IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940



International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 11, November 2026

DOI: 10.17148/IJARCCE.2026.151185

Token-Aware Context Optimization for Retrieval-Augmented Generation

Ved Sahasrabudhe[1], Rishi Patwa[2], Pushkar Pawar[3], Piyush Raut[4], Balwant A. Sonkamble[5]

Department of Computer Engineering, SCTR's Pune Institute of Computer Technology, Pune, India[1-5]

Abstract: Large language models (LLM) can leverage knowledge sources using retrieval-augmented generation (RAG). Conventional RAG systems retrieve a few passages and pass full text to the generator. This often involves duplicate content and a higher total number of input tokens. This paper reviews ten works that cover fundamental retrieval and generation adaptive and corrective retrieval hierarchical retrieval and evidence selection and context compression. For each work, we describe the mechanism report the results and point out the limitation. In addition, we compare these works in terms of pipeline stage unit of operation adaptivity evidence selection and compression. We observe that in general, the development of adaptive retrieval and context reduction has mostly been done separately. In addition, the token budget is often viewed as a compression ratio rather than a common constraint among competing pieces of evidence. Furthermore, the reviewed studies rarely compare methods in the same context budget. We identify a research direction for the selection of evidence with limited budgets. We do not make any experimental claims.

Keywords: Retrieval-Augmented Generation, Large Language Models, Adaptive Retrieval, Evidence Selection, Context Compression, Token Budget

I. INTRODUCTION

Large Language Models (LLMs) have done well in question answering. They can handle tasks like summarization and dialogue. They are limited by the knowledge they had during training. This means a model might not have the information. It also might not have information that's important to a specific organization. When there is no information to support a statement the model can give answers. This can lead to answers without evidence. Retrieval-Augmented Generation (RAG) is one way to fix this. RAG works by combining a Large Language Model with a retrieval system. The system finds passages from a large collection of documents. It then feeds these passages along with the query to the LLM. RAG was introduced by Lewis et al. As a way to mix a network with a retrieval system, it uses a retriever and a sequence-to-sequence generator. It also includes a non-parametric memory [1]. The simplest version of RAG gets a fixed number of passages for each query. It adds these passages directly to the prompt without changing them. This is easy to do. It has problems. Different queries need amounts of supporting evidence. Some retrieved passages might have information but only a small part. Some passages might be repeated. When the full prompt is sent to the LLM it includes a lot of text. This increases the number of tokens used slows down the process and raises costs. There has been research trying to solve these issues. Some approaches include retrieval and self-reflective retrieval [3], [4], [6]. Others focus on evidence selection and hierarchical indexing [5]. There is also work on compression techniques [7]–[10]. This paper looks at these works. Tries to answer one main question: How is the context that reaches the generator built? How well is the size of that context controlled?

This survey contributes to:

- a description of ten works, including their mechanisms, results and limitations;
- a comparison of these works by pipeline stage, unit of operation and how they handle adaptivity, selection and compression;
- an identification of open research gaps and a short outlook, on the challenge of limited budget context construction.

II. BACKGROUND

A simplified RAG pipeline can be written as

$$q \rightarrow \mathrm{Ret}(q) \rightarrow D_q \rightarrow G(q, D_q) \rightarrow y, \quad (1)$$

where q is the query, Ret is the retrieval operation, Dq is the retrieved evidence, G is the language model and y is the response. In the indexing stage documents are split into passages converted to vectors and stored in an index. At inference time the query is embedded in the space and compared with the indexed passages commonly using cosine similarity:

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940



International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 11, November 2026

DOI: 10.17148/IJARCCE.2026.151185

$$\mathrm{sim}(q, d) = \frac{q \cdot d}{\|q\|\|d\|}. \quad (2)$$

In conventional RAG the retrieved set is the k highest-ranked passages, for a fixed k:

$$D_q = \mathrm{TopK}(\mathrm{Ret}(q), k). \quad (3)$$

The selected passages are concatenated with the query and given to the generator. This design is predictable and easy to implement. It assumes that the same amount of evidence suits every query.

