PolyU NLP

COMP5423 - NATURAL LANGUAGE PROCESSING

HONG KONG POLYTECHNIC UNIVERSITY

DEPARTMENT OF COMPUTING

Group 16 Project Report

Group Members:
Zhang Yuanlin (25104351g),
CHEN Zhuokai (25106193g),
LIU Huawei (25104825g),
LUO Senhang (25123978g),

Date: November 29, 2025

1 Introduction

Retrieval-Augmented Generation (RAG) has become a key paradigm in modern NLP systems, enabling Large Language Models (LLMs) to incorporate external knowledge during generation. Compared to standalone generation-based models, RAG offers enhanced factual accuracy, transparency, and interpretability, making it particularly suitable for applications requiring grounded reasoning.
In this project, we implemented a complete RAG system following the COMP5423 course requirements, covering all major components including retrieval (BM25 and dense retrieval), hybrid search, multi-vector retrieval, generation, multi-turn dialogue, agentic workflow, and a user interface. Our implementation strictly aligns with the provided template and is fully executable.

The goal of the project is to:

- Build a robust and modular RAG system using modern retrieval techniques.
- Integrate an LLM (Qwen2.5-0.5B-Instruct) as the core generator.
- Support intermediate reasoning features such as query rewriting, sub-question decomposition, reranking, and self-checking.
- Provide both structured output and transparent workflow reasoning.
- Design a terminal-based user interface to support interactive querying.

Our system is designed to be fully modular, reproducible, and extensible. The architecture follows the official project specification and README design. All required features, including Dense Retrieval, Instruction Dense Retrieval, Hybrid Pipeline, Multi-vector Representation, Multi-turn Dialogue, and Agentic Workflow are successfully completed and evaluated.

2 System design method

This system integrates the Retrieval-Augmented Generation (RAG) method, employing six different retrieval methods and three generation modes, The overall architecture of the system is shown in the figure:

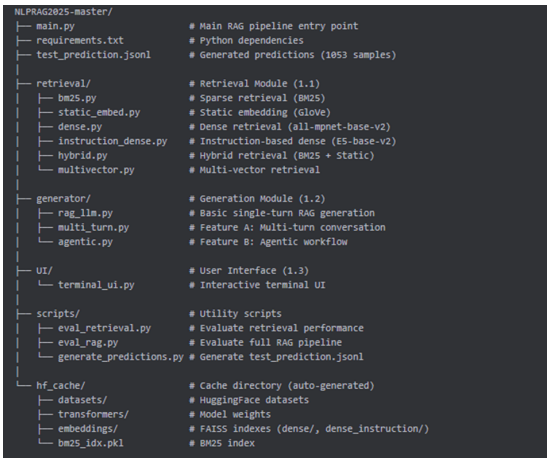

Figure 1: System Architecture.

2.1 Retrieval Module

The Retrieval Module provides first stage candidate generation for downstream reranking and QA. We implemented six retrieval methods covering lexical, lightweight semantic, contextual semantic, token level matching, and hybrid fusion approaches. Below are more details for each of them.

2.1.1 Sparse retrieval: BM25

BM25 is a fast and interpretable first-stage retriever that produces candidate documents for reranking or hybrid fusion. It scores documents by IDF-weighted term matches with term-frequency saturation and document-length normalization, and i for a query Q and document D the score is computed as shown below:

$$\text{score}(D,Q) = \sum_{i=1}^{n} \text{IDF}(q_i) \cdot \frac{f(q_i,D)\,(k_1+1)}{f(q_i,D) + k_1\left(1 - b + b\dfrac{|D|}{\text{avgdl}}\right)}$$

The top-N candidates are returned to a downstream reranker or fused with dense retrieval. BM25 performs well on lexical/sparse queries and when compute is limited, and it serves as a robust, explainable baseline. Its limitation is weak handling of synonyms and semantic matches—combine it with dense methods or query expansion to improve semantic coverage.

