

FlexRAG: A Flexible and Comprehensive Framework for Retrieval-Augmented Generation

Zhuocheng Zhang^{1,2}, Yang Feng^{1,2,3*}, Min Zhang⁴

¹Key Laboratory of Intelligent Information Processing,
Institute of Computing Technology, Chinese Academy of Sciences (ICT/CAS)

²University of Chinese Academy of Sciences, China

³Key Laboratory of AI Safety, Chinese Academy of Sciences

⁴Institute of Computing and Intelligence, Harbin Institute of Technology (Shenzhen), China
zhangzhuocheng20z, fengyang@ict.ac.cn minzhang@suda.edu.cn

Abstract

Retrieval-Augmented Generation (RAG) plays a pivotal role in modern large language model applications, with numerous existing frameworks offering a wide range of functionalities to facilitate the development of RAG systems. However, we have identified several persistent challenges in these frameworks, including difficulties in algorithm reproduction and sharing, lack of new techniques, and high system overhead. To address these limitations, we introduce **FlexRAG**, an open-source framework specifically designed for research and prototyping. FlexRAG supports text-based, multimodal, and network-based RAG, providing comprehensive lifecycle support alongside efficient asynchronous processing and persistent caching capabilities. By offering a robust and flexible solution, FlexRAG enables researchers to rapidly develop, deploy, and share advanced RAG systems. Our toolkit and resources are available at <https://github.com/ictnlp/FlexRAG>.

1 Introduction

With the rapid advancement of large language models (LLMs) (OpenAI et al., 2024; Dubey et al., 2024; Yang et al., 2024), they are increasingly playing a pivotal role across various domains. However, numerous application scenarios necessitate that these models maintain accurate, comprehensive, and up-to-date knowledge (Gao et al., 2024; Zhao et al., 2024). Continuously retraining LLMs to integrate new information is not only computationally expensive but also poses challenges such as catastrophic forgetting. To address these limitations, retrieval-augmented generation (RAG) has emerged as a promising solution, enabling models to dynamically retrieve relevant information from external sources, thereby enhancing their factual accuracy and adaptability.

Given the vast application potential of RAG across various domains, numerous frameworks

have emerged in recent years to facilitate rapid construction of RAG systems (Jin et al., 2024; Hoshi et al., 2023; Feng et al., 2024; Zhang et al., 2024b; Kim et al., 2024a; Yu et al., 2024b). However, a comprehensive analysis of existing frameworks reveals that these tools still fail to adequately address several core challenges in RAG research. First, due to the complexity of RAG systems, which involve multiple components and intricate environment configurations, researchers often struggle to precisely reproduce existing studies or effectively share their own work with others. Second, constructing a RAG system is inherently complex, requiring researchers to address numerous engineering challenges, which significantly diverts their focus from scientific inquiry. Furthermore, as RAG technology evolves rapidly, many researchers are exploring advanced topics such as multimodal retrieval, web-based retrieval, and document chunking. However, most existing frameworks are designed to address only a single aspect of RAG research, specifically retrieval strategies. More importantly, both the retrieval and generation components in RAG systems impose substantial computational costs, limiting the ability of resource-constrained researchers to conduct effective investigations.

To address these issues, we present FlexRAG, a novel open-source framework designed to facilitate the rapid reproduction, development, and evaluation of RAG systems. The proposed framework offers comprehensive support for diverse RAG scenarios, including text-based, multimodal, and web-accessible RAG applications, while providing end-to-end pipeline support from data preparation to system evaluation. FlexRAG enables researchers to efficiently share their work with the community and quickly develop demonstrative prototypes based on their algorithms. Notably, FlexRAG incorporates four key distinguishing features that set it apart from existing frameworks, which are as follows.

*Corresponding author.

