EncouRAGe: Evaluating RAG Local, Reliable, and Efficient

Jan Strich, Martin Semmann, Chris Biemann
Hub of Computing and Data Science (HCDS)
University of Hamburg, Germany

Correspondence: {first_name}.{last_name}@uni-hamburg.de

Abstract

We introduce **EncouRAGe**, a comprehensive Python library designed to streamline the development and evaluation of Retrieval-Augmented Generation (RAG) systems using Large Language Models (LLMs) and Embedding Models. EncouRAGe comprises five modular and extensible components: *Type Manifest*, *RAG Factory*, *Inference*, *Vector Store*, and *Metrics*, facilitating flexible experimentation and extensible development. Each component helps to make development RAG evaluation and emphasizes **scientific reproducibility**, **diverse evaluation metrics**, and **local deployment**, enabling researchers to efficiently assess datasets within RAG workflows. This paper presents implementation details and an extensive evaluation across multiple benchmark datasets, including *25k QA pairs* and *over 51k documents*. Our results show that RAG still underperforms compared to the *Oracle Context*, while *Hybrid BM25* consistently achieves the best results across all four datasets.
Code: github.com/uhh-hcds/encourage

1 Introduction

Retrieval-Augmented Generation (RAG) (Lewis et al., 2020; Huang and Huang, 2024; Nikishina et al., 2025) has emerged as a dominant approach for enhancing large language models (LLMs) with external knowledge sources. By combining retrieval and generation, RAG systems can provide more factual, contextually grounded, and up-to-date responses (Shuster et al., 2021; Sahoo et al., 2024). However, the rapid development of RAG architectures (Huang and Huang, 2024; Sarthi et al., 2024), methods (Gao et al., 2021, 2023), and LLM-as-a-Judge metrics (Es et al., 2024a) has created a need for an implementation-focused RAG evaluation library to create a reliable way to compare performance on domain-specific datasets.

Assessing a RAG system for a specific dataset requires analyzing the performance of both the retriever and generator, as well as their interaction. It is particularly challenging because LLM outputs can be lengthy, complex, and ambiguous, making it difficult to define clear correctness or objectively measure factual accuracy (Ru et al., 2024). Recent RAG evaluation frameworks, such as RAGAS (Saad-Falcon et al., 2024), RAGChecker (Ru et al., 2024), and TruLens (Snowflake Inc., 2024), have begun addressing these challenges by providing libraries to jointly analyze retrieval and generation quality, reflecting the interdependence between the two components. Instead of relying solely on traditional metrics like BLEU (Papineni et al., 2002) or ROUGE (Lin, 2004), they use LLM-based or embedding-based measures to directly evaluate factual correctness, relevance, and consistency of the generated text (Es et al., 2024a). Additionally, they often provide enhanced UI and monitoring features for RAG in production environments.

Despite recent advances, a lack of the following features in state-of-the-art (SOTA) frameworks remains, hindering the effective comparison of RAG methods for domain-specific datasets:
(1) Lack of comparability of scientific methods. Existing libraries in this field focus solely on implementing metrics, with no standardized implementation of SOTA RAG methods. This heterogeneity hinders meaningful and reproducible comparisons across different RAG approaches.
(2) Limited flexibility of LLM-as-a-judge metrics. LLM-as-a-judge metrics are sensitive to domain and dataset, and must be flexible to be effective. Therefore, custom prompts for individually designing evaluations are crucial and are not currently available in existing frameworks.
(3) Primarily cloud-oriented, with limited flexibility for integrating new metrics or methods. Many existing frameworks are designed with a narrow focus on specific local setups and lack modu-

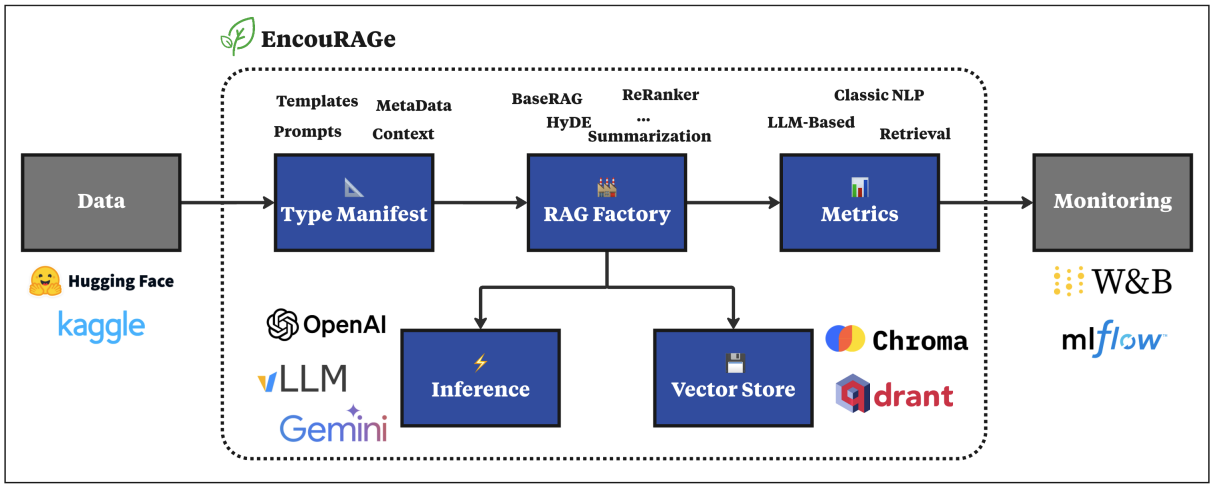

Figure 1: System overview of the **EncouRAGe** Python library. Input data in any format must be transformed to fit the **Type Manifest**. The **RAG Factory** provides various RAG methods, while the **Metrics** component implements all evaluation metrics. **Inference** can be executed locally or via cloud providers using the OpenAI SDK, and supported **Vector Store** includes Chroma and Qdrant. The output fits popular monitoring systems.

larity for broader integration. A flexible, plug-and-play architecture is essential to support new RAG methods and additional new metrics.

