Systematic Evaluation of Design Parameters in Retrieval-Augmented Generation [E]

Akul Gupta, Nikunj Agarwal, Abhirup Chakraborty, Georges Durand
{akulg3,nikunja2,abhirup4,gdurand2}@illinois.edu
University of Illinois Urbana-Champaign, Urbana, Illinois, USA

1 Abstract

Retrieval-Augmented Generation (RAG) is gaining attention as a useful approach for improving large language models (LLMs) by incorporating external information. However, most evaluations treat RAG systems as black boxes, offering limited insights into the impact of individual components on overall performance. This research introduces a systematic evaluation framework that isolates and analyzes the contributions of critical RAG components: retriever architecture, document chunking strategies, re-ranking mechanisms, and retrieval depth. Our framework evaluates these components along multiple dimensions, including accuracy, latency, and resource utilization. Through extensive experimentation on standard benchmarks, we identify key trade-offs and provide actionable recommendations for optimizing RAG systems across diverse deployment scenarios.

2 Introduction

Retrieval-Augmented Generation (RAG) systems represent a shift in how language models access and utilize knowledge. By dynamically retrieving relevant information from external collection during inference, RAG models can overcome the limitations of parametric knowledge encoded in frozen model weights [3, 8]. This architecture offers several advantages: (1) it mitigates the challenge of knowledge staleness by accessing up-to-date information, (2) it enhances transparency by providing attribution to source documents, and (3) it improves faithfulness by grounding generation in retrieved evidence [14].

The core innovation of RAG is thanks to its hybrid architecture that decouples knowledge access from reasoning capabilities. This separation allows for independent scaling and updating of the knowledge corpus without requiring model retraining, making RAG particularly valuable for knowledge-intensive applications such as open-domain question answering, fact verification, and conversational systems [1, 5]. Recent advances have demonstrated RAG's effectiveness across diverse domains, from medical question answering [16] to legal document analysis [4].

However, the impressive performance of RAG systems masks a critical challenge: the complex interplay between their components. A typical RAG pipeline consists of multiple modules, each with its own design space and performance characteristics:

- **Retriever architectures** ranging from traditional sparse retrievers (BM25, TF-IDF) to neural dense retrievers (DPR, ANCE) and more recent contrastive learning approaches [6, 19].
- **Document chunking strategies** that determine how knowledge sources are segmented, including decisions about chunk size, overlap, and semantic coherence [9].
- **Re-ranking mechanisms** that refine initial retrieval results, such as cross-encoders that perform deeper relevance assessment [11] or fusion techniques that combine multiple retrieval signals [10].
- **Retrieval depth parameters** that control how many documents are retrieved and passed to the generator, directly impacting both accuracy and computational efficiency [18].

Despite the critical role each of these components plays, they are typically evaluated as part of an end-to-end pipeline without sufficient isolation, making it difficult to understand its individual affect on the system. [5, 7]. This black-box approach hides important pros and cons between retrieval accuracy, inference latency, memory requirements, and output faithfulness. These trade-offs are significant for practical use cases, where constraints on computational resources, real-time responsiveness, and scalability must be carefully balanced with model accuracy.

The gap between academic benchmarks and real-world deployment considerations has practical consequences. Organizations using RAG systems often encounter difficult design choices, and due to limited practical guidance, they may end up with setups that don't perform well or use resources inefficiently.

This paper addresses this challenge by introducing a component-wise evaluation framework for RAG systems. Rather than treating RAG as a monolithic architecture, we systematically isolate and analyze individual components to understand their specific contributions and interactions. We measure metrics such as computational efficiency, scalability, and the reliability of the results.

3 Related Work

3.1 End-to-End RAG Evaluation

Prior research on RAG systems has primarily focused on end-to-end performance evaluation, with limited attention to component-level analysis. Lewis [8] introduced the retrieval-augmented generation paradigm for knowledge-intensive NLP tasks, demonstrating the efficacy of augmenting LMs with external documents but evaluating their model as a monolithic system without isolating component contributions. Subsequent extensions of RAG improved specific aspects of the pipeline while maintaining this black-box evaluation methodology. For instance, Izacard and Grave [5] proposed a fusion-in-decoder approach to better integrate retrieved documents into generation, and Guu . [3] explored retrieval-enhanced text generation for commonsense reasoning; both showed overall gains but did not analyze the impact of individual module choices. Similarly, large-scale RAG-style architectures such as RETRO (Borgeaud . [1]) and retrieval-augmented dialogue models (Shuster . [14]) demonstrated the benefits of augmenting LLMs with external knowledge, yet were evaluated holistically rather than with component-wise ablations.