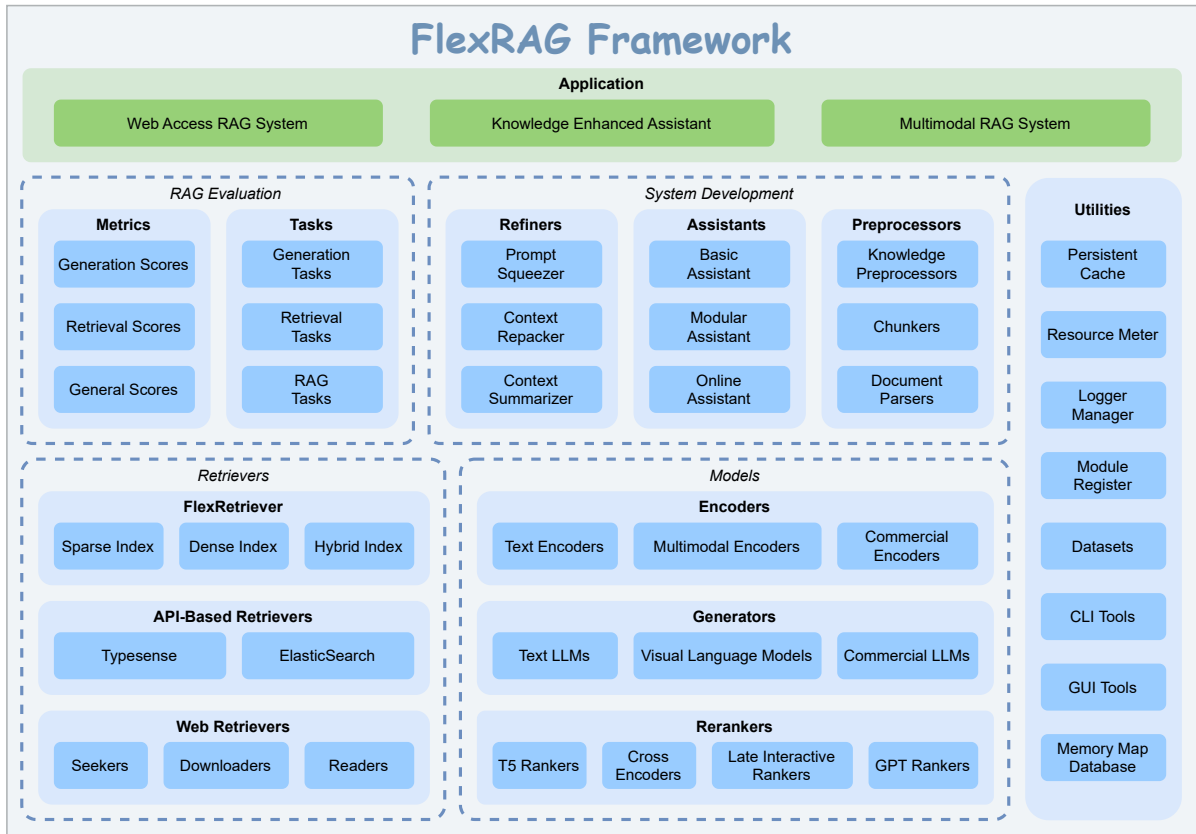


Figure 1: The architecture of FlexRAG. The light blue boxes represent modules, while the dashed boxes indicate collections of modules with relevant functions.

Research Oriented Design FlexRAG provides a unified configuration management system and a standardized RAG evaluation process to ensure fair and convenient performance assessment. By integrating with the Hugging Face Hub, FlexRAG enables researchers to share their retrievers with the community, fostering collaborative research efforts. Moreover, FlexRAG offers an example repository that facilitates algorithm comparison and reproduction, supporting rigorous scientific inquiry.

Extensive Infrastructure and Tooling To reduce the engineering burden on researchers, FlexRAG provides complete bilingual documentation and pre-built retrievers available on the Hugging Face Hub, facilitating the rapid implementation of algorithms. Additionally, FlexRAG provides a comprehensive command-line toolkit that facilitates a wide range of tasks, including data preprocessing, retriever construction, system evaluation, and the development of GUI prototypes, as illustrated in Figure 2.

Comprehensive Technical Support FlexRAG not only supports text-based RAG but also extends

to multimodal and web-based RAG, enabling broad applicability across various data types. Additionally, the framework provides end-to-end support for the entire RAG pipeline, including document parsing, chunking, and other essential processes, facilitating the development of comprehensive RAG systems.

Superior Performance FlexRAG employs a modular design and leverages asynchronous functions for computationally intensive components, facilitating the development of high-throughput RAG system prototypes. Moreover, it employs a persistent caching mechanism to further reduce retrieval overhead and enhance retrieval efficiency. Most importantly, FlexRAG incorporates advanced indexing techniques and memory map mechanism, consuming only one-tenth of the CPU and memory resources required by comparable frameworks when performing large-scale retrieval tasks.

In summary, FlexRAG is a comprehensive and flexible framework that addresses the core challenges of RAG research, providing researchers with a powerful tool to develop, evaluate, and deploy RAG systems.

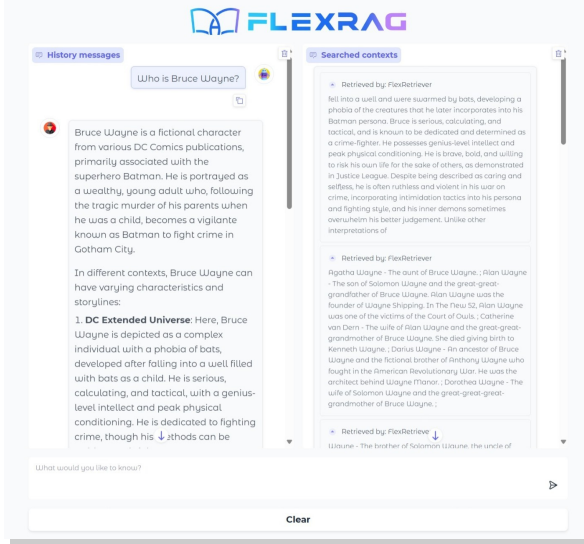


Figure 2: The GUI demonstration provided by FlexRAG. The left panel displays the messages exchanged between the user and the assistant, while the right panel shows the retrieved contexts. The GUI is designed to facilitate user interaction with the RAG system, allowing users to input queries and receive responses in a user-friendly manner.

2 The Architecture of FlexRAG

As illustrated in Figure 1, FlexRAG comprises twelve core modules, each serving a distinct function in the RAG pipeline. For clarity, we categorize them into four functional groups: models, retrievers, system development, and evaluation, along with auxiliary utility tools. This section first introduces the modules within these four categories, followed by a detailed discussion of the remaining components.

2.1 Models

In RAG systems, models are employed across various components. For instance, dense retrievers utilize encoders to transform knowledge pieces into dense vector representations, while generators are required to produce final responses. FlexRAG incorporates three fundamental model categories: encoders, generators, and rerankers.

Encoders The encoder functions to convert input queries or documents into dense vectors for similarity search in vector space. The encoders in FlexRAG can be classified into text encoders (Devlin et al., 2019; Izacard et al., 2022; Karpukhin et al., 2020; Lin et al., 2023) and multimodal encoders (Radford et al., 2021) based on input data types. Additionally, FlexRAG also supports calling

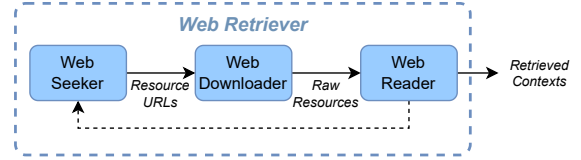


Figure 3: The core components of the *WebRetriever* module in FlexRAG and its typical workflow.

commercial encoders via API calls^{1,2,3}, and deploying encoders using famous frameworks^{4,5}.

Rerankers Rerankers optimize the initially retrieved document list through reordering mechanisms, effectively filtering out irrelevant content to reduce noise and enhance input quality for generators. FlexRAG supports various rerankers, including cross-encoder rerankers(Chen et al., 2024), late-interaction rerankers(Khattab and Zaharia, 2020; Santhanam et al., 2022; Jha et al., 2024), T5-style rerankers(Nogueira et al., 2020), and GPT-style rerankers(Sun et al., 2023). In addition, FlexRAG also supports calling online rerankers via APIs^{1,2,6,7}.

