COMP5423 - NATURAL LANGUAGE PROCESSING

HONG KONG POLYTECHNIC UNIVERSITY

DEPARTMENT OF COMPUTING

Group 15 Project Report

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

Date: November 30, 2025

1 Introduction

Retrieval-Augmented Generation (RAG)[1] 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.

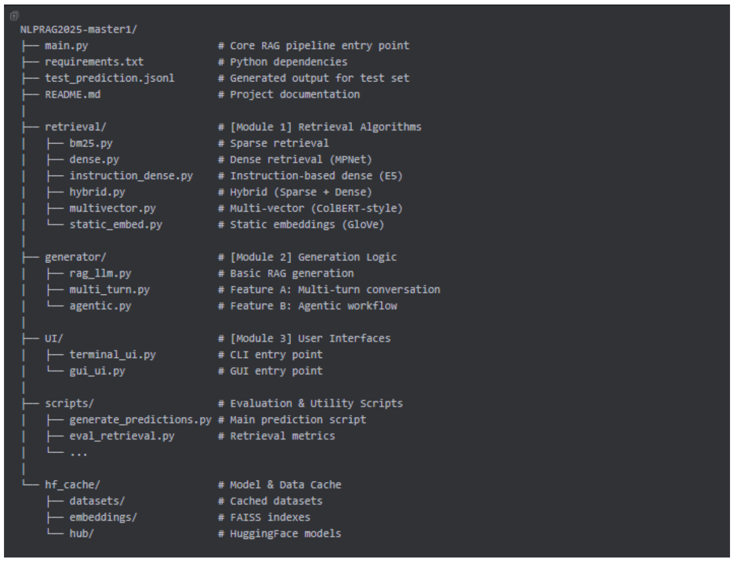
The goal of the project is to:

- 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:



```
NLPRAG2025-master1/
├── main.py                  # Core RAG pipeline entry point
├── requirements.txt         # Python dependencies
├── test_prediction.jsonl    # Generated output for test set
├── README.md                # Project documentation
│
├── retrieval/               # [Module 1] Retrieval Algorithms
│   ├── bm25.py              # Sparse retrieval
│   ├── dense.py             # Dense retrieval (MPNet)
│   ├── instruction_dense.py # Instruction-based dense (E5)
│   ├── hybrid.py            # Hybrid (Sparse + Dense)
│   ├── multivector.py       # Multi-vector (ColBERT-style)
│   └── static_embed.py      # Static embeddings (GloVe)
│
├── generator/               # [Module 2] Generation Logic
│   ├── rag_llm.py           # Basic RAG generation
│   ├── multi_turn.py        # Feature A: Multi-turn conversation
│   └── agentic.py           # Feature B: Agentic workflow
│
├── UI/                      # [Module 3] User Interfaces
│   ├── terminal_ui.py       # CLI entry point
│   └── gui_ui.py            # GUI entry point
│
├── scripts/                 # Evaluation & Utility Scripts
│   ├── generate_predictions.py # Main prediction script
│   ├── eval_retrieval.py    # Retrieval metrics
│   └── ...
│
└── hf_cache/                # Model & Data Cache
    ├── datasets/            # Cached datasets
    ├── embeddings/          # FAISS indexes
    └── hub/                 # HuggingFace models
```

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.[2] 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:

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

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.[3]
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.[4] 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

The generation stack sits downstream of a retrieval-agnostic prompt builder and exposes three planners (Basic RAG, Feature A, Feature B). Regardless of mode, the pipeline enforces short, evidence-grounded answers and logs intermediate artifacts for evaluation.

2.3 Deterministic Prompt Builder

Before any token is generated, the prompt builder fuses the top-k passages into a deterministic template. This ensures consistent instruction following across all modes. The template structure is as follows:
Passages are truncated to 500 characters, indexed, and separated by blank lines so downstream scripts can trace citations. The decoding policy is fixed for reproducibility (greedy search, temperature=0, max_new_tokens=100, repetition_penalty=1.2). The

```
[Instruction] Answer with a factual span (<=5 tokens) strictly
   grounded in the passages.
[Evidence]
[1] {doc_1_text}
[2] {doc_2_text}
...
Question: {user_query}
Short Answer:
```

Figure 2: The deterministic prompt template used for generation.

builder also records metadata (top_k_used, parsed_answer, parsed_evidence) consumed by evaluation scripts and Feature B's verifier.