In order to solve these challenges, we present **EncouRAGe**, an open-source Python library for comprehensive, reproducible, and extensible RAG evaluation (Fig. 1). It can be integrated with any dataset into a parsable, object-oriented structure for effective and traceable workflows. EncouRAGe includes 10 RAG methods, manages LLM and embedding inference, and provides metrics in three categories:
1. **Generator Metrics**: Evaluating the performance end-to-end with classic NLP metrics (ROUGE, BLEU, etc.).
2. **Retrieval Metrics**: Evaluating the effectiveness of the retriever in finding relevant information from the knowledge base.
3. **LLM-as-a-Judge**: Evaluating the effectiveness of the RAG pipeline enhanced by custom prompts.

To address the rapid pace and volume of new RAG and LLM research, reproducing and comparing methods has become increasingly time-consuming. With EncouRAGe, evaluating new approaches or verifying others claims becomes straightforward and extensible. The library enables quick benchmarking across diverse methods, models, and datasets with minimal manual effort, requiring mainly GPU time rather than complex reimplementation. We demonstrate this through experiments comparing RAG methods on four QA datasets.

The main contributions of this paper are as follows:

- We present **EncouRAGe**, a novel RAG evaluation library that consists of **9** SOTA RAG methods, an object-oriented `Type Manifest`, and over 20 `Metrics` to get transparent and reliable results.
- We conduct a case study across four datasets with **25k QA pairs** and **50k documents**, demonstrating the workflow, effectiveness, and usefulness of the framework.

2 Related Work

Retrieval-Augmented Generation. LLMs excel at text generation but face challenges such as outdated knowledge and hallucinations (Shuster et al., 2021; Huang and Huang, 2024). RAG addresses these issues by incorporating external knowledge, improving accuracy and factuality (Lewis et al., 2020; Sarthi et al., 2024). This approach is critical in domains such as law (Cui et al., 2024), medicine (Xiong et al., 2024; Chen et al., 2025b), and finance (Chen et al., 2025b), where precision is crucial. RAG systems have achieved strong results in tasks such as open-domain question answering (Kim and Lee, 2024), code generation (Wang et al., 2025), and dialogue (Chen et al., 2025a). They have also been adopted in real-world applications, including LlamaIndex (Liu, 2022), where they help integrate external data sources into large language model workflows for more context-aware responses.

Evaluation of RAG. (1) Lightweight and Dataset-Based Evaluation Tools. RAGAS (Es et al., 2024a) and RAGChecker (Ru et al., 2024) provide systematic metrics for assessing RAG pipelines. RAGAS was the first library to provide LLM-as-a-Judge metrics and offers additional classic NLP evaluation metrics. In contrast, RAGChecker employs a modular approach that decomposes RAG performance into retrieval and generation components, allowing detailed error attribution and interpretability. But both lack implementation of RAG methods and are cloud-focused.

(2) Monitoring Frameworks for Production RAG Applications. TruLens (Snowflake Inc., 2024) and Opik (Comet ML, Inc., 2025) extend evaluation capabilities to deployed RAG and LLM-based applications. TruLens enables real-time tracking, custom evaluation functions, and integration with human feedback loops, making it suited for agent-oriented systems in which RAG serves as an auxiliary mechanism. Opik provides end-to-end monitoring and analytics for production environments, supporting model observability for improvement workflows. Both systems are application-focused and not designed for research.

(3) General LLM Evaluation Frameworks. DeepEval (AI, 2024) and OpenAI evals (OpenAI, 2025a) offer broader evaluation capabilities that extend beyond RAG-specific scenarios. DeepEval provides standardized testing interfaces and model-agnostic benchmarking tools for assessing reasoning, factual accuracy, and robustness.

3 EncouRAGe

As shown in Fig. 1, EncouRAGe is an end-to-end RAG evaluation library in Python consisting of five main components. To showcase our library, we define the formal idea in Subsec. 3.1. It supports any dataset containing queries, answers, and golden context triples. The data is standardized via the *Type Manifest*, as explained in Subsec. 3.2, ensuring type safety of all elements during processing. Users can select from a wide range of RAG methods defined in the *RAG Factory* (Subsec. 3.3), which utilize the *Inference Handler* and *Vector Store* described in Subsec. 3.4. For evaluation, EncouRAGe uses over 20 *Metrics* to determine whether the performance loss originates from the retriever or the generator. EncouRAGe is a fully tested codebase and supports all LLMs and embedding models available on HuggingFace.