Generators The generator synthesizes natural language responses based on the retrieved documents and user queries. FlexRAG implements traditional LLMs (Yang et al., 2024; Dubey et al., 2024) and Vision Language Models (VLMs) (Wang et al., 2024b; Steiner et al., 2024) to serve as generators. Similarly, FlexRAG supports calling commercial generators via API calls^{3,8}, or deploying generators using fast inference engines^{4,9}.

2.2 Retrievers

The retriever constitutes one of the most critical components in RAG systems, serving to rapidly identify relevant information based on user queries. In FlexRAG, retrievers are classified into three types: the *Web Retriever*, which gathers information directly from the internet; the *API-Based Retriever*, which connects to external retrieval systems via APIs; and the *FlexRetriever*, developed in-house by the FlexRAG team, which stores the

¹<https://jina.ai/>

²<https://cohere.com/>

³<https://www.openai.com/>

⁴<https://ollama.com/>

⁵<https://sbnet.net/>

⁶<https://www.mixedbread.com/>

⁷<https://www.voyageai.com/>

⁸<https://www.anthropic.com/>

⁹<https://github.com/vllm-project/vllm>

knowledge base locally and builds indexes using sparse, dense, or hybrid techniques.

2.2.1 Web Retrievers

Web retrievers are designed to retrieve information from the internet, typically through search engines or walking through web pages. With internet access, web retrievers has significant advantages in both the timeliness of retrieval and the breadth of information it can access, making them particularly suitable for building personal assistants.

As shown in Figure 3, FlexRAG designs three key roles to support the construction of web retrievers. The *Web Seeker* is responsible for locating online resources. It can be implemented as either a search engine interface or a web crawler. The *Web Downloader* handles the downloading of web resources. Since web resources are typically in HTML format, which is challenging for LLMs to process directly, the *Web Reader* is designed to extract content from raw web pages.

To further streamline the development process of RAG systems, FlexRAG provides two built-in web retrievers: the *SimpleWebRetriever*, which leverages search engines to locate web pages and employs a Web Reader to convert them into an LLM-friendly format, and the *WikipediaRetriever*, specifically designed for direct entity querying from Wikipedia knowledge bases.

2.2.2 FlexRetriever

FlexRetriever is a versatile retriever that supports both MultiField and MultiIndex retrieval paradigms. It enables documents to be decomposed into multiple semantic fields, such as title, abstract, and content, with dedicated indexes constructed for each field. Moreover, FlexRetriever facilitates hybrid retrieval across multiple indexes, allowing for flexible and fine-grained retrieval strategies that can be tailored to address complex information needs. The system supports both sparse and dense retrieval approaches (Lù, 2024; Douze et al., 2025; Guo et al., 2020), making it applicable to a wide spectrum of retrieval tasks.

Notably, FlexRetriever employs memory map and the empirical formula (Aumüller et al., 2018) designed for Inverted File and Product Quantization (IVFPQ) indexing techniques as its default configuration, achieving significantly lower memory overhead and superior retrieval efficiency compared to alternative frameworks.

Furthermore, FlexRetriever is fully integrated

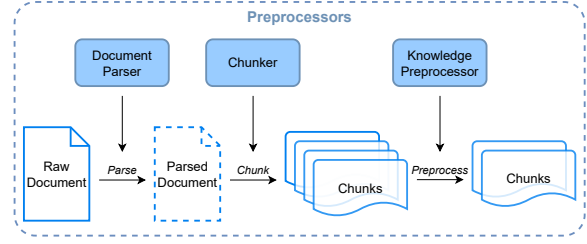


Figure 4: Architecture of the *Preprocessors* module in FlexRAG and its typical workflow.

with the Hugging Face ecosystem, enabling seamless publication, sharing, and reuse of retrievers via the Hugging Face Hub¹⁰. This integration promotes community collaboration and lowers the barrier to leveraging and contributing retrieval pipelines with minimal configuration overhead.

2.2.3 API-Based Retriever

FlexRAG also supports two API-Based Retrievers, namely TypesenseRetriever¹¹, and ElasticSearchRetriever¹², enabling users to implement their RAG systems by leveraging mature and feature-rich retrieval systems.

2.3 System Development

Beyond the two fundamental modules of a RAG system, namely the retriever and the model, additional components are essential for constructing a complete RAG pipeline. To address this requirement, FlexRAG introduces three modules that collectively enhance the pipeline’s functionality. The **Preprocessors** module is responsible for preparing and structuring the knowledge base, ensuring that relevant information is efficiently organized for retrieval. The **Refiners** module enhances the retrieved contexts through refinement and post-processing, improving the quality and relevance of the input provided to the model. Lastly, the **Assistants** module serves as a unified framework that encapsulates the entire RAG pipeline, facilitating seamless integration and operation.

2.3.1 Preprocessors

In modern computing systems, a substantial proportion of knowledge resources are stored and disseminated through document file formats (e.g., PDF, DOCX), as opposed to plain text. While these semi-structured data maintains human interpretability, it present significant parsing challenges for LLMs