III. LIMITATIONS OF CONVENTIONAL RAG

Fixed retrieval depth A short factual question might need one passage while a question that requires several steps might need several passages. Using a fixed k can leave some queries without enough evidence and can give too much evidence for other queries.

Irrelevant evidence A high similarity score does not mean that the whole passage will help. A passage can discuss the topic yet only one or two sentences may actually answer the question. The rest of passage might have redundant information.

Redundant evidence Multiple paragraphs can have the same facts. Adding all of them in the context wastes the context window and does not provide any new information. Therefore relevance alone can not be enough to choose which evidence to keep.

Context Length and Cost The text that is retrieved becomes part of the input to the language model. When the context size increases, latency grows along with cost increase. Long contexts can also make it harder for the model to locate the information.

These limitations inspire the works that are studied in the following section. Those works operate at stages of the pipeline as shown in Figure 1.

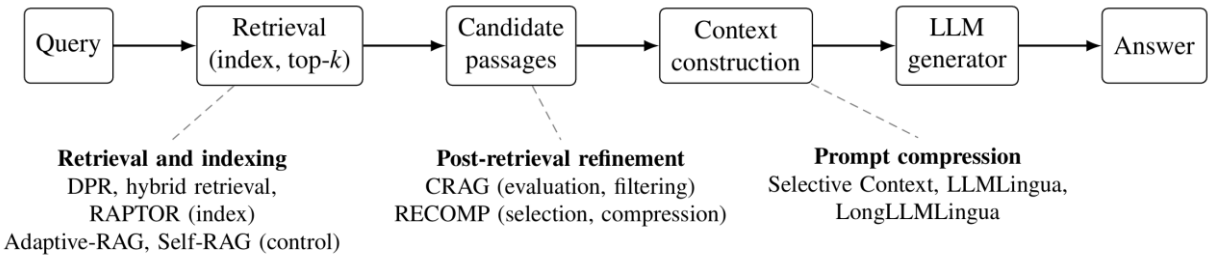


Fig. 1 Pipeline stages at which the reviewed works intervene.

IV. REVIEW OF EXISTING APPROACHES

A. Foundations: RAG and Dense Retrieval

RAG. Lewis et al. [1] proposed a general fine-tuning recipe in which a pre-trained sequence-to-sequence generator (BART) is combined with a dense vector index of Wikipedia that serves as non-parametric memory. A DPR-style retriever fetches the top-k documents for the input, and the generator conditions on them. Two variants are described: RAG-Sequence uses the same retrieved document for the whole output, whereas RAG-Token can draw on a different document for each generated token. The query encoder and generator are fine-tuned jointly, and the approach was evaluated on open-domain question answering, fact verification, and knowledge-intensive generation, where it gave state-of-the-art results on several open-domain QA benchmarks. Retrieval uses a fixed number of documents, so the system inherits the fixed-depth limitation described above.

Dense Passage Retrieval. Karpukhin et al. [2] replaced sparse term matching with a dual-encoder design in which two BERT encoders map questions and passages into one vector space and passages are ranked by vector similarity. The encoders are trained using in-batch negatives together with hard negatives obtained from BM25. On open-domain QA datasets, DPR outperformed a strong BM25 baseline by roughly 9–19% absolute in top-20 passage retrieval accuracy, and this improvement carried over to end-to-end QA accuracy. Dense retrieval finds passages that are semantically related to the query even when they share few exact words. Sparse methods such as TF-IDF and BM25 remain effective when query terms appear explicitly, and hybrid retrieval combines the two signals:



International Journal of Advanced Research in Computer and Communication Engineering
Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 11, November 2026
DOI: 10.17148/IJARCCE.2026.151185

$$S_{\text{hybrid}} = \lambda S_{\text{sparse}} + (1 - \lambda) S_{\text{dense}}. \quad (4)$$

Hybrid scoring can improve coverage but adds ranking stages. In all cases, relevance is scored for whole passages, so the useful portion of a passage is not isolated.