2.1.2 Static embedding retrieval: IDF-weighted GloVe + FAISS

Static embeddings is a semantic first-stage retriever that produces candidate documents for reranking or hybrid fusion. It represents the query Q and document D as fixed

vectors emb(Q) and emb(D) and scores documents by vector similarity. In our implementation we use IDF-weighted GloVe average vectors and cosine similarity, and for a query Q and document D the score is computed as shown below:

$$\text{score}(D, Q) = \frac{\text{emb}(Q) \cdot \text{emb}(D)}{\|\text{emb}(Q)\| \, \|\text{emb}(D)\|}$$

The top-N candidates are returned to a downstream reranker or fused with lexical retrieval (e.g., BM25). Static embeddings improve recall for paraphrase and synonym-rich queries but require precomputed vectors, a FAISS ANN index, and additional storage/compute.

2.1.3 Dense retrieval: Sentence-transformers/all-mpnet-base-v2 + FAISS

Dense retrieval is a semantic first-stage retriever that encodes passages with a pretrained sentence transformer and returns nearest-neighbor passages via an ANN index. In our implementation, documents are split into fixed passages (150-token chunks with 50-token overlap), passages are encoded with sentence-transformers/all-mpnet-base-v2, L2-normalized, and indexed with FAISS.
The top N passages are returned to a downstream reranker or fused with lexical methods (e.g., BM25). Dense retrieval improves recall for semantic and paraphrase queries but requires a heavy encoder, precomputed vectors, and FAISS indexing; tune passage chunking, index type, and to balance recall, latency and storage.

2.1.4 Dense retrieval with instruction: intfloat/e5-base-v2 + FAISS

Dense retrieval with instruction is a semantic first-stage retriever that encodes passages using an Instruction-tuned sentence transformer and returns nearest-neighbor passages via a FAISS index. In our implementation, documents are split into overlapping passages (150-token chunks with 50-token overlap), passages are prefixed with an instruction token (”passage: ...”) and encoded using intfloat/e5-base-v2 (sentence-transformers). Embeddings are L2-normalized and indexed with FAISS.
When retrieving, queries are prefixed with ”query: ” to match the instruction format used at encoding. The top N passages are returned to a downstream reranker or fused with lexical methods. Instruction tuning usually improves alignment between query intent and passage representations, enhancing semantic matching for instruction-style or task-oriented queries, but requires a suitable instruction-tuned encoder and the usual precompute/indexing costs.

2.1.5 Multi-vector retrieval (ColBERT-style, lightweight)

Multi-vector retrieval applies a ColBERT-style scoring function that matches each query token (or local span) to its best-matching document token/span and sums those best

scores. In our lightweight implementation, we first use the dense retriever to produce a candidate set, then compute span-level embeddings on-the-fly for the query and candidates and apply ColBERT-style scoring. For query tokens q_i and document tokens d_j the score is:

$$\text{score}(Q, D) = \sum_{i \in Q_{\text{tokens}}} \max_{j \in D_{\text{tokens}}} \frac{\langle q_i, d_j \rangle}{\|q_i\| \, \|d_j\|}$$

The top N candidates are returned to a downstream reranker or used for final ranking. This approach captures fine-grained token-level matches and often improves precision on queries requiring local alignment, but it is more costly at query time (encoding and token-level matching) and is implemented here as a reranking step over a limited candidate set to balance cost and quality.

2.1.6 Hybrid (BM25+E5)

Hybrid retrieval combines BM25's lexical precision with instruction-tuned E5's semantic recall to produce robust candidate sets. Concretely, for each query we retrieve candidates from BM25 and from the instruction-dense retriever, aggregate passage scores to document level, then fuse the two signals either by normalized score interpolation or by Reciprocal Rank Fusion (RRF). In score fusion the combined document score is:

$$\text{score}_{\text{hybrid}}(D, Q) = \alpha \, \text{norm}\Big(\text{score}_{\text{BM25}}(D, Q)\Big) + (1 - \alpha) \, \text{norm}\Big(\text{score}_{\text{e5}}(D, Q)\Big), \qquad \alpha \in [0, 1]$$

2.2 Generation Module

2.2.1 Basic RAG

Under this mode, the system will pass the retrieved documents together with the user's query to the generation model (Qwen2.5-0.5B-Instruct) to generate concise and accurate answers.