¹⁰<https://huggingface.co/FlexRAG>

¹¹<https://github.com/typesense/typesense>

¹²<https://github.com/elastic/elasticsearch>

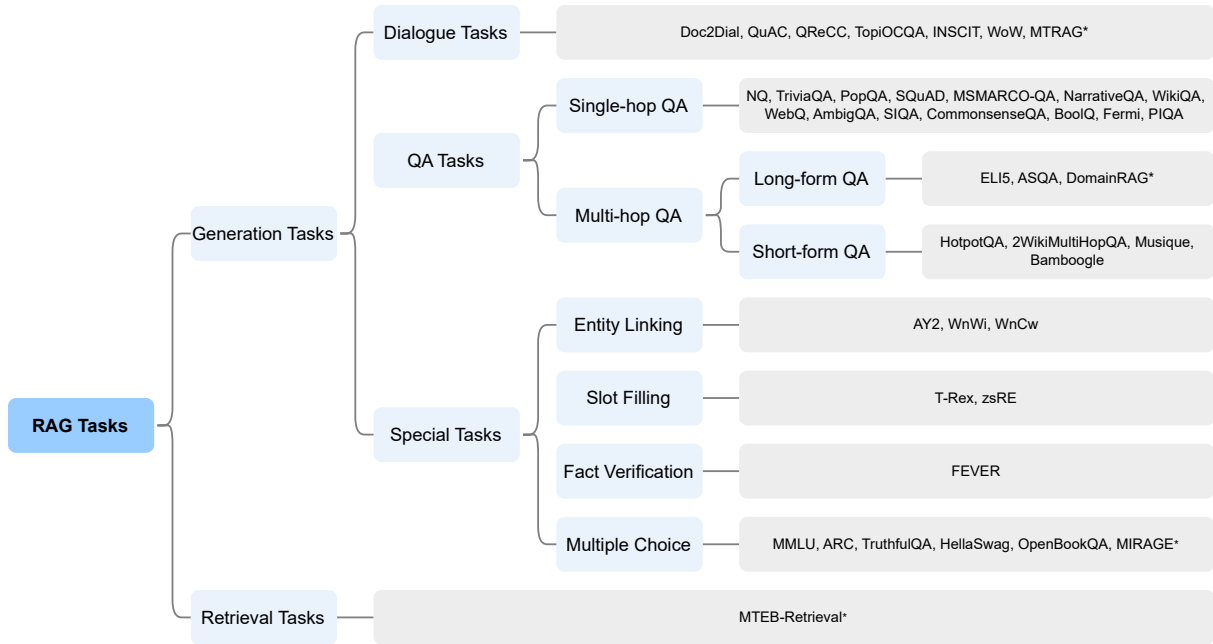


Figure 5: RAG Evaluation Tasks. Tasks without an asterisk (*) correspond to individual datasets, while those marked with an asterisk indicate benchmarks that may comprise multiple datasets.

during information extraction. To address this limitation, document preprocessing pipelines are required to transform these heterogeneous formats into standardized structured representations that are computationally tractable for LLM processing. As illustrated in Figure 4, FlexRAG’s preprocessing module comprises three specialized roles, namely *Document Parser*, *Chunker*, and *Knowledge Preprocessor*, to facilitate this critical format conversion.

Concretely, the *Document Parser* is responsible for extracting LLM readable content from various document formats, including PDF, DOCX, and HTML. Once the content is extracted, the *Chunker* segments it into smaller, more manageable chunks, enabling efficient processing by both the retriever and the generator. To further improve knowledge base quality, the *Knowledge Preprocessor* is designed to preprocess and filter the extracted content, ensuring that it is well-structured and optimized for retrieval.

2.3.2 Refiners

Existing research indicates that the quality of responses generated by LLMs is closely associated with the relevance, sequencing, and quantity of contextual information provided in the prompt (Shi et al., 2023; Zhang et al., 2024a). However, relying solely on retrievers does not guarantee that the retrieved context aligns with the preferences

of LLMs. Therefore, further processing of the retrieved context is a critical step in constructing a high-performance RAG system. To address this issue, FlexRAG incorporates three specialized sub-modules: *Prompt Squeezer*, *Context Repacker*, and *Context Summarizer*.

Concretely, the *Prompt Squeezer* is designed to optimize the prompt provided to the generator, ensuring that it is concise and relevant to the user query (Jiang et al., 2023; Pan et al., 2024; Jiang et al., 2024). To prevent critical information from being overlooked by the LLM, the *Context Repacker* reorganizes the retrieved context for better coherence. Additionally, the *Context Summarizer* enhances the quality of the retrieved context by condensing it into a more concise and informative format (Xu et al., 2023; Kim et al., 2024b; Li et al., 2023), thereby decreasing the inference overhead.

2.3.3 Assistants

In FlexRAG, the RAG assistant encapsulates the entire RAG process. This encapsulation standardizes the interaction between the RAG pipeline and the user, while also streamlining the evaluation of the pipeline. Specifically, the RAG assistant should provide a chat interface that accepts user input, generates appropriate responses, and returns both the retrieved results and generated responses, along with relevant metadata.

Methods	PopQA(%)			NQ(%)			TriviaQA(%)			Average(%)		
	F1	EM	Succ	F1	EM	Succ	F1	EM	Succ	F1	EM	Succ
BM25s(Lù, 2024)	57.88	52.75	68.48	38.79	30.00	54.74	65.93	58.02	61.98	54.20	46.92	61.73
Contriever*(Izacard et al., 2022)	64.14	59.04	80.77	49.67	39.03	75.65	70.36	62.55	68.26	61.39	53.54	74.89
E5 base(Wang et al., 2024a)	59.74	54.25	77.20	50.05	39.56	78.84	71.66	63.79	70.63	60.48	52.53	75.56
BGE M3(Chen et al., 2024)	63.65	58.76	83.42	50.98	40.36	80.00	71.92	63.85	71.10	62.18	54.32	78.17
FLAT	63.65	58.40	82.20	49.20	39.11	77.95	70.61	62.70	80.03	61.15	53.40	80.06
Faiss*(Douze et al., 2025)	64.14	59.04	81.42	49.62	39.11	77.87	70.48	62.57	79.80	61.41	53.57	79.70
ScaNN(Guo et al., 2020)	63.26	58.11	82.13	49.31	39.25	77.76	70.50	62.64	79.93	61.02	53.33	79.94
BGE-reranker-M3(Chen et al., 2024)	66.02	60.76	86.92	50.94	40.53	81.91	74.58	66.71	84.81	63.85	56.00	84.55
colbert-v2(Santhanam et al., 2022)	65.44	60.47	83.56	47.18	37.06	77.53	72.13	64.24	81.47	61.58	53.92	80.85
InRanker-base(Laitz et al., 2024)	66.05	60.90	86.63	48.77	38.50	79.78	73.38	65.47	83.20	62.73	54.96	83.20
rankGPT(Sun et al., 2023)	63.11	58.26	77.91	49.50	39.06	75.90	70.13	62.31	79.11	60.91	53.21	77.64
Qwen2-7B*(Yang et al., 2024)	64.14	59.04	81.42	49.62	39.11	77.87	70.48	62.57	79.80	61.41	53.57	79.70
Llama3.1-8B(Dubey et al., 2024)	63.20	55.83	81.42	47.58	35.73	77.87	71.75	62.97	79.80	60.84	51.51	79.70
ChatQA2-7B(Xu et al., 2024)	60.36	53.82	81.42	49.84	39.09	77.87	71.84	62.67	79.80	60.68	51.86	79.70

Table 1: The experimental results of the *ModularAssistant* on three widely used RAG tasks. The experiment was divided into four groups, each investigating the impact of modifying the retriever, index, re-ranker, and generator on the overall RAG system. Items marked with an asterisk in the table indicate the default configuration for this experiment. We did not use rerankers except in the experiments investigating the differences between them.

