

Agentic Multi-Stage Retrieval-Augmented Generation System with Hybrid Retrieval and Self-Evaluation

Mayank Patoliya

Student, Dept of Artificial Intelligence Engineering, Shri Shankaracharya Institute Of Professional Management & Technology, Chhattisgarh, India

Abstract- Retrieval-Augmented Generation (RAG) improves the factual accuracy of Large Language Models by incorporating external knowledge during inference. However, traditional RAG systems suffer from limitations such as poor retrieval quality, redundancy, and hallucinated responses[14]. This paper proposes a Hybrid RAG framework that combines sparse retrieval (BM25) and dense retrieval (FAISS embeddings) using Reciprocal Rank Fusion (RRF)[2], followed by Cross-Encoder reranking for enhanced semantic relevance. Maximal Marginal Relevance (MMR) is applied to ensure diversity among retrieved documents. Additionally, a confidence scoring mechanism[22] and failure detection module are introduced to improve reliability and identify insufficient information scenarios. Experimental evaluation across multiple query types[2] demonstrates that the proposed system achieves improved retrieval performance (MRR = 0.8056) while maintaining balanced generation quality. The results highlight trade-offs between accuracy, latency, and reasoning complexity, contributing toward the development of reliable and scalable RAG systems for real-world applications[1].

Key Words: Retrieval-Augmented Generation (RAG), Agentic AI, Multi-Hop Reasoning, Hybrid Retrieval, Large Language Models (LLMs), Cross-Encoder Re-ranking

1. INTRODUCTION

Large Language Models (LLMs) have significantly advanced natural language processing tasks[17] such as question answering, summarization, and conversational AI[10]. Despite their capabilities, these models rely on static training data[8] and often generate hallucinated responses[10] when confronted with unfamiliar queries[2]. Retrieval-Augmented Generation (RAG) addresses this limitation[1] by integrating external knowledge retrieval mechanisms[3] into the generation pipeline[4], allowing models to ground their responses in retrieved evidence.

However, conventional RAG systems face several challenges. Sparse retrieval methods such as BM25 rely on lexical matching and fail to capture semantic similarity[15], while dense retrieval methods rely on embeddings and may overlook exact keyword matches. Additionally, traditional pipelines lack reranking[15], diversity optimization[6], and mechanisms for evaluating response [1].

To address these issues, this work proposes a Hybrid RAG system[1] that integrates sparse and dense retrieval using Reciprocal Rank Fusion (RRF). Cross-Encoder reranking is used to refine relevance[17], while Maximum Marginal Relevance (MMR) ensures diversity. A confidence scoring mechanism and failure detection module[23] are introduced to enhance reliability[7].

2. Related Work

The idea of using information to help Large Language Models is not new[1]. For a time, people have been working on ways to combine the knowledge that a model has with information from outside sources[4]. One way to do this is to use a method called retrieval which finds information based on keywords. Another way is to use a method called retrieval, which finds information based on the meaning of the text.

Recently, people have been working on ways to combine these two methods[1][2]. One way to do this is to use a method called Reciprocal Rank Fusion, which combines the results of searches. Another way is to use a method called Cross-Encoder, which checks if the results are relevant to the question. We are going to use these methods in our system.

3. Mathematical Formulation

To make our system work we need to use some formulas. These formulas help us combine the results of searches, check if the results are relevant and detect when we do not have enough information.

3.1. Reciprocal Rank Fusion (RRF)

First, we use a formula called Reciprocal Rank Fusion to combine the results of searches. This formula takes the results of each search. Combine them into one list. The list is ranked, so the relevant results are at the top.

$$RRF(d) = \sum_{i=1}^n \frac{1}{k + rank_i(d)}$$

3.2. Final Hybrid Score

The final scoring function combines the retrieval score with the Cross-Encoder score. The Cross-Encoder checks how well a query and document match by encoding them. This gives an idea of how relevant they are. The parameters alpha and beta decide how much each part, retrieval, and semantic re-ranking contribute to the score. This scoring system makes sure that both how well a document is retrieved and how semantically relevant it is are considered when selecting documents.

$$Score(d) = \alpha \cdot RRF(d) + \beta \cdot CE(d, q)$$

3.3. Confidence Score

The confidence score shows how reliable the generated response is. It does this by averaging the scores of the documents that were used to create the response. The system also uses a function to make sure the confidence score is lower when the system is not being truthful or when it has to use a default response. This way the system can tell users how much they can trust what it says. The confidence score is very important because it helps users know if they can believe the generated response. The system uses the confidence score to give users an idea of how reliable the responses.

