reasoning embeds retrieval into a reasoning cycle to enable multi-hop and self-correction. The four stages have followed one another, each compensating for the shortcomings of its predecessor while introducing new costs. The distinction between adaptive routing and agentic reasoning is that the former inserts a judgment step within a single retrieval-generation cycle, whereas the latter embeds retrieval as a reusable tool into a reasoning loop, allowing evidence to accumulate across multiple steps.

DOI: 10.54254/2755-2721/2026.39116

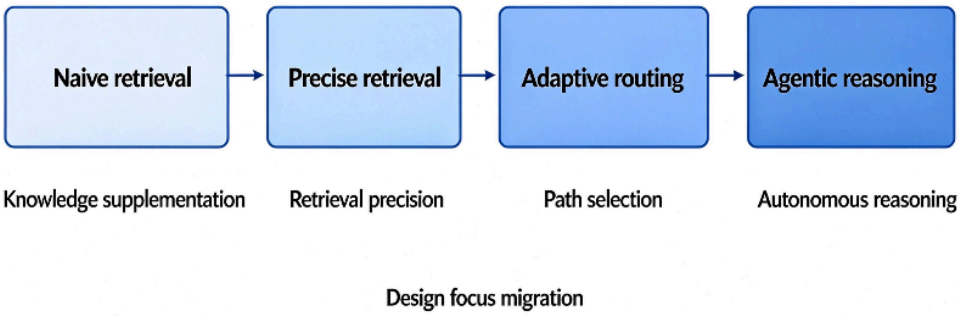


Figure 1. Schematic of the RAG four-stage paradigm transition and design-focus migration (picture credit: original)

3.1. Naive retrieval stage

The early system ran along the one-way pipeline of "document segmentation → vector index → similar recall → condition generation" [9]. Lewis et al. Proposed two marginalization strategies, RAG-Sequence and RAG-Token. The former weighted the retrieval fragments at the sequence level, and the latter dynamically selected the reference fragments at each token position [1]. Izacard and Grave proposed FiD to solve the problem of high coding complexity of multi-passage long sequence: each segment is encoded independently after splicing with the query. The decoder accesses all segments in a unified way at the cross-attention layer, and its performance continues to improve with the increase of the number of recalled segments [11].

At this stage the architecture is straightforward, the deployment threshold is low, and knowledge-base updates only require rebuilding the index. The weaknesses are equally obvious: the quality ceiling is almost entirely determined by the first recall. If semantic units are severed during chunking, or if vector similarity only captures surface lexical overlap, the irrelevant fragments mixed in will interfere with generation. External knowledge reduces the incidence of hallucination but does not eradicate it [12].

3.2. Precise retrieval stage

For the recall noise, the improvement is carried out in three directions: the retriever, the reranker, and the query side. DPR uses a two-tower structure to encode queries and documents into dense vectors respectively, so that paragraphs with similar semantics but different words can be matched [2]; However, two-tower independent encoding is difficult to capture word-level fine-grained interaction, and the BERT cross-encoder re-ranks the coarse candidates to compensate for this accuracy with higher computational overhead [3]. On the query side, HyDE first asks the model to write a hypothetical document, and then uses the vector of the document as a retrieval query to fill in the expression gap between the user's short query and the formal document [4].

Hybrid retrieval has gradually become the default practice in engineering at this stage. BM25 mainly manages the accurate hit of proper names, values and code fragments. Dense retrieval compensates for the weakness of semantic generalization. The two results are sent to reorder after normalization. The cost of doing so mainly lies in the delay and computational power: the cross-encoder has to encode the candidate pairs one by one, and the offline vectorization of all documents also brings the burden of storage and update. Up to now, there is no general solution for how to tune the weights when the two score distributions are inconsistent. The actual deployment often requires

DOI: 10.54254/2755-2721/2026.39116

repeated offline trials for specific business sets. In addition, the coverage of rare words and professional terms in dense retrieval has a great relationship with the distribution of training corpus. Vertical domains such as law and medicine often have to cooperate with the continuous training in the field.

3.3. Adaptive routing stage

The fixed pipeline applies the same retrieval depth to all queries, wasting computation on simple questions and under-retrieving on complex ones. Adaptive routing inserts a lightweight judgment module before retrieval and dynamically selects the execution path according to query complexity.