Furthermore, FlexRAG also incorporates several built-in RAG assistants:

- *ModularAssistant*: A modular assistant that can be arbitrarily configured through configuration files.
- *OnlineAssistant*: An assistant that leverages online API calls to access RAG services provided by commercial companies.

2.4 Evaluation

Tasks Given the sustained scholarly interest in RAG, researchers have proposed a variety of tasks to assess RAG systems and their individual components. After a comprehensive review of existing evaluation benchmarks (Yu et al., 2024a; Petroni et al., 2021; Jin et al., 2024; Katsis et al., 2025; Muennighoff et al., 2023), we found that these tasks can be categorized into multi-turn dialogue tasks, single-turn question-answering tasks, specialized tasks, and retrieval tasks. As shown in Figure 5, these tasks can be further classified into two categories: generative tasks and retrieval-based tasks. Accordingly, we provide two command-line tools in FlexRAG to evaluate these two types of tasks. To ensure a fairer evaluation process, we have developed pre-configured retrievers for the widely used Wikipedia knowledge base. These retrievers have been made publicly available on the Hugging Face Hub, providing researchers with a convenient and standardized resource for their evaluations.

Metrics FlexRAG supports a variety of evaluation metrics for assessing the performance of RAG systems. These metrics can be broadly categorized into two types: retrieval metrics, and generation metrics. To ensure the accuracy and reliability of the evaluation results, we employed the widely adopted `pytrec_eval`¹³, `sacreBLEU`¹⁴, and `Rouge`¹⁵ for metric computation. Additionally, FlexRAG also supports several LLM-as-a-Judge metrics for evaluating the quality of generated responses.

3 Empirical Study

To demonstrate the advantages of FlexRAG in research and prototype development, we conducted several experiments on *ModularAssistant*, a highly flexible RAG pipeline within FlexRAG. We evaluated the performance of the pipeline on three widely used RAG tasks: *Natural Questions*(Kwiatkowski et al., 2019), *TriviaQA*(Joshi et al., 2017), and *PopQA*(Mallen et al., 2023). We employed the Wikipedia knowledge base provided by Karpukhin et al. (2020). In the experiment, we fixed the other components of the *ModularAssistant* and independently varied its retriever, indexer, re-ranker, and generator to demonstrate the roles played by each component in the RAG task. Additionally, the number of contexts fed into the generator is fixed at 10. When the reranker is employed, we retrieve 100 contexts from the retriever and em-

¹³https://github.com/cvangysel/pytrec_eval

¹⁴<https://github.com/mjpost/sacrebleu>

¹⁵<https://github.com/pltrdy/rouge>

ploy the reranker to select the top 10 most relevant ones. We employed the F1 and Exact Match (EM) scores to evaluate the generation quality, and the Success Rate (Succ) to evaluate the retrieval quality.

As shown in Table 1, the results demonstrate that the choice of retriever, indexer, re-ranker, and generator significantly impacts the overall performance of the RAG system. For more detailed information about the experiments and the experimental findings, please visit our benchmark pages¹⁶.

4 Resource Overhead Analysis

To further validate the advantages of FlexRAG in terms of system resource efficiency, we evaluated its dense retrieval performance on the *MS_MARCO Passage Retrieval* (Bajaj et al., 2016) task using a server equipped with 256 GB of RAM, two Intel Xeon Silver 4214R CPUs, and eight GeForce RTX 3090 GPUs. FlashRAG, whose architecture is most similar to that of FlexRAG, was selected as the baseline for comparison. All experiments were conducted under default parameter settings. This evaluation primarily focuses on the following four system resource metrics:

- **Average Wall-Clock Time:** The average time taken to complete a single retrieval operation. This metric is crucial for assessing the actual latency experienced by users during the retrieval process.
- **Total CPU Time:** The total CPU time consumed during the retrieval process. This metric provides insight into the computational efficiency of the retrieval operation.
- **Average Memory Usage:** The average memory usage during the retrieval process. This metric reflects the memory efficiency of the retrieval operation, which is particularly important for large-scale retrieval tasks.
- **Total Memory Usage:** The total memory usage during the retrieval process.

As shown in Figure 6, under varying batch sizes, FlexRAG consistently exhibits significantly lower overhead compared to FlashRAG in both average wall-clock time and total CPU time, with performance gaps reaching up to an order of magnitude.

¹⁶https://github.com/ictnlp/FlexRAG/blob/master/benchmarks/singlehop_qa.md

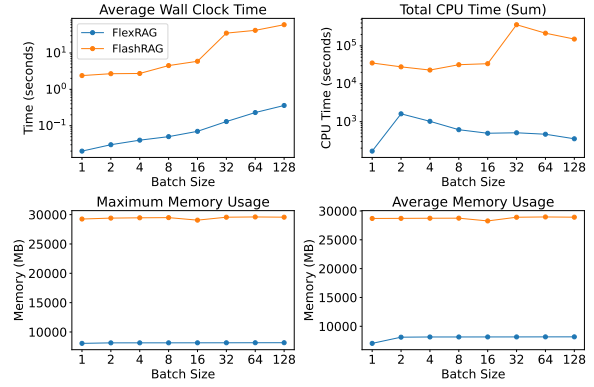


Figure 6: The resource overhead of FlexRAG and FlashRAG under different batch sizes.