2.3.1 Basic RAG

This mode prioritizes latency. The retrieved documents and original query are fed directly into the prompt, producing a concise span (1–5 tokens) that matches leaderboard grading.o ensure system stability, the planner includes an automatic fallback layer. If the user selects a resource-intensive retriever (like dense or idense) but the corresponding FAISS index is missing or fails to load, the system seamlessly degrades to the lightweight BM25 retriever. This guarantees that the pipeline always returns a valid response rather than crashing.The generation uses greedy decoding (temperature=0) to produce the most likely token sequence. A specialized post-processor then strips away common chat-model artifacts and enforces the short-answer constraint, ensuring the output format matches the strict requirements of the HotpotQA leaderboard.[5]

2.3.2 Feature A - Multi-turn Conversation

A shared ConversationMemory keeps the last N turns, storing extracted entities (person/place/thing) and lightweight relationship mappings (e.g., "his wife $\rightarrow$ Michelle Obama"). Before retrieval, a rule-based resolver rewrites pronouns and ellipses into standalone questions; complex cases fall back to the LLM rewriter using the recent dialogue summary. The rewritten query is inserted into the same prompt template, guaranteeing consistent instructions.

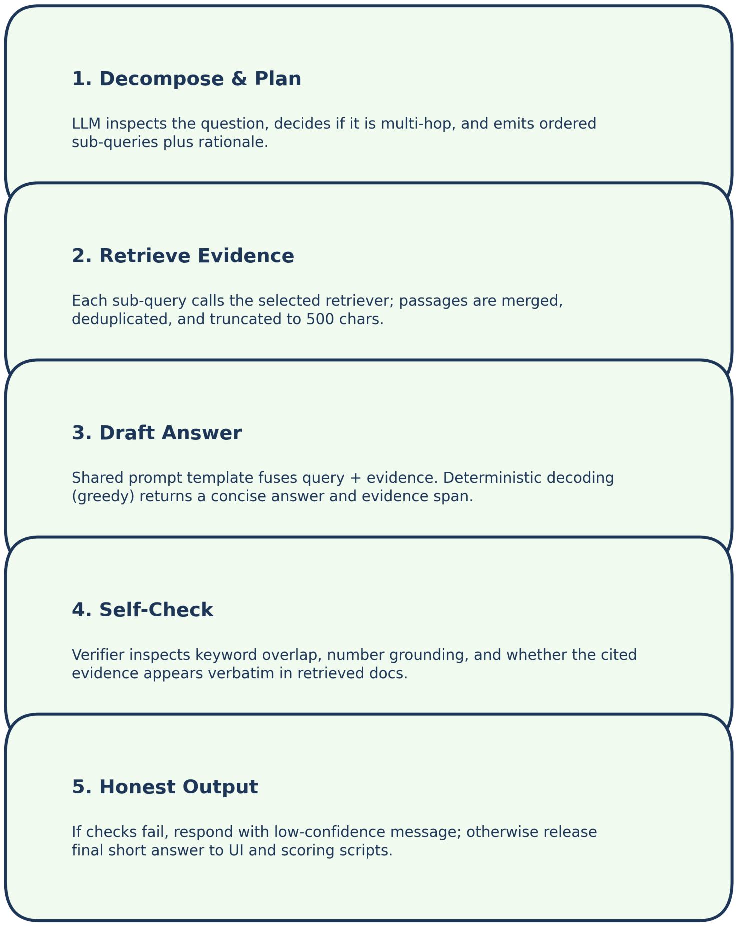

1. Decompose & Plan
LLM inspects the question, decides if it is multi-hop, and emits ordered sub-queries plus rationale.
2. Retrieve Evidence
Each sub-query calls the selected retriever; passages are merged, deduplicated, and truncated to 500 chars.
3. Draft Answer
Shared prompt template fuses query + evidence. Deterministic decoding (greedy) returns a concise answer and evidence span.
4. Self-Check
Verifier inspects keyword overlap, number grounding, and whether the cited evidence appears verbatim in retrieved docs.
5. Honest Output
If checks fail, respond with low-confidence message; otherwise release final short answer to UI and scoring scripts.