3.1 Task Formulation

We formalize the framework underlying our library as follows. **EncouRAGe** provides a collection of RAG methods $\mathcal{R}$ and evaluation metrics $\mathcal{M}$, which can be applied to any dataset $\mathcal{D}$ that is defined as $\mathcal{D} = \{(q_i, a_i, c_i)\}_{i=1}^{N}$, where q_i denotes a question, a_i its corresponding answer, and c_i the associated gold context, which may also include additional elements not paired with a q_i and a_i.

All contexts c_i are embedded and stored in a vector store $\mathcal{V} = \text{Embed}(\{c_i\}_{i=1}^{N})$, which serves as the retrieval space for each RAG method $\mathcal{R}$. A RAG method is denoted as $\mathcal{R} = (R, G)$, where R represents the retriever and G the generator. Given a query q_i, the retriever R accesses the vector store $\mathcal{V}$ to return a set of top-k relevant contexts

$$C_k = [c_1, c_2, \ldots, c_k] = R(q_i; \mathcal{V}), \tag{1}$$

where c_1 is the most relevant context and c_k the least among the top k.

C_k is then passed to the generator G to produce a predicted answer

$$\hat{a}_i \sim P(a \mid q_i, C_k). \tag{2}$$

The quality of the predicted answer $\hat{a}_i$ is evaluated against the ground-truth a_i using a set of metrics $\mathcal{M} = \{m_1, m_2, \ldots, m_J\}$, producing scores

$$s_{i,j} = m_j(\hat{a}_i, a_i), \quad \forall i \in [1, N],\ j \in [1, J]. \tag{3}$$

Each score $s_{i,j}$ reflects the performance of the retriever, the generator, and their interaction.

3.2 Type Manifest

The *Type Manifest* is a collection of object-oriented Python structures designed to streamline evaluation, improve reliability, and enhance maintainability, as illustrated in Fig. 2. Each data point is represented as a `Document` object containing a unique identifier, textual content, and an associated `MetaData` object. The `MetaData` object is a dictionary for meta information not used in processing, enabling flexible organization and analysis. A `Context` may include multiple `Documents`, reflecting the capability of RAG methods to retrieve several documents. Each `Context` is injected into a `Jinja2` template, allowing users to define prompt structures and incorporate document content via template variables. The `Prompt` object contains an identifier, conversation history (sys and user messages), a `Context`, and a `MetaData` object.

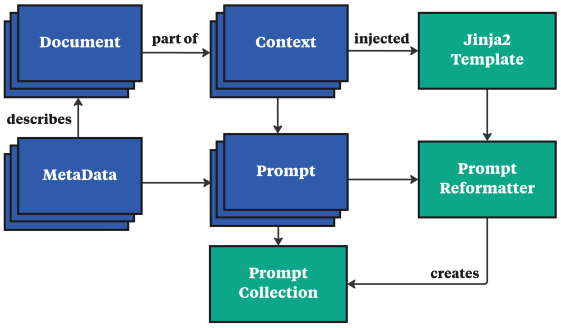

Figure 2: *Type Manifest*. Gold **Documents** link to the **Context**, combined via **Jinja2 templates** with the prompt to form the **Prompt Collection. Metadata** ensures traceability for documents and prompts.

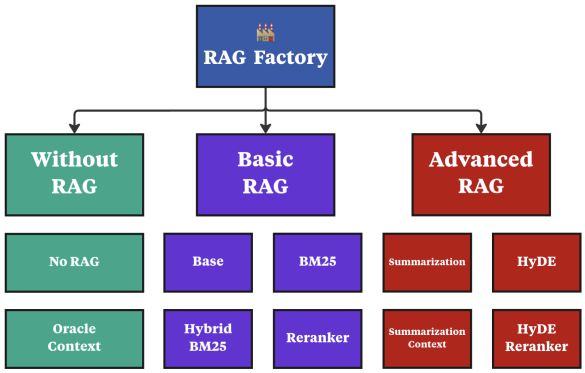

Figure 3: Overview of *RAG Factory*. RAG Factory is organized into three categories: Without RAG, Basic RAG, and Advanced. In total, EncouRAGe supports **10 methods**, with more to be added in the future.

When constructing a `PromptCollection`, the `PromptReformatter` renders the `Jinja2` template, and the related template variables. This ensures consistent usage across the evaluation pipeline, allowing seamless integration with the *RAG Factory* and reproducible metric computation.

3.3 RAG Factory

The *RAG Factory* is the core of our Python library, defining all available methods for dataset evaluation (Fig. 3). It currently provides ten methods across three categories: **Without RAG**, **Basic RAG**, and **Advanced RAG**, with a standardized interface for easy extension and a *Type Manifest* for consistent integration.

Without RAG. In the *Pretrained-Only* setup, no retriever is employed, and models must answer questions solely based on their pretraining knowledge. Conversely, the *Oracle Context* setting assumes that the relevant context is directly passed to the generator.

Basic RAG Methods. The *Base RAG* implementation follows the original RAG approach (Lewis et al., 2020), where only the question is embedded to retrieve the top-k documents, which are then passed unchanged to the generator. In addition, a *Standard BM25* (Robertson and Zaragoza, 2009) retriever is provided as a purely lexical baseline, relying on term-frequency and inverse document-frequency weighting to rank documents based on keyword overlap. *Hybrid BM25* (Gao et al., 2021) combines sparse lexical retrieval using BM25 with dense vector retrieval, leveraging both methods to improve recall and relevance. Finally, the *Reranker* method (Glass et al., 2022) can apply any cross-encoder model after initial retrieval to reorder documents in a shared embedding space.