B. Adaptive and Corrective Retrieval

Adaptive-RAG. Jeong et al. [3] observe that single-step retrieval is wasteful for simple questions and insufficient for complex multi-hop ones. A smaller language model is trained as a classifier that predicts the complexity of an incoming question and routes it to one of three strategies: answering directly without retrieval, single-step retrieval, or iterative multi-step retrieval. The classifier labels are obtained automatically, from the outcomes of the strategies and from biases in existing datasets, so no manual annotation is needed. The authors report a better balance between accuracy and cost than always applying a single strategy. The adaptation, however, concerns the choice of retrieval strategy; the content of the retrieved passages is passed on unchanged.

Self-RAG. Asai et al. [4] train a language model to retrieve on demand and to critique its own output. The model generates special reflection tokens that indicate whether retrieval is needed and whether a retrieved passage is relevant, whether the generated text is supported by the passage, and whether the response is useful overall. Training data for these tokens is created using a critic model, and the tokens can be used at inference time to control behavior. Experiments with 7B and 13B models reported gains over strong baselines, including ChatGPT and retrieval-augmented Llama2-chat, on open-domain QA, reasoning, and fact verification, with better factuality and citation accuracy. The approach treats retrieval as a decision rather than a fixed step, but it requires fine-tuning the generator and still works at the passage level.

CRAG. Yan et al. [6] address the case in which retrieval returns poor results. A lightweight retrieval evaluator estimates the quality of the retrieved documents and triggers one of three actions: use the documents (correct), discard them and fall back on web search (incorrect), or combine both (ambiguous). A decompose-then-recompose step then splits documents into small strips, scores them, and keeps only the relevant ones, which removes part of the irrelevant text. CRAG is designed as a plug-and-play addition to existing RAG pipelines. It adds an evaluator and, in some cases, a dependence on web search, and its filtering is not driven by an explicit limit on context size.

C. Hierarchical Retrieval

RAPTOR. Sarthi et al. [5] build a retrieval tree over a document collection. Text chunks are embedded and clustered, each cluster is summarized by an LLM, and the process repeats on the summaries until a tree with several levels of abstraction is obtained. At query time, retrieval can draw on both detailed leaf chunks and higher-level summaries, which helps with questions that need information from many parts of a long document. Combined with GPT-4, RAPTOR improved accuracy on the QuALITY benchmark by about 20% in absolute terms. Because the summaries are created at indexing time, they do not depend on the query, and building the tree adds indexing cost.

D. Evidence Selection and Context Compression

RECOMP. Xu et al. [7] compress retrieved documents into a short text before they are placed in the prompt. Two compressors are studied: an extractive compressor that selects useful sentences, and an abstractive compressor that generates a summary. Both are trained so that the compressed text helps the language model perform the task. The compressors can return an empty output when the retrieved documents are irrelevant, which the authors call selective augmentation. Compression rates as low as about 6% were reported with small loss of performance on language modeling and open-domain QA. The approach requires training dedicated compressors, and compression can still remove evidence that the generator needs.

Selective Context. Li et al. [8] remove content that is predictable and therefore carries little information. A small language model computes the self-information of lexical units (tokens, phrases, or sentences), and units with low self-information are pruned. In experiments on summarization, question answering, and response generation, a 50% reduction in context was reported to cut memory use by about 36% and inference time by about 32% with only minor quality loss. The score does not depend on the question, so the method is not designed specifically for retrieval-augmented settings or for choosing evidence relevant to a given query.

LLMLingua. Jiang et al. [9] propose coarse-to-fine prompt compression. A budget controller assigns different compression ratios to different prompt components, such as the instruction, demonstrations, and question; a token-level iterative algorithm then compresses the text while accounting for dependencies between tokens; and a distribution-alignment step adapts the small language model that performs the compression to the target LLM. The authors report compression of up to 20 times with little loss of performance on reasoning and in-context learning benchmarks.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940



International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 11, November 2026

DOI: 10.17148/IJARCCE.2026.151185

LLMLingua is a general prompt compressor rather than a RAG-specific method, and its token-level output is meant for the model rather than for readers.

