Improving Retrieval Accuracy in Two Knowledge Graph RAG Frameworks via BM25 Integration

Koki Takami[1], Tetsuya Takiguchi[1], Yasuo Ariki[1], Yusuke Iwasaki[2], Motoki Kojima[2] and Masaki Yamamoto[2]

[1]*Graduate School of System Informatics, Kobe University, Kobe, Japan*
[2]*DENSO TEN Limited, Kobe, Japan*

Abstract

Knowledge graph RAG (KG-RAG) enables multi-hop evidence retrieval over graph structures, but its internal retrieval often relies heavily on embedding similarity. Consequently, it may miss evidence whose relevance depends on exact lexical matches, particularly rare proper nouns; later graph exploration may not recover candidates discarded early. We propose a general design principle that injects BM25 lexical signals before a KG-RAG system produces its final ranking, and instantiate it in two structurally distinct frameworks, MiniRAG and LinearRAG. For MiniRAG, we replace embedding similarity with BM25 during important-edge selection so that lexical evidence influences path voting and chunk aggregation. For LinearRAG, BM25 initializes passage weights before Personalized PageRank (PPR), propagating lexical relevance through its hierarchical graph. We evaluate these changes on LiHua-World against graph-free baselines and post-hoc fusion with an independent BM25 retriever. Internal injection improves MiniRAG (chunk variant) from 0.657 to 0.752 in all-evidence recall@10 (Recall@10) and from 0.453 to 0.591 in nDCG@10. LinearRAG improves from 0.710 to 0.808 and from 0.543 to 0.658, respectively (all Holm-adjusted $p < 0.001$). Reranking the union of candidates from the injected graph retriever and independent BM25 retriever achieves 0.869/0.796 for MiniRAG and 0.886/0.818 for LinearRAG (Recall@10/nDCG@10), exceeding post-hoc fusion without internal injection, although at this stage only LinearRAG's nDCG@10 gain is statistically significant. These results indicate that early lexical injection complements direct lexical retrieval across different graph designs.

Keywords

Knowledge Graph RAG, BM25, Lexical Retrieval, Hybrid Retrieval

CEUR-WS.org/Vol-4316/pd-short-3.pdf

1. Introduction

Retrieval-Augmented Generation (RAG) [1], which supplements large language models (LLMs) with external knowledge through retrieval, has been widely adopted. Among RAG approaches, knowledge graph RAG (KG-RAG), which maintains entities and relations extracted from documents as a graph, is well suited for exploring evidence that spans multiple documents. Various methods with different objectives and graph structures have been proposed, including GraphRAG [2], LightRAG [3], MiniRAG [4], and LinearRAG [5].

These methods employ embedding similarity when selecting seed entities, important edges, passages, and other elements. While dense retrieval captures semantic similarity, it has been reported to fail on queries where rare proper nouns are crucial [6, 7]. If correct candidates are scored low at the early stages of graph exploration, it is difficult to recover them through the subsequent structural exploration alone.

In this work, we introduce the lexical retrieval method BM25 [8] into the internal retrieval of KG-RAG. Rather than a single algorithm tied to a specific graph structure, we formulate the approach as a design principle: *inject the lexical signal before graph retrieval finalizes its ranking*. We validate this principle on two frameworks, MiniRAG and LinearRAG. We further evaluate configurations that combine internal injection with post-hoc fusion using an independent BM25 retrieval, which supplements direct lexical matches that bypass the graph. The contributions of this work are threefold.

IJCKG 2026 Posters, Demos and Workshops, November 19–21, 2026, Bangkok, Thailand
261x043x@stu.kobe-u.ac.jp (K. Takami); takigu@kobe-u.ac.jp (T. Takiguchi); ariki@kobe-u.ac.jp (Y. Ariki); yuusuke.iwasaki.j7p@jpgr.denso.com (Y. Iwasaki); motoki.kojima.j3h@jpgr.denso.com (M. Kojima); masaki.yamamoto.j3h@jpgr.denso.com (M. Yamamoto)
© 2026 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

CEUR Workshop Proceedings ceur-ws.org ISSN 1613-0073
published 2026-11-06

- We present a unified design that introduces BM25 into different retrieval stages: the important-edge selection of MiniRAG and the initial passage weights of LinearRAG.
- We apply post-hoc fusion via Reciprocal Rank Fusion (RRF) or reranking to configurations with and without internal injection, isolating the contribution of each component.
- We show that both KG-RAG frameworks improve in Recall@10 and nDCG@10 over their original versions. LinearRAG surpasses the strong BM25-only baseline on both metrics through internal injection alone, whereas MiniRAG requires post-hoc fusion to do so.