CRAG deploys a relevance evaluator between retrieval and generation, and determines the retrieval results as correct, ambiguous and incorrect: the Correct tier decomposes and recombines existing documents; the Ambiguous tier supplements evidence via web search; the Incorrect tier discards the original retrieval results and falls back to web search [6]. Self-RAG internalizes the judgment into the generation model, using four types of reflection tokens to let the model decide during generation whether to trigger retrieval or discard passages, and to self-evaluate whether the output is supported by evidence [5]. This kind of mechanism exchanges the output reliability with additional calculation, but the effectiveness depends on the accuracy of judging the module itself; Once the evaluation is wrong, the subsequent actions will be amplified along the wrong branch.

Another approach is the complexity classifier: the lightweight model predicts the query complexity, simple queries bypass retrieval and go straight to generation, the medium query triggers a single retrieval, and the complex query enters the multi-step iterative retrieval. Hierarchical scheduling can reduce the average delay in high-concurrency services at the cost of the error rate of the classifier itself becoming a new bottleneck.

3.4. Agentic reasoning stage

Some queries require multi-hop reasoning across multiple documents, in which case a single retrieval-generation pass cannot obtain all required evidence at once. Agentic retrieval places the retrieval action within a cycle of thought, action, and observation: the model first observes the results from the previous round, determines the next retrieval direction accordingly, and gives the final answer when evidence is sufficient [7, 8].

The architecture is divided into single-agent and multi-agent. The former performs planning, retrieval, and generation internally by a single LLM agent, with compact structure and intuitive debugging; The latter divides the work according to planning, retrieval, verification and response, and achieves stability on complex tasks through information passing between roles [8]. Liang et al. further distinguished between pre-defined reasoning and autonomous reasoning: the former followed a fixed modular pipeline and approximated a fast intuitive system; The latter is multi-step planning by the model, independently determining the retrieval time and tool call, and accompanied by self-reflection [13]. GraphRAG first extracts entities and relationships from documents to build a hierarchical knowledge map, and then generates summaries according to communities, which is superior to traditional vector recall in the query of cross-document synthesis of global topics [14].

The cost is also the most prominent. Multiple rounds of tool calls push up the delay and computing power by an order of magnitude. If the link is too long, it may fall into an invalid cycle, and the overall performance is highly dependent on the quality of model decision-making; In the whole process, problems such as tool call failure, retrieval direction deviation and reflection miscalculation will accumulate along the link.

DOI: 10.54254/2755-2721/2026.39116

4. RAG hierarchical evolution analysis framework

Most of the existing discussions are carried out along the technical module, and few place the problem level and evolution stage together for cross-reference. This paper classifies problems encountered in engineering implementation into six levels and then maps them onto the four stages, yielding the hierarchical evolution framework in Table 1. The layering is essentially determined by the direction of data in the pipeline: the retrieval and knowledge layers address what to retrieve; the fusion layer manages how to combine; the generation layer handles how to answer; the conflict arbitration layer governs how to reconcile contradictory evidence; and the engineering evaluation layer answers whether it is worth it. The first five layers are all concerned with quality, while only the last layer is concerned with cost; these two directions often trade off against each other in practice.

Table 1. RAG hierarchical evolution analysis framework

Problem level	Naive retrieval	Precise retrieval	Adaptive routing	Agentic reasoning
Retrieval layer	Direct vector recall, no fine re-ranking	DPR dense retrieval + re-ranking + hybrid retrieval	CRAG assessment and correction	Multi-hop autonomous retrieval
Knowledge layer	Fixed-length chunking, semantic fragmentation	Semantic-boundary-aware chunking	Dynamic knowledge source switching	graph-based structured knowledge (GraphRAG)
Fusion layer	Direct context splicing	FiD cross-attention fusion	Evaluation before integration	Long-chain evidence accumulation management
Generation layer	Susceptible to interference from irrelevant content	Prompt-engineering constraints	Reflection-token verification	Multi-agent cross-validation
Conflict arbitration layer	No processing, direct splicing of multi-source evidence	Filter by score but no arbitration; relevance ≠ consistency	Evaluator judges relevance/ambiguity/error, but not multi-source contradiction	Multi agent cross verification, unified arbitration mechanism is not yet mature
Engineering evaluation layer	No unified indicator	Offline recall-rate evaluation	RAGAS automatic indicators	End-to-end benchmarks missing

In the six levels, the core contradiction of the retrieval level is how to balance the recall rate and accuracy rate, and the problem of contradictory document recognition has not been really solved through the four stages. Most of the current evaluators tend to judge "correlation" rather than "consistency", and there is no arbitration mechanism for the corpus encountering source conflict, which is the most obvious blank in Table 1. In addition, the segmentation granularity of the knowledge layer has been in common use, the incremental index has been mature in engineering, the long context attentional dilution (lost in the middle [15]) of the fusion layer, and the faithfulness of the generation layer to the retrieval evidence. Although these directions have made progress at all stages, they still lack a unified conclusion across tasks.