In terms of memory consumption, FlexRAG also demonstrates substantially lower average and peak memory usage, outperforming the baseline by several times. These results highlight the tangible performance benefits achieved through the incorporation of a memory-mapping mechanism into the system architecture, as well as the optimization of dense index parameters using the ANN-Benchmark toolkit.

Additionally, we observe a general trend wherein system latency increases with larger batch sizes, while total CPU overhead tends to decrease. A particularly noteworthy case arises when the batch size is set to 1: under this configuration, FlexRAG achieves the lowest computational overhead across all settings. Further investigation reveals that this outcome stems from the Tokenizer component operating in a single-process mode, thereby avoiding the additional overhead associated with inter-process scheduling.

5 Conclusion

In this paper, we introduce FlexRAG, an open-source framework designed to facilitate the rapid reproduction, development, and evaluation of RAG systems. In general, FlexRAG significantly reduces the barrier to building RAG systems, streamlines collaboration, and enables a seamless transition from research to prototyping with an integrated pipeline design.

References

- Martin Aumüller, Erik Bernhardsson, and Alexander Faithfull. 2018. *ANN-Benchmarks: A Benchmarking Tool for Approximate Nearest Neighbor Algorithms*. *arXiv preprint*. ArXiv:1807.05614 [cs].

- Payal Bajaj, Daniel Campos, Nick Craswell, Li Deng, Jianfeng Gao, Xiaodong Liu, Rangan Majumder, Andrew McNamara, Bhaskar Mitra, Tri Nguyen, and 1 others. 2016. Ms marco: A human generated machine reading comprehension dataset. *arXiv preprint arXiv:1611.09268*.
- Jianlv Chen, Shitao Xiao, Peitian Zhang, Kun Luo, Defu Lian, and Zheng Liu. 2024. [BGE M3-Embedding: Multi-Lingual, Multi-Functionality, Multi-Granularity Text Embeddings Through Self-Knowledge Distillation](#). *arXiv preprint*. ArXiv:2402.03216 [cs].
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. [BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding](#). *arXiv preprint*. ArXiv:1810.04805 [cs].
- Matthijs Douze, Alexandr Guzhva, Chengqi Deng, Jeff Johnson, Gergely Szilvasy, Pierre-Emmanuel Mazaré, Maria Lomeli, Lucas Hosseini, and Hervé Jégou. 2025. [The Faiss library](#). *arXiv preprint*. ArXiv:2401.08281 [cs].
- Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, and 514 others. 2024. [The Llama 3 Herd of Models](#). *arXiv preprint*. ArXiv:2407.21783 [cs].
- Zhangchi Feng, Dongdong Kuang, Zhongyuan Wang, Zhijie Nie, Yaowei Zheng, and Richong Zhang. 2024. [EasyRAG: Efficient Retrieval-Augmented Generation Framework for Automated Network Operations](#). *arXiv preprint*. ArXiv:2410.10315.
- Jia Fu, Xiaoting Qin, Fangkai Yang, Lu Wang, Jue Zhang, Qingwei Lin, Yubo Chen, Dongmei Zhang, Saravan Rajmohan, and Qi Zhang. 2024. [AutoRAG-HP: Automatic Online Hyper-Parameter Tuning for Retrieval-Augmented Generation](#). *arXiv preprint*. ArXiv:2406.19251.
- Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, Qianyu Guo, Meng Wang, and Haofen Wang. 2024. [Retrieval-Augmented Generation for Large Language Models: A Survey](#). *arXiv preprint*. ArXiv:2312.10997 [cs].
- Ruiqi Guo, Philip Sun, Erik Lindgren, Quan Geng, David Simcha, Felix Chern, and Sanjiv Kumar. 2020. [Accelerating Large-Scale Inference with Anisotropic Vector Quantization](#). *arXiv preprint*. ArXiv:1908.10396 [cs].
- Yasuto Hoshi, Daisuke Miyashita, Youyang Ng, Kento Tatsuno, Yasuhiro Morioka, Osamu Torii, and Jun Deguchi. 2023. [RaLLe: A Framework for Developing and Evaluating Retrieval-Augmented Large Language Models](#). *arXiv preprint*. ArXiv:2308.10633.
- Gautier Izacard, Mathilde Caron, Lucas Hosseini, Sebastian Riedel, Piotr Bojanowski, Armand Joulin, and Edouard Grave. 2022. [Unsupervised Dense Information Retrieval with Contrastive Learning](#). *arXiv preprint*. ArXiv:2112.09118 [cs].
- Rohan Jha, Bo Wang, Michael Günther, Georgios Mastrotras, Saba Sturua, Isabelle Mohr, Andreas Koukounas, Mohammad Kalim Akram, Nan Wang, and Han Xiao. 2024. [Jina-ColBERT-v2: A General-Purpose Multilingual Late Interaction Retriever](#). *arXiv preprint*. ArXiv:2408.16672 [cs].
- Huiqiang Jiang, Qianhui Wu, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. 2023. [LLMLingua: Compressing Prompts for Accelerated Inference of Large Language Models](#). In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 13358–13376, Singapore. Association for Computational Linguistics.
- Huiqiang Jiang, Qianhui Wu, Xufang Luo, Dongsheng Li, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. 2024. [LongLLMLingua: Accelerating and Enhancing LLMs in Long Context Scenarios via Prompt Compression](#). In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1658–1677, Bangkok, Thailand. Association for Computational Linguistics.
- Jiajie Jin, Yutao Zhu, Xinyu Yang, Chenghao Zhang, and Zhicheng Dou. 2024. [FlashRAG: A Modular Toolkit for Efficient Retrieval-Augmented Generation Research](#). *arXiv preprint*. ArXiv:2405.13576 [cs] version: 1.
- Mandar Joshi, Eunsol Choi, Daniel Weld, and Luke Zettlemoyer. 2017. [TriviaQA: A Large Scale Distantly Supervised Challenge Dataset for Reading Comprehension](#). In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1601–1611, Vancouver, Canada. Association for Computational Linguistics.
- Vladimir Karpukhin, Barlas Oğuz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. [Dense Passage Retrieval for Open-Domain Question Answering](#). *arXiv preprint*. ArXiv:2004.04906 [cs].
- Yannis Katsis, Sara Rosenthal, Kshitij Fadnis, Chulaka Gunasekara, Young-Suk Lee, Lucian Popa, Vraj Shah, Huaiyu Zhu, Danish Contractor, and Marina Danilevsky. 2025. [MTRAG: A Multi-Turn Conversational Benchmark for Evaluating Retrieval-Augmented Generation Systems](#). *arXiv preprint*. ArXiv:2501.03468 [cs].
- Omar Khattab and Matei Zaharia. 2020. [ColBERT: Efficient and Effective Passage Search via Contextualized Late Interaction over BERT](#). *arXiv preprint*. ArXiv:2004.12832 [cs].