$$Confidence = \frac{1}{k} \sum_{i=1}^k Score(d)_i \cdot 1_{faithful}$$

3.4. Maximal Marginal Relevance (MMR)

The MMR formulation is a way to balance the relevance and differentness of the documents. It picks documents that're very relevant to the query and not similar to the documents that were picked before. The parameter λ is what controls this balance. The system can decide how important it is to have different documents compared to how relevant they are to the query. This helps to stop the system from picking the kind of documents over and over which makes the responses that are generated a lot better. The MMR formulation does this by making sure the documents are not too similar to each other. The parameter λ is important because it helps the system find the balance between relevance[16] and diversity. This balance is what makes the MMR formulation so useful for generating responses. The MMR formulation and the

parameter λ work together to make sure the documents that are picked are both relevant to the query and different, from each other.

$$MMR = \arg \max_{d \in D} [\lambda Sim(d, q) - (1 - \lambda) maxSim(d, \{d\})]$$

3.5. Failure Rate

The failure detection metric[14] looks at how the system can find queries that it cannot answer because it does not have enough information. If the system has a high failure detection rate that means it is good at knowing what it cannot do. It gives a good response when it cannot answer something. The failure detection metric is important because it helps the system avoid giving answers. The system uses the failure detection to know when it should say that it does not know something. The failure detection metric is a part of the system and it helps the system to be better, at the failure detection.

$$Failure Rate = \frac{Correct\ Insufficient\ Info\ Responses}{Total\ Failure\ Queries}$$

4. Dataset and Preprocessing

To test our system we need a lot of text data. We use a dataset that has different types of text, including articles and books. We preprocess the data by removing any characters and splitting it into smaller chunks[6]. This makes it easier for our system to find the information it needs[11].

We hope that our system will be able to give accurate answers, than other systems. We also hope that it will be able to detect when it does not have information to give a good answer. In the section we will talk about how we tested our system and what the results were.

Each document is broken down into pieces and then processed in two separate ways at the same time. For the way the system uses something called BM25 indexing to figure out how relevant the document is based on how often certain words appear and how rare they are in other documents. For the way the system uses a special kind of model to create embeddings, which are like maps of what the document is about and stores them in a special index called FAISS that makes it easy to find similar documents. The system also keeps track of things like where the document came from and what it's about so that it can provide more accurate and useful results. This two-part approach lets the system use both the words in the document and what they mean to find the results.

5. Methodology

The method we are using has a lot of steps to make sure we get the information[2]. First we take the user's

question. Get it ready for the system to look at. We then send this question to two systems that find information[9]: one system looks for keywords, and the other system looks for things that mean the same thing[2].

The system that looks for keywords finds documents that have the words as the question. The other system finds documents that're similar in meaning even if they do not have the same words. We then combine the results from both systems[2], which makes sure that the documents that both systems think are important get a priority.

We then use another system to look at the question and the documents together[15] which helps us figure out how relevant each document is to the question[3]. This system is better at understanding what the question means than systems.

After that we pick documents that are not only relevant but also different from each other. This makes sure that we do not get the information over and over again[5]. We then use these documents to create the answer[20].

At the time we are also checking how sure we are about the answer. If we are not sure the system will say so instead of making something up[3]. This makes sure that the information we give is not relevant but also trustworthy. The method is designed to make sure that the information retrieval system gets the information and is also reliable[1].

We use the information retrieval system to get the documents and the language model to create the final answer. The information retrieval system has a lot of parts[7] including the keyword system and the system that looks for meanings. The language model uses the documents to create the answer. This is how the method works to get the information.

6. Result and Analysis

6.1. Retrieval Performance

The Hybrid RAG system does really well when it comes to finding the information[22]. It gets the Mean Reciprocal Rank with a score of 0.8056. This is better than the Basic RAG system which has a score of 0.7931 and the Agentic RAG system which has a score of 0.7533. The Hybrid RAG system is good because it combines two ways of searching for information. It looks at the words and what they mean.

The Basic RAG system is a little better at picking the documents. It gets a Precision@5 score of 0.2867.. The Hybrid RAG system is still better at ranking all documents[11]. The Hybrid RAG system is good at finding the information because it uses both the Hybrid RAG system method and the Hybrid RAG system way of

searching. The Hybrid RAG system is just better than the Basic RAG system and the Agentic RAG system.