LongLLMLingua. Jiang et al. [10] extend this approach to long-context settings, where important information is scattered throughout surrounding text and its position in the prompt affects how well the model uses it. The method is question-aware in that it ranks and compresses documents using the question, applies dynamic compression ratios, reorders documents to highlight important information in more desirable locations, and appends a recovery step for restoration of key subsequences. On NaturalQuestions, performance was reported to improve by up to 21.4% while using approximately 4 times fewer tokens. The approach mainly targets long-context efficiency, and it produces token-level compressed text.

V. COMPARATIVE ANALYSIS

Table I compares the reviewed works. The Adapt., Select., and Compr. columns indicate whether a method explicitly adapts retrieval to the query, selects among pieces of evidence, or shortens the context (Partial means the capability is present only as a side effect or in limited form). These ratings are the authors' assessment based on the descriptions in the original papers.

TABLE I COMPARISON OF REVIEWED WORKS

Method	Stage	Core mechanism	Unit	Adapt.	Select.	Compr.	Main limitation
RAG [1]	Retrieval, generation	Dense retriever with seq2seq generator, fine-tuned jointly	Passage	No	No	No	Fixed number of retrieved documents
DPR [2]	Retrieval	Dual-encoder dense passage ranking	Passage	No	No	No	Whole passages scored; no refinement
Adaptive-RAG [3]	Retrieval control	Complexity classifier selects no, single-step, or multi-step retrieval	Query	Yes	No	No	Adapts strategy, not content
Self-RAG [4]	Retrieval, generation	Reflection tokens decide retrieval and critique output	Passage	Yes	Partial	No	Requires fine-tuning the generator
CRAG [6]	Post-retrieval	Evaluator triggers use, discard, or web search; strip-level filtering	Document, strip	Yes	Partial	Partial	Extra evaluator; web-search dependence
RAPTOR [5]	Indexing	Recursive clustering and summarization into a tree	Chunk, summary	Partial	Partial	Yes	Query-independent summaries; indexing cost
RECOMP [7]	Post-retrieval	Extractive or abstractive compressor, may output nothing	Sentence, summary	No	Yes	Yes	Needs trained compressors; may drop evidence
Selective Context [8]	Context pruning	Removes low self-information units	Token, phrase, sentence	No	Partial	Yes	Query-agnostic; not RAG-specific
LLMLingua [9]	Prompt compression	Budget controller with token-level iterative compression	Token	Partial	Partial	Yes	General prompt compression
LongLLMLingua [10]	Prompt compression	Question-aware compression, reordering, dynamic ratios	Document, token	Partial	Partial	Yes	Targets long contexts; token-level output

Three patterns emerge from the table. First, the literature shows a progression from fixed retrieval [1], [2] toward adaptive and corrective retrieval [3], [4], [6] and, in parallel, toward reducing the context that reaches the generator [7]–[10]. Second, these lines of work act on different stages (Fig. 1): a system can adapt how many passages are retrieved while

 This work is licensed under a Creative Commons Attribution 4.0 International License

DOI: 10.17148/IJARCCE.2026.151185

leaving unnecessary text inside them, and a compressor can shorten the context without deciding how competing pieces of evidence should share the available space. Third, the unit of operation varies from whole queries and passages to sentences and tokens, and finer units give more control over what is kept but are harder to evaluate for answer quality. Methods that compress heavily also carry the risk, noted by several of the authors, of removing information that the answer depends on.

VI. RESEARCH GAPS AND OUTLOOK

This section discusses the research gaps identified in the reviewed works and highlights possible directions for future research.

A. Research Gaps

- **Retrieval and compression are mostly treated separately:** Adaptive-RAG and Self-RAG adapt the retrieval to the query [3], [4], RECOMP, Selective Context and LLMLingua are all about selection or compression of the retrieved content [7]–[9]. The remaining option is to jointly optimise retrieval depth along with passage level compression.
- **Limited use of a shared evidence budget:** LLMLingua and LongLLMLingua papers, control the amount of information that is retained during compression [9], [10]. The problem with that is that you end up having to jointly decide how many passages to retrieve and how much information to retain from each of those passages which again requires a shared token budget.
- **Differences in evaluation settings:** Existing methods use a variety of datasets, retrieval schemes, compression settings and evaluation metrics. A proper comparison of all of them under a common token budget ,would give a far clearer picture of how effectively they use the context to produce accurate answers.

