FlashRAG: A Modular Toolkit for Efficient Retrieval-Augmented Generation Research

Jiajie Jin
Yutao Zhu*
jinjiajie@ruc.edu.cn
ytzhu@ruc.edu.cn
Gaoling School of Artificial Intelligence
Renmin University of China
Beijing, China

Zhicheng Dou*
Guanting Dong
Xinyu Yang
dou@ruc.edu.cn
Gaoling School of Artificial Intelligence
Renmin University of China
Beijing, China

Chenghao Zhang
Tong Zhao
Zhao Yang
Ji-Rong Wen
Gaoling School of Artificial Intelligence
Renmin University of China
Beijing, China

Abstract

With the advent of large language models (LLMs) and multimodal large language models (MLLMs), the potential of retrieval-augmented generation (RAG) has attracted considerable research attention. However, the absence of a standardized framework for implementation, coupled with the inherently complex RAG process, makes it challenging and time-consuming for researchers to compare and evaluate these approaches in a consistent environment. In response to this challenge, we develop FlashRAG, an efficient and modular open-source toolkit designed to assist researchers in reproducing and comparing existing RAG methods and developing their own algorithms within a unified framework. Our toolkit has implemented 16 advanced RAG methods and gathered and organized 38 benchmark datasets. It has various features, including a customizable modular framework, a rich collection of pre-implemented RAG works, comprehensive datasets, efficient auxiliary pre-processing scripts, and extensive and standard evaluation metrics. Our toolkit and resources are available at https://github.com/RUC-NLPIR/FlashRAG.

CCS Concepts

• **Computing methodologies** → *Natural language generation.*

Keywords

Retrieval Augmented Generation, Toolkit, RAG, Efficient, Research

ACM Reference Format:
Jiajie Jin, Yutao Zhu, Zhicheng Dou, Guanting Dong, Xinyu Yang, Chenghao Zhang, Tong Zhao, Zhao Yang, and Ji-Rong Wen. 2025. FlashRAG: A Modular Toolkit for Efficient Retrieval-Augmented Generation Research. In *Companion of the 16th ACM/SPEC International Conference on Performance Engineering (WWW Companion '25), April 28-May 2, 2025, Sydney, NSW, Australia.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3701716.3715313

*Corresponding authors.

1 Introduction

In the era of large language models (LLMs), retrieval-augmented generation (RAG) [5] has emerged as an effective solution to mitigate hallucination issues by leveraging external knowledge bases. However, with the introduction of a large number of new algorithms and models in recent years, comparing these methods in a consistent setting has become increasingly challenging.

Existing methods are often not open-source or require specific configurations for implementation, making adaptation to custom data and components challenging. Datasets and retrieval corpora frequently vary, with resources scattered and necessitating significant pre-processing efforts. Additionally, the inherent complexity of RAG systems—comprising indexing, retrieval, and generation—often demands extensive technical implementation. While some RAG toolkits, such as LangChain[1] and LlamaIndex[2], are available, they prove to be complex and cumbersome, limiting researchers' ability to tailor processes to their specific needs. Thus, there is a clear demand for a unified, research-focused RAG toolkit to simplify method development and facilitate comparative studies.

To address the aforementioned issues, we introduce FlashRAG, an open-source library that empowers researchers to reproduce, benchmark, and innovate within the RAG domain efficiently. This library offers built-in pipelines for replicating existing work, customizable components for crafting tailored RAG workflows, and streamlined access to organized datasets and corpora to accelerate research processes. FlashRAG provides a more researcher-friendly solution compared to existing toolkits. To summarize, the key features of our FlashRAG library include:

A comprehensive, customizable, and efficient modular RAG framework. FlashRAG offers a highly modular setup at both the component and pipeline levels, featuring 5 core modules and 16 diverse RAG subcomponents that can be independently integrated or combined into pipelines. Additionally, we provide 9 standardized RAG processes and auxiliary scripts for tasks such as downloading and chunking Wikipedia for corpus construction, building retrieval indexes, and preparing retrieval results, resulting in an efficient and user-friendly end-to-end RAG framework.

[1]https://www.langchain.com
[2]https://www.llamaindex.ai

Support for multi-modal RAG scenarios. FlashRAG covers a diverse range of scenarios, encompassing both text-only and multi-modal modalities in RAG system deployment, providing researchers with technical support across a variety of application scenarios.