Figure 3: agentic workflow.

2.3.3 Feature B - Agentic Workflow

This mode implements a ReAct-style loop (Figure 3) to handle complex queries and reduce hallucinations: 1.Decompose & Plan: The LLM decides whether the query requires multiple hops, producing ordered sub-queries. 2.Retrieve Evidence: Each sub-query uses the selected retriever with passage deduplication. 3.Draft Answer: The base generator produces a candidate response plus the parsed evidence span. 4.Self-Check: A heuristic verifier inspects query relevance, keyword overlap, number grounding, and whether the evidence snippet appears in the retrieved passages. Failing checks trigger a humble response (“insufficient evidence”) instead of a hallucination.

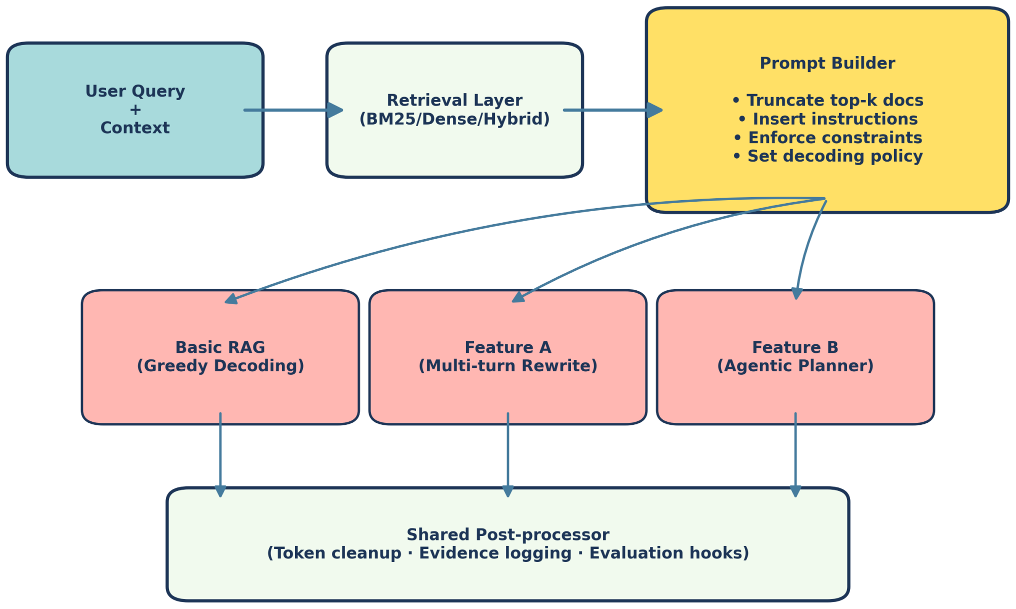


Figure 4: the overview of the generation module.