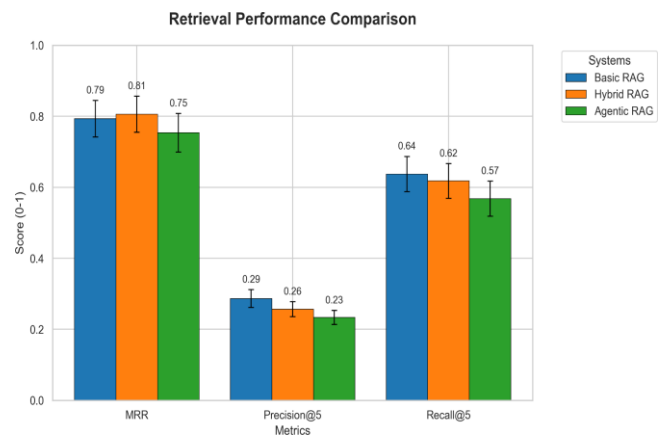


Chart -1: Retrieval Performance Comparison

6.2. Generation Quality

The results of how things were made show that Agentic RAG makes the most complete and relevant answers[13]. It got scores of 0.5655 for being complete and 0.565 for being relevant.. Basic RAG did the best when it came to being faithful to what it found. It got a score of 0.455 which means its answers were more based on what it found. Hybrid RAG did a job with all the measures. This shows that using ways to find information helps make better answers without losing too much faithfulness to what was found. Rag and Basic RAG and Hybrid RAG all have their good points. Agentic RAG makes answers and relevant answers. Basic RAG is good at being faithful to what it found. Hybrid RAG does a bit of everything.

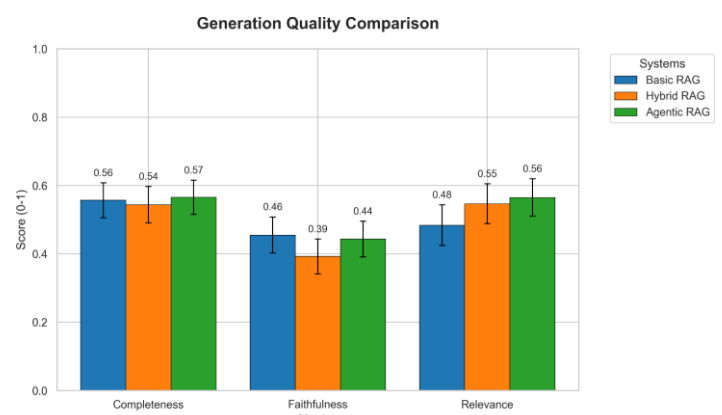


Chart -2: Generation Quality Comparison

6.3. Failure Detection

The Basic RAG system is really good at finding failures; it gets it most of the time about 0.9167. The Hybrid RAG system is next it is good too but not as good as Basic RAG it gets it right 0.85 of the time. The Agentic RAG system is not as good at finding failures; it only gets it right about 0.6833 of the time.

This shows that the simpler systems, like Basic RAG and Hybrid RAG are better at finding out when something is missing.

But when it comes to failures the Agentic RAG system is actually really good. It can find all of them which is great when you have to think about really hard problems, like complex reasoning scenarios and the Agentic RAG system can handle them.

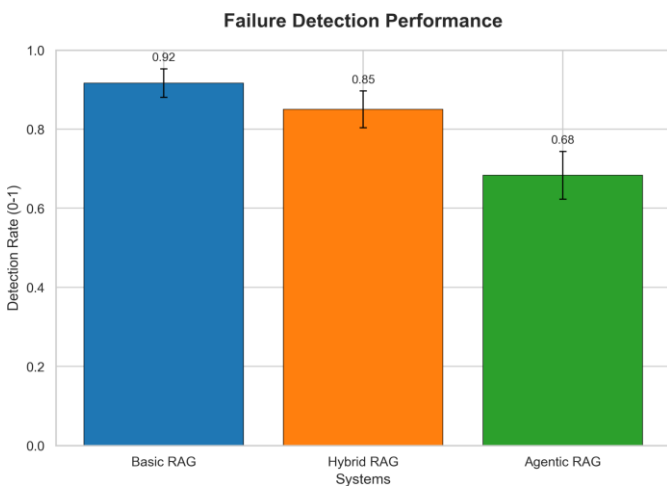


Chart -3: Failure Detection Performance

6.4. Latency Comparison

The Basic RAG system is really fast. It takes 9.7356 seconds to respond. The Hybrid RAG system is a bit slower. It takes around 12.8824 seconds[18] to respond because it has to do some work like fusion and reranking.

