

# Component-Level Evaluation of Cascaded Retrieval and Reranking in RAG: Ablations Across Financial, Biomedical, and Medical Domains

R.Mabubasha<sup>1</sup>, Navuluri Sindhu<sup>2</sup>, MuppalaVenkata Pujitha<sup>3</sup>, Jonnalagadda Abhishek Vardhan<sup>4</sup>

<sup>1</sup> Assistant Professor, Department of CSE, RVR & JC College of Engineering, Chowdavaram, Guntur, A.P, India.

<sup>2,3,4</sup> B. Tech Students, Department of CSE, RVR & JC College of Engineering, Chowdavaram, Guntur, A.P, India.

\*\*\*

**Abstract** - Multi-stage reranking has been applied in RAG models across a number of studies, but what's missing is any serious accounting of how much each component actually contributes. The question of weight assignment methods is similarly underexplored. This study takes a closer look at retrieval across multiple stages, evaluating performance on three BEIR benchmark datasets FiQA, SciFact, and NFCorpus. It examines what tri-modal hybrid search does to performance scores and compares the effect of cross-encoder and LLM reranking modules directly. The ablation experiments tell a clear story: cross-encoder reranking consistently beats the baseline before it. LLM reranking is a different matter it's genuinely useful for financial question answering but loses its edge on biomedical data. The domain dependency is hard to ignore. On top of that, LLM reranking adds latency without matching the performance of well-tuned static weights. Perhaps most surprisingly, simpler retrieval pipelines outperform tri-modal retrieval outright more complexity, worse results.

**Keywords:** Hybrid Retrieval, Cross-Encoder Reranking, Reciprocal Rank Fusion (RRF), Large Language Model Reranking, Retrieval Augmented Generation (RAG) Dynamic Weight Assignment, Agentic AI, BEIR Benchmarks

## 1. INTRODUCTION

Retrieval-Augmented Generation (RAG) has become the standard approach for building AI systems that need to stay factually grounded. Rather than relying on what the model memorized during training, RAG systems retrieve relevant documents at query time and use them to generate answers which cuts hallucination rates substantially. That matters especially in finance, medicine, and law, where a confidently wrong output can cause real harm [16, 19, 27, 32]. The catch is that answer quality is now hostage to retrieval quality. A broken retrieval stage poisons everything downstream, and as RAG systems move from research into production, a question the literature has been surprisingly slow to answer is: which parts of the pipeline actually pull their weight?

A standard RAG pipeline runs in three stages. The retriever identifies candidate documents using similarity-based signals. A reranker then reorders those candidates using a more precise model. Finally, the language model

reads the selected documents and generates a response. For retrieval, three main approaches are used in practice dense retrieval with neural embeddings for semantic matching [4], sparse retrieval with lexical methods like BM25 for exact term matching [1, 3, 31], and graph-based retrieval that exploits entity relationships for structurally complex queries [15, 23, 26, 30]. These are combined through fusion methods, with Reciprocal Rank Fusion (RRF) being the most widely used [1, 3, 31]. Reranking is handled predominantly by cross-encoders [4, 5], with growing interest in prompting large language models directly as zero-shot rankers [7, 21, 22].

Several questions in this area remain genuinely open. Most published work evaluates systems end-to-end, which tells you almost nothing about what individual components are doing. Whether tri-modal retrieval actually beats bi-modal is unclear, and whether an LLM reranking step adds anything meaningful over a cross-encoder remains equally unresolved [11, 12, 13, 17, 32]. Agentic RAG gets a lot of credit for letting models weigh information sources dynamically based on the query [2, 8, 9] — appealing in theory, rarely verified in practice. And the latency cost of calling an LLM mid-pipeline tends to get either understated or ignored entirely [18, 24, 27].