- Dongkyu Kim, Byoungwook Kim, Donggeon Han, and Matouš Eibich. 2024a. [AutoRAG: Automated Framework for optimization of Retrieval Augmented Generation Pipeline](#). *arXiv preprint*. ArXiv:2410.20878.
- Jaehyung Kim, Jaehyun Nam, Sangwoo Mo, Jongjin Park, Sang-Woo Lee, Minjoon Seo, Jung-Woo Ha, and Jinwoo Shin. 2024b. [SuRe: Summarizing Retrievals using Answer Candidates for Open-domain QA of LLMs](#). *arXiv preprint*. ArXiv:2404.13081 [cs].
- Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. 2019. [Natural questions: A benchmark for question answering research](#). *Transactions of the Association for Computational Linguistics*, 7:452–466.
- Thiago Laitz, Konstantinos Papakostas, Roberto Lotufo, and Rodrigo Nogueira. 2024. [InRanker: Distilled Rankers for Zero-shot Information Retrieval](#). *arXiv preprint*. ArXiv:2401.06910 [cs].
- Yucheng Li, Bo Dong, Chenghua Lin, and Frank Guerin. 2023. [Compressing Context to Enhance Inference Efficiency of Large Language Models](#). *arXiv preprint*. ArXiv:2310.06201 [cs].
- Sheng-Chieh Lin, Akari Asai, Minghan Li, Barlas Oguz, Jimmy Lin, Yashar Mehdad, Wen-tau Yih, and Xilun Chen. 2023. [How to Train Your DRAGON: Diverse Augmentation Towards Generalizable Dense Retrieval](#). *arXiv preprint*. ArXiv:2302.07452 [cs].
- Xing Han Lù. 2024. [BM25S: Orders of magnitude faster lexical search via eager sparse scoring](#). *arXiv preprint*. ArXiv:2407.03618 [cs].
- Alex Mallen, Akari Asai, Victor Zhong, Rajarshi Das, Daniel Khashabi, and Hannaneh Hajishirzi. 2023. [When Not to Trust Language Models: Investigating Effectiveness of Parametric and Non-Parametric Memories](#). *arXiv preprint*. ArXiv:2212.10511 [cs].
- Niklas Muennighoff, Nouamane Tazi, Loïc Magne, and Nils Reimers. 2023. [MTEB: Massive Text Embedding Benchmark](#). *arXiv preprint*. ArXiv:2210.07316 [cs].
- Rodrigo Nogueira, Zhiying Jiang, and Jimmy Lin. 2020. [Document Ranking with a Pretrained Sequence-to-Sequence Model](#). *arXiv preprint*. ArXiv:2003.06713 [cs].
- OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haiming Bao, Mohammad Bavarian, Jeff Belgum, and 262 others. 2024. [GPT-4 Technical Report](#). *arXiv preprint*. ArXiv:2303.08774 [cs].
- Zhuoshi Pan, Qianhui Wu, Huiqiang Jiang, Menglin Xia, Xufang Luo, Jue Zhang, Qingwei Lin, Victor Rühle, Yuqing Yang, Chin-Yew Lin, H. Vicky Zhao, Lili Qiu, and Dongmei Zhang. 2024. [LLMLingua-2: Data Distillation for Efficient and Faithful Task-Agnostic Prompt Compression](#). In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 963–981, Bangkok, Thailand. Association for Computational Linguistics.
- Fabio Petroni, Aleksandra Piktus, Angela Fan, Patrick Lewis, Majid Yazdani, Nicola De Cao, James Thorne, Yacine Jernite, Vladimir Karpukhin, Jean Maillard, Vassilis Plachouras, Tim Rocktäschel, and Sebastian Riedel. 2021. [KILT: a Benchmark for Knowledge Intensive Language Tasks](#). *arXiv preprint*. ArXiv:2009.02252 [cs].
- Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. 2021. [Learning Transferable Visual Models From Natural Language Supervision](#). *arXiv preprint*. ArXiv:2103.00020 [cs].
- Keshav Santhanam, Omar Khattab, Jon Saad-Falcon, Christopher Potts, and Matei Zaharia. 2022. [ColBERTv2: Effective and Efficient Retrieval via Lightweight Late Interaction](#). *arXiv preprint*. ArXiv:2112.01488 [cs].
- Freda Shi, Xinyun Chen, Kanishka Misra, Nathan Scales, David Dohan, Ed Chi, Nathanael Schärli, and Denny Zhou. 2023. [Large Language Models Can Be Easily Distracted by Irrelevant Context](#). *arXiv preprint*. ArXiv:2302.00093 [cs].
- Andreas Steiner, André Susano Pinto, Michael Tschanen, Daniel Keysers, Xiao Wang, Yonatan Bitton, Alexey Gritsenko, Matthias Minderer, Anthony Sherbondy, Shangbang Long, Siyang Qin, Reeve Ingle, Emanuele Bugliarello, Sahar Kazemzadeh, Thomas Mesnard, Ibrahim Alabdulmohsin, Lucas Beyer, and Xiaohua Zhai. 2024. [PaliGemma 2: A Family of Versatile VLMs for Transfer](#). *arXiv preprint*. ArXiv:2412.03555 [cs].
- Weiwei Sun, Lingyong Yan, Xinyu Ma, Shuaiqiang Wang, Pengjie Ren, Zhumin Chen, Dawei Yin, and Zhaochun Ren. 2023. [Is ChatGPT Good at Search? Investigating Large Language Models as Re-Ranking Agents](#). *arXiv preprint*. ArXiv:2304.09542 [cs].
- Liang Wang, Nan Yang, Xiaolong Huang, Binxing Jiao, Linjun Yang, Daxin Jiang, Rangan Majumder, and Furu Wei. 2024a. [Text Embeddings by Weakly-Supervised Contrastive Pre-training](#). *arXiv preprint*. ArXiv:2212.03533 [cs].
- Peng Wang, Shuai Bai, Sinan Tan, Shijie Wang, Zhihao Fan, Jinze Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Yang Fan, Kai Dang, Mengfei Du, Xuancheng Ren, Rui Men, Dayiheng Liu, Chang Zhou, Jingren Zhou, and Junyang Lin. 2024b.