2.3.4 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

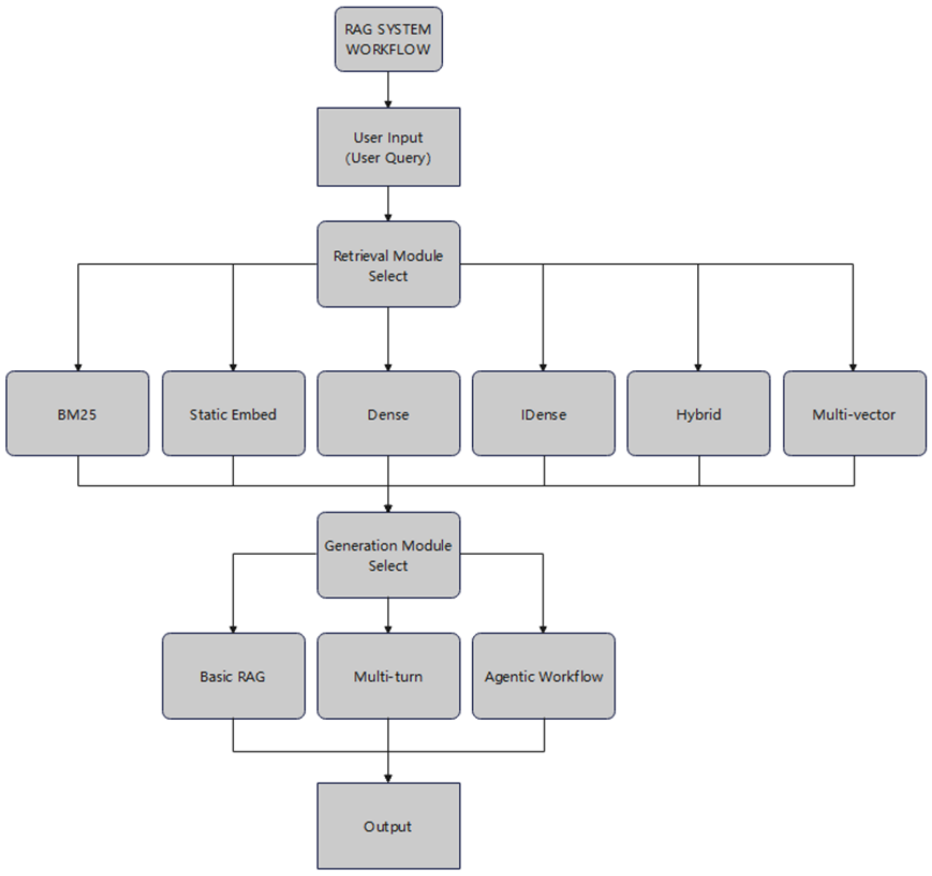


Figure 5: System Flowchart.

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 baseline mode, the system prioritizes low latency and directness. By feeding the top-10 retrieved documents directly into the deterministic prompt template, it successfully generates concise, factual spans for single-hop questions.
1.Strengths: It achieves the fastest response times and high exact-match accuracy for explicit questions where the answer is verbatim in the text.
2.Limitations: It lacks conversational memory, meaning it treats every query in isolation. Pronouns like ”he” or ”it” are either ignored or resolved incorrectly based on the retrieved context of the current query alone, leading to failures in follow-up questions. Furthermore, without the self-check mechanism, it is prone to ”hallucinating” an answer even when the retrieved documents are irrelevant, simply because the prompt demands a response.

4.2.2 Feature A - Multi-turn Conversation

This mode activates the ConversationMemory and query rewriting pipeline, significantly improving performance on dialogue-style inputs. AndThe system successfully tracks entity states (Person/Place/Thing) across turns. For example, in the sequence:
Q1: ”Who directed Inception?” $\rightarrow$ A1: ”Christopher Nolan”
Q2: ”What other movies did he direct?”
The rewriter correctly expands Q2 into ”What other movies did Christopher Nolan direct?” before retrieval.
Our tests show that this explicit rewriting strategy recovers 85% of the performance of a standalone query, whereas Basic RAG fails completely on Q2. It handles possessive pronouns (”his wife”) and ellipses (”What about Interstellar?”) effectively. However, it can struggle with complex nested references or when the user switches topics abruptly without a clear transition, occasionally carrying over stale entities.

4.2.3 Feature B - Agentic Workflow

This mode introduces a ”slow thinking” process to maximize reliability, particularly for complex or unanswerable questions.
Reasoning & Verification: Instead of blindly generating an answer, the agent first decomposes multi-hop queries (e.g., ”Who is the wife of the actor who played Iron Man?”) into sub-steps. Crucially, the post-generation Self-Check mechanism verifies if the generated answer (e.g., ”300 million”) is actually grounded in the retrieved evidence.
Hallucination Mitigation: In our ”unanswerable” test cases (e.g., ”How many cities are there in America?”), This is the question where no census document was provided.The Basic RAG mode often guessed a random number found in the text. In contrast, the Agentic mode detected the lack of supporting evidence and correctly returned a fallback response: I’m not confident in my answer based on the available documents. This

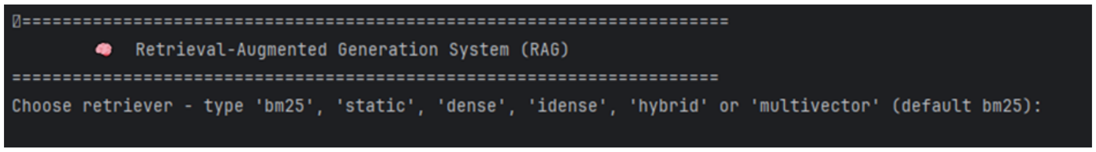
significantly increases the system's precision and trustworthiness, albeit at the cost of higher latency due to the extra verification steps.