Pre-implemented advanced RAG algorithms and benchmark evaluation. FlashRAG provides the most comprehensive implementation of existing work, featuring 16 advanced RAG algorithms that encompass 4 paradigms, facilitating fair evaluation under consistent settings and enhancing reproducibility. Additionally, we have gathered 38 commonly used datasets and standardized their formats while offering a broad set of RAG evaluation metrics.

2 The Toolkit: FlashRAG

FlashRAG is designed for RAG research and has three hierarchical modules (Figure 1): environment, component, and pipeline. The environment module offers resources like datasets, hyperparameters, and evaluation metrics. The component module features RAG components tailored for specific functions, such as retrieval and generation. The pipeline module integrates these into a full RAG process. This paper focuses on the component and pipeline modules, with further details in our library's documentation.

2.1 Component Module

FlashRAG is organized into five main components, each designed to function autonomously or within a combined application, enhancing both flexibility and specificity in the RAG process.

Judger functions determines whether a query needs retrieval. Given the limited studies in this domain, we implement a judger based on SKR [23], which utilizes LLM self-knowledge data to determine the necessity of retrieval.

Retriever implementations are extensively covered by our toolkit. For the text-only retrieval, we integrate the Pyserini library [15] to support sparse retrieval methods like BM25 [19]. For dense retrieval, we consider various BERT-based embedding models along with T5-based models. In the multimodal domain, we deploy the widely used CLIP family [18] for cross-modal retrieval. We employ FAISS for efficient vector database operations and integrate sentence-transformers library to improve the overall adaptability.

Reranker aims at refining the order of retrieved results. FlashRAG supports many cross-encoder models and facilitates the use of bi-encoder models for scenarios using embedding models for reranking. Rerankers can be seamlessly integrated with any retriever through a decorator, enabling simple and flexible combinations.

Refiner processes input text to optimize it for generation by reducing token usage and noise. We have implemented three types of refiners: extractive, abstractive, perplexity-based [8]. Each type employs different methods to handle retrieved passages, such as semantic extraction or summarization.

Generator is the final component in the RAG process. We integrate two advanced LLM acceleration libraries, vLLM [13] and FastChat [30]. Moreover, we develop inference frameworkds for both open-source and closed-source MLLMs, leveraging the Hugging Face ecosystem and providing a native interface to the Transformers library [24] to enhance system robustness. This module also includes encoder-decoder models and supports fusion-in-decoder technique to improve processing efficiency with retrieved content.

Table 1: Comparison between FlashRAG and other toolkits.

Toolkit	Automatic Evaluation	Multi modal	Corpus Helper	# Provided Dataset	# Support Methods
Langchain	✗	✓	✓	-	2
LlamaIndex	✓	✓	✓	-	2
Haystack [17]	✓	✗	✗	-	-
FastRAG [6]	✗	✗	✗	2	1
LocalRQA [27]	✓	✗	✗	3	-
AutoRAG [11]	✓	✗	✗	4	2
RAGLab [29]	✓	✗	✓	10	6
FlashRAG (ours)	✓	✓	✓	**38**	**16**

2.2 Pipeline Module

To systematically implement the operational logic of various RAG tasks, we identify four primary types of RAG process flows for FlashRAG: Sequential, Branching, Conditional, and Loop, based on an in-depth analysis of RAG-related literature [4].

Sequential Pipeline follows a linear execution flow: "query → retriever → post-retrieval (reranker, refiner) → generator." After configuring the settings, the library automatically loads the required components and process logic. FlashRAG supports both text-only and multimodal modes, providing researchers with more deployment options.

Branching Pipeline executes multiple paths in parallel for a single query (often one path per retrieved passage) and merges the results from all paths to form the final output.

Conditional Pipeline utilizes a judger to direct the query into different execution paths based on predefined criteria. Queries requiring retrieval follow the standard sequential process, while others bypass retrieval and proceed directly to generation. FlashRAG provides utility functions to split and merge the input dataset based on the judger's decisions, ensuring batch processing and enhanced pipeline efficiency. Additionally, the conditional pipeline supports integration with various types of pipelines, enabling dynamic execution based on judger's results.

Loop Pipeline involves complex interactions between retrieval and generation processes, often containing multiple cycles of retrieval and generation. Compared to the previous ones, this pipeline is more flexible, and thus can often yield better performance.