2. Related Work

2.1. Knowledge Graph RAG

GraphRAG [2] aggregates entities and relations extracted from documents into communities and uses them to answer global questions. MiniRAG [4] exploits a heterogeneous graph of text chunks and entities together with its topology, realizing lightweight retrieval that works even with small language models. LinearRAG [5] constructs an entity–sentence–passage hierarchical graph that requires no relation extraction, and retrieves passages through local entity activation and global importance aggregation. Our work differs from these in that it does not propose a new graph construction method but adds a lexical signal to the candidate selection inside existing KG-RAGs.

2.2. Combining Dense and Lexical Retrieval

Dense retrieval, represented by Dense Passage Retrieval [9], and BM25 exploit complementary information: semantic matching and lexical matching. Reciprocal Rank Fusion (RRF) [10] can combine multiple retrieval results using only ranks, without requiring score calibration across retrievers. Whereas typical hybrid retrieval fuses the outputs of final retrievers, our work also injects the lexical signal *inside* the graph retrieval so that it is reflected in the subsequent graph processing.

3. Proposed Method

3.1. Problem Setting

Let q denote a query and D the set of text chunks to be retrieved. A KG-RAG assigns initial scores to entities, edges, passages, and other elements relevant to q, and outputs a ranking over D through propagation or path aggregation based on the graph structure. Hereafter, we follow each framework's convention for the retrieval target texts, which are called chunks in MiniRAG and passages in LinearRAG. In this work, we replace some internal scores previously determined solely by embedding similarity with BM25-based scores and reconstruct the internal ranking; we refer to this operation as "injection." Figure 1 shows the positions of internal injection in both frameworks and the overall configuration including post-hoc fusion with an independent BM25 retrieval.

3.2. BM25 and Rank Fusion

BM25 for a query q and a document d is computed as follows.

$$\mathrm{BM25}(q,d) = \sum_{t \in q} \mathrm{IDF}(t) \frac{\mathrm{tf}(t,d)(k_1+1)}{\mathrm{tf}(t,d) + k_1(1 - b + b\,\ell_d/\bar{\ell})} \tag{1}$$

Here, $\mathrm{tf}(t,d)$ is the frequency of term t in d, $\mathrm{IDF}(t)$ is the inverse document frequency of t, ℓ_d and $\bar{\ell}$ are the document length and the average document length, and k_1 and b are parameters.

Let $r_{\mathrm{BM25}}(d)$ be the rank of d obtained by sorting the scores in Eq. (1) in descending order (rank 1 is highest). We fuse it with the dense retrieval rank $r_{\mathrm{dense}}(d)$ using RRF.

$$s_{\mathrm{RRF}}(d) = \frac{1}{k + r_{\mathrm{dense}}(d)} + \frac{1}{k + r_{\mathrm{BM25}}(d)} \tag{2}$$