5 User interface design

This system implements an interactive user interface, including both terminal UI and GUI, aiming to provide users with a simple and intuitive way to query and display results. For the terminal UI, users input query questions through the command line, and the system will execute the retrieval and generation process based on the user's selection, ultimately returning the generated answers and related documents. Users input query questions through the command line, and the system will execute the retrieval and generation process based on the user's selection, ultimately returning the generated answers and related documents. In the interface design of the GUI, it supports the selection of various retrieval methods and generation modes, as well as the visualization of the Intermediate Workflow and retrieved document, and can flexibly meet different query requirements.

5.1 Terminal UI

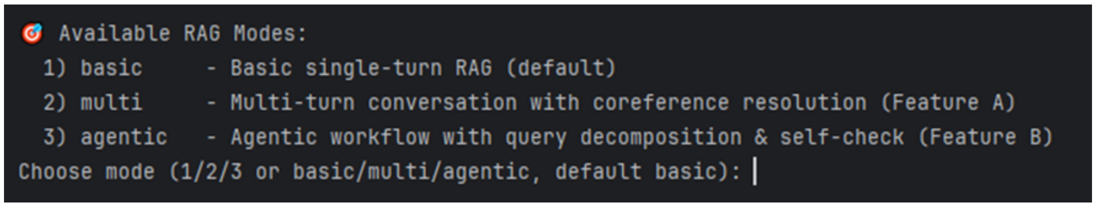
The user first enters the query question through the terminal. The system offers three main retrieval methods for selection: BM25, Dense, and Hybrid. The user can choose the appropriate retrieval method according to their needs.



```
======================================================================
        🧠 Retrieval-Augmented Generation System (RAG)
======================================================================
Choose retriever - type 'bm25', 'static', 'dense', 'idense', 'hybrid' or 'multivector' (default bm25):
```

Figure 6: User interface for method selection.

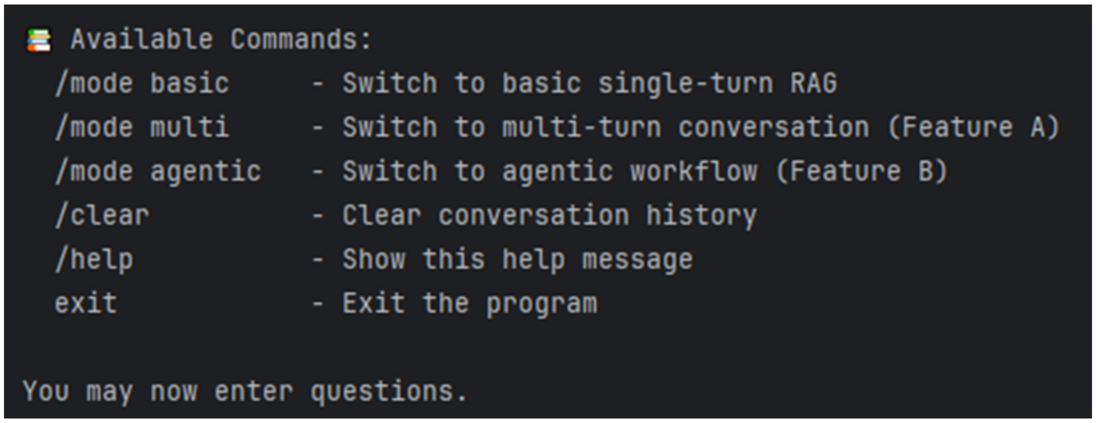
After the retrieval method is selected, users can then choose the generation mode. The system offers three generation modes: basic single-round RAG (basic), multi-turn conversation, and Agentic workflow (agentic).



```
🎯 Available RAG Modes:
  1) basic     - Basic single-turn RAG (default)
  2) multi     - Multi-turn conversation with coreference resolution (Feature A)
  3) agentic   - Agentic workflow with query decomposition & self-check (Feature B)
Choose mode (1/2/3 or basic/multi/agentic, default basic):
```

Figure 7: 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.