2.3 Datasets and Evaluation Metric

2.3.1 Datasets for RAG. We collect and pre-process 38 benchmark datasets, covering a wide range of RAG scenarios. Each dataset has been standardized into a unified JSONL format, comprising four fields per item: ID, question, golden answer, and metadata. These processed datasets are readily accessible on HuggingFace. For multimodal datasets, image and text are set as two columns respectively, and image is directly included in the dataset.

2.3.2 Retrieval Corpus. We use Wikipedia and MS MARCO as our retrieval corpus.

Wikipedia passages: This collection includes passages from English Wikipedia. We offer scripts for easy downloading and pre-processing, as well as chunking functions for custom segmentation, ensuring compatibility with various Wikipedia versions.

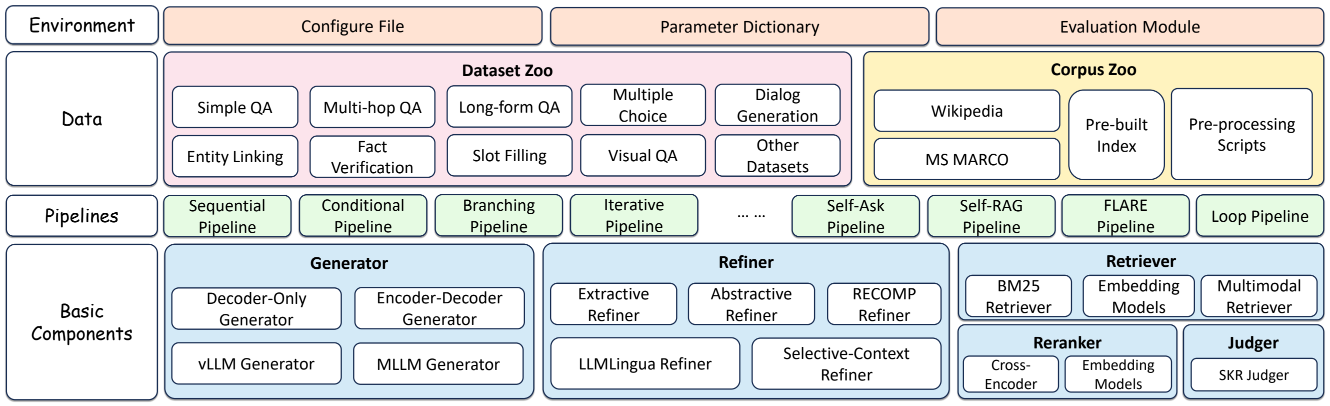

Figure 1: An overview of FlashRAG.

Table 2: The benchmarking results. *Optimize component* represents the primary component optimized by the method. Methods marked with $*$ denote the use of a trained generator. Seq., Branch., and Cond. denote Sequential, Branching, and Conditional pipelines, respectively.

Method	Optimize component	Pipeline type	NQ (EM)	TriviaQA (EM)	HotpotQA (F1)
Naive Generation	-	Seq.	22.6	55.7	28.4
Standard RAG	-	Seq.	35.1	58.8	35.3
AAR [28]	Retriever	Seq.	30.1	56.8	33.4
LongLLMLingua [9]	Refiner	Seq.	32.2	59.2	37.5
RECOMP-abs. [25]	Refiner	Seq.	33.1	56.4	37.5
Trace [2]	Refiner	Seq.	30.7	50.2	34.0
Spring [31]	Generator	Seq.	37.9	64.6	42.6
Ret-Robust* [26]	Generator	Seq.	42.9	68.2	35.8
SuRe [12]	Flow	Branch.	37.1	53.2	33.4
REPLUG [21]	Generator	Branch.	28.9	57.7	31.2
SKR [23]	Judger	Cond.	33.2	56.0	32.4
Adaptive-RAG [7]	Judger	Cond.	35.1	56.6	39.1
Self-RAG* [1]	Flow	Loop	36.4	38.2	29.6
FLARE [10]	Flow	Loop	22.5	55.8	28.0
Iter-RetGen [20]	Flow	Loop	36.8	60.1	38.3
IRCoT [22]	Flow	Loop	33.3	56.9	41.5

MS MARCO passages [16]: Comprising 8.8 million passages from Bing, this dataset is smaller and pre-processed compared to Wikipedia. It is available on HuggingFace, with direct links provided in our library for easy access.