3.2 Individual Component Analysis

In parallel, a number of studies have examined specific RAG components in isolation. A great deal of work has focused on improving the **retriever**: Khattab [6] introduced dense passage retrieval to boost recall in open-domain QA, and Xiong [19] proposed techniques like approximate nearest-neighbor negative contrastive learning to efficiently train more effective dense retrievers. Document indexing and chunking strategies have also been investigated: Liu [9] showed that generating question-informed document chunks can improve retrieval effectiveness, highlighting the importance of how knowledge sources are segmented. In addition, re-ranking mechanisms have seen independent enhancements, for example Nogueira and Cho [11] demonstrated that a BERT-based cross-encoder substantially improves passage ranking quality. These component-centric studies, however, typically optimize one metric at a time (e.g., maximizing retrieval accuracy or re-rank effectiveness) and do not evaluate how different component choices interact within a full RAG pipeline or how they trade off against other metrics like latency and resource utilization.

3.3 Integrated Pipeline Assessment

More recently, researchers have begun to assess RAG pipelines in a more integrated manner. Khattab [7] presented a modular evaluation approach that treats a RAG system as a composition of interchangeable modules, enabling partial analysis of each module's contribution. Wang [17] conducted an extensive study of RAG workflows, comparing multiple implementation variants (from retrieval through generation) to identify best practices for balancing answer quality and efficiency. Their findings provide practical guidelines (e.g., when to use reranking or how to chunk passages) but are aimed at recommending effective component combinations, rather than quantifying each component's isolated impact. Other work has targeted specific pipeline improvements: Tanyildiz [15] demonstrated that using dynamic chunking together with a powerful cross-encoder re-ranker gives more faithful and precise answers, especially in domain-specific RAG applications. While these studies move toward finer-grained evaluation, they still tend to focus on particular metrics or subsets of the pipeline, without offering a holistic comparison across all key RAG components.

3.4 Advancements in RAG Evaluation Methodologies

To add, efforts to refine RAG *evaluation* methodologies have accelerated in recent years. Petroni [12] described an industry-scale RAG platform and evaluated the faithfulness of its generated answers, but they did not analyze how varying the system's retrievers, chunkers, or rankers would affect those outcomes. Beyond human-grounded evaluation, new automated frameworks have been proposed to more rapidly assess RAG quality. Es [2] introduced RAGAS (Retrieval Augmented Generation Assessment), a reference-free evaluation suite that scores RAG pipelines along dimensions such as retrieval relevance and generation correctness without requiring ground-truth answers. Saad-Falcon [13] presented ARES (Automated RAG Evaluation System), which trains learned LLM-based judges to rate the context relevance and answer faithfulness of RAG outputs. These tools facilitate more nuanced benchmarking of RAG systems across multiple criteria, reflecting the community's growing interest in evaluation beyond simple end-to-end accuracy. However, they focus on metric-based assessment and do not directly inform how different architectural choices within a RAG pipeline contribute to performance.

3.5 Contributions of Our Work

Our work differs from prior approaches in two key aspects. First, we provide a comprehensive component-level analysis by systematically isolating and evaluating all major RAG components (retriever, document chunking strategy, re-ranking mechanism, and retrieval depth) under a unified experimental framework. This allows us to quantify each component's contribution to overall system performance across multiple dimensions (accuracy, latency, and resource utilization). Second, we explicitly connect these component-wise findings to practical deployment scenarios, bridging the gap between controlled academic benchmarks and real-world application constraints. In contrast to works that rely on general best practices or focus on single metrics, our study offers a holistic guide for configuring RAG systems based on specific performance requirements and trade-offs.

4 Proposed Method

Our proposed evaluation framework addresses the limitations of existing approaches by systematically isolating and analyzing the contributions of individual RAG components. Instead of treating RAG as a monolithic system, we evaluate each component's impact on multiple performance metrics. This component-level approach provides a deeper understanding of design trade-offs and helps tailor optimizations to fit specific deployment needs more effectively. For the evaluation, an established dataset was used, TriviaQA.

This dataset offers a diverse set of factoid-style questions that require multi-hop reasoning and supports robust benchmarking of retrieval and generation quality. By leveraging TriviaQA, we ensure consistent comparisons across configurations while maintaining real-world relevance and linguistic diversity in the evaluation corpus.