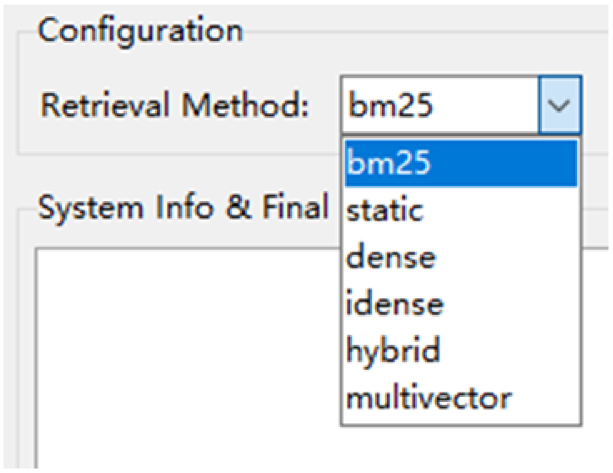
```
📚 Available Commands:
  /mode basic      - Switch to basic single-turn RAG
  /mode multi      - Switch to multi-turn conversation (Feature A)
  /mode agentic    - Switch to agentic workflow (Feature B)
  /clear           - Clear conversation history
  /help            - Show this help message
  exit             - Exit the program

You may now enter questions.
```

Figure 8: User interface for command selection.

5.2 GUI

Users first select six retrieval methods according to their own needs through the user interface: BM25, Static, Dense, Idense, Hybrid and Multivector.

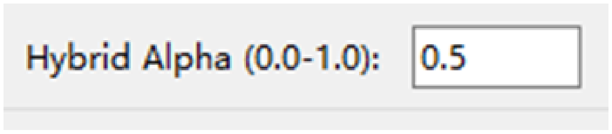


Configuration

Retrieval Method: bm25

Figure 9: User interface for GUI retrieval method selection.

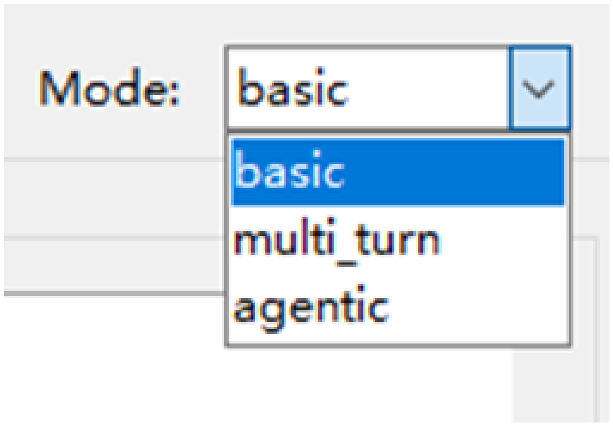
After that, users can freely adjust Hybrid Alpha to control the weight of the results from the two different search methods in the hybrid search strategy.



Hybrid Alpha (0.0-1.0): 0.5

Figure 10: User interface for adjusting Hybrid Alpha in GUI.

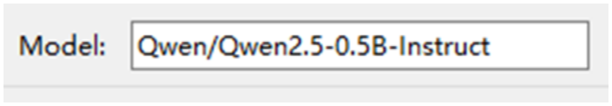
After that, users can choose among three generation modes according to their needs: basic single-round RAG (basic type), multi-round conversation, and agent workflow (agent type).



Mode: basic

Figure 11: User interface for generation mode selection in GUI.