Advanced RAG Methods. This category consists of methods that modify the query, transform retrieved contexts, or employ iterative retrieval strategies. The *HyDE* method (Gao et al., 2023) generates hypothetical answers for each question, using them as refined queries to retrieve more relevant documents. *Summarization* reduces noise by summarizing each retrieved context using an LLM, focusing on essential information. *SumContext* applies a similar summarization step, retaining the original full documents for generation, aiming to reduce distractions while preserving the content.

3.4 Inference and Vector Store

For *Inference*, we focus on local deployment, primarily leveraging `vLLM` (Kwon et al., 2023), which is one of the fastest LLM engines available. Communication with the LLM is handled through the OpenAI Python SDK (OpenAI, 2025b), allowing seamless integration of HuggingFace, OpenAI, and Gemini. Also structured output is possible through providing a pydantic model to the runner. Other integrations are planned.

For the *Vector Store*, we utilize a serverless version of Chroma (Chroma Team, 2025), which creates an in-memory `SQLite3` database stored locally. As an alternative, we also support Qdrant (Qdrant Team, 2025), which can handle large-scale document collections. Both implementations share a unified interface, simplifying the integration of additional vector database backends in the future.

3.5 Metrics

Evaluating different RAG methods is a core feature of our library, implemented through the *Metrics* module. To provide a transparent and comprehensive assessment, we include metrics for all components of RAG across multiple datasets, grouped into three categories: generator metrics, retrieval metrics, and LLM-based metrics. To align with existing research, we primarily use the `evaluate` (HuggingFace, 2025) library for most generator metrics. For retrieval metrics, we used the `ir_measures` (MacAvaney et al., 2022) library, which is specialized for information retrieval. LLM-based metrics are fully implemented and easily customizable with prompts and examples, enabling flexible use across datasets. Appendix B gives an overview of the metrics of the library.

The *Type Manifest* typing system integrates with all metrics, enabling automatic computation once a RAG method is selected. Outputs are stored for easy integration with `mlflow` (Zaharia et al., 2018) or `wandb` (Biewald, 2020).

4 Case Study

To demonstrate our library, we conduct a case study analyzing four popular datasets using each RAG method implemented in EncouRAGe. The datasets used are summarized in Table 1 and Appendix A. Each QA pair has at least one golden document, enabling comparisons of RAG methods across generation and retrieval metrics.

4.1 Implementation

Type Manifest. For each dataset, we followed the following implementation steps. Each sample was converted to a list of `user_prompts`, a collection of typed `Document` objects, where each document corresponds to a single context and has a unique identifier used for retrieval metrics, and an associated collection of `MetaData` objects.

RAG Factory. For the initialization of each method, it is required to provide the `Document` objects, the embedding function identifier from Hugging Face, a `InferenceRunner` instance, and the related prompt template to render the content. Each method also has optional parameters such as filter queries for the vector db and batch sizes for insert the documents or query them. Running each RAG method requires passing `user_prompts`, a system prompt, and a set of `retrieval_prompts` for querying the vector db. It is also possible to just get the retrieval results for debugging. This modular design allows swapping prompts, methods, and runners easily, without modifying the surrounding evaluation pipeline.

Subset	#QA	Q Tok.	#Docs	Doc Tok.
BioSQA	4,012	13.5	34,634	3,222.8
FinQA	6,237	45.1	2,110	1,080.4
FeTaQA	7,324	20.4	6,929	520.0
HotPotQA	7,395	21.4	7,395	1,421.2
Total	**24,968**	25.7	**51,068**	2,506.7

Table 1: Comparison of QA pairs, documents, and average token lengths across subsets. HotPotQA (Yang et al., 2018) and FeTaQA (Nan et al., 2022) are based on Wikipedia articles, while FinQA (Chen et al., 2021) and BioSQA (Krithara et al., 2023) use PDF documents. Avg. tokens are computed with Gemma-3 tokenizer.

Inference and Vector Store. We initialized a `BatchInferenceRunner` with standard `SamplingParams` (e.g., temperature = 0) and chose Gemma3 27B (Kamath et al., 2025) as the generator. For retrieval, we selected E5-Large Instruct (Wang et al., 2024), which has an embedding space of 1024 tokens. We used a pydantic model for providing structured output, having three fields: reasoning steps (`list[str]`), list of supporting facts (`list[str]`) and final answer (`str`) to make evaluation more transparent. All experiments were conducted on NVIDIA H100 GPUs.

Metrics. For retrieval evaluation, metrics such as *MRR@10*, *MAP*, and *Recall@10* are computed by comparing ranked document lists against the annotated gold documents. The metrics objects can be initialized with parameter such as k for MRR or Recall, making them flexible for any use case. *MRR@10* is used for tasks with a single relevant document, while *MAP* is used when multiple documents are required to answer a query. *Recall@10* complements these metrics by measuring whether relevant documents appear in the retrieved set, independent of rank.

For generation evaluation, task-specific metrics are applied depending on the answer format. Short textual answers are evaluated using *F1*, while numerical answers are evaluated using *NM* (Strich et al., 2025). Since all predictions are returned as responses, metrics can be applied consistently across datasets and RAG methods.