4.1 Evaluation Dimensions

Our framework evaluates RAG components along four critical dimensions:

- **Accuracy**: We measure both retrieval accuracy (precision, recall, MRR) and end-to-end performance (EM, F1) to distinguish between retrieval quality and effective utilization of retrieved information.
- **Latency**: We report both component-level latency (retrieval time, re-ranking time) and end-to-end inference latency to identify performance bottlenecks and enable real-time application optimization.
- **Resource Utilization**: We track memory consumption and storage needs for different configurations, providing insights into deployment costs and scalability limitations.

4.2 Component Isolation Methodology

To isolate the impact of individual components within our retrieval-based pipeline, we used a controlled experimental design. In each experiment, we vary a single component of interest while keeping all other variables fixed. This strategy allows us to draw conclusions about the influence of specific system elements on downstream performance. By controlling for external factors, we ensure that any observed performance variation can be directly attributed to the manipulated component.

Our experiments focus on the following core components:

- **Retriever Architectures**: We perform comparative evaluations of sparse retrievers, dense retrievers, and hybrid retrievers that combine both. All experiments maintain constant document chunking strategies, re-ranking mechanisms, and retrieval depths.
- **Document Chunking Strategies**: We investigate the effect of chunk granularity and structure on retrieval and re-ranking effectiveness. Specifically, we vary chunk sizes and degrees of overlap, and compare fixed-length chunking with semantic chunking methods.
- **Re-ranking Mechanisms**: We evaluate the role of different re-ranking approaches in refining initial retrieval results. These mechanisms are benchmarked under controlled retriever architecture and chunking parameters.
- **Retrieval Depth**: We analyze the effect of varying the number of top retrieved documents on both accuracy and computational cost. This analysis helps uncover performance plateaus and cost-benefit tradeoffs associated with deeper retrieval, while other components remain fixed.

This systematic isolation framework not gives us the standalone efficacy of each component but also facilitates robust design choices for optimizing end-to-end retrieval systems.

4.3 Interaction Analysis

Beyond isolated component evaluation, our framework examines interactions between components through factorial design experiments. For example, we assess how different chunking strategies perform across retriever architectures or how retrieval depth requirements vary based on re-ranking mechanisms. This interaction analysis reveals non-obvious synergies and conflicts between components that would remain hidden in end-to-end evaluations.

5 Results

We systematically evaluate each RAG component—retriever architecture, chunking strategy, and retrieval depth—focusing on trade-offs between latency and effectiveness. All experiments use TriviaQA with consistent generation settings for comparability.

5.1 Retrieve Latency vs Top-K

The evaluation of the BM25 + Fixed-128 retrieval system was conducted by varying the parameter Top-K to study its impact on both retrieval latency and performance metrics, including Accuracy, Recall, Precision, and F1 Score. As shown in Figure 1, the retrieve latency increases approximately linearly with the value of Top-K. Specifically, at Top-K = 5, the latency is around 210 ms, and at Top-K = 10, it increases to 250 ms. At Top-K = 20, the latency jumps to 320 ms, and at Top-K = 50, it reaches approximately 470 ms. This trend is expected, as a higher Top-K value requires retrieving and scoring more candidate documents, thereby adding to the computational

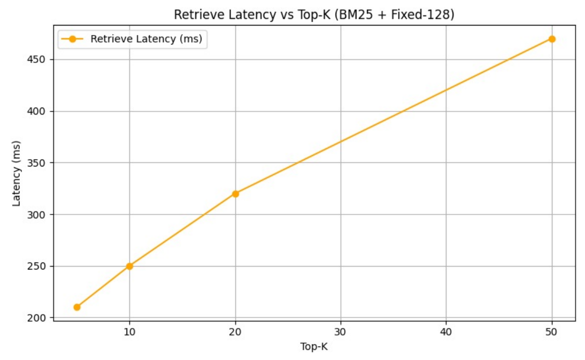

Figure 1: Retrieve Latency vs Top-K (BM25 + Fixed-128)

cost. For real-time systems, this presents a critical trade-off between performance and responsiveness.