2.2.2 Feature A - Multi-turn Conversation:

Support multiple rounds of conversation and be capable of handling coreference resolution for pronouns and entities. For instance, when a user mentions "he", the system can automatically identify it as the previously mentioned character and generate the corresponding answer.

2.2.3 Feature B - Agentic Workflow:

Under this mode, before generating the answer, the system conducts query decomposition, self-checking and reasoning. Through the self-checking mechanism, the system

verifies whether the generated answer is accurate, thereby avoiding the generation of incorrect answers.

Figure 2: Gerneration Module.

2.2.4 Module integration

The retrieval module and the generation module of the system are integrated through a unified pipeline. After the user inputs a query, the system first uses the selected retrieval method (such as BM25, Dense, Hybrid, etc.) to retrieve relevant documents from the document library and passes these documents together with the query to the generation module. The generation module generates answers based on different patterns: in the basic single-round RAG mode, the system combines the query and the retrieved documents to generate concise answers; in the multi-round conversation mode, the system automatically resolves pronouns and generates continuous conversation answers based on the context; while in the Agentic workflow mode, the generation module verifies the accuracy of the answers through reasoning and self-checking mechanisms, ensuring that the generated answers are consistent with the retrieved evidence and returns a ”low confidence” prompt in case of uncertainty. Through this approach, the system can flexibly adjust different retrieval methods and generation modes, thereby providing more accurate and reasonable answers.

3 System flowchart

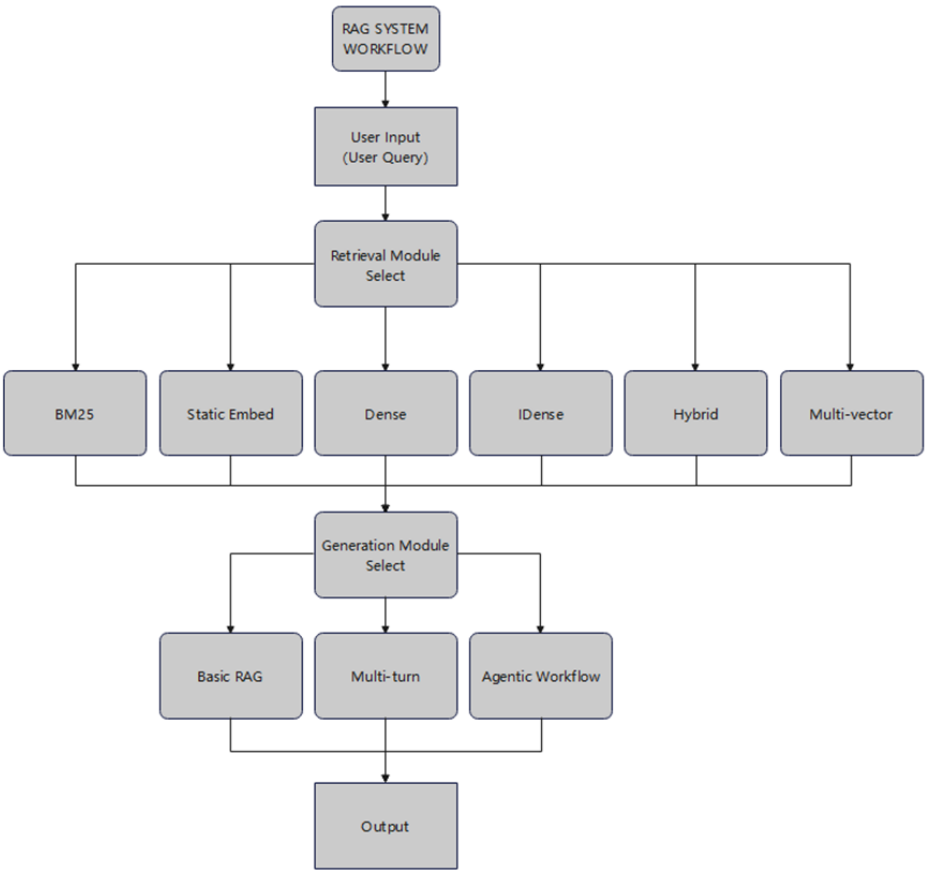

Figure 3: System Flowchart.