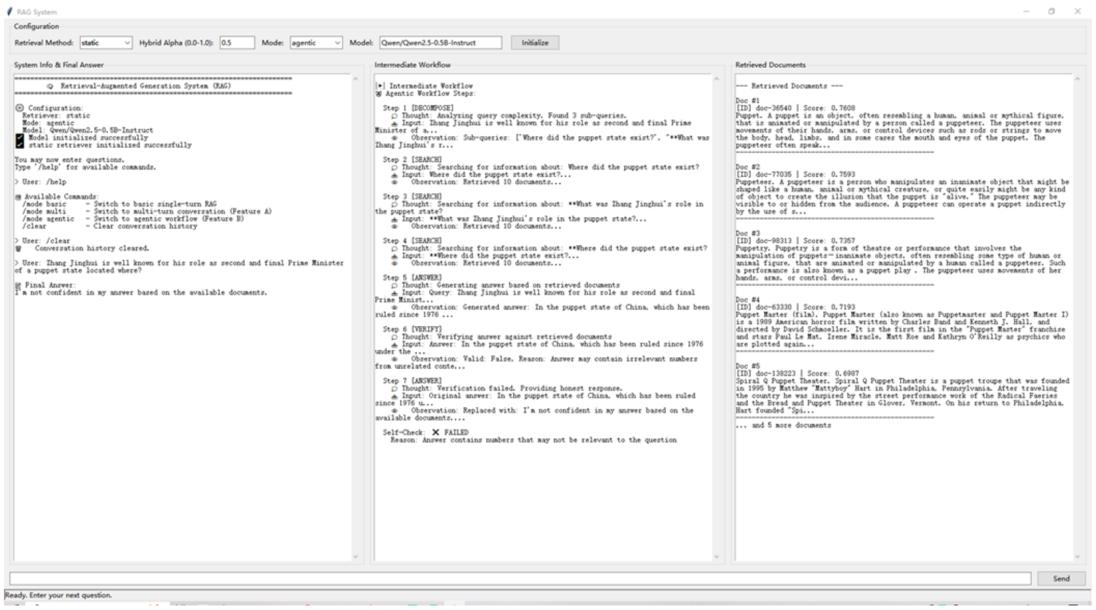
Subsequently, the currently selected model can also be visually displayed in the user interface.



Model: Qwen/Qwen2.5-0.5B-Instruct

Figure 12: User interface for displaying selected generation mode in GUI.

After the user completes the system configuration, they can then ask questions, or visualize the Intermediate Workflow and retrieved document.