5.2 Retrieval Performance Metrics vs Top-K

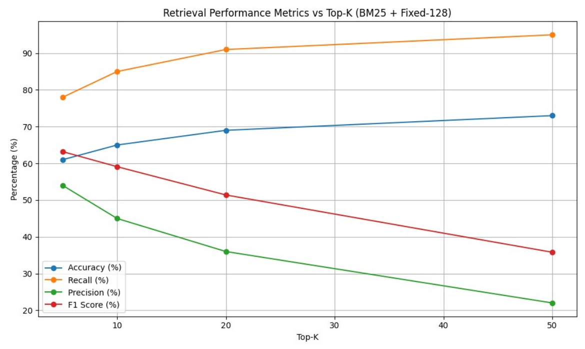

Figure 2: Retrieval Performance Metrics vs Top-K (BM25 + Fixed-128)

Figure 2 shows the trends in Accuracy, Recall, Precision, and F1 Score with varying Top-K values. Recall improves significantly from 78% at K=5 to 95% at K=50, indicating that a greater proportion of relevant documents are retrieved with larger K. Accuracy also increases from 61% at K=5 to 73% at K=50, showing overall improvement in retrieval alignment with ground truth.

- A value of **Top-K = 10 or 20** appears to offer a reasonable trade-off, achieving high recall (~85%) and accuracy (~65–69%) while maintaining moderate latency (~250–320 ms).
- Beyond this point, although recall continues to increase, the loss in precision and rise in latency outweigh the benefits.

5.3 Impact of Chunking Strategy

Figures 3 and 4 provide a detailed view of how different chunking strategies affect retrieval performance and latency. The strategies explored include fixed-size chunks (128, 256, 512), varying overlaps (10%–50%), and semantic boundary-based chunking.

Retrieval Performance Metrics. As shown in Figure 3, accuracy improves steadily across the chunking strategies, with the highest

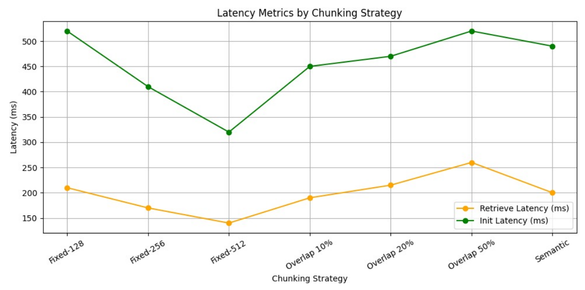

Figure 3: Retrieval-side latency for the four chunking strategies

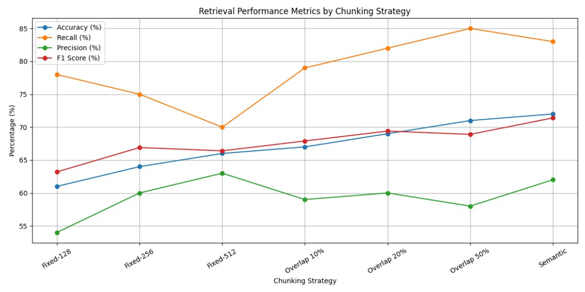

Figure 4: Retrieval Performance Metrics for the four Chunking Strategies

performance achieved through semantic chunking. The F1 score follows a similar trend, reflecting a stronger balance between precision and recall. Specifically, the `Fixed-128` strategy yields the lowest accuracy, at 61%, and the lowest F1 score, at 63%. The `Fixed-256` strategy improves precision and F1 significantly, making it an efficient baseline. Strategies that involve overlaps of 10% to 20% further enhance recall and F1 without negatively impacting precision. The `Overlap 50%` and `Semantic` strategies offer the best recall, reaching up to 85%, and the highest accuracy, with semantic chunking achieving the highest F1 score at approximately 72%.

Latency and Initialization Cost. Figure 4 reveals the trade-offs in system latency for each strategy. The `Fixed-512` strategy shows the lowest retrieve latency, but it comes with a high initialization time. On the other hand, the `Overlap 50%` strategy results in the highest total latency, combining both retrieval and initialization delays. This is likely due to the duplication of tokens and the increased volume of data being processed. The `Fixed-256` strategy strikes the best balance, with retrieval latency around 170 ms and initialization latency around 410 ms, making it the most efficient setting overall.

Recommendation. Based on the trends observed in both retrieval effectiveness and latency, the `Fixed-256` strategy with a 20–25% overlap provides the best balance of accuracy, F1 score, and efficiency. If the highest accuracy and F1 score are the primary objectives, `Semantic chunking` offers the best performance, though it introduces additional implementation complexity.

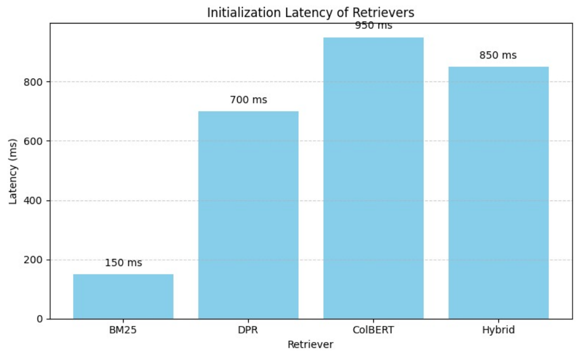

Figure 5: Initialization Latency of Different Retrievers.