4 Result analysis

4.1 The effectiveness of the retrieval module

We conducted validation experiments on various retrieval methods using the validation split of the HQ small dataset. The two tables below summaries the top-10 results and at smaller cutoffs.

Table 1: Retrieval results (top-10) on the validation split.

Method	MAP@10	nDCG@10	Recall@10	Precision@10
BM25	0.6210	0.7212	0.8246	0.1649
Static (GloVe IDF)	0.3121	0.3967	0.4567	0.0913
Dense (all-mpnet)	0.4598	0.5609	0.6354	0.1271
Instruction–dense (E5)	0.6743	0.7672	0.8696	0.1739
Multivector (ColBERT–style)	0.4639	0.5689	0.6563	0.1313
Hybrid (BM25 + idense)	0.6880	0.7828	0.8946	0.1789

Table 2: Retrieval results at smaller cutoffs (MAP and Precision).

Method	MAP@2	MAP@5	Precision@2	Precision@5
BM25	0.4965	0.5987	0.5329	0.2945
Static	0.2569	0.2974	0.2892	0.1558
Dense	0.3731	0.4419	0.4050	0.2230
Instruction–dense	0.5398	0.6560	0.5779	0.3193
Multivector	0.3694	0.4430	0.4063	0.2277
Hybrid	0.5460	0.6674	0.5804	0.3258

The first table summarizes ranking metrics (MAP, nDCG), recall and precision at common cutoffs. The hybrid method (BM25 + instruction-tuned dense retrieval) performs best across MAP@10 (0.6880), nDCG@10 (0.7828) and Recall@10 (0.8946), and also yields the highest precision at common cutoffs (Precision@2 = 0.5804, Precision@5 = 0.3258). Instruction-tuned E5 is the strongest single dense retriever (MAP@10 = 0.6743, Recall@10 = 0.8696) and outperforms BM25 on most measures, while BM25 remains a strong lexical baseline (MAP@10 = 0.6210, Recall@10 = 0.8246). These results indicate that fusing BM25's lexical exact-match signal with instruction-tuned semantic embeddings provides complementary coverage: BM25 secures exact matches, while the E5 encoder recovers paraphrases and intent-aligned passages that lexical matching misses. Practically, the hybrid raises both the fraction of queries with at least one supporting document in the top-10 (Hit@10: 0.9883) and the average number of supporting documents per query (AvgHits: 1.7892), which should benefit downstream QA and reranking.

4.2 The effect of the generation module

In terms of the generation module, we have verified it through experiments in different modes:

4.2.1 Basic RAG

In this mode, the system can generate concise and accurate answers, which are suitable for directly responding to queries. However, it is unable to handle complex pronouns and multi-round conversations.

4.2.2 Feature A - Multi-turn Conversation

Through the analysis of pronouns and context tracking, the system can conduct effective conversations in multiple rounds. For instance, when a user asks ”Where was Barack Obama born?”, and then follows up with ”Where was his wife born?”, the system can automatically interpret ”his” as ”Barack Obama” and provide the correct answer.

4.2.3 Feature B - Agentic Workflow

This mode incorporates reasoning and self-checking functions, enabling verification of the generated answers. For instance, when querying ”How many cities are there in America?”, the system can identify the ”uncertainty” of the answer and return an ”uncertain” response instead of generating an incorrect one.

5 User interface design

This system implements an interactive user interface based on the terminal, aiming to provide users with a simple and intuitive way to query and display results. Users input query questions through the command line, and the system will execute the retrieval and generation process according to the user's selection, ultimately returning the generated answers and related documents. The interface design supports various retrieval methods and generation modes, and can flexibly respond to different query requirements.
The user first enters the query question through the terminal. The system offers three main retrieval methods for selection: BM25, Dense, and Hybrid. The user can choose the appropriate retrieval method according to their needs.

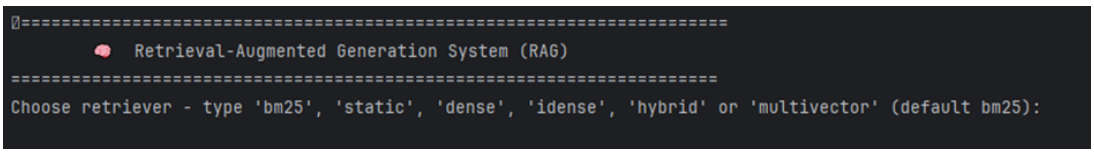

Figure 4: User interface for method selection.