RAG Method	HotPotQA			FeTaQA			FinQA			BioSQA		
	F1	MRR	R@10	F1	MRR	R@10	NM	MRR	R@10	F1	MAP	R@10
+ Pretrained-Only	36.7	–	–	29.3	–	–	9.6	–	–	39.3	–	–
+ Oracle Context	43.4	100	100	49.4	100	100	72.9	100	100	54.5	100	100
+ Vanilla RAG	37.1	68.8	82.5	49.8	87.5	**92.7**	47.8	45.6	71.9	47.2	42.1	50.1
+ BM25	40.5	29.1	**98.4**	39.0	18.6	68.5	50.8	45.5	77.2	44.3	26.7	44.3
+ Hybrid BM25	**40.8**	70.4	96.9	**50.1**	**87.6**	92.4	**51.8**	46.7	**82.1**	**49.9**	43.6	**55.7**
+ Reranker	36.1	63.4	78.4	49.6	86.9	92.2	50.2	45.5	71.7	44.5	42.1	50.1
+ HyDE	30.2	40.6	61.0	48.4	82.8	89.5	41.7	37.3	61.4	49.2	**45.9**	54.7
+ Summarization	31.7	**71.0**	83.3	40.0	83.9	90.5	45.9	**48.5**	74.6	45.0	43.2	51.0
+ SumContext	37.7	**70.9**	83.3	48.6	84.0	90.5	48.6	48.1	74.6	47.8	43.0	50.4

Table 2: Overall Performance (F1, MRR/MAP, R@10) for Gemma3 with E5-Large on all four datasets. Cells in **Bold** indicate the highest value over all RAG methods, and underlined across RAG method categories.

4.2 Main Results

Without RAG Runs. The *Pretrained-Only* results for HotPotQA, FeTaQA, and BioSQA are comparatively high, indicating that these datasets are likely part of the LLM's pretraining corpus (e.g., Wikipedia-based sources). In contrast, FinQA shows a substantially lower score, suggesting that its domain-specific financial questions are less represented in the model's training data. However, when provided with *Oracle Context*, all datasets achieve higher performance than in the *Pretrained-Only*, confirming that the model can effectively reason when supplied with relevant context. Notably, the *Oracle Context* results for HotPotQA (Yang et al., 2018) and FeTaQA (Nan et al., 2022) are slightly below the original papers, as our setup relies solely on a zero-shot setting.

Base RAG. When incorporating retrieval, *Base-RAG* performs close to the *Oracle Context* on HotPotQA, FeTaQA, and BioSQA, but shows a larger performance gap on FinQA. Interestingly, the traditional *BM25* retriever performs comparably to the dense retriever in terms of F1, but exhibits substantially lower MRR on HotPotQA and BioSQA. Combining both retrieval methods (*Hybrid BM25*) yields the most consistent improvements, achieving the highest F1 and Recall@10 across all datasets. Adding a reranker does not yield measurable gains, as the MRR remains similar to base RAG when the top-10 retrieved documents are reordered.

Advanced RAG. Among the advanced retrieval enhancements, *HyDE* offers modest improvements only for BioSQA, which are insufficient to justify its additional computational cost. *Summarization*-based retrieval increases MRR and Recall@10 for all datasets except FeTaQA, but leads to lower F1 scores, likely due to information loss during summarization. The combined approach, *SumContext*, which integrates both summarized and original documents, achieves a balanced trade-off, yielding results similar to *Base-RAG* with slightly better retrieval performance. However, given its additional retrieval and generation overhead, its practical benefit remains limited.

Overall, the results demonstrate that hybrid retrieval remains the most effective RAG method across diverse QA domains. In contrast, advanced generation-based retrieval strategies offer only small improvements at a higher computational cost.

5 Conclusion

In this paper, we introduce **EncouRAGe**, a Python library for evaluating RAG methods locally, reliably, and efficiently. EncouRAGe enables researchers to systematically investigate retrieval-augmented generation techniques on their own datasets, providing insights into which configurations yield the best results for specific domains. The library includes over 10 RAG methods, more than 20 metrics, and supports models from Hugging Face, OpenAI, and Gemini. Furthermore, users can flexibly choose between Chroma and Qdrant as the vector store backend.

To demonstrate the capabilities of EncouRAGe, we conducted experiments on four widely used datasets, revealing that the optimal RAG method varies across domains. Overall, our results indicate that the *Hybrid BM25* approach delivers the best performance among the tested configurations.

With EncouRAGe, we aim to contribute to the RAG research community by providing a flexible, extensible evaluation library that enables deeper analysis and supports research and product development for RAG.

Ethical Statement

We do not foresee ethical risks in our work. The dedicated use of this library is to evaluate the performance of RAG methods on multiple metrics. We believe that the library can help identify errors in datasets and uncover biases or common mistakes when applying RAG to a dataset.

Acknowledgement

This work is supported by the Genial4KMU project at the Universität Hamburg, funded by the BMFTR (grant no. 16IS24044B).

References

Confident AI. 2024. DeepEval: The Open-Source LLM Evaluation Framework. Https://deepeval.com.

Lukas Biewald. 2020. Experiment Tracking with Weights and Biases. Https://www.wandb.com/.