We ran a controlled study across three benchmarks covering genuinely different domains: FiQA, SciFact, and NFCorpus [17]. The pipeline uses tri-modal hybrid retrieval, cascaded cross-encoder and LLM reranking, Reciprocal Rank Fusion [31] for combining scores, and Claude Sonnet 4.6 via AWS Bedrock [14] for dynamic weight assignment. What we found, bluntly, is that complexity doesn't pay off. A simpler pipeline good retrieval, effective reranking beats more elaborate fusion approaches. And static retrieval weights, tuned once on a validation set, outperform LLM-based dynamic assignment on every dataset we tested, with considerably less latency.

Four things fall out of this. Cross-encoder reranking accounts for most of the performance improvement; LLM reranking only helps in some domains. Static weights beat dynamic LLM assignment consistently, and the latency gap is real. The cost of LLM reranking is disproportionate to what it actually returns in precision. And a domain-level error analysis shows where each retrieval modality tends to fall apart.

## II. RELATED WORK

In terms of advancements in RAG research, there have been three main directions that the field has followed in recent years: the creation of hybrid retrieval mechanisms based on multimodal information; the usage of cascaded reranking for progressive improvements; and finally, the capability of LLMs to govern the entire retrieval process. Each direction has given birth to notions that carry weight but pose open-ended questions at the same time, issues that will be discussed in the following sections.

### A. Hybrid Retrieval Mechanisms

An approach of fusion of modalities in information retrieval which means the strengths of a dense retrieval and a sparse retrieval are combined which are complementary to each other and have their own weakness. Retrievals that are dense fail to work on queries that contain rare words and/or very specific numbers. Retrievals that are sparse fail to capture the semantics behind any query. This is especially true in the case of paraphrases, which often just asks for different terms for similar concepts. In Blended RAG [1], it was presented that mixing neural representations with BM25 leads to a considerable gain in overall performance for mixed query types, and this is the same idea we utilize. However, we came to know that it was more complicated than we thought. A recent paper on hybrid search balancing showed that naive fusion may result in degraded performance due to not setting the modal weights incorrectly. There are unequivocally benefits of using one or both methods in tandem, and their efficacy no doubt lies in their proper tuning. However, the specifics of tuning your methods depend on the domain and query-types it is deployed for. A significant portion of the related papers simply state that hybrid search works better than its unimodal counterparts without discussing or specifying either the reason or the necessary conditions.

Knowledge graph augmentation. An area to explore that is somewhat unrelated to the previous topic is knowledge graph augmentation. This type of research aims to compensate for the weaknesses of text retrieval systems by introducing a knowledge graph of the information system that exists within the same document collection. For example, customer service documents [15], legislative documents [23], and archival documents [30] all utilize knowledge graphs to enhance retrieval capabilities.

Architecture decisions. The architectures used to augment knowledge graphs with text retrieval systems are also a topic of research. One study [26] examines various methods for integrating knowledge graphs into information retrieval systems. These methods include expanding and filtering search results using the knowledge graph, as well as using the knowledge graph to

rerank search results. Additionally, some research studies have used neural networks to learn relationships within knowledge graphs [25]. These neural networks allow for the creation of even more accurate search results but come at a cost of training complexity and inference time.

Specific domain results. There are three papers in this domain-based research area that are of particular interest to our investigation into the weaknesses of text retrieval systems. One study of an enterprise retrieval system [11] shows that different types of documents can only be retrieved using separate information retrieval systems. Another study of a clinical decision support system [12] shows that text retrieval models struggle to process the abbreviations and specific terminologies used in those documents. Finally, a study of a financial question-answering system [13] shows that text retrieval models that use sparse retrieval methods still achieve high accuracy by focusing on the exact matching of specific keywords and financial terms. The common weakness across all three papers can be identified by noting that although complex systems were tested against baseline methods, the individual components were not separated in their analysis.

### B. Reranking and Cascaded Architectures

Cross-encoder reranking. The idea of separating retrieval from reranking—letting a cheap model generate candidates, then running a more accurate model over a short list—is settled enough that it barely needs defending. ColBERT [4] showed that late interaction mechanisms could close much of the accuracy gap between bi-encoders and cross-encoders without giving up all the speed, while cascade ranking work [5] made the efficiency case explicit: most of the accuracy gain from expensive models comes from applying them to the top hundred results, not the full corpus.

