FIRE-QA: Full Interaction Retrieval and Enhanced Question Answering

Hamad Alrashid Nabeel Bukhari
Abdullah Altamimi Abrar Alabdulwahab
Samer Almontasheri

Abstract

Question-answering (QA) remains one of the core challenges in Natural Language Processing (NLP). While most QA & Information Retrieval (IR) systems perform well on single-hop questions, where answers are explicitly stated in a single passage, they often struggle with multi-hop questions that require multiple reasoning steps across different documents. FIRE-QA (Full Interaction Retrieval and Enhanced Question Answering) addresses this gap by enabling multi-hop question-answering across a large corpus of documents while explaining its reasoning process. The architecture focuses on improving information retrieval by combining dual-encoder late-interaction models along with cross-encoders for high-recall retrieval and precise evidence selection, respectively. Large Language Models (LLMs) handle both query transformation and answer generation. FIRE-QA is implemented in Python and includes a web interface that allows users to interact with the QA system easily.

1 Introduction

Since most of the existing QA and IR systems that rely on single-hop techniques perform effectively, they often struggle when dealing with multi-hop questions that require complex reasoning across multiple sources. To address this limitation, FIRE-QA (Full Interaction Retrieval and Enhanced Question Answering) is introduced as a modular pipeline designed to improve multi-hop reasoning and retrieval performance on the HotpotQA dataset.

The main contributions are as follows:

- A unified multi-hop QA architecture (FIRE-QA) that integrates query decomposition, late-interaction retrieval, and cross-encoder reranking for high-recall and high-precision evidence selection.
- A context-grounded reasoning and generation module that applies Chain-of-Thought prompting to produce answers strictly from retrieved evidence, ensuring factual accuracy and reducing hallucinations.

The HotpotQA dataset [1] is used to evaluate the system's ability to locate and connect evidence across multiple documents before producing an answer and an interpretable reasoning path.

To get a high-recall retrieval and precise evidence selection, our framework focuses on improving information retrieval by combining the late-interaction

dense retriever, modeled after the ColBERT family, which preserves token-level signals while remaining indexable at scale, with a cross-encoder re-ranker that jointly scores (query, passage) pairs to select compact, faithful evidence[2],[3], [4],[5]. In addition, a lightweight LLM, Qwen3[6], performs answer generation and optional query rewriting to refine retrieval in hard cases. We also included a web interface that has a minimal design just to let the users interact with the system easily. Our work shows a significant improvement where the system can answer a multi-hop question with accurate evidence a reasoning.

2 Related Work

This section reviews existing approaches to multi-hop QA and information retrieval systems.

2.1 Benchmarks for multi-hop QA

HotpotQA introduced natural, multi-sentence questions with labeled supporting facts and distractor/full-wiki evaluation, becoming the canonical MHQA testbed [1]. Follow-up datasets (e.g., 2WikiMultiHopQA, MuSiQue) further stress compositional reasoning, but HotpotQA remains the most widely used benchmark for explainable multi-document QA and is the focus of this work.

2.2 Retrieval foundations: sparse, dense, and late interaction

Sparse retrieval (BM25) excels when queries share vocabulary with answers and is often used as a baseline or part of hybrid stacks [2]. Dense passage retrieval (DPR) maps queries and passages into a shared vector space and improves top-k recall on open-domain QA benchmarks [3]. For MHQA, multi-hop dense retrieval (MDR) conditions later retrieval steps on previously found evidence, improving results on HotpotQA and other multi-evidence tasks [8]. In parallel, late-interaction retrieval (e.g., ColBERT/ColBERTv2) encodes tokens independently and computes MaxSim interactions at query time, preserving fine-grained token matches that are valuable for multi-hop chaining and entity bridging while remaining indexable and efficient [4, 5]. These properties motivate FIRE-QA's choice of a late-interaction retriever as the first stage.