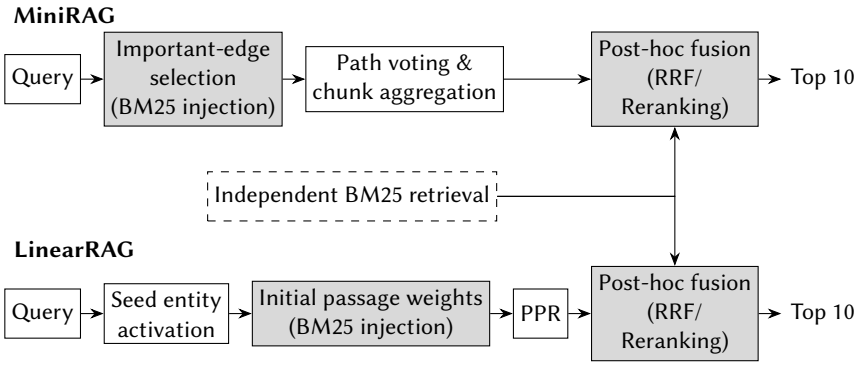


Figure 1: Overall architecture of the proposed method. Shaded blocks indicate the stages where the lexical signal of BM25 is integrated. Internal injection (important-edge selection and initial passage weights) reflects the lexical signal before the graph retrieval finalizes its ranking, and the dashed independent BM25 retrieval supplements candidates matched directly without going through the graph, via post-hoc fusion.

Here, k is a constant that smooths rank differences, and the two retrievers are weighted equally.

3.3. Injection into Important-Edge Selection in MiniRAG

MiniRAG selects important edges based on the embedding similarity between the query and edge information, and retrieves chunks by prioritizing candidate paths that contain the selected edges. In this work, we replace embedding similarity with BM25 ranking as the selection criterion for important edges. As the retrieval target of BM25, we compare two variants: (1) **edge information**, which concatenates relation keywords, the names of both endpoint entities, and the relation description, and (2) the **chunk** from which the relation was extracted. The path voting and chunk aggregation after important-edge selection are identical to the original method, so that lexically matched edges are reflected in the graph exploration before the final chunk selection.

3.4. Pre-Propagation Injection into LinearRAG

LinearRAG activates, through sentences, the seed entities obtained from the question, and feeds the initial weights of entities and passages into Personalized PageRank (PPR) to obtain the final passage ranking. The original initial passage weights were based on dense retrieval scores computed as the inner product between the question embedding and passage embeddings.

In the proposed method, we build a BM25 index treating each passage as a document and, as in MiniRAG, replace the dense retrieval score underlying the initial weights with BM25. The BM25 scores are normalized, and, as in the original method, a bonus based on the occurrence of activated entities is added to form the initial weights of passage nodes, after which PPR is executed as before. Thus, rather than fusing the final results of LinearRAG with BM25 post hoc, our method injects the lexical signal into the *passage prior distribution before PPR*.

3.5. Post-Hoc Fusion with Independent BM25 Retrieval

Apart from internal injection, we fuse the graph retrieval ranking—after path aggregation in MiniRAG and after PPR in LinearRAG—with the chunk ranking of an independent BM25 retrieval that does not use the graph. In post-hoc RRF, the final ranking is obtained by replacing $r_{\text{dense}}(d)$ in Eq. (2) with the graph retrieval rank $r_{\text{graph}}(d)$. In post-hoc reranking, for the union of the top 10 results of the graph retrieval and the top 10 of the independent BM25 retrieval (at most 20 candidates), a cross-encoder reranker recomputes the relevance to the query, and the top 10 are taken as the final result. Post-hoc fusion serves to keep candidates matched directly without going through the graph in the final result, and can be applied regardless of whether internal injection is used; our experiments evaluate their combinations.

4. Experiments

4.1. Dataset and Evaluation Metrics

We use LiHua-World, a dataset for local RAG released with MiniRAG [4]. This dataset contains one year of chat logs of a virtual user, LiHua, and QAE (Question-Answer-Evidence) pairs consisting of questions answerable from those logs, their answers, and the ground-truth evidence. Using all chat logs as the knowledge base, each method outputs its top 10 retrieval results for each question. As evaluation metrics, we use the all-evidence recall@10 (hereafter Recall@10), which is the proportion of questions for which all ground-truth evidence is retrieved within the top 10, and the standard nDCG@10 with binary relevance ($rel_i = 1$ if the item at rank i is ground-truth evidence for that question and 0 otherwise), which normalizes DCG@10 $= \sum_{i=1}^{10} rel_i / \log_2(i+1)$ by the IDCG@10 of the ideal ranking and thus also evaluates the ranks of the ground-truth evidence. Answer generation is not evaluated.