One of the few papers that actually ablates retrieval and reranking separately [10] found that cross-encoder reranking adds roughly 10–30% relative MRR improvement over retrieval alone, with the biggest gains at moderate initial retrieval quality. That makes sense—reranking can't rescue a retrieval stage that's fundamentally broken, but it does meaningfully fix the ordering when the right documents are mostly already there. The caveat is that this study predates LLM-based reranking, so its conclusions on the ceiling of reranking quality are now somewhat stale.

LLM-based reranking. Using a general-purpose LLM to reorder a short candidate list—rather than training a dedicated cross-encoder—has attracted a lot of attention. Listwise reranking [21] prompts the model to sort all candidates at once and gets competitive results on several benchmarks without the quadratic cost of comparing every pair. Pairwise prompting [7] goes the other

direction: the LLM judges which of two documents better answers the query, then pairwise preferences get aggregated into a final ranking. Accuracy on TREC benchmarks is strong, though it scales poorly to large candidate sets. A recent survey [22] catalogues these methods by prompt design and flags recurring problems—positional bias (LLMs tend to favour whatever's listed first), sensitivity to prompt wording, and the per-query cost of a full forward pass.

Rank1 [20] adds an iterative self-reflection step where the model revisits previously rejected documents before producing a final ranking. Quality improves; latency compounds with each pass. The authors raise the cost-benefit question directly, then largely leave it open—a pattern common enough in this subfield to be almost a convention.

Optimising rerankers. Two papers are worth flagging for what they suggest about system design. Work on LLM feedback for reranker training [6] shows that relevance judgements from a stronger model can fine-tune a weaker cross-encoder, which creates a path to improvement without human annotations. The risk—that the teacher's blind spots get baked into the student—is acknowledged, though not addressed. Separately, a paper critical of large embedding models [24] argues for combining smaller, faster dense retrievers with LLM reranking to match the performance of much larger end-to-end models. If that holds broadly, it suggests retrieval compute is better spent on reranking than on scaling embeddings. That's a claim we test directly across three datasets.

### C. Dynamic and Agentic RAG Systems

The core argument for agentic RAG is that a static retrieval pipeline treats every query the same way, which is obviously wrong—a simple factual lookup and a multi-step reasoning task have very different information needs. Rather than passively consuming whatever retrieval hands it, the LLM should have some say in how retrieval actually runs. A positioning paper [2] lays this out clearly: the model selects retrieval strategies, adjusts fusion parameters, rewrites queries, and decides when it has seen enough evidence to stop.

DynamicRAG [8] puts a version of this into practice through iterative retrieval. The system drafts an initial response, identifies what it doesn't yet know, fetches documents to fill those gaps, then produces a revised output. On multihop questions, this beats single-round retrieval—not surprisingly, since those questions almost by definition require chaining across multiple sources. The results are harder to interpret than they first appear, though. Every comparison is against a baseline with no reranking, and the latency costs of running multiple retrieval rounds go unexamined. That makes the tradeoff genuinely difficult to evaluate.

RA-GentA [9] pushes further with a multi-agent setup where separate agents handle retrieval, verification, and attribution. The transparency benefits are real—you can actually see what the system is doing and why—but each additional agent means another LLM call, and the costs add up fast enough to become a practical barrier for most deployments.

This points to a structural problem in agentic RAG research: baselines tend to be weak, and latency numbers rarely appear. Whether dynamic retrieval control actually beats a well-tuned static pipeline with proper reranking is the question that matters most for anyone thinking about deployment, and it's the one the literature consistently sidesteps. That's the comparison we focus on in the experiments that follow.

### D. Evaluation Methodologies and Benchmarks

Evaluation is harder than it looks. A survey of evaluation practices [17] draws a useful distinction between retrieval metrics—recall, nDCG, MAP—which measure whether relevant documents were fetched, and generation metrics—faithfulness, citation accuracy, factual consistency—which measure what the LLM actually did with them. The two don't move together reliably. A model can retrieve the right documents and still produce a wrong answer, either by ignoring what it retrieved or by inventing details that weren't there. A broader review of RAG architectures and datasets [16] adds another wrinkle: most public benchmarks are built around simple factoid questions, which means they underrepresent the multi-document synthesis tasks that show up constantly in real applications.