Yifu Chen, Shengpeng Ji, Haoxiao Wang, Ziqing Wang, Siyu Chen, Jinzheng He, Jin Xu, and Zhou Zhao. 2025a. WavRAG: Audio-Integrated Retrieval Augmented Generation for Spoken Dialogue Models. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12505–12523, Vienna, Austria. Association for Computational Linguistics.

Zhe Chen, Yusheng Liao, Shuyang Jiang, Pingjie Wang, YiQiu Guo, Yanfeng Wang, and Yu Wang. 2025b. Towards Omni-RAG: Comprehensive Retrieval-Augmented Generation for Large Language Models in Medical Applications. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15285–15309, Vienna, Austria. Association for Computational Linguistics.

Zhiyu Chen, Wenhu Chen, Charese Smiley, Sameena Shah, Iana Borova, Dylan Langdon, Reema Moussa, Matt Beane, Ting-Hao Huang, Bryan Routledge, and William Yang Wang. 2021. FinQA: A Dataset of Numerical Reasoning over Financial Data. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3697–3711, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Chroma Team. 2025. Chroma: Open-source search and retrieval database for AI applications. Https://github.com/chroma-core/chroma.

Comet ML, Inc. 2025. Comet — The AI Developer Platform. Https://www.comet.com/site/.

Jiaxi Cui, Munan Ning, Zongjian Li, Bohua Chen, Yang Yan, Hao Li, Bin Ling, Yonghong Tian, and Li Yuan. 2024. Chatlaw: A Multi-Agent Collaborative Legal Assistant with Knowledge Graph Enhanced Mixture-of-Experts Large Language Model. *arXiv preprint*. ArXiv:2306.16092.

Shahul Es, Jithin James, Luis Espinosa Anke, and Steven Schockaert. 2024a. RAGAs: Automated Evaluation of Retrieval Augmented Generation. In *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics: System Demonstrations*, pages 150–158, St. Julians, Malta. Association for Computational Linguistics.

Shahul Es, Jithin James, Luis Espinosa Anke, and Steven Schockaert. 2024b. RAGAs: Automated evaluation of retrieval augmented generation. In *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics: System Demonstrations*, pages 150–158, St. Julians, Malta. Association for Computational Linguistics.

Luyu Gao, Zhuyun Dai, Tongfei Chen, Zhen Fan, Benjamin Van Durme, and Jamie Callan. 2021. Complementing Lexical Retrieval with Semantic Residual Embedding. *arXiv preprint*. ArXiv:2004.13969.

Luyu Gao, Xueguang Ma, Jimmy Lin, and Jamie Callan. 2023. Precise Zero-Shot Dense Retrieval without Relevance Labels. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1762–1777, Toronto, Canada. Association for Computational Linguistics.

Michael Glass, Gaetano Rossiello, Md Faisal Mahbub Chowdhury, Ankita Naik, Pengshan Cai, and Alfio Gliozzo. 2022. Re2G: Retrieve, rerank, generate. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2701–2715, Seattle, Washington, United States. Association for Computational Linguistics.

Yizheng Huang and Jimmy Huang. 2024. A Survey on Retrieval-Augmented Text Generation for Large Language Models. *arXiv preprint*. ArXiv:2404.10981.

HuggingFace. 2025. Evaluate: A library for easily evaluating machine learning models and datasets. Https://github.com/huggingface/evaluate.

Kalervo Järvelin and Jaana Kekäläinen. 2002. Cumulated gain-based evaluation of IR techniques. *ACM Trans. Inf. Syst.*, 20(4):422–446.

Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, Alexandre Ramé, Morgane Rivière, Louis Rouillard, and et al. Mesnard. 2025. Gemma 3 Technical Report. *arXiv preprint*. ArXiv:2503.19786.

Kiseung Kim and Jay-Yoon Lee. 2024. RE-RAG: Improving Open-Domain QA Performance and Interpretability with Relevance Estimator in Retrieval-Augmented Generation. In *Proceedings of the*

2024 Conference on Empirical Methods in Natural Language Processing, pages 22149–22161, Miami, Florida, USA. Association for Computational Linguistics.

Anastasia Krithara, James G. Mork, Anastasios Nentidis, and Georgios Paliouras. 2023. The Road From Manual To Automatic Semantic Indexing Of Biomedical Literature: a 10 Years Journey. *Frontiers in Research Metrics and Analytics*, 8.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient Memory Management for Large Language Model Serving with PagedAttention. In *Proceedings of the 29th Symposium on Operating Systems Principles*, SOSP '23, pages 611–626, Koblenz, Germany. Association for Computing Machinery.

Patrick S. H. 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. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020*, virtual.

Chao Li, Zhiyuan Liu, Mengmeng Wu, Yuchi Xu, Huan Zhao, Pipei Huang, Guoliang Kang, Qiwei Chen, Wei Li, and Dik Lun Lee. 2019. Multi-Interest Network with Dynamic Routing for Recommendation at Tmall. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, CIKM '19, pages 2615–2623, Beijing, China. Association for Computing Machinery.

Chin-Yew Lin. 2004. ROUGE: A Package for Automatic Evaluation of Summaries. In *Proceedings of the ACL-04 Workshop*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

Jerry Liu. 2022. LlamaIndex. Https://github.com/jerryjliu/llama_index.