4.2. Experimental Setup

For MiniRAG, we use all-MiniLM-L6-v2 [11] as the embedding model and Qwen3-1.7B [12] for entity extraction. For LinearRAG, we use all-MiniLM-L6-v2 and spaCy [13] (en_core_web_trf), keeping all graph construction and PPR settings other than BM25 identical to the original method. For both frameworks, BM25 uses $k_1 = 1.5$ and $b = 0.75$, RRF uses $k = 60$, and Qwen3-Reranker-4B [14] is used for post-hoc reranking.

4.3. Compared Methods

As graph-free baselines, we evaluate NaiveRAG, which performs only dense retrieval over chunks (with all-MiniLM-L6-v2 as the embedding model), and BM25 alone. Furthermore, as a point of comparison for post-hoc fusion, we also evaluate a graph-free hybrid that fuses NaiveRAG and the independent BM25 via post-hoc RRF or post-hoc reranking (denoted NaiveRAG+BM25 in Table 1). For MiniRAG, we compare the original method (without BM25) with variants whose BM25 internal injection targets either edge information or chunks. For LinearRAG, we compare the original method (without BM25) with the variant that introduces pre-propagation injection, replacing the retrieval score for the initial passage weights with BM25. In addition, for both frameworks, we apply post-hoc RRF or reranking to configurations with and without internal injection.

4.4. Statistical Analysis

For comparability with the original evaluation, Table 1 retains all 572 QAE entries, and Table 3 groups the 569 entries with available lexical coverage from the same evaluation, without deduplication.[1] Paired tests use 570 distinct questions after removing only the two duplicate rows; all retained per-entry scores from the saved runs remain unchanged, and Table 2 is computed on this set. We examine four comparisons: original versus internal injection alone for MiniRAG (chunk) and LinearRAG, and the corresponding comparisons with post-hoc reranking applied to both configurations. For Recall@10, each entry contributes a binary all-evidence success indicator, and we use a two-sided exact McNemar test. For nDCG@10, we use a two-sided paired permutation test of the mean difference with 99,999 random sign-flip resamples and master seed 5402026. We apply Holm correction jointly to the eight tests (four comparisons times two metrics), with significance level $\alpha = 0.05$. These tests assess variation across the distinct questions for the fixed retrieval runs; they do not assess variation across graph construction runs or datasets.

[1] Two rows in the distributed query set are exact duplicates. Removing them changes Recall by at most 0.30 percentage points and nDCG by at most approximately 0.0016 across the 19 methods.

Table 1
Evidence retrieval accuracy on LiHua-World. Bold indicates the best value within each of the MiniRAG and LinearRAG frameworks.

Method	Recall@10	nDCG@10
Baselines		
NaiveRAG (dense retrieval)	0.696	0.567
BM25 alone	0.740	0.621
NaiveRAG+BM25 (RRF)	0.806	0.680
NaiveRAG+BM25 (Reranking)	0.844	0.774
MiniRAG		
Original (without BM25)	0.657	0.453
+Post-hoc RRF	0.797	0.647
+Post-hoc Reranking	0.853	0.786
BM25 internal (edge)	0.701	0.534
+Post-hoc RRF	0.820	0.671
+Post-hoc Reranking	0.853	0.788
BM25 internal (chunk)	0.752	0.591
+Post-hoc RRF	0.834	0.688
+Post-hoc Reranking	**0.869**	**0.796**
LinearRAG		
Original (without BM25)	0.710	0.543
+Post-hoc RRF	0.809	0.680
+Post-hoc Reranking	0.876	0.802
BM25 internal (pre-propagation)	0.808	0.658
+Post-hoc RRF	0.844	0.724
+Post-hoc Reranking	**0.886**	**0.818**

4.5. Results

Table 1 shows the results. Among the baselines, BM25 alone (0.740/0.621) outperformed NaiveRAG (0.696/0.567) and the original versions (without BM25) of both frameworks, and the graph-free hybrid reached 0.844/0.774 with post-hoc reranking. With internal injection, the chunk variant of MiniRAG improved from 0.657 to 0.752 (+0.095) in Recall@10 and from 0.453 to 0.591 (+0.138) in nDCG@10, and the edge variant (0.701/0.534) also improved over the original method. The pre-propagation injection of LinearRAG likewise improved Recall@10 from 0.710 to 0.808 (+0.098) and nDCG@10 from 0.543 to 0.658 (+0.115). Adding post-hoc fusion with the independent BM25, MiniRAG (chunk) achieved 0.869/0.796 and LinearRAG achieved 0.886/0.818 with post-hoc reranking, the best values within each framework. Moreover, for both fusion methods, combining internal injection with post-hoc fusion yielded higher scores on both metrics than post-hoc fusion alone for MiniRAG (chunk) and LinearRAG.