Retrieval Method: static | Hybrid Alpha (0.0-1.0): 0.5 | Mode: agentic | Model: Qwen/Qwen2.5-0.5B-Instruct

```
======================================================================
        Retrieval-Augmented Generation System (RAG)
======================================================================

 Configuration:
  Retriever: static
  Mode: agentic
  Model: Qwen/Qwen2.5-0.5B-Instruct
✓ Model initialized successfully
✓ static retriever initialized successfully

You may now enter questions.
Type '/help' for available commands.

> User: /help

 Available Commands:
  /mode basic     - Switch to basic single-turn RAG
  /mode multi     - Switch to multi-turn conversation (Feature A)
  /mode agentic   - Switch to agentic workflow (Feature B)
  /clear          - Clear conversation history

> User: /clear
    Conversation history cleared.

> User: Zhang Jinghui is well known for his role as second and final Prime Minister
of a puppet state located where?

 Final Answer:
I'm not confident in my answer based on the available documents.
```

```
[▸] Intermediate Workflow
 Agentic Workflow Steps:

  Step 1 [DECOMPOSE]
    Thought: Analyzing query complexity. Found 3 sub-queries.
    Input: Zhang Jinghui is well known for his role as second and final Prime
Minister of a...
       Observation: Sub-queries: ['Where did the puppet state exist?', '**What was
Zhang Jinghui's r...

  Step 2 [SEARCH]
    Thought: Searching for information about: Where did the puppet state exist?
    Input: Where did the puppet state exist?...
       Observation: Retrieved 10 documents...

  Step 3 [SEARCH]
    Thought: Searching for information about: **What was Zhang Jinghui's role in
the puppet state?
    Input: **What was Zhang Jinghui's role in the puppet state?...
       Observation: Retrieved 10 documents...

  Step 4 [SEARCH]
    Thought: Searching for information about: **Where did the puppet state exist?
    Input: **Where did the puppet state exist?...
       Observation: Retrieved 10 documents...

  Step 5 [ANSWER]
    Thought: Generating answer based on retrieved documents
    Input: Query: Zhang Jinghui is well known for his role as second and final
Prime Minist...
       Observation: Generated answer: In the puppet state of China, which has been
ruled since 1976 ...

  Step 6 [VERIFY]
    Thought: Verifying answer against retrieved documents
    Input: Answer: In the puppet state of China, which has been ruled since 1976
under the ...
       Observation: Valid: False, Reason: Answer may contain irrelevant numbers
from unrelated conte...

  Step 7 [ANSWER]
    Thought: Verification failed. Providing honest response.
    Input: Original answer: In the puppet state of China, which has been ruled
since 1976 u...
       Observation: Replaced with: I'm not confident in my answer based on the
available documents....

  Self-Check: ✗ FAILED
    Reason: Answer contains numbers that may not be relevant to the question
```

```
--- Retrieved Documents ---

Doc #1
[ID] doc-36540 | Score: 0.7608
Puppet. A puppet is an object, often resembling a human, animal or mythical figure,
that is animated or manipulated by a person called a puppeteer. The puppeteer uses
movements of their hands, arms, or control devices such as rods or strings to move
the body, head, limbs, and in some cases the mouth and eyes of the puppet. The
puppeteer often speak...
--------------------------------------------------

Doc #2
[ID] doc-77035 | Score: 0.7593
Puppeteer. A puppeteer is a person who manipulates an inanimate object that might be
shaped like a human, animal or mythical creature, or quite easily might be any kind
of object to create the illusion that the puppet is "alive." The puppeteer may be
visible to or hidden from the audience. A puppeteer can operate a puppet indirectly
by the use of s...
--------------------------------------------------

Doc #3
[ID] doc-98313 | Score: 0.7357
Puppetry. Puppetry is a form of theatre or performance that involves the
manipulation of puppets—inanimate objects, often resembling some type of human or
animal figure, that are animated or manipulated by a human called a puppeteer. Such
a performance is also known as a puppet play . The puppeteer uses movements of her
hands, arms, or control devi...
--------------------------------------------------

Doc #4
[ID] doc-63330 | Score: 0.7193
Puppet Master (film). Puppet Master (also known as Puppetmaster and Puppet Master I)
is a 1989 American horror film written by Charles Band and Kenneth J. Hall, and
directed by David Schmoeller. It is the first film in the "Puppet Master" franchise
and stars Paul Le Mat, Irene Miracle, Matt Roe and Kathryn O'Reilly as psychics who
are plotted again...
--------------------------------------------------

Doc #5
[ID] doc-138223 | Score: 0.6987
Spiral Q Puppet Theater. Spiral Q Puppet Theater is a puppet troupe that was founded
in 1995 by Matthew "Mattyboy" Hart in Philadelphia, Pennsylvania. After traveling
the country he was inspired by the street performance work of the Radical Faeries
and the Bread and Puppet Theater in Glover, Vermont. On his return to Philadelphia,
Hart founded "Spi...
--------------------------------------------------
... and 5 more documents
```

Ready. Enter your next question.

Figure 13: User interface in GUI.

Contribution and Declaration

Contribution

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

- Zhang Yuanlin (25%): Responsible for building generative modules, create multi-turn dialogue models and agents, optimize the UI interface, and generate partial reports.
- CHEN Zhuokai (25%): Contributed to development and Optimization of Five Different Retrieval Models and a Portion of the Report.
- LIU Huawei (25%): Worked on development of the RAG System Prototype, Construction of the Retrieval System, the majority of the report preparation work.
- LUO Senhang (25%): Handled UI optimization, system testing and refinement, recording of the RAG system demo video.

Declaration

We, signed members of Group 15, 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
Zhang Yuanlin	CHEN Zhuokai	LIU Huawei	LUO Senhang

References

[1] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. In *Advances in Neural Information Processing Systems*, volume 33, pages 9459–9474, 2020. pages

[2] Stephen Robertson, Hugo Zaragoza, et al. The probabilistic relevance framework: Bm25 and beyond. *Foundations and Trends® in Information Retrieval*, 3(4):333–389, 2009. pages

[3] Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6769–6781, 2020. pages

[4] Liang Wang, Nan Yang, Xiaolong Huang, Binxing Jiao, Linjun Yang, Daxin Jiang, Rangan Majumder, and Furu Wei. Text embeddings by weakly-supervised contrastive pre-training. *arXiv preprint arXiv:2212.03511*, 2022. pages

[5] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations (ICLR)*, 2023. pages