Sean MacAvaney, Craig Macdonald, and Iadh Ounis. 2022. Streamlining Evaluation with ir-measures. In *Advances in Information Retrieval: 44th European Conference on IR Research, ECIR 2022, Stavanger, Norway, April 10–14, 2022, Proceedings, Part II*, page 305–310, Berlin, Heidelberg. Springer-Verlag.

Andrew Mutton, Mark Dras, Stephen Wan, and Robert Dale. 2007. GLEU: Automatic Evaluation of Sentence-Level Fluency. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 344–351, Prague, Czech Republic. Association for Computational Linguistics.

Linyong Nan, Chiachun Hsieh, Ziming Mao, Xi Victoria Lin, Neha Verma, Rui Zhang, Wojciech Kryściński, Hailey Schoelkopf, Riley Kong, Xiangru Tang, Mutethia Mutuma, Ben Rosand, Isabel Trindade, Renusree Bandaru, Jacob Cunningham, Caiming Xiong, Dragomir Radev, and Dragomir Radev. 2022. FeTaQA: Free-form Table Question Answering. *Transactions of the Association for Computational Linguistics*, 10:35–49.

Irina Nikishina, Özge Sevgili, Mahei Manhai Li, Chris Biemann, and Martin Semmann. 2025. Creating a Taxonomy for Retrieval Augmented Generation Applications. *arXiv preprint*. ArXiv:2408.02854.

OpenAI. 2025a. OpenAI Evals — A framework for evaluating LLMs. Https://github.com/openai/evals.

OpenAI. 2025b. openai-python: Official Python library for the OpenAI API. Https://github.com/openai/openai-python.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a Method for Automatic Evaluation of Machine Translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Qdrant Team. 2025. Qdrant: High-performance, massive-scale Vector Database and Vector Search Engine for the next generation of AI. Https://github.com/qdrant/qdrant.

Stephen Robertson and Hugo Zaragoza. 2009. The Probabilistic Relevance Framework: BM25 and Beyond. *Foundations and Trends in Information Retrieval*, 3(4):333–389.

Dongyu Ru, Lin Qiu, Xiangkun Hu, Tianhang Zhang, Peng Shi, Shuaichen Chang, Cheng Jiayang, Cunxiang Wang, Shichao Sun, Huanyu Li, Zizhao Zhang, Binjie Wang, Jiarong Jiang, Tong He, Zhiguo Wang, Pengfei Liu, Yue Zhang, and Zheng Zhang. 2024. RAGChecker: A Fine-grained Framework for Diagnosing Retrieval-Augmented Generation. In *Advances in Neural Information Processing Systems*, volume 37, pages 21999–22027. Curran Associates, Inc.

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: Human Language Technologies (Volume 1: Long Papers)*, pages 338–354, Mexico City, Mexico. Association for Computational Linguistics.

Pranab Sahoo, Prabhash Meharia, Akash Ghosh, Sriparna Saha, Vinija Jain, and Aman Chadha. 2024. A Comprehensive Survey of Hallucination in Large Language, Image, Video and Audio Foundation Models. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 11709–11724, Miami, Florida, USA. Association for Computational Linguistics.

Parth Sarthi, Salman Abdullah, Aditi Tuli, Shubh Khanna, Anna Goldie, and Christopher D. Manning. 2024. RAPTOR: Recursive Abstractive Processing for Tree-Organized Retrieval. In *The Twelfth International Conference on Learning Representations*, Vienna, Austria. The Association for Computational Linguistics.

Kurt Shuster, Spencer Poff, Moya Chen, Douwe Kiela, and Jason Weston. 2021. Retrieval Augmentation Reduces Hallucination in Conversation. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 3784–3803, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Snowflake Inc. 2024. TruLens Eval. Https://www.trulens.org.

Marina Sokolova and Guy Lapalme. 2009. A systematic analysis of performance measures for classification tasks. *Information Processing & Management*, 45(4):427–437.

Jan Strich, Enes Kutay Isgorur, Maximilian Trescher, Chris Biemann, and Martin Semmann. 2025. T^2-RAGBench: Text-and-Table Benchmark for Evaluating Retrieval-Augmented Generation. *Preprint*, arXiv:2506.12071.

Ellen M. Voorhees. 1993. Using WordNet to disambiguate word senses for text retrieval. In *Proceedings of the 16th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '93, pages 171–180, Pittsburgh, Pennsylvania, USA. Association for Computing Machinery.

Liang Wang, Nan Yang, Xiaolong Huang, Linjun Yang, Rangan Majumder, and Furu Wei. 2024. Multilingual E5 Text Embeddings: A Technical Report. *arXiv preprint*. ArXiv:2402.05672.

Yuchen Wang, Shangxin Guo, and Chee Wei Tan. 2025. From Code Generation to Software Testing: AI Copilot With Context-Based Retrieval-Augmented Generation. *IEEE Software*, 42(4):34–42.

Guangzhi Xiong, Qiao Jin, Zhiyong Lu, and Aidong Zhang. 2024. Benchmarking Retrieval-Augmented Generation for Medicine. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 6233–6251, Bangkok, Thailand. Association for Computational Linguistics.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018. HotpotQA: A Dataset for Diverse, Explainable Multi-hop Question Answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2369–2380, Brussels, Belgium. Association for Computational Linguistics.