Engineering and evaluation are concerned with cost, latency, and measurability. RAGAS proposed automatic indicators such as faithfulness and context precision [16], and Yu et al. organized existing benchmarks [17]. However, most of these benchmarks are based on academic question answering and do not cover enterprise documents or frequently updated knowledge bases.

DOI: 10.54254/2755-2721/2026.39116

Mature systems generally use automatic indicator regression testing, supplemented by small-scale manual sampling for periodic calibration.

5. Engineering selection and applicable boundaries

RAG is not always optimal. In the following three cases, the necessity of the retrieval link should be reconsidered.

When the knowledge itself is already well mastered by the model, retrieval only adds latency. Format conversion, general rewriting, and chitchat queries show limited output-quality differences with and without retrieval, yet they must bear the cost of vector-store maintenance. When the context window is sufficient to accommodate the full corpus, the marginal benefit of chunked retrieval decreases: the long-context model can place the entire document or even a small knowledge base directly into the prompt, bypassing the semantic damage caused by chunking. At this point, the value of retrieval lies more in saving tokens than in improving quality. When the knowledge base is small and updates are sparse, the engineering complexity of the vector index may outweigh its benefits; fine-tuning or simply writing the document into the system prompt is often more cost-effective.

After confirming the need for retrieval, selection is based on query complexity and latency sensitivity. Table 2 compares the four stages along three dimensions.

Table 2. Comparison of accuracy, computational cost, and robustness across the four stages

Evolution stage	Accuracy	Computation and time cost	Robustness
Naive retrieval	Lower; ceiling depends on first recall	Very low; single retrieval	Poor; sensitive to redundant and irrelevant context
Precise retrieval	High, dense retrieval and reordering [2, 3] combined with query rewriting, hybrid retrieval and semantic chunking, better than BM25	Medium; additional latency from cross-encoder re-ranking	Medium; shallow noise can be filtered, but deep logical correlation remains limited
Adaptive routing	High, Self-RAG reflective on-demand retrieval and faithfulness self-assessment [5], CRAG falls back to web search for low-quality retrieval [6], superior to RAG systems of the same scale, relying on the accuracy of reflection	From medium to high, reflection tokens, the evaluator, and external search incur additional overhead	High; active identification and dynamic filtering capability
Agentic reasoning	Complex multi hop tasks are the most advantageous [7, 8]	Highest; multi-round interactive and iterative retrieval	Local errors can be recovered, but long-chain decision errors still accumulate

According to this, simple fact query with controllable knowledge base, simple retrieval and one layer of rearrangement can be used; Adaptive routing is more cost-effective when the intention is changeable and large and heterogeneous knowledge bases; Multi hop reasoning, cross document synthesis, noisy library or adversarial query are worthy of agent loops and GraphRAG. In practice, a more common form is to stack a layer of lightweight routing on the precise retrieval link, rather than directly build a complete agent architecture [8].

When calculating cost, note that the overhead of a link is not only the tokens in the inference stage, but also offline index construction, vector-store storage, monitoring and operations, and

DOI: 10.54254/2755-2721/2026.39116

failure retry. Internal systems with low daily traffic and stable query modes often lose more than they gain by pursuing the latest paradigm; for products with open users, long-tail query distribution, and low fault tolerance, the early investment in adaptive and verification mechanisms is usually recovered in later operations and maintenance.

6. Core challenges and future prospects

For the future, on-device retrieval and model compression are feasible cost reduction directions. The small parameter encoder and quantized vector index are deployed on the terminal, which can complete the preliminary retrieval under the premise that the data is not local, and has practical significance for privacy sensitive scenes. For this kind of load, it is more realistic to sink the retrieval link to the terminal and aim at privacy protection and low latency.

The agentic RAG pipeline is very long. How to set reliable termination conditions and fallback strategies in a limited number of steps to avoid invalid cycles is the key to its large-scale application. There is no built-in termination criterion for the "think → act → observe" cycle proposed by ReAct, which can only be realized by the maximum number of iteration steps plus human fallback in practical engineering [7]. Although the multi-agent division of labor architecture can find downstream errors in the verification process and fall back, the process of transferring information between roles will also magnify the error of single-step decision [8]. Liang et al. Pointed out that the reliability boundaries of the two types of reasoning are different, and the unified step budget and fallback mechanism need to be specially studied [13].