Table 2 reports the paired-test results for the four comparisons specified in Section 4.4. Internal injection alone significantly improved both evaluated metrics over the original method in each framework (all Holm-adjusted $p < 0.001$). With post-hoc reranking, the additional improvement was significant for LinearRAG's nDCG@10 ($p \approx 0.009$), but not for its Recall@10 or either metric in MiniRAG (adjusted $p \approx 0.26$). Thus, the injection-only improvements have statistical support, whereas the smaller gains after reranking are not uniformly significant.

4.6. Exploratory Query Analysis

To examine which queries benefit from internal injection, we stratify the saved per-entry scores by lexical overlap between the question and its gold evidence. We split CamelCase, lowercase alphabetic tokens, and remove a fixed stopword list, numbers, and the ubiquitous protagonist tokens *Li* and *Hua*, without stemming. For each gold evidence item, lexical coverage is the fraction of distinct question content tokens also present in that evidence; we average this fraction across the required evidence items. Reference texts are recovered from timestamp-tagged passages across all 19 logs, retaining the longest

Table 2
Holm-adjusted p-values for internal injection on 570 distinct questions after removing two duplicate rows, with per-entry scores unchanged. Reranking is absent from both configurations or applied to both. Values are rounded; very small permutation estimates are reported as < 0.001. Bold indicates significance at $\alpha = 0.05$ across all eight tests.

Framework	Post-hoc reranking	Recall@10	nDCG@10
MiniRAG (chunk)	No	$\mathbf{2.24 \times 10^{-8}}$	$\mathbf{< 0.001}$
LinearRAG	No	$\mathbf{1.82 \times 10^{-11}}$	$\mathbf{< 0.001}$
MiniRAG (chunk)	Yes	0.26	0.26
LinearRAG	Yes	0.26	**0.009**

Table 3
Exploratory gains from internal injection alone over the original framework. ΔRecall is in percentage points; ΔnDCG is the difference in nDCG@10. The upper block groups 569 of the original 572 entries by lexical coverage. The lower block restricts these groups to Single entries, which require one gold evidence item.

		MiniRAG (chunk)		LinearRAG	
Coverage	N	ΔRecall	ΔnDCG	ΔRecall	ΔnDCG
Low	191	+7.33	+0.0970	+6.28	+0.0784
Medium	222	+9.91	+0.1486	+10.36	+0.1376
High	156	+11.54	+0.1767	+13.46	+0.1329
Single: low	130	+3.08	+0.1235	+5.38	+0.0629
Single: medium	217	+9.22	+0.1451	+10.60	+0.1376
Single: high	156	+11.54	+0.1767	+13.46	+0.1329

observed version where shorter versions are verified prefixes. Three entries have unresolved evidence labels and are excluded only from the lexical groups. We obtain coverage terciles from the remaining 567 distinct questions, without using retrieval scores: low coverage is at or below the first threshold (approximately 0.6611), medium coverage is above it and at or below the second (approximately 0.8333), and high coverage is above the second. Table 3 retains the original evaluation entries and keeps tied coverage values together, so the groups differ in size.

Recall gains are larger in the high-coverage group in both frameworks. All 156 high-coverage entries are Single, whereas Multi entries require multiple evidence items; nevertheless, the same pattern remains within Single entries (Table 3). This is a descriptive association rather than evidence of a causal effect of lexical coverage; query specificity and evidence length are not controlled. With post-hoc reranking, the high-coverage group gains 2.56 Recall percentage points in both frameworks, whereas the low-coverage group changes by 0.00 points in MiniRAG and −0.52 points in LinearRAG. Thus, the additional benefit after reranking is not uniform across query groups, consistent with the qualifications in the overall analysis.