Qwen2-VL: Enhancing Vision-Language Model’s Perception of the World at Any Resolution. *arXiv preprint*. ArXiv:2409.12191 [cs].

Fangyuan Xu, Weijia Shi, and Eunsol Choi. 2023. **RECOMP: Improving Retrieval-Augmented LMs with Compression and Selective Augmentation**. *arXiv preprint*. ArXiv:2310.04408 [cs].

Peng Xu, Wei Ping, Xianchao Wu, Chejian Xu, Zihan Liu, Mohammad Shoeybi, and Bryan Catanzaro. 2024. **ChatQA 2: Bridging the Gap to Proprietary LLMs in Long Context and RAG Capabilities**. *arXiv preprint*. ArXiv:2407.14482 [cs].

An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, Haoran Wei, Huan Lin, Jialong Tang, Jialin Wang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Ma, and 43 others. 2024. **Qwen2 Technical Report**. *arXiv preprint*. ArXiv:2407.10671 [cs].

Hao Yu, Aoran Gan, Kai Zhang, Shiwei Tong, Qi Liu, and Zhaofeng Liu. 2024a. Evaluation of retrieval-augmented generation: A survey. In *CCF Conference on Big Data*, pages 102–120. Springer.

Xiao Yu, Yunan Lu, and Zhou Yu. 2024b. **LocalRQA: From Generating Data to Locally Training, Testing, and Deploying Retrieval-Augmented QA Systems**. *arXiv preprint*. ArXiv:2403.00982.

Taolin Zhang, Dongyang Li, Qizhou Chen, Chengyu Wang, Longtao Huang, Hui Xue, Xiaofeng He, and Jun Huang. 2024a. **R4: Reinforced Retriever-Reorder-Responder for Retrieval-Augmented Large Language Models**. *arXiv preprint*. ArXiv:2405.02659 [cs].

Xuanwang Zhang, Yunze Song, Yidong Wang, Shuyun Tang, Xinfeng Li, Zhengran Zeng, Zhen Wu, Wei Ye, Wenyuan Xu, Yue Zhang, Xinyu Dai, Shikun Zhang, and Qingsong Wen. 2024b. **RAGLAB: A Modular and Research-Oriented Unified Framework for Retrieval-Augmented Generation**. *arXiv preprint*. ArXiv:2408.11381 [cs].

Siyun Zhao, Yuqing Yang, Zilong Wang, Zhiyuan He, Luna K. Qiu, and Lili Qiu. 2024. **Retrieval Augmented Generation (RAG) and Beyond: A Comprehensive Survey on How to Make your LLMs use External Data More Wisely**. *arXiv preprint*. ArXiv:2409.14924 [cs].

A Comparison with Existing RAG Frameworks

To further illustrate the uniqueness of FlexRAG, we conducted a comparative analysis of a wide range of related works. The results of this comparison are summarized in Table 2. As a heavyweight framework, LangChain and LlamaIndex offer the most comprehensive functionalities. However, the research-oriented design brings FlexRAG distinct advantages in algorithm reproducibility and knowledge sharing. At the same time, its lightweight architecture ensures a smoother learning curve, making it more accessible to researchers and developers alike.

Among lightweight frameworks, FlashRAG has made notable contributions to the reproducibility of existing researches. Beyond this, FlexRAG offers a more extensive set of fundamental components, supports web access, integrates seamlessly with Hugging Face, and features a well-structured preprocessing module. UltraRAG incorporates numerous cutting-edge techniques. In contrast, the modular architecture of FlexRAG allowing researchers to efficiently extend and customize it to meet their evolving needs. Meanwhile, AutoRAG and AutoRAG-HP focus primarily on automated hyperparameter tuning, while several other frameworks in this category have been discontinued.

¹<https://www.langchain.com/>

²<https://www.llamaindex.ai/>

³<https://github.com/OpenBMB/UltraRAG>

Frameworks	Web Access	Multimodal	Preprocess	Evaluation	Training Scripts	Research Oriented	Still Maintain
LangChain ¹	✓	✓	✓	✓	✓	✗	✓
LlamaIndex ²	✓	✓	✓	✓	✓	✗	✓
FlashRAG(Jin et al., 2024)	✗	✓	✗	✓	✗	✓	✓
RAGLab(Zhang et al., 2024b)	✗	✗	✗	✓	✓	✓	✗
AutoRAG(Kim et al., 2024a)	✗	✗	✓	✓	✗	✓	✓
AutoRAG-HP(Fu et al., 2024)	✗	✗	-	✓	✗	✓	-
RaLLe(Hoshi et al., 2023)	✗	✗	✗	✓	✗	✓	✗
LocalRQA(Yu et al., 2024b)	✗	✗	✓	✓	✓	✓	✗
EasyRAG(Feng et al., 2024)	✓	✗	✗	✗	✗	✓	✗
UltraRAG ³	✗	✓	✗	✓	✓	✓	✓
FlexRAG (Ours)	✓	✓	✓	✓	✗	✓	✓

Table 2: Comparison with existing retrieval-augmented generation frameworks. We evaluate each framework based on the following criteria: (1) support for internet access, (2) multimodal RAG capabilities, (3) inclusion of preprocessing modules, (4) availability of evaluation modules, (5) provision of training scripts, (6) research-oriented design, and (7) active maintenance status (defined as having commits within the last three months). "-" indicates the framework is not currently public available.