The Agentic RAG system is really slow. It takes 65.5583 seconds to respond. This is because the Agentic RAG system has to do a lot of thinking and it has to do it in steps. The Basic RAG system is the fastest. The Hybrid RAG system is, in the middle. The Agentic RAG system is the slowest.

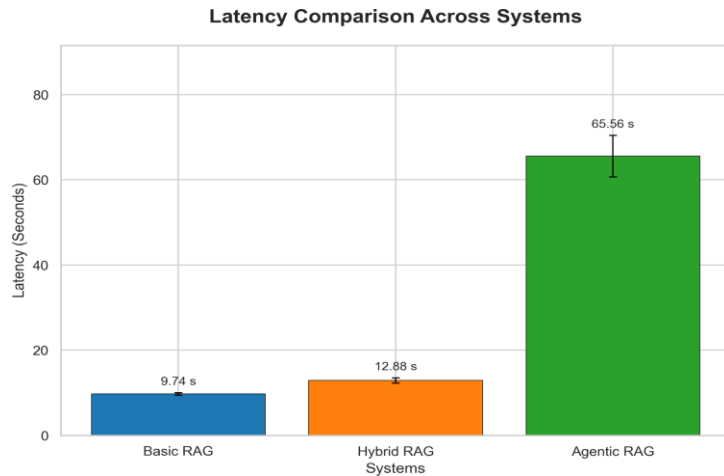


Chart -4: Latency Comparison

6.4. Overall Comparison

The results really show the bad points of each system. Hybrid RAG is the best at finding the information. Basic RAG is great when it comes to being accurate and fast. On the hand Agentic RAG is very good, at reasoning but it takes longer to get the results. Hybrid RAG and Basic RAG and Agentic RAG all have their strengths and weaknesses.

Metric	Basic	Hybrid	Agentic
MRR	0.7931	0.8056	0.7533
Precision@5	0.2867	0.2567	0.2333
Recall@5	0.6372	0.6178	0.5678
Faithfulness	0.4550	0.3922	0.4433
Relevance	0.4838	0.5467	0.5650
Completeness	0.5567	0.5438	0.56555
Latency	9.73	12.88	65.56
Failure Detection	0.9167	0.8500	0.6833

Table: Overall Comparison

7. Challenges

This work had a problem. We had to make sure that the system could find what we were looking for and do it quickly. When we combined systems to find things and

then rearranged the results to get the best ones and make them different it took longer and got more complicated[8]. The system also had to be able to tell when it was giving answers. We had to try things to get it just right like figuring out the best numbers to use for things, like RRF constants and MMR lambda values and reranking weights.

8. Applications

The new system can be used in different areas where we need to trust the artificial intelligence and understand how it works[4]. We can use this system in schools to help with research in companies to keep track of what they know and in computer programs that talk to customers[7]. It is also useful in systems that help lawyers and doctors find the information because in these areas it is very important to be accurate and trustworthy. The system is really useful when we need to be sure that the artificial intelligence is working correctly like in the systems that lawyers and doctors use[9] to find information.

9. Limitations

The system has some limitations. It really needs quality data to work properly. The Cross-Encoder reranking method uses a lot of computer power, which means it is not very good, for applications that need to work in time. The system also uses a confidence scoring mechanism that is based on guesses[12]. This may not be able to find all the subtle cases of hallucination detection[16]. The hallucination detection is something that the system is supposed to do. It is not always able to do it well because of these limitations. The system and the hallucination detection are closely. The limitations of the system can affect the hallucination detection[19].

10. Future Work

The future work will be, about making the system work faster. This can be done by using model distillation and other methods to get the information we need quickly[11]. We also want to use reinforcement learning to help the system[7] decide what information is important[6]. This will help the system get better at finding what we are looking for. We also need to work on stopping the system from giving us information. This can be done by using verification models to check the information.

Another thing we can do is make the system work with types of data like pictures and videos. This is something we can look into in the future.

11. Conclusion

This paper is about a Hybrid RAG framework. The Hybrid RAG framework does a lot of things. It combines retrieval

fusion and semantic reranking. It also does diversity optimization and confidence-based reliability estimation. The system is really good at finding the information and creating text that is balanced and makes sense. It also helps to stop hallucinations[13].