2.3.3 Evaluation Metrics. FlashRAG supports several commonly used evaluation metrics to measure the quality of the RAG process:

Retrieval-aspect metrics: FlashRAG supports four metrics including recall@k, precision@k, F1@k, and mean average precision (MAP) to evaluate retrieval quality. Different from evaluation in standalone retrieval systems, the passages retrieved in the RAG process often lack golden labels (*e.g.*, related or unrelated tags). Therefore, we consider the presence of the golden answer within retrieved passages as an indicator of relevance.

Generation-aspect metrics: To evaluate the quality of generation, FlashRAG supports five metrics: token-level F1 score, exact match, accuracy, BLEU, and ROUGE-L. Moreover, FlashRAG also support llm-as-a-judge method for evaluation. To accommodate custom evaluation metrics, FlashRAG provides a metric template for users. As our library automatically saves intermediate results, users can conveniently evaluate results from intermediate components.

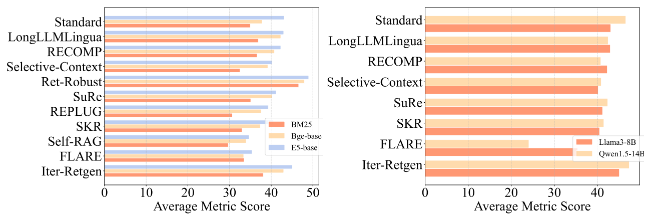

Figure 2: Average results on NQ, TriviaQA, and HotpotQA datasets for baseline methods with different settings. Left: Results with three retrievers. Right: Results with two generator models of varying parameter scales.

3 Experiments

Backbones and Methods. To assess FlashRAG, we conduct experiments using `LLaMA-3-8B-instruct` as the generator and `E5-base-v2` as the retriever, evaluating various RAG methods. For the multimodal domain, we use `Qwen2-VL` and `InternVL2` as MLLM backbones. Our RAG baselines include: AAR [28] focuses on the retriever; LongLLMLingua [9], RECOMP [25], and SC [14] refine and compress the input; Ret-Robust [26] and REPLUG [21] enhance the generator and decoding strategies; SKR [23] and Adaptive-RAG [7] assess the necessity of retrieval for a query; and SuRe [12], Self-RAG [1], FLARE [10], Iter-RetGen [20], and ITRG [3], IRCOT [22] optimize the entire RAG flow, including multi-turn retrieval and generation processes.

Benchmarking Results. The main results are shown in Table 2. Overall, RAG methods significantly outperform the direct generation baseline. This demonstrates the benefits of incorporating external knowledge into the generation process. Key observations include: (1) Standard RAG with advanced retrievers and generators is a strong baseline, showing robust performance across six datasets. (2) All three methods employing refiners exhibit significant improvements, particularly on multi-hop datasets(e.g., HotpotQA). This is potentially because complex problems result in

Table 3: The performance comparison of different MLLM backbones with and without FlashRAG's multimodal retrieval module, which combines BM25 and CLIP to recall Top-1 knowledge from the corresponding training sets.

Method	Use Retriever?	MMQA (EM.)	GAOKAO-MM (Acc.)
Qwen2-VL-2B	✗	0.274	0.268
Qwen2-VL-2B	✓	0.313	0.268
Qwen2-VL-7B	✗	0.304	0.304
Qwen2-VL-7B	✓	0.300	0.398
InternVL2.5-8B	✗	0.343	0.409
InternVL2.5-8B	✓	0.378	0.374

less accurate passage retrieval, introducing more noise and highlighting the necessity for refiner optimization. (3) As for generator optimization, Ret-Robust fine-tunes the `LLaMA-2-13B` model via LoRA, significantly enhancing the generator's capability of understanding retrieved passages and outperforming other training-free approaches.

Multimodal Results. To validate FlashRAG's plug-and-play capability in multimodal scenarios, we evaluate various MLLM backbones on three VQA benchmarks (Table 3). Our findings are: (1) Larger MLLM backbones with multimodal retrieval modules show stronger VQA performance. (2) Multimodal retrieval improves performance in common-sense VQA tasks but not in complex reasoning tasks like GAOKAO-MM, where accuracy and consistency in retrieved knowledge are crucial.

Impact of Retrievers and Generators We investigate the effect of retrievers and generators. As shown in Figure 2, retrieval quality greatly impacts performance, with a nearly 10% gap between BM25 and E5 retrievers, likely due to noise from sparse retrieval. Interestingly, larger generators cannot always outperform smaller ones, as seen in FLARE and RECOMP, suggesting that RAG performance is more dependent on generation capabilities than model size.

