NeutronRAG: Towards Understanding the Effectiveness of RAG from a Data Retrieval Perspective

Peizheng Li*
Northeastern University,China
Shengyang, China
lipeizheng@stumail.neu.edu.cn

Chaoyi Chen*
Northeastern University,China
Shengyang, China
chenchaoy@stumail.neu.edu.cn

Hao Yuan
Northeastern University,China
Shengyang, China
yuanhao@stumail.neu.edu.cn

Zhenbo Fu
Northeastern University,China
Shengyang, China
fuzhenbo@stumail.neu.edu.cn

Hang Shen
Northeastern University,China
Shengyang, China
shenhang@stumail.neu.edu.cn

Xinbo Yang
Northeastern University, China
Shengyang, China
yangxinbo@stumail.neu.edu.cn

Qiange Wang
National University of Singapore
Singapore, Singapore
wang.qg@nus.edu.sg

Xin Ai
Northeastern University, China
Shengyang, China
aoxin0@stumail.neu.edu.cn

Yanfeng Zhang†
Northeastern University,China
Shengyang, China
zhangyf@mail.neu.edu.cn

Yingyou Wen
Neusoft AI Magic Technology Research
Shengyang, China
wenyingyou@mail.neu.edu.cn

Ge Yu
Northeastern University,China
Shengyang, China
yuge@mail.neu.edu.cn

Abstract

Retrieval-Augmented Generation (RAG) has been widely used to enhance the generation quality of large language models (LLMs), particularly in domain-specific tasks. As application requirements become increasingly diverse, various RAG methods have been proposed to optimize the retrieval and generation quality of different task scenarios, such as VectorRAG, GraphRAG, and HybridRAG. However, RAG faces two key challenges in these retrieval methods: evaluating different retrieval methods and optimizing parameter configurations. First, different retrieval methods show different performances. How to uniformly compare and analyze the advantages and disadvantages of these retrieval methods remains an open research problem. Second, the effectiveness of RAG is highly sensitive to key parameter configurations. Optimizing these parameters is challenging due to the complexity of the parameter space and the difficulty in identifying the sources of errors in the generated responses. Existing RAG tools typically use a single retrieval method, lacking analytical capabilities and multi-strategy support. To address these challenges, we introduce NeutronRAG, a demonstration of understanding the effectiveness of RAG from a data retrieval perspective. NeutronRAG supports hybrid retrieval strategies and helps researchers iteratively refine RAG configuration to improve retrieval and generation quality through systematic analysis, visual feedback, and parameter adjustment advice. It facilitates data-driven decisions to enhance LLM generation quality while exploring effective retrieval strategies in RAG systems. The web app is available at: https://github.com/iDC-NEU/NeutronRAG.

CCS Concepts

• **Information systems** → *Multidimensional range search*; **Evaluation of retrieval results**.

Keywords

Retrieval-Augmented Generation; VectorRAG; GraphRAG; HybridRAG

ACM Reference Format:
Peizheng Li, Chaoyi Chen, Hao Yuan, Zhenbo Fu, Hang Shen, Xinbo Yang, Qiange Wang, Xin Ai, Yanfeng Zhang, Yingyou Wen, and Ge Yu. 2025. NeutronRAG: Towards Understanding the Effectiveness of RAG from a Data Retrieval Perspective. In *Companion of the 2025 International Conference on Management of Data (SIGMOD-Companion '25), June 22–27, 2025, Berlin, Germany*. ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3722212.3725119

1 Introduction

Retrieval-augmented generation (RAG) has emerged as an effective approach to mitigate the hallucination problem in large language models (LLMs). By integrating external knowledge corpus, RAG enhances the generation capability of LLMs, enabling them to generate more accurate and contextually relevant responses. The core of

*Equal contribution.
†Corresponding author.

SIGMOD-Companion '25, Berlin, Germany.

ACM ISBN 979-8-4007-1564-8/25/6
https://doi.org/10.1145/3722212.3725119

RAG is to combine the strengths of traditional information retrieval methods with the generative capabilities of LLMs. As application requirements become increasingly diverse, various RAG methods have been proposed to optimize the retrieval and generation quality for different task scenarios. Specifically, VectorRAG [7] exploits vector-based indexing for efficient semantic retrieval. GraphRAG [2, 3, 6] exploits graph structures to enhance the capabilities to retrieve global information. HybridRAG [5, 10] combines the two retrieval strategies for better adaptability and generation quality.