The results show that using an approach is very important for building strong and trustworthy RAG systems. This is especially true for real-world applications where you need to be accurate and reliable[1]. The Hybrid RAG framework is a help in these situations because it uses a hybrid approach. The Hybrid RAG framework is very useful for building RAG systems that're robust and trustworthy.

REFERENCES

- [1] Mishra, Saroj, et al. "Sok: Agentic retrieval-augmented generation (rag): Taxonomy, architectures, evaluation, and research directions." arXiv preprint arXiv:2603.07379 (2026).
- [2] Cento, Ramiro López. "Agentic Retrieval Augmented Generation for Estimation of Distribution Algorithms." Algorithms (2025).
- [3] Wei, Tianxin, et al. "Agentic reasoning for large language models." arXiv preprint arXiv:2601.12538 (2026).
- [4] Cheng, Mingyue, et al. "A survey on knowledge-oriented retrieval-augmented generation." arXiv preprint arXiv:2503.10677 (2025)..
- [5] Aggarwal, Vedanth. "Empowering Large Language Model Reasoning: Hybridizing Layered Retrieval Augmented Generation and Knowledge Graph Synthesis." Int. J. High Sch. Res 6 (2024): 80-92.
- [6] e Aquino, Gustavo de Aquino, et al. "From rag to multi-agent systems: A survey of modern approaches in llm development." (2025).
- [7] Gong, Ming. Toward A Self-Evolving Agent In Multi-Turn Dialogue Question-Answering Systems. Diss. University of Dayton, 2025.
- [8] Gao, Huan-ang, et al. "A Survey of Self-Evolving Agents: What, When, How, and Where to Evolve on the Path to Artificial Super Intelligence." arXiv preprint arXiv:2507.21046 (2025).
- [9] Gautam, Aditya. "Multi-agent systems for misinformation lifecycle: Detection, correction and source identification." arXiv preprint arXiv:2505.17511 (2025).
- [10] Daull, Xavier, et al. "Complex QA and language models hybrid architectures, Survey." arXiv preprint arXiv:2302.09051 (2023).
- [11] Lensu, Urho. "Impact of Chunking Granularity on Accuracy and Token Consumption in Retrieval-Augmented Generation for Question-Answering." (2025).

- [12] Raza, Shaina, et al. "Responsible Agentic Reasoning and AI Agents: A Critical Survey." Authorea Preprints (2025).
- [13] Baibakova, Viktoriia, and Alexey Serov. "Agentic framework for programmatic crystal structure generation using a fine-tuned worker-supervisor large language model." *Energy and AI* (2026): 100710.
- [14] Wegener, Gregor. "SORT-AI: A Structural Safety and Reliability Framework for Advanced AI Systems with Retrieval-Augmented Generation as a Diagnostic Testbed." (2025).
- [15] Singh, Divyansh. "Bridging Knowledge and Generation: A Survey on Retrieval-Augmented Generation."
- [16] Li, Xiaopeng, et al. "A survey of personalization: From rag to agent." *ACM Transactions on Information Systems* (2025).
- [17] Gao, Yunfan, et al. "Synergizing rag and reasoning: A systematic review." *arXiv preprint arXiv:2504.15909* (2025).
- [18] Wang, Zhichao, Cheng Wan, and Dong Nie. "Review of Inference-Time Scaling Strategies: Reasoning, Search and RAG." *arXiv preprint arXiv:2510.10787* (2025).
- [19] Wang, Ziqi, et al. "A survey on parallel reasoning." *arXiv preprint arXiv:2510.12164* (2025).
- [20] Aggarwal, Vedanth. "Empowering Large Language Model Reasoning: Hybridizing Layered Retrieval Augmented Generation and Knowledge Graph Synthesis." *Int. J. High Sch. Res* 6 (2024): 80-92.
- [21] Tang, Fei, et al. "A survey on (m) llm-based gui agents." *arXiv preprint arXiv:2504.13865* (2025).
- [22] Ioannis, et al. "From illusion to insight: A taxonomic survey of hallucination mitigation techniques in LLMs." *AI* 6.10 (2025): 260. Cento, Ramiro López. "Agentic Retrieval Augmented Generation for Estimation of Distribution Algorithms." *Algorithms* (2025).
- [23] Wei, Tianxin, et al. "Agentic reasoning for large language models." *arXiv preprint arXiv:2601.12538* (2026).
- [24] Baig, Afnan, and D. Sc Mikko Raatikainen. "Autonomous LLM Monitoring and Remediation Framework." (2025)