4 Conclusion

In this study, we introduce FlashRAG, a modular toolkit designed to address replicability and high development costs in RAG research. FlashRAG provides benchmark datasets, advanced RAG methods, pre-processing tools, and standardized evaluation metrics, enabling both replication of existing techniques and the development of new approaches. Our experiments across various datasets validate its effectiveness, aiming to lower technical barriers, enhance reproducibility, and accelerate innovation in the RAG domain.

Acknowledgments

This work was supported by the National Natural Science Foundation of China No. 62272467, Beijing Natural Science Foundation No. L233008, and Beijing Municipal Science and Technology Project No. Z231100010323009.

References

[1] A. Asai, Z. Wu, Y. Wang, A. Sil, and H. Hajishirzi. 2023. Self-RAG: Learning to Retrieve, Generate, and Critique through Self-Reflection. *ArXiv preprint* (2023).

[2] J. Fang, Z. Meng, and C. Macdonald. 2024. TRACE the Evidence: Constructing Knowledge-Grounded Reasoning Chains for Retrieval-Augmented Generation. *ArXiv preprint* (2024).

[3] Z. Feng, X. Feng, D. Zhao, M. Yang, and B. Qin. 2023. Retrieval-Generation Synergy Augmented Large Language Models. *ArXiv preprint* (2023).

[4] Y. Gao, Y. Xiong, X. Gao, K. Jia, J. Pan, Y. Bi, Y. Dai, J. Sun, M. Wang, and H. Wang. 2023. Retrieval-Augmented Generation for Large Language Models: A Survey. *ArXiv preprint* (2023).

[5] K. Guu, K. Lee, Z. Tung, P. Pasupat, and M.-W. Chang. 2020. REALM: Retrieval-Augmented Language Model Pre-Training. *ArXiv preprint* (2020).

[6] P. Izsak, M. Berchansky, D. Fleischer, and R. Laperdon. 20f. fastRAG: Efficient Retrieval Augmentation and Generation Framework. *ArXiv preprint* (20f).

[7] S. Jeong, J. Baek, S. Cho, S. Hwang, and J. Park. 2024. Adaptive-RAG: Learning to Adapt Retrieval-Augmented Large Language Models through Question Complexity. In *Proc. of NAACL-HLT*.

[8] H. Jiang, Q. Wu, C.-Y. Lin, Y. Yang, and L. Qiu. 2023. LLMLingua: Compressing Prompts for Accelerated Inference of Large Language Models. In *Proc. of EMNLP*.

[9] H. Jiang, Q. Wu, X. Luo, D. Li, C.-Y. Lin, Y. Yang, and L. Qiu. 2023. LongLLMLingua: Accelerating and Enhancing LLMs in Long Context Scenario. *ArXiv preprint* (2023).

[10] Z. Jiang, F. Xu, L. Gao, Z. Sun, Q. Liu, J. Dwivedi-Yu, Y. Yang, J. Callan, and G. Neubig. 2023. Active Retrieval Augmented Generation. In *Proc. of EMNLP*.

[11] D. Kim, B. Kim, D. Han, and M. Eibich. 2024. AutoRAG: Automated Framework for optimization of Retrieval Augmented Generation Pipeline. *ArXiv preprint* (2024).

[12] J. Kim, J. Nam, S. Mo, J. Park, S.-W. Lee, M. Seo, J.-W. Ha, and J. Shin. 2024. SuRe: Summarizing Retrievals using Answer Candidates for Open-domain QA of LLMs. In *Proc. of ICLR*.

[13] W. Kwon, Z. Li, S. Zhuang, Y. Sheng, L. Zheng, C. H. Yu, J. Gonzalez, H. Zhang, and I. Stoica. 2023. Efficient Memory Management for Large Language Model Serving with PagedAttention. In *Proc. of SOSP*.

[14] Y. Li. 2023. Unlocking Context Constraints of LLMs: Enhancing Context Efficiency of LLMs with Self-Information-Based Content Filtering. *ArXiv preprint* (2023).

[15] J. Lin, X. Ma, S.-C. Lin, J.-H. Yang, R. Pradeep, and R. F. Nogueira. 2021. Pyserini: A Python Toolkit for Reproducible Information Retrieval Research with Sparse and Dense Representations. In *Proc. of SIGIR*.