RAG faces two key challenges in data retrieval: evaluating different retrieval strategies and optimizing parameter configurations. First, different retrieval methods, such as GraphRAG, VectorRAG, and HybridRAG, exhibit varying performance on the same tasks. A unified framework for comparing these methods, analyzing their strengths and weaknesses, and identifying the key factors affecting their performance remains an open research problem. Second, the effectiveness of RAG is highly sensitive to key parameter configurations, such as `top-k` and `chunksize` for vector retrieval, `k-hop` and `pruning` for graph retrieval, and fusion strategy for HybridRAG. These parameters directly affect retrieval quality and generation accuracy. However, optimizing these parameters is challenging due to the complexity of the parameter space and the difficulty in identifying the sources of errors in the generated responses. For example, identifying the source of incorrect answers or understanding why certain retrieval strategies fail often requires manual inspection and domain-specific expertise. Therefore, these challenges highlight the need for powerful tools to comprehensively evaluate the effectiveness of different retrieval methods.

Despite the growing use of RAG, there is a lack of effective tools for analyzing the effectiveness of different retrieval strategies, making it challenging for developers to optimize RAG methods efficiently. Existing RAG tools [4, 8, 9] have several limitations. First, most tools focus on a single RAG method and do not support the exploration of hybrid strategies, that is, combining the advantages of multiple RAG methods to obtain more accurate and relevant results, limiting the ability to compare different retrieval strategies and configurations. Second, these tools often lack analytical capabilities and do not provide sufficient support for parameter tuning, making it difficult to identify and resolve problems in the retrieval process. These limitations highlight the need for a comprehensive tool that enables deeper analysis, comparison, and optimization of various RAG configurations.

In this demonstration, we introduce NeutronRAG, a tool that aims to fill these gaps by providing a comprehensive RAG pipeline analysis platform. Unlike existing tools, NeutronRAG supports flexible parameter configuration of RAG methods, allowing researchers to experiment and fine-tune key parameters to optimize performance for specific downstream tasks. The tool provides systematic performance evaluation, including metrics for retrieval quality and generation accuracy, and enables intuitive visual analysis of the retrieval process by providing detailed visualizations, error analysis, and parameter tuning suggestions. Overall, NeutronRAG provides an intuitive way to visualize and evaluate the contribution of each form of data to the overall model performance. This tool can provide an operational platform and guidance for effective knowledge management and improving retrieval of different data formats in RAG methods.

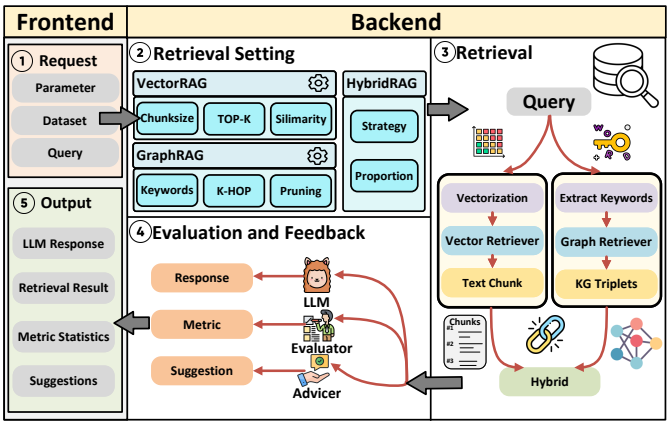

Figure 1: Architecture of NeutronRAG.

2 System Overview

We provide a comprehensive overview of NeutronRAG. Figure 1 presents the architecture of NeutronRAG and the interaction between its frontend and backend. The frontend serves as an interactive interface, enabling users to input data, configure parameters, and visualize RAG performance metrics along with enhancement recommendations. The backend performs a comprehensive analysis of the RAG conversation history, including accuracy statistics, error analysis, and improvement suggestions.

2.1 Frontend

The frontend is designed to support user interaction and visualize evaluation results. It consists of two components: the input interface and the output interface.

Input. The input interface handles user input, allowing users to customize retrieval parameters for various RAG methods based on their research needs, and to select the dataset and specific queries for evaluation. This component is connected to the retrieval settings component of the backend, enabling users to influence the retrieval process by adjusting parameters.