Retrieval Examples. We inspect both improvements and failures among Single entries, selecting examples near the median nDCG change within each framework's Recall gains and losses. For "What does Li Hua plan to do to clear their head?", MiniRAG moves the designated evidence from outside the top 10 to rank 2. The evidence describes a walk and a visit to a new café, including the phrase "clear my head", illustrating a gain associated with matching content words. For "Who is Li Hua meeting with to discuss homepage design updates?", LinearRAG moves the designated evidence to rank 3 from outside the top 10; it contains *meeting*, *homepage*, and *updates*. The original method already retrieves another meeting with the same person, so this example demonstrates improved retrieval of the designated evidence, not necessarily improved answer quality.

Conversely, for "What is one characteristic of Real Madrid that JakeWatson mentions?", MiniRAG loses evidence previously at rank 7. Its new top result discusses Messi's skills and mentions both JakeWatson and Real Madrid, but is a different conversation from the designated evidence about the team's comebacks. For "What video does LiHua send to the band group chat on 20260624?", LinearRAG loses evidence previously at rank 5 about playing the intro to *Stairway to Heaven*. The new results

include unrelated group chats and a video about AI robots. These cases illustrate that matching names or topic words does not ensure that the retrieved conversation satisfies the question's subject and temporal constraints. The logs expose final retrieval changes, but do not establish which internal graph paths caused them.

4.7. Discussion

Effectiveness and Injection Targets. The consistent improvements from both internal injection and post-hoc fusion across two frameworks with different graph structures and injection positions suggest that the benefits of BM25-based augmentation may not be limited to a specific KG-RAG implementation. The higher accuracy of targeting chunks rather than edge information in MiniRAG is presumably because chunks containing concrete utterances preserve the named entities in questions and capture lexical matches better than the summary-like attributes of edges.

Complementarity of Injection and Fusion. While the pre-propagation injection of LinearRAG outperformed BM25 alone on both metrics, the chunk variant of MiniRAG fell below it in nDCG@10, and the edge variant in both metrics (Table 1). Post-hoc fusion is therefore necessary for MiniRAG to outperform BM25 alone on both metrics, whereas LinearRAG does so with internal injection alone. Compared with the graph-free hybrid, configurations combining internal injection with post-hoc fusion outperformed their counterparts using the same fusion method, except that the edge variant with post-hoc RRF was slightly lower in nDCG@10. Internal injection also yielded higher scores in every fusion setting for MiniRAG (chunk) and LinearRAG than fusion without injection. In MiniRAG, chunk injection moved post-hoc RRF from below the graph-free hybrid to above it. These numerical gains suggest that internal lexical signals can improve the candidates passed to fusion and complement direct matches that bypass the graph. However, the reranking comparisons in Table 2 support an additional gain only for LinearRAG's nDCG@10 at the chosen significance level. The remaining non-significant results do not establish the absence of an effect, but limit the statistical evidence for an additional benefit after reranking.

Furthermore, post-hoc reranking was consistently more accurate than post-hoc RRF in both frameworks, suggesting that re-evaluating the lexically complemented candidate set by semantic relevance is effective in improving the ranking.

Limitations. First, the evaluation uses a single dataset, and generality across domains and languages remains unverified. Second, the statistical analysis covers four comparisons using fixed retrieval runs; we did not evaluate variation across independent graph construction runs. The query analysis is exploratory, uses gold evidence to define lexical groups, and does not isolate the effect of rare named entities. Third, since the two frameworks differ in entity extraction and graph construction methods, absolute scores cannot be compared across frameworks. Future work includes validation on multiple datasets including multi-hop QA, analysis of named-entity overlap and rarity, and evaluation of answer generation quality as well as retrieval time and memory usage.

5. Conclusion

We injected BM25 into MiniRAG's important-edge selection and LinearRAG's pre-PPR passage weights, and isolated its contribution through post-hoc fusion with independent BM25 retrieval. Both frameworks improved in Recall@10 and nDCG@10 on LiHua-World, with statistically significant injection-only gains on both evaluated metrics. With reranking, MiniRAG achieved 0.869/0.796 and LinearRAG achieved 0.886/0.818, numerically outperforming fusion alone, although the additional gains were significant only for LinearRAG's nDCG@10. Future work will validate the approach on multiple datasets and ablate fusion positions and weights.