B. Outlook: Budget-Constrained Context Construction

- **Evidence selection under a token budget:** Context construction can be seen as the process of selecting evidence units on the basis of their relevance, their token cost and the amount of additional information they provide us, all within a given budget.
- **Balancing relevance and novelty:** Evidence can be selected on the basis of relevance to the query and the amount of new information it adds on top of what's already been retrieved.A greedy approach based on marginal utility per token would be a simple baseline.
- **Joint retrieval and compression:** Each passage could have a number of representations (the original text, a selection of sentences, a compressed version etc). The system would then decide which passages to retrieve and how much of each to take.
- **Evaluation:** This approach can be evaluated against fixed-k RAG and a variety of other retrieval and compression baselines in terms of its effect on answer quality and context usage.

VII. CONCLUSION

The study examined ten papers concerning context construction in RAG. The initial research established dense retrieval and the retrieve-then-generate approach [1], [2]. Later work has been based on this foundation. Some of the more recent studies have concerned themselves with self-reflective and self-correcting methods in order to make retrieval more adaptive [3], [4], [6]. RAPTOR, on the other hand, devised a tree of summaries [5]. Another kind of work has been concerned with the selection and compression of the retrieved context before it is fed into the generator [7]–[10].

The various stages of the RAG pipeline—namely retrieval, evidence selection and compression—were subjected to evaluation. The main point that emerges is that most of the methods work better when applied separately to each of these stages. Yet they are connected with one another; the more passages you retrieve, the less room there is for detail in each individual passage. This raises a most interesting question: can retrieval and compression be adjusted together rather than separately, without losing the context even when the token budget is fixed?

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," in Advances in Neural Information Processing Systems, vol. 33, 2020.

[2] V. Karpukhin, B. Oguz, S. Min, P. Lewis, L. Wu, S. Edunov, D. Chen, and W.-t. Yih, "Dense Passage Retrieval for Open-Domain Question Answering," in Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), 2020, pp. 6769–6781.



International Journal of Advanced Research in Computer and Communication Engineering
Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 11, November 2026
DOI: 10.17148/IJARCCE.2026.151185

[3] S. Jeong, J. Baek, S. Cho, S. J. Hwang, and J. Park, "Adaptive-RAG: Learning to Adapt Retrieval-Augmented Large Language Models through Question Complexity," in Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT), 2024, pp. 7036–7050.

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

[5] P. Sarthi, S. Abdullah, A. Tuli, S. Khanna, A. Goldie, and C. D. Manning, "RAPTOR: Recursive Abstractive Processing for Tree-Organized Retrieval," in International Conference on Learning Representations (ICLR), 2024.

[6] S.-Q. Yan, J.-C. Gu, Y. Zhu, and Z.-H. Ling, "Corrective Retrieval Augmented Generation," arXiv preprint arXiv:2401.15884, 2024.

[7] F. Xu, W. Shi, and E. Choi, "RECOMP: Improving Retrieval-Augmented LMs with Compression and Selective Augmentation," in International Conference on Learning Representations (ICLR), 2024.

[8] Y. Li, B. Dong, F. Guerin, and C. Lin, "Compressing Context to Enhance Inference Efficiency of Large Language Models," in Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP), 2023, pp. 6342–6353.

[9] H. Jiang, Q. Wu, C.-Y. Lin, Y. Yang, and L. Qiu, "LLMLingua: Compressing Prompts for Accelerated Inference of Large Language Models," in Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP), 2023, pp. 13358–13376.

[10] H. Jiang, Q. Wu, X. Luo, D. Li, C.-Y. Lin, Y. Yang, and L. Qiu, "LongLLMLingua: Accelerating and Enhancing LLMs in Long Context Scenarios via Prompt Compression," in Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (ACL), 2024, pp. 1658–1677.

 This work is licensed under a Creative Commons Attribution 4.0 International License