Output. The output interface presents the evaluation feedback of NeutronRAG, including LLM responses, retrieval result visualization, evaluation metric statistics, and RAG parameter optimization suggestions. These feedback components support users in iteratively improving the performance of RAG methods. Specifically, the LLM response section provides services by invoking the LLM interface. The retrieval result visualization module presents the outputs of both vector-based and graph-based retrieval, namely text chunks and subgraphs, emphasizing the most relevant content to enhance readability and interpretability. To facilitate user understanding of graph-based results, we employ `Cytoscape.js`, a JavaScript library specialized for interactive graph rendering. This component enables intuitive exploration of entities and their relationships within the knowledge graph, offering clearer insights into how graph-structured information contributes to response generation. The evaluation metrics section consists of two components: the historical records dashboard and the evaluation dashboard. The historical records provide a comprehensive record of all conversation histories, allowing users to track the performance of different RAG

methods across the dataset. The evaluation dashboard provides a comprehensive view of the evaluation metrics and performance of the RAG methods. It presents real-time statistics across three key aspects: the overall accuracy of each method, categorized analysis of error cases, and detailed metrics for both retrieval and generation stages, including retrieval precision, relevance, recall, and generation fidelity. The chart display is implemented by `Chart.js` and `Gauge.js` to enhance information readability and intuitive comparison.

2.2 Backend

The backend manages the entire RAG pipeline, including the hybrid mechanism and the evaluation of individual RAG methods. Figure 1 illustrates the backend workflow that consists of three components: retrieval settings, retrieval, and evaluation and feedback.

Retrieval Settings. In the retrieval settings, the backend provides a flexible configuration interface to support the retrieval requirements of various RAG methods. In VectorRAG, users can configure the following key parameters: (1) The `chunksize` controls the length of text chunks. A smaller `chunksize` enables fine-grained retrieval, while a larger value preserves more contextual information. (2) The similarity (θ) determines the number of retrieved text chunks, as only those with a similarity score above θ are included in the results. (3) The `top-k` specifies the number of retrieved chunks fed into the LLM. A smaller `top-k` may lead to missing critical information, while a larger value introduces more irrelevant content, degrading the quality of generation. In GraphRAG, users can configure the following key parameters: (1) The `keywords` parameter determines the number of entities used for retrieval. (2) The `k-hop` parameter specifies the retrieval hops for each entity. As the number of hops increases, more contextual information is retrieved, but it may also introduce a significant amount of irrelevant information. (3) The `pruning` parameter controls the size of the retrieved subgraph by retaining the most relevant entities and relationships based on similarity. In HybridRAG, the backend allows users to choose different hybrid strategies (e.g., union, intersection, or routing) and adjust the ratio between vector-based and graph-based results. This helps combine the strengths of VectorRAG and GraphRAG to generate more complete and accurate answers.

Retrieval. In the retrieval component, information relevant to the user's query is retrieved from the database based on the parameters configured in the retrieval settings. In VectorRAG, we first use an embedding model to convert the query into a vector representation. Then, we retrieve text chunks from the vector database whose similarity scores are above a predefined threshold (θ). Finally, in the post-retrieval stage, we select the `top-k` most similar text chunks. In GraphRAG, we first use an LLM to extract keywords from the query. These keywords serve as starting nodes to retrieve their k-hop subgraphs from the graph database. We then prune the retrieved subgraphs based on the similarity between the triplets and the query, ensuring that only the most relevant triples are retained. In HybridRAG, we combine the retrieval results from VectorRAG and GraphRAG based on a user-defined hybrid strategy and ratio. In addition, NeutronRAG retains the retrieval results of all three RAG methods for subsequent data analysis and comparison.

Evaluation and Feedback. In the evaluation and feedback component, the user-selected LLM is first used to generate responses. The evaluator then analyzes the performance of the RAG methods, and the advisor provides targeted optimization suggestions. Specifically, the evaluator assesses the quality of retrieved information, as well as the correctness and faithfulness of the generated responses. It also performs error-type analysis for incorrect answers. For common types of errors, we have designed a set of recommendation templates. The advisor module leverages these templates to offer tailored suggestions, while for unclassified errors, it invokes the LLM to generate adaptive recommendations.

3 Demonstration Scenario

Figure 2 illustrates our demonstration scenarios, which are snapshots of the frontend. To demonstrate its functionality, we present a workflow using the RGB dataset [1]. RGB is an evaluation dataset for RAG methods, constructed from recent news articles and external documents retrieved from search engines.

Step1: Initial Selection. When users interact with our tool, they are directed to the interface, as shown in Figure 2. The first step involves selecting the LLM to run in the backend. Users can either choose from the available models or provide their own API-KEY to call a third-party LLM. Once the model is selected, users can proceed to configure various parameters in the left-side Setting bar to adjust the retrieval and generation settings based on their requirements.