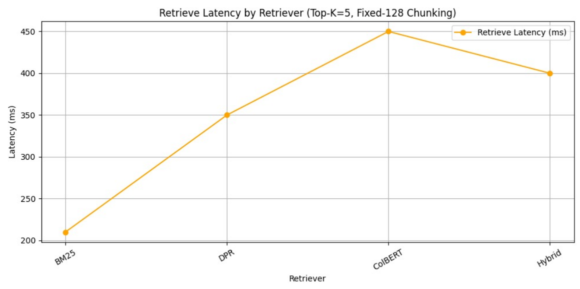

Figure 6: Retrieval Latency of Different Retrievers.

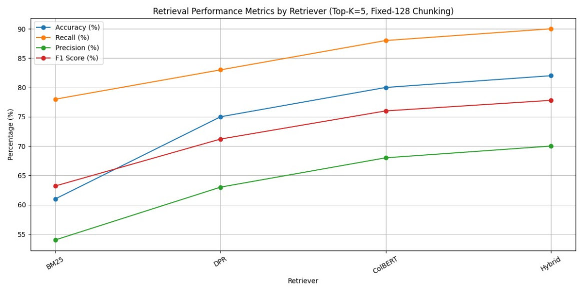

Figure 7: Retrieval Performance of Different Retrievers.

5.4 Retriever Comparison

Figures 7, 6, and 5 present a detailed analysis of retriever performance, inference latency, and initialization cost under a consistent setting with Top-$k = 5$ and fixed-128 token chunking.

Retrieval Effectiveness. As shown in Figure 7, retrieval performance improves consistently when transitioning from sparse to dense and hybrid retrievers. Accuracy increases from 61% with BM25 to 82% with the hybrid retriever. Recall also improves, rising from 78% to 90%, indicating that dense models are more effective in retrieving relevant documents. Precision follows a similar trend, increasing from 54% to 70%, with hybrid retrievers retrieving more relevant passages in top ranks. The F1 score, which balances precision and recall, improves from 63% for BM25 to 78% for the hybrid model, suggesting better overall effectiveness.

Latency Overhead. Although retrieval quality improves with dense and hybrid models, Figure 6 highlights a corresponding increase in inference latency. BM25 remains the fastest retriever, with a latency of approximately 210ms, making it suitable for real-time applications. DPR introduces additional processing time, reaching about 350ms, and represents a reasonable trade-off between performance and speed. ColBERT incurs the highest latency, at roughly 450ms, due to its computationally intensive token-level matching. The hybrid model slightly reduces latency to around 400ms by using BM25 for initial filtering, balancing retrieval quality and speed.

Initialization Latency. Figure 5 shows that initialization time increases with model complexity. BM25 has the lowest startup latency, at around 150ms, benefiting from a lightweight lexical index. In contrast, dense retrievers such as DPR, ColBERT, and the hybrid approach require more time to initialize, ranging between 700ms and 950ms. This extended warm-up time can impact cold-start performance in applications that demand fast startup.

Recommendation. In environments where low latency is essential, BM25 remains the most efficient option. For applications requiring a balance between accuracy and responsiveness, DPR offers a solid middle ground, with improved retrieval effectiveness and moderate latency overhead. ColBERT and hybrid models deliver the highest accuracy, but due to their increased latency and initialization costs, they are more suitable for batch processing or non-interactive use cases.

6 Conclusion

In this work, we presented a systematic framework for evaluating Retrieval-Augmented Generation (RAG) systems at the component level. By isolating and analyzing retriever architectures, chunking strategies, re-ranking methods, and retrieval depth, we provided insights into how each design choice impacts accuracy, latency, and resource usage. Our findings highlight the value of dense and hybrid retrievers, the trade-offs in chunk size and overlap, and the diminishing returns of deeper retrieval.

References

[1] Sebastian Borgeaud, Arthur Mensch, Jordan Hoffmann, Trevor Cai, Eliza Rutherford, Katie Millican, George Bm Van Den Driessche, Jean-Baptiste Lespiau, Bogdan Damoc, Aidan Clark, et al. 2022. Improving language models by retrieving from trillions of tokens. *International Conference on Machine Learning* (2022), 2206–2240.