After the retrieval method is selected, users can then choose the generation mode. The system offers three generation modes: basic single-round RAG (basic), multi-turn conversation, and Agentic workflow (agentic).

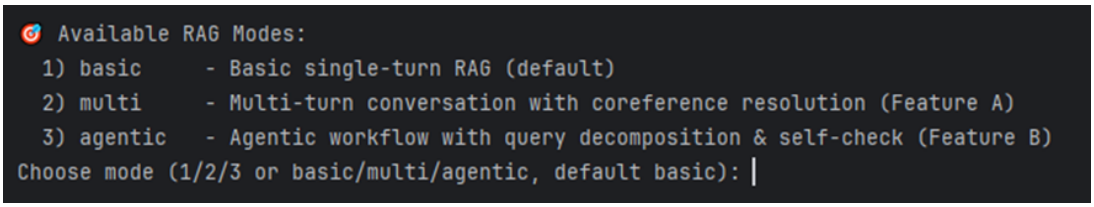

Figure 5: User interface for generation mode selection.

After the system is completed, a selection interface will be provided for users to make choices or to exit.

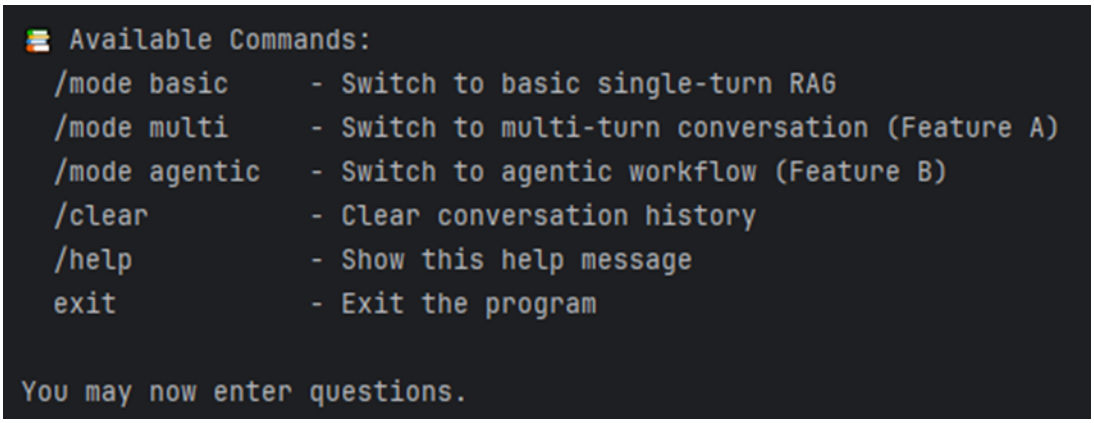

Figure 6: User interface for command selection.

Contribution and Declaration

Contribution

The table below specifies the contribution roles and percentage allocation for each group member:

- Zhang Yuanlin (%): Responsible for ...
- CHEN Zhuokai (%): Contributed to...
- LIU Huawei (%): Worked on...
- LUO Senhang (%): Handled...

Declaration

We, signed members of Group 16, hereby make the following declarations:

- We confirm that the work presented in this project is the original result of our collective efforts, except where explicitly acknowledged through proper citation.
- We guarantee the authenticity of all data, the accuracy of all analyses, and the integrity of the results presented herein. We affirm that the work has been conducted in accordance with accepted academic principles of the university.
- We have collectively reviewed and critically evaluated all materials submitted for this project, acknowledge and fully agree with the contribution distribution stated above, and hereby give our full approval to this final version of submission.
- We understand and acknowledge that any form of academic misconduct, including but not limited to plagiarism, fabrication, falsification, or other practices that violate academic integrity norms, will be subject to disciplinary actions in line with university policies.

By signing below, each member confirms their consent to the statements above:

Zhang Yuanlin	CHEN Zhuokai	LIU Huawei	LUO Senhang

References