Step2: LLM Response. In the second step, users can click the "RUN" button to start the backend LLM processing. Then, our tool will automatically run the LLM using the selected dataset and configured parameters. Our tool automatically calls VectorRAG, GraphRAG, and HybridRAG to process the same query at the same time, and the responses are displayed on the interface. Furthermore, users can enter queries individually, review the retrieved results, and gain insights. If a standard answer is provided, the tool analyzes the generated response and offers detailed performance insights. If the answer is unknown, users can select a preferred model response, allowing the tool to seamlessly transition to the evaluation stage in both cases.

Step3: Evaluation. In the third step, NeutronRAG automatically loads the results of the previous step and processes the data to present it in an interactive and user-friendly interface. This is shown in the `Query and Feedback Area` and `Metric Area` in Figure 2. The `Query and Feedback Area` contains several key components. The `Answer Block` displays the answers generated by the LLM, the `Retrieval Result Block` visualizes the retrieval results, allowing users to view the relevant documents or subgraphs used to generate the model response, and the `Suggestions Block` displays parameter optimization suggestions provided to users to help users adjust settings to improve accuracy. The `Metric Area` consists of two main parts. The `History Block` provides a summary of the entire question and answer history. The `Dashboard` provides a detailed evaluation of the model performance and provides users with in-depth metric and visualizations.

In our scenario, when users discover an incorrect answer in the `History Block`, such as the query with id 286, they can click on the specific query to further investigate. Upon comparing the

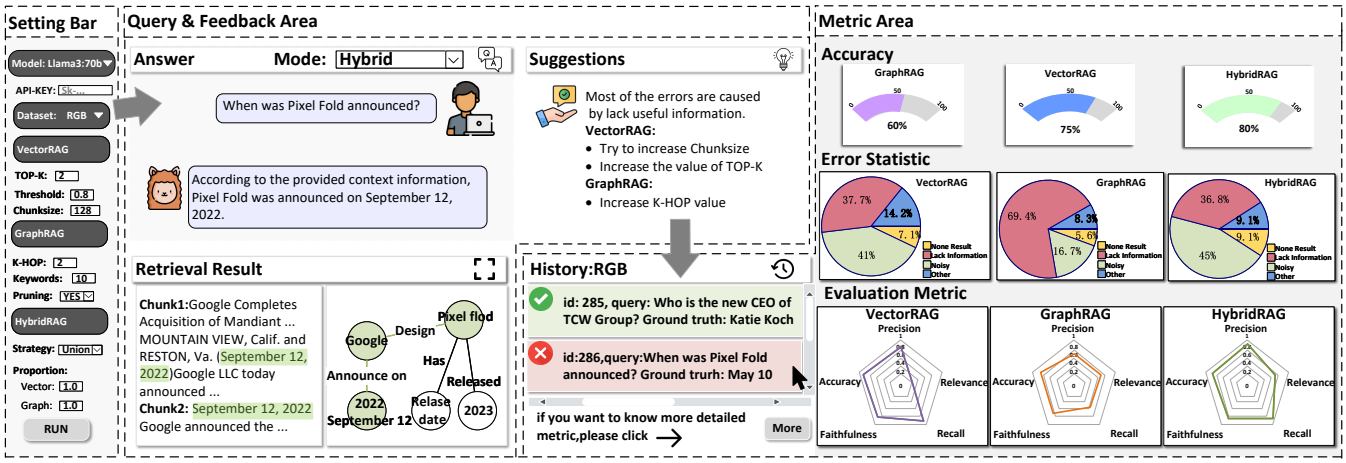

Figure 2: Interface of NeutronRAG.

LLM's response in the `Answer Block`, the user notices that the correct answer is "May 10", while the LLM responsed "September 12". To investigate the cause of this error, the user examines the highlighted sections in the `Retrieval Result Block` to identify where the LLM response comes from. This process reveals that the error occurred because the relevant knowledge for this question was not retrieved, leading to the incorrect answer.

After identifying the errors and their causes for a specific query, users can gain deeper insights by accessing the Dashboard by clicking the "More" button in the lower right corner. This takes them to a more detailed view of the evaluation metrics. The dashboard provides users with a comprehensive breakdown of metrics such as retrieval recall, precision, relevance, and generation accuracy, in addition to statistics on the causes of the errors.

Step4: Refine Suggestion. After gaining a deeper understanding of the overall performance of the RAG method, users can adjust various parameters based on their own insight to further optimize the performance of different RAG modes. To help users optimize their configurations, in `Suggestions Block`, we provide actionable parameter modification suggestions based on the analysis of retrieval and generation indicators to provide users with some inspiration. For example, when users learn that some questions are answered incorrectly because no relevant knowledge is retrieved, our tool will suggest increasing the `k-hop` and `chunksize` parameters to improve the retrieval range.