Matei A. Zaharia, Andrew Chen, Aaron Davidson, Ali Ghodsi, Sue Ann Hong, Andy Konwinski, Siddharth Murching, Tomas Nykodym, Paul Ogilvie, Mani Parkhe, Fen Xie, and Corey Zumar. 2018. Accelerating the Machine Learning Lifecycle with MLflow. *IEEE Data Eng. Bull.*, 41:39–45.

A Dataset Overview

For evaluation, we selected four datasets from different domains to demonstrate EncouRAGe's flexibility in handling diverse datasets.

HotPotQA. The dataset (Yang et al., 2018) contains 7,395 general knowledge question–answer pairs sourced from Wikipedia, emphasizing multi-hop reasoning. Each question is linked to a single gold document, resulting in 7,395 papers in total, with an average document length of 1,421.2 tokens. Questions average 21.4 tokens, reflecting concise query formulation. HotPotQA challenges models to combine information across multiple articles to answer a single question accurately. For evaluation, we employ *MRR@10* for retrieval and *F1-Score* for answer generation, consistent with multi-hop reasoning tasks.

FetaQA. The dataset (Nan et al., 2022) includes over 7,300 QA pairs that require complex reasoning over tabular data, extracted from Wikipedia articles. The dataset emphasizes semantic understanding and reasoning across multiple table cells. Each question is accompanied by a textual explanation and a gold-standard answer derived from structured tables. We use the training split and all table documents related to these QA pairs, resulting in a total of 6,929 documents to process. Evaluation employs *MRR@10*, *Recall@10*, and *F1-Score* to measure both retrieval effectiveness and answer quality.

FinQA. We use a variant from FinQA that is focused on retrieval from T^2-RAGBench (Strich et al., 2025). The dataset comprises approximately 6,200 finance-related question–answer pairs derived from company reports and other financial documents. Each question is associated with a single financial document containing a combination of text and tables. On average, there are roughly three QA pairs per document, resulting in a total of 2,110 documents for retrieval. We used a modified version of the training split, reformulating questions to make them more suitable for RAG. For reasons of anonymity, the original paper is not cited in this submission. For evaluation, we employ *MRR@10*, *Recall@10*, and *Number Match (NM)*, a metric specifically designed to compare numerical values, like Exact Match, up to the second decimal place.

BioASQ. The dataset (Krithara et al., 2023) contains approximately 4,000 biomedical question–answer pairs derived from scientific literature in the life sciences. It is designed to evaluate factual biomedical question answering in combination with semantic retrieval. The dataset is updated annually, and the version used in our evaluation is BioASQ11. Each QA pair includes multiple reference documents, over 34k in total, making it particularly challenging for the retriever to locate the relevant information. Questions are grouped into four types: list, yes/no, factoid, and summary, each requiring different response formats. For evaluation, we use *MAP* and *Recall@10* for retrieval, and *F1-Score* for generation.

B Metrics Overview

Metric	Description
Generator Metrics	
Exact Match	Checks whether the generated text exactly matches the reference answer without any differences in wording or formatting.
Number Match (Anonym)	Verifies whether numeric values in the generated text are approximately equal to the reference values within a small tolerance.
BLEU (Papineni et al., 2002)	Measures the overlap of n-grams between generated and reference text, emphasizing precision of matching word sequences.
ROUGE (Lin, 2004)	Measures overlap of n-grams and longer sequences, often used to evaluate recall of important content.
GLEU (Mutton et al., 2007)	Variant of BLEU that balances recall and precision, providing a more symmetric similarity measure.
Precision (Sokolova and Lapalme, 2009)	Fraction of generated tokens that are correct compared to the reference text.
Recall (Sokolova and Lapalme, 2009)	Fraction of reference tokens that are successfully generated by the model.
F1 (Sokolova and Lapalme, 2009)	Harmonic mean of precision and recall, balancing correctness and completeness.
Retrieval Metrics	
Mean Reciprocal Rank (MRR) (Voorhees, 1993)	Evaluates how highly the first relevant document is ranked by averaging reciprocal ranks over multiple queries.
Mean Average Precision (MAP) (Voorhees, 1993)	Computes the mean of average precision values across queries, reflecting ranking quality over all relevant documents.
nDCG (Järvelin and Kekäläinen, 2002)	Measures ranking quality by assigning higher importance to relevant documents appearing at top positions.
Recall@k (Robertson and Zaragoza, 2009)	Measures the proportion of relevant documents retrieved within the top-k ranked results.
HitRate@k (Li et al., 2019)	Indicates whether at least one relevant document appears within the top-k retrieved results.
LLM-Based Metrics	
Answer Relevance (Es et al., 2024b)	Evaluates whether the generated answer is relevant and directly addresses the given query.
Answer Faithfulness (Es et al., 2024b)	Measures whether the generated answer is factually consistent with the provided context.
Non-Answer Critic (Es et al., 2024b)	Detects responses that fail to answer the query or contain only irrelevant information.
Answer Similarity (Es et al., 2024b)	Measures semantic similarity between the generated answer and a reference answer or alternative outputs.
Context Recall (Es et al., 2024b)	Evaluates whether the retrieved context contains the information required to produce the answer.
Context Precision (Es et al., 2024b)	Measures how much of the retrieved context is actually relevant for answering the query.

Table 3: Overview of the most used RAG evaluation metrics that we implemented in *EncouRAGe*.