Safety and compliance are also inseparable, and the existing reviews are limited [8, 9]. The knowledge base poisoning attack embeds carefully constructed malicious paragraphs in the document to induce the model to output the specified content under a specific query; The retrieval process itself may also bring the sensitive information in the user query into the third-party search engine, resulting in data leakage. Combined with Ji et al's analysis of the source of hallucinations, the filtering and auditing at the generation end can only constrain the last link of the output [12]; If the search entry lacks access control over the external corpus, the contaminated evidence will still be transmitted to the generated results along the link.

7. Conclusion

This paper focuses on three aspects of the evolution of RAG. Along the thread of design-focus shifts, the technical route is divided into four stages—naive retrieval, precise retrieval, adaptive routing, and agentic reasoning—and the core mechanisms and new costs of each stage are organized. On this basis, a hierarchical evolution framework is proposed that cross-compares six levels against the four stages, identifying the unresolved gap in multi-source contradictory-document arbitration that runs through all stages. The applicable boundaries and selection logic of the approach are also discussed. In general, the evolution direction is not that more complex is always better, but rather a constant search for balance among accuracy, computational cost, and robustness. The agent paradigm sits at the forefront of complex multi-hop tasks, but for most production scenarios, moderate retrieval depth and clear routing rules remain the most cost-effective choice.

References

[1] Lewis, P., Perez, E., Piktus, A. et al. (2020). Retrieval-augmented generation for knowledge-intensive NLP tasks. *Advances in Neural Information Processing Systems, 33*, 9459-9474. https://arxiv.org/abs/2005.11401

DOI: 10.54254/2755-2721/2026.39116

[2] Karpukhin, V., Oguz, B., Min, S. et al. (2020). Dense passage retrieval for open-domain question answering. *Proceedings of EMNLP*, 6769-6781. https://arxiv.org/abs/2004.04906
[3] Nogueira, R. & Cho, K. (2019). Passage re-ranking with BERT. arXiv preprint arXiv:1901.04085.
[4] Gao, L., Ma, X., Lin, J. et al. (2023). Precise zero-shot dense retrieval without relevance labels. *Proceedings of ACL*, 1762-1777. https://arxiv.org/abs/2212.10496
[5] Asai, A., Wu, Z., Wang, Y. et al. (2024). Self-RAG: Learning to retrieve, generate, and critique through self-reflection. *International Conference on Learning Representations*. https://arxiv.org/abs/2310.11511
[6] Yan, S., Gu, J., Zhu, Y. et al. (2024). Corrective retrieval augmented generation. arXiv preprint arXiv:2401.15884.
[7] Yao, S., Zhao, J., Yu, D. et al. (2023). ReAct: Synergizing reasoning and acting in language models. *International Conference on Learning Representations*. https://arxiv.org/abs/2210.03629
[8] Singh, A., Ehtesham, A., Kumar, S. et al. (2025). Agentic retrieval-augmented generation: A survey on agentic RAG. arXiv preprint arXiv:2501.09136.
[9] Gao, Y., Xiong, Y., Gao, X. et al. (2023). Retrieval-augmented generation for large language models: A survey. arXiv preprint arXiv:2312.10997.
[10] Vaswani, A., Shazeer, N., Parmar, N. et al. (2017). Attention is all you need. *Advances in Neural Information Processing Systems, 30*. https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html
[11] Izacard, G. & Grave, E. (2021). Leveraging passage retrieval with generative models for open domain question answering. *Proceedings of EACL*, 874-880. https://arxiv.org/abs/2007.01282
[12] Ji, Z., Lee, N., Frieske, R. et al. (2023). Survey of hallucination in natural language generation. *ACM Computing Surveys, 55*(12), 1-38. https://doi.org/10.1145/3571730
[13] Liang, J., Su, G., Lin, H. et al. (2025). Reasoning RAG via System 1 or System 2: A survey on reasoning agentic retrieval-augmented generation for industry challenges. *Proceedings of IJCNLP-AACL*, 1954-1966. https://arxiv.org/abs/2506.10408
[14] Edge, D., Trinh, H., Cheng, N. et al. (2024). From local to global: A graph RAG approach to query-focused summarization. arXiv preprint arXiv:2404.16130.
[15] Liu, N. F., Lin, K., Hewitt, J. et al. (2024). Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics, 12*, 157-173. https://arxiv.org/abs/2307.03172
[16] Es, S., James, J., Espinosa-Anke, L. et al. (2024). RAGAS: Automated evaluation of retrieval augmented generation. *Proceedings of EACL: System Demonstrations*, 150-158. https://arxiv.org/abs/2309.15217
[17] Yu, H., Gan, A., Zhang, K. et al. (2024). Evaluation of retrieval-augmented generation: A survey. *CCF Conference on Big Data*, 102-120. Springer. Singapore. https://arxiv.org/abs/2405.07437