4 Conclusion

In this demonstration, we introduce NeutronRAG, an interactive tool designed to optimize retrieval strategies and parameter configurations in RAG methods. NeutronRAG provides comprehensive evaluation and analysis of different retrieval methods, including retrieval effectiveness evaluation, generation accuracy analysis, and error diagnosis. It supports real-time parameter tuning, hybrid retrieval strategy configurations, and detailed visualizations, enabling researchers to iteratively refine RAG methods. By facilitating data-driven decision-making and offering intuitive feedback, NeutronRAG enhances retrieval performance and advances research in RAG-based retrieval and generation techniques. In addition, its modular design allows seamless integration with existing RAG frameworks, making it a practical and adaptable solution for improving RAG methods.

Acknowledgments

This work is supported by the National Key R&D Program of China (2023YFB4503601), the National Natural Science Foundation of China (U2241212, 62461146205, and 62202088), the Distinguished Youth Foundation of Liaoning Province (2024021148-JH3/501) and the Fundamental Research Funds for the Central Universities (N2416004).

References

[1] Jiawei Chen, Hongyu Lin, Xianpei Han, and Le Sun. 2023. Benchmarking Large Language Models in Retrieval-Augmented Generation. arXiv:2309.01431 [cs.CL] https://arxiv.org/abs/2309.01431

[2] Darren Edge, Ha Trinh, Newman Cheng, Joshua Bradley, Alex Chao, Apurva Mody, Steven Truitt, and Jonathan Larson. 2024. From Local to Global: A Graph RAG Approach to Query-Focused Summarization. *CoRR* abs/2404.16130 (2024).

[3] Zirui Guo, Lianghao Xia, Yanhua Yu, Tu Ao, and Chao Huang. 2024. Lightrag: Simple and fast retrieval-augmented generation. (2024).

[4] Jiajie Jin, Yutao Zhu, Xinyu Yang, Chenghao Zhang, Zhicheng Dou, and Ji-Rong Wen. 2024. FlashRAG: A Modular Toolkit for Efficient Retrieval-Augmented Generation Research. *CoRR* abs/2405.13576 (2024). https://arxiv.org/abs/2405.13576

[5] Rishi Kalra, Zekun Wu, Ayesha Gulley, Airlie Hilliard, Xin Guan, Adriano Koshiyama, and Philip Colin Treleaven. 2024. HyPA-RAG: A Hybrid Parameter Adaptive Retrieval-Augmented Generation System for AI Legal and Policy Applications. In *Proceedings of the 1st Workshop on Customizable NLP: Progress and Challenges in Customizing NLP for a Domain, Application, Group, or Individual (CustomNLP4U)*. Association for Computational Linguistics, 237–256. doi:10.18653/v1/2024.customnlp4u-1.18

[6] Lei Liang, Mengshu Sun, Zhengke Gui, Zhongshu Zhu, Zhouyu Jiang, Ling Zhong, Yuan Qu, Peilong Zhao, Zhongpu Bo, Jin Yang, et al. 2024. KAG: Boosting LLMs in Professional Domains via Knowledge Augmented Generation. *arXiv preprint arXiv:2409.13731* (2024).

[7] Xinbei Ma, Yeyun Gong, Pengcheng He, Hai Zhao, and Nan Duan. 2023. Query Rewriting for Retrieval-Augmented Large Language Models. arXiv:2305.14283 [cs.CL] https://arxiv.org/abs/2305.14283

[8] David Mezzetti. 2020. *txtai: the all-in-one embeddings database.* https://github.com/neuml/txtai

[9] Malte Pietsch, Timo Möller, Bogdan Kostic, Julian Risch, Massimiliano Pippi, Mayank Jobanputra, Sara Zanzottera, Silvano Cerza, Vladimir Blagojevic, Thomas Stadelmann, Tanay Soni, and Sebastian Lee. 2019. *Haystack: the end-to-end NLP framework for pragmatic builders.* https://github.com/deepset-ai/haystack

[10] Bhaskarjit Sarmah, Benika Hall, Rohan Rao, Sunil Patel, Stefano Pasquali, and Dhagash Mehta. 2024. HybridRAG: Integrating Knowledge Graphs and Vector Retrieval Augmented Generation for Efficient Information Extraction. arXiv:2408.04948 [cs.CL] https://arxiv.org/abs/2408.04948