3 Methodology

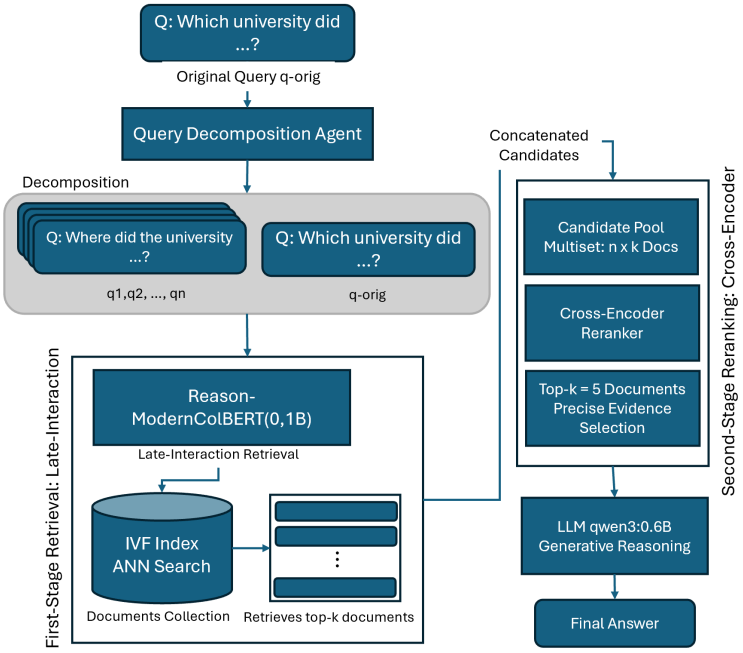

Figure 1: The architecture of FIRE-QA

Our system architecture, $\mathcal{F}$, for multi-hop question answering is a multi-stage pipeline designed to find the optimal answer A from a document index I given an original query q_{orig}:

$$A = \mathcal{F}(q_{\text{orig}}, I) \tag{1}$$

As displayed in Figure 1, the overall process consists of three main phases: query decomposition, multi-stage retrieval and reranking, and a generative answer. The core is a novel multi-stage retrieval pipeline that uses query decomposition followed by late-interaction encoders for high-recall retrieval and cross-encoder rerankers for precise evidence selection.

3.1 1. Query Decomposition

The process begins by transforming the original query to sub-queries. We we a query decomposition agent [7] to break down the original query q_{orig} into a set of targeted sub-queries $Q_d = \{q_1, q_2, \ldots, q_n\}$, which also includes the original query. This approach creates multiple search targets to ensure that we create a large pool of candidate documents that contain all all necessary evidence.

3.2 2. Retrieval and Reranking

This phase is a multi-stage process designed to effectively find the most relevant documents from a large corpus.

3.2.1 First-Stage Retrieval: Late-Interaction

First, we trigger a retrieval for each sub-query $q_i \in Q_d$. Instead of using traditional single-vector models that compress documents into a single `[CLS]` token or an average-pooled vector, we used a late-interaction retrieval model. This allows for full, token-level interaction between the query (q) and document (d) token embeddings. We use **Reason-ModernColBERT (0.1b)**, a model fine-tuned on the reasonir-hq dataset for QA and based on ModernBERT. This is a bi-encoder model that generates token-level embeddings for all documents during offline indexing. For each query at run-time, a relevance score is computed against the retrieved document vectors via a MaxSim operation, similar to the original late-interaction ColBERT model as shown in Figure 2. This operation compares every query token embedding with all document token embeddings to capture contextualized token information.

For efficient search, documents are indexed offline using an Inverted File Index (IVF) data structure, which provides an efficient interface for the approximate nearest neighbor (ANN) vector search. We index all of the sentences in the dev-split, totaling to 275966 sentences and 9950730 embeddings. For the IVF hyper-parameters, we use the default settings in PLAID.

$$D_{\text{top-}k_1 i} = \text{ANN}(q_i, I, k_1) \tag{2}$$

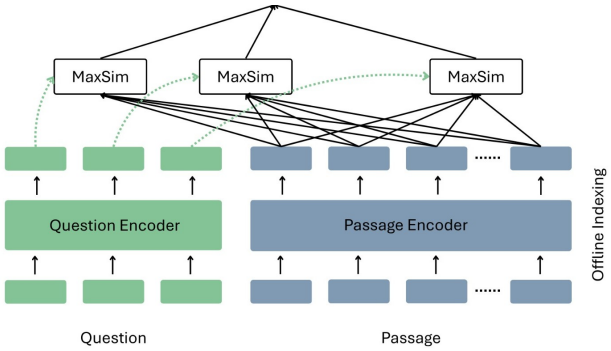

Figure 2: The original figure of the ColBERT model, showcasing the question and passage dual-encoder architecture with late-interaction mechanism via the MaxSim operation