[16] T. Nguyen, M. Rosenberg, X. Song, J. Gao, S. Tiwary, R. Majumder, and L. Deng. 2016. MS MARCO: A Human Generated MAchine Reading COmprehension Dataset. In *Proc. of NeurIPS Workshop), Barcelona, Spain, December 9, 2016 (CEUR Workshop Proceedings)*.

[17] M. Pietsch, T. Möller, B. Kostic, J. Risch, M. Pippi, M. Jobanputra, S. Zanzottera, S. Cerza, V. Blagojevic, T. Stadelmann, T. Soni, and S. Lee. 2019. Haystack: the end-to-end NLP framework for pragmatic builders.

[18] A. Radford, J. Wook Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, G. Krueger, and I. Sutskever. 2021. Learning Transferable Visual Models From Natural Language Supervision. In *Proc. of ICML (Proceedings of Machine Learning Research)*.

[19] S. E. Robertson and H. Zaragoza. 2009. The Probabilistic Relevance Framework: BM25 and Beyond. *Found. Trends Inf. Retr.* 4 (2009).

[20] Z. Shao, Y. Gong, Y. Shen, M. Huang, N. Duan, and W. Chen. 2023. Enhancing Retrieval-Augmented Large Language Models with Iterative Retrieval-Generation Synergy. In *Proc. of EMNLP 2023 Findings*.

[21] W. Shi, S. Min, M. Yasunaga, M. Seo, R. James, M. Lewis, L. Zettlemoyer, and W. Yih. 2024. REPLUG: Retrieval-Augmented Black-Box Language Models. In *Proc. of NAACL-HLT*.

[22] H. Trivedi, N. Balasubramanian, T. Khot, and A. Sabharwal. 2023. Interleaving Retrieval with Chain-of-Thought Reasoning for Knowledge-Intensive Multi-Step Questions. In *Proc. of ACL*.

[23] Y. Wang, P. Li, M. Sun, and Y. Liu. 2023. Self-Knowledge Guided Retrieval Augmentation for Large Language Models. In *Proc. of EMNLP 2023 Findings*.

[24] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, J. Davison, S. Shleifer, P. von Platen, C. Ma, Y. Jernite, J. Plu, C. Xu, T. Le Scao, S. Gugger, M. Drame, Q. Lhoest, and A. Rush. 2020. Transformers: State-of-the-Art Natural Language Processing. In *Proc. of EMNLP*.

[25] F. Xu, W. Shi, and E. Choi. 2023. RECOMP: Improving Retrieval-Augmented LMs with Compression and Selective Augmentation. *ArXiv preprint* (2023).

[26] O. Yoran, T. Wolfson, O. Ram, and J. Berant. 2023. Making Retrieval-Augmented Language Models Robust to Irrelevant Context. *ArXiv preprint* (2023).

[27] X. Yu, Y. Lu, and Z. Yu. 2024. LocalRQA: From Generating Data to Locally Training, Testing, and Deployin. *ArXiv preprint* (2024).

[28] Z. Yu, C. Xiong, S. Yu, and Z. Liu. 2023. Augmentation-Adapted Retriever Improves Generalization of Language Models as Generic Plug-In. In *Proc. of ACL*.

[29] X. Zhang, Y. Song, Y. Wang, S. Tang, X. Li, Z. Zeng, Z. Wu, W. Ye, W. Xu, Y. Zhang, X. Dai, S. Zhang, and Q. Wen. 2024. RAGLAB: A Modular and Research-Oriented Unified Framework for Retrieval-Augmented Generation. In *Proc. of the EMNLP: Demo*, Delia Irazu Hernandez Farias, Tom Hope, and Manling Li (Eds.).

[30] L. Zheng, W.-L. Chiang, Y. Sheng, S. Zhuang, Z. Wu, Y. Zhuang, Z. Lin, Z. Li, D. Li, E. P. Xing, H. Zhang, J. E. Gonzalez, and I. Stoica. 2023. Judging LLM-as-a-Judge with MT-Bench and Chatbot Arena. In *Proc. of NeurIPS*.

[31] Y. Zhu, Z. Huang, Z. Dou, and J.-R. Wen. 2024. One Token Can Help! Learning Scalable and Pluggable Virtual Token. *ArXiv preprint* (2024).