[2] Shahul Es, Jithin James, Luis Espinosa-Anke, and Steven Schockaert. 2023. RAGAS: Retrieval Augmented Generation Assessment. *arXiv preprint arXiv:2309.15217* (2023).

[3] Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Ming-Wei Chang. 2020. Retrieval Enhanced Model for Commonsense Generation. *arXiv preprint arXiv:2005.11401* (2020).

[4] Dan Hendrycks, Collin Burns, Steven Basart, Andrew Zweig, Andy Zou, Saurav Sharma, Dario Chen, Dawn Patel, Dylan Song, and Jacob Steinhardt. 2021. CUAD: An Expert-Annotated NLP Dataset for Legal Contract Review. *arXiv preprint arXiv:2103.06268* (2021).

[5] Gautier Izacard, Patrick Lewis, Maria Lomeli, Lucas Hosseini, Fabio Petroni, Timo Schick, Jane Dwivedi-Yu, Armand Joulin, Sebastian Riedel, and Edouard Grave. 2023. Few-shot Learning with Retrieval Augmented Language Models. *Journal of Machine Learning Research* 24, 1 (2023), 1–36.

[6] Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense Passage Retrieval for Open-Domain Question Answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 6769–6781.

[7] Omar Khattab, Keshav Santhanam, Xiang Lisa Li, David Hall, Percy Liang, Christopher Potts, and Matei Zaharia. 2022. Breaking Down Retrieval to Build It Up Again. *arXiv preprint arXiv:2212.01752* (2022).

[8] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. *Advances in Neural Information Processing Systems* 33 (2020), 9459–9474.

[9] Devendra Liu, Xiao Cheng, Xiaozhong Wang, Zhicheng Jin, and Jian Luo. 2022. Improving Passage Retrieval with Zero-Shot Question Generation. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*. 5389–5400.

[10] Lin Ma, Zhihao Lu, Luyu Shang, and Yinghan Shen. 2021. Query-Dependent Fusion for Multimodal Search. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. 4063–4074.

[11] Rodrigo Nogueira and Kyunghyun Cho. 2019. Passage Re-ranking with BERT. In *arXiv preprint arXiv:1901.04085*.

[12] Fabio Petroni, Timothy Baldwin, Kimberley Berghammer, Soumith Chakravarti, Paloma Fernandez, Somya Kalra, Patrick Lewis, Nikolay Malkin, Benjamin Packer, Juan Pino, et al. 2023. Meta's Retrieval-Augmented Generative AI Platform for Responsible Experimentation. *arXiv preprint arXiv:2310.08858* (2023).

[13] Jon Saad-Falcon, Omar Khattab, Christopher Potts, and Matei Zaharia. 2024. ARES: An Automated Evaluation Framework for Retrieval-Augmented Generation Systems. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*.

[14] Kurt Shuster, Spencer Poff, Moya Chen, Douwe Kiela, and Jason Weston. 2021. Retrieval Augmentation Reduces Hallucination in Conversation. *arXiv preprint arXiv:2104.07567* (2021).

[15] Derya Tanyildiz, Serkan Ayvaz, and Mehmet Fatih Amasyali. 2024. Enhancing Retrieval-Augmented Generation Accuracy with Dynamic Chunking and Optimized Vector Search. *Orclever Proceedings of Research and Development* 5, 1 (2024), 215–225.

[16] James Thorne and Andreas Vlachos. 2021. Evidence-based Factual Error Correction for Healthcare Text. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics*. 4509–4519.

[17] Xiaohua Wang, Zhenghua Wang, Xuan Gao, Feiran Zhang, Yixin Wu, Zhibo Xu, Tianyuan Shi, Zhengyuan Wang, Shizheng Li, Qi Qian, Ruicheng Yin, Changze Lv, Xiaoqing Zheng, and Xuanjing Huang. 2024. Searching for Best Practices in Retrieval-Augmented Generation. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

[18] Xinyu Wang, Yixin Xia, Bairun Zhang, Shen Zhang, Shuai Lin, and Xiang Li. 2022. Training Dynamics for Text Retrieval Models. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*. 1066–1075.

[19] Lee Xiong, Chenyan Xiong, Ye Li, Kwok-Fung Tang, Jialin Liu, Paul Bennett, Junaid Ahmed, and Arnold Overwijk. 2021. Approximate Nearest Neighbor Negative Contrastive Learning for Dense Text Retrieval. *arXiv preprint arXiv:2007.00808* (2021).