3.2.2 Second-Stage Reranking: Cross-Encoder

The top-k_1 documents retrieved from all sub-queries, $Q_d = \{q_1, \ldots, q_n\}$, are grouped to form a single candidate pool. This pool is a multiset formed by concatenating the top-k_1 results from each sub-query, resulting in

$$|D_{\text{pool}}| = |Q_d| \times k_1$$

total number of retrieved documents before reranking. The pool is then passed to a more powerful cross-encoder to rerank and obtain the top-k_2 documents. We use the bge-reranker-v2-m3 (0.6b) cross-encoder model. Unlike dual-encoders that process query and document separately, cross-encoders process both inputs by concatenating them (`[CLS] query [SEP] document [SEP]`), allowing for full token interaction between both inputs. Cross-encoders are significantly more accurate than dual-encoders, but much slower since each query-document pair must be passed to the model [5].

The reranker filters the entire candidate pool by scoring the relevancy of each query-document pair, sorts them, and obtains the top-k_2 documents.

$$D_{\text{top-}k_2} = \text{Rerank}(q_{\text{orig}}, \bigcup_{q_i \in Q_d} \text{ANN}(q_i, I, k_1), k_2) \tag{3}$$

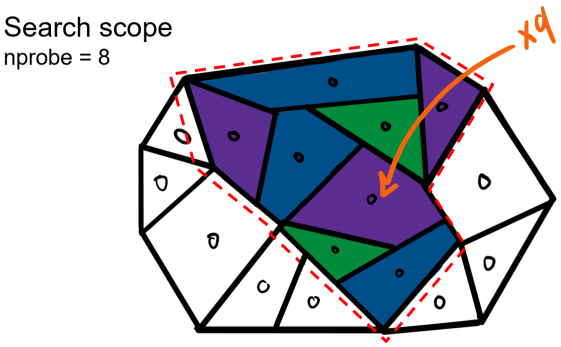

Figure 3: A visualization of an IVF index, where the query vector is searched against the nearest nprobe clusters only.

where ANN is the first-stage retrieval of k_1 documents for each query, and Rerank is the reranking, which applies the set union operator over all documents and selects the final k_2 documents.

3.3 3. Answer Generation

Lastly, the **top 5 documents** (i.e., $k_2 = 5$) from the reranker are passed along the original query to a Large Language Model with Chain-of-Thought reasoning output for improved generation [8]. The LLM is prompted to only answer from the given context and fallback to "INSUFFICIENT EVIDENCE" when the context does not contain the enough information to answer the query.

3.4 Implementation Details

Our system is implemented in Python using the HotPotQA dataset under the Fullwiki [1] setting. We used the Pylate [9] library for efficient late-interaction retrieval. The retriever component employs the ColBERT model[2] from Pylate, which performs semantic search over a pre-built PLAID index [10] to find the most relevant passages. After retrieval, the CrossEncoder from the Sentence-Transformers library reranks the candidate documents by relevance, ensuring that the most accurate and meaningful passages are passed to the answer generator. For the answer generation stage, we used qwen3:0.6b [6], a lightweight yet capable model, integrated through Ollama for local deployment and offline operation. Finally, we built a simple user interface using Gradio[11], allowing users to input questions and receive generated answers along with supporting reasoning.

4 Experiments, Results, and Discussion

This section presents the experimental setup, evaluation metrics, results, and analysis.

Table 1: Performance of FIRE-QA configurations with Qwen3 1.7B LLM on HotpotQA dev set.