Declaration on Generative AI

The authors used Codex and Claude Code to assist with manuscript preparation, including review, translation, proofreading, and formatting. The authors reviewed and revised the resulting content and take full responsibility for the final content of the paper.

References

[1] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, et al., Retrieval-augmented generation for knowledge-intensive NLP tasks, in: Advances in Neural Information Processing Systems, volume 33, Curran Associates, Inc., 2020, pp. 9459–9474.

[2] D. Edge, H. Trinh, N. Cheng, J. Bradley, A. Chao, A. Mody, et al., From local to global: A graph RAG approach to query-focused summarization, arXiv preprint 2404.16130, 2024. doi:10.48550/arXiv.2404.16130.

[3] Z. Guo, L. Xia, Y. Yu, T. Ao, C. Huang, LightRAG: Simple and fast retrieval-augmented generation, in: Findings of the Association for Computational Linguistics: EMNLP 2025, Association for Computational Linguistics, 2025, pp. 10746–10761. doi:10.18653/v1/2025.findings-emnlp.568.

[4] T. Fan, J. Wang, X. Ren, C. Huang, MiniRAG: A lightweight RAG system with small language models, in: Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), Association for Computational Linguistics, 2026, pp. 37121–37144. doi:10.18653/v1/2026.acl-long.1721.

[5] L. Zhuang, S. Chen, Y. Xiao, H. Zhou, Y. Zhang, H. Chen, et al., LinearRAG: Linear graph retrieval augmented generation on large-scale corpora, in: International Conference on Learning Representations, 2026, pp. 147053–147075. URL: https://openreview.net/forum?id=mCtfkypdm6.

[6] C. Sciavolino, Z. Zhong, J. Lee, D. Chen, Simple entity-centric questions challenge dense retrievers, in: Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, 2021, pp. 6138–6148. doi:10.18653/v1/2021.emnlp-main.496.

[7] W. X. Zhao, J. Liu, R. Ren, J.-R. Wen, Dense text retrieval based on pretrained language models: A survey, ACM Transactions on Information Systems 42 (2024) 89:1–89:60. doi:10.1145/3637870.

[8] S. E. Robertson, S. Walker, S. Jones, M. Hancock-Beaulieu, M. Gatford, Okapi at TREC-3, in: Proceedings of the Third Text REtrieval Conference (TREC-3), National Institute of Standards and Technology, 1994, pp. 109–126. doi:10.6028/NIST.SP.500-225.routing-city.

[9] V. Karpukhin, B. Oguz, S. Min, P. Lewis, L. Wu, S. Edunov, et al., Dense passage retrieval for open-domain question answering, in: Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, 2020, pp. 6769–6781. doi:10.18653/v1/2020.emnlp-main.550.

[10] G. V. Cormack, C. L. A. Clarke, S. Büttcher, Reciprocal rank fusion outperforms Condorcet and individual rank learning methods, in: Proceedings of the 32nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, ACM, 2009, pp. 758–759. doi:10.1145/1571941.1572114.

[11] Sentence Transformers, all-MiniLM-L6-v2 model card, 2022. URL: https://huggingface.co/sentence-transformers/all-MiniLM-L6-v2, accessed 31 July 2026.

[12] A. Yang, A. Li, B. Yang, B. Zhang, B. Hui, B. Zheng, et al., Qwen3 technical report, arXiv preprint 2505.09388, 2025. doi:10.48550/arXiv.2505.09388.

[13] M. Honnibal, I. Montani, S. Van Landeghem, A. Boyd, spaCy: Industrial-strength NLP in Python, 2020. doi:10.5281/zenodo.1212303.

[14] Y. Zhang, M. Li, D. Long, X. Zhang, H. Lin, B. Yang, et al., Qwen3 embedding: Advancing text embedding and reranking through foundation models, arXiv preprint 2506.05176, 2025. doi:10.48550/arXiv.2506.05176.