Configuration	Answer				Supporting Facts			
	EM	F1	Prec	Recall	SP EM	SP F1	SP Prec	SP Recall
Direct LLM only	0.020	0.032	0.032	0.036	0.000	0.000	0.000	0.000
Retriever only	0.204	0.282	0.285	0.327	0.061	0.386	0.422	0.363
Retriever + Query Rewriter	0.203	0.278	0.282	0.316	0.057	0.384	0.444	0.354
Retriever + Reranker	0.219	0.306	0.308	0.356	0.115	0.471	0.518	0.443
Retriever + Reranker + Query Rewriter	0.217	0.299	0.300	0.345	0.084	0.462	**0.580**	0.406

Table 2: Performance of FIRE-QA configurations with Qwen 0.6B LLM on HotpotQA dev set.

Configuration	Answer				Supporting Facts			
	EM	F1	Prec	Recall	SP EM	SP F1	SP Prec	SP Recall
Direct LLM only	0.002	0.005	0.005	0.005	0.000	0.000	0.000	0.000
Retriever only	0.112	0.189	0.186	0.235	0.000	0.189	0.118	0.497
Retriever + Query Rewriter	0.098	0.170	0.170	0.210	0.000	0.192	0.122	0.479
Retriever + Reranker	0.142	0.217	0.213	0.268	0.000	0.200	0.125	0.525
Retriever + Reranker + Query Rewriter	0.110	0.193	0.183	0.251	0.000	0.213	**0.138**	0.511

The FIRE-QA system was evaluated on the HotPotQA dataset in the full-wiki setting using the dev split only. We evaluate only on the first 1000 queries, due to time constraints. We evaluate the system on two versions of Qwen3: 0.6 billion (Table 2) and 1.7 billion (Table 1) parameters. For each, we evaluate on four configurations: LLM-only, basic retriever only, retriever with query rewriter, retriever with reranker, and the full setup combining retriever, reranker, and query rewriter. We set $k_1 = 100$ and $k_2 = 10$. We run the evaluation setup on a machine with RTX 5070 Ti and 32GB of RAM. All models are

loaded to the GPU, while the PLAID index is loaded to the CPU due to its larger size.

4.1 Evaluation Metrics

Performance was measured using standard HotpotQA metrics: Exact Match (EM) and F1 for answer overlap with ground-truth answers. Supporting facts metrics (SP EM, SP F1, SP Prec, SP Recall) assess retrieval of ground-truth supporting sentences. Higher values indicate better performance across recall retrieval and precise generation.

4.2 Discussion

The baseline is the LLM-only answer the query. The poor results demonstrate how LLM-based QA systems struggle with multi-hop questions. This highlights the need for having a powerful retrieval component in the loop. Reading the tables from top to bottom, the scores increase across metrics as we add the late-interaction model, followed by a cross-encoder and a query-rewriter. In Figure 1, the full FIRE-QA pipeline achieves the highest supporting facts precision of 0.580. This is mainly due to how the retrieval captures token-level semantics across multiple document embeddings.

Our FIRE-QA system outputs a result to the user, as shown in Figure 4.

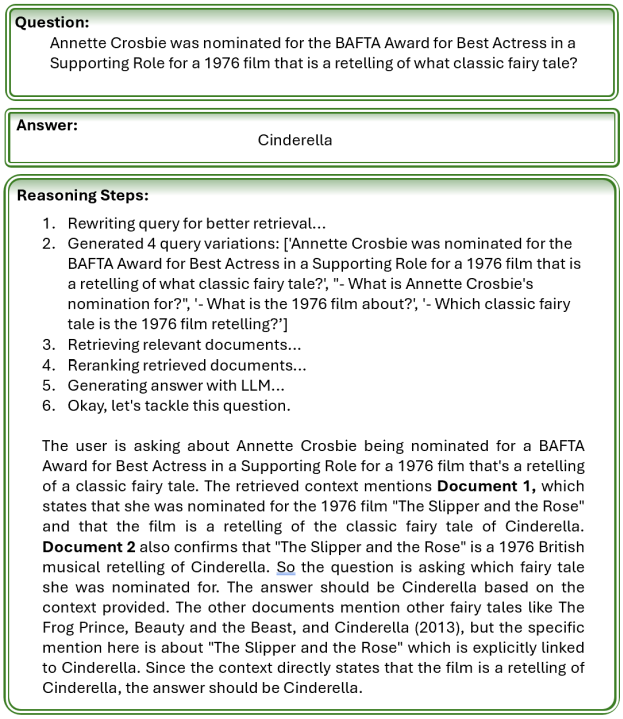

Figure 4: The output illustrates the question, the corresponding answer, and the reasoning supported by evidence.

5 Conclusion

FIRE-QA demonstrates that a retrieval-focused design can make multi-hop question-answering both practical and transparent. By pairing late-interaction retrieval to capture token-level semantics across documents with a cross-encoder re-ranker to select a small evidence set, the system consistently produces well-grounded contexts for the generator. The pipeline is fully modular and implemented in Python, making it straightforward to extend and evaluate across components.

The current prototype relies on fixed hyperparameters and a single re-ranking pass. It does not yet explore end-to-end optimization of multi-hop retrieval decisions or advanced query transformation. Our evaluation scope, including dataset coverage and models selection and tuning, also remains limited.

Improvements for future work include:

- Add an iterative query-rewriting loop with stopping criteria to recover missed hops on difficult questions.
- Research end-to-end multi-hop retrieval methods such as beam search.
- Use larger multilingual models where they can provide the greatest benefit, including more powerful cross-encoder re-rankers and stronger LLMs for answer generation, while exploring smaller or distilled variants for efficiency.

- Compare alternative re-rankers & dual-encoders for QA.
- Research effective indexing strategies for improved multi-vector retrieval.

Overall, FIRE-QA shows that a carefully engineered multi-stage retrieval system, grounded in token-level interaction and selective re-ranking, can improve both the answerability and the evidence recall of multi-hop question answering.

References

[1] Z. Yang, P. Qi, S. Zhang, Y. Bengio, W. W. Cohen, R. Salakhutdinov, and C. D. Manning, "Hotpotqa: A dataset for diverse, explainable multi-hop question answering," *arXiv preprint arXiv:1809.09600*, 2018.

[2] O. Khattab and M. Zaharia, "Colbert: Efficient and effective passage search via contextualized late interaction over bert," in *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval*, 2020, pp. 39–48.

[3] K. Santhanam, O. Khattab, J. Saad-Falcon *et al.*, "Colbertv2: Effective and efficient retrieval via lightweight late interaction," *arXiv preprint arXiv:2112.01488*, 2021.

[4] R. Nogueira and K. Cho, "Passage re-ranking with bert," *arXiv preprint arXiv:1901.04085*, 2019.

[5] R. Nogueira, W. Yang, K. Cho, and J. Lin, "Multi-stage document ranking with bert," *arXiv preprint arXiv:1910.14424*, 2019.

[6] A. Yang, A. Li, B. Yang, B. Zhang, B. Hui, B. Zheng, B. Yu, C. Gao, C. Huang, C. Lv *et al.*, "Qwen3 technical report," *arXiv preprint arXiv:2505.09388*, 2025.

[7] P. J. L. Ammann, J. Golde, and A. Akbik, "Question decomposition for retrieval-augmented generation," *arXiv preprint arXiv:2507.00355*, 2025. [Online]. Available: https://arxiv.org/abs/2507.00355

[8] J. Wei, X. Wang, D. Schuurmans, M. Bosma, B. Ichter, F. Xia, E. Chi, Q. Le, and D. Zhou, "Chain-of-thought prompting elicits reasoning in large language models," *arXiv preprint arXiv:2201.11903*, 2022. [Online]. Available: https://arxiv.org/abs/2201.11903

[9] A. Chaffin and R. Sourty, "Pylate: Flexible training and retrieval for late interaction models," *arXiv preprint arXiv:2508.03555*, 2025.

[10] K. Santhanam, O. Khattab, C. Potts, and M. Zaharia, "Plaid: an efficient engine for late interaction retrieval," in *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, 2022, pp. 1747–1756.

[11] A. Abid, A. Abdalla, A. Abid, D. Khan, A. Alfozan, and J. Zou, "Gradio: Hassle-free sharing and testing of ml models in the wild," *arXiv preprint arXiv:1906.02569*, 2019.