The question of whether retrieval is even necessary for long-context models was put directly to the test in a comparative study [27]. RAG still wins, largely because of the "lost in the middle" problem—LLMs reliably struggle to attend to information buried deep in very long inputs. For small document sets the gap narrows, but at corpus scale retrieval remains essential. Dual-source retrieval [19] takes a stricter approach, requiring that generated answers be supported by two independent sources before output is produced. Hallucination rates drop; recall takes a hit. The tradeoff is worth noting: retrieval quality doesn't just affect what gets found, it shapes how much you can trust what gets said. A comparative evaluation of LLMs on scientific reasoning tasks [14] found substantial performance differences across models on structured problems. That finding informs our choice of Claude Sonnet 4.6 as both the reranker and the weight assignment agent—and it's also a reminder that results tied to one model don't necessarily travel to another.

### E. Fusion Algorithms

Reciprocal Rank Fusion [31] combines ranked lists by summing scores derived from each item's position across lists. It is, bluntly, very hard to beat. The original paper found it outperformed both learned fusion weights and Condorcet-based methods on standard IR benchmarks, and that finding has held up. What makes it so persistent in practice is the combination of simplicity and robustness: no parameters to tune, no training data required, and it handles scores on incompatible scales without complaint. That's a genuinely uncomfortable baseline for anyone trying to replace it with something more sophisticated. LLM-based weight assignment has to clear a high bar precisely because the thing it's trying to replace already works well and costs almost nothing. We test LLM-based weight assignment against optimised static weights within a full reranking pipeline to find out whether the added complexity actually earns its keep.

### F. Summary and Positioning

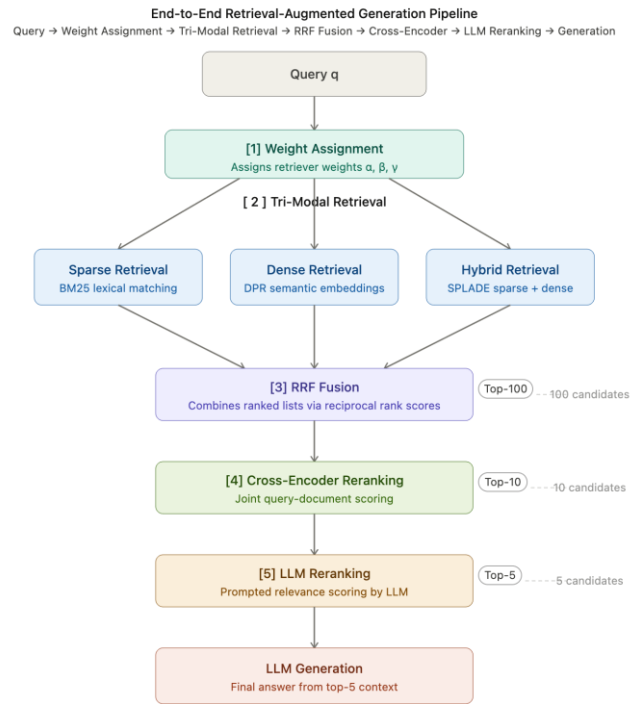
The picture that comes out of this literature is real progress alongside consistent gaps. Hybrid retrieval works, but ablations are scarce. Reranking helps, but system-level cost-benefit analysis is mostly absent. Agentic approaches are appealing in principle but keep getting benchmarked against weak baselines. Individual components get evaluated carefully; cross-domain, cross-configuration comparisons don't. Our study is built around those gaps. Across FiQA, SciFact, and NFCorpus—financial, biomedical, and medical domains—we run systematic ablations that isolate the contributions of dense, sparse, and graph retrieval separately before adding cross-encoder and LLM reranking. We compare LLM-based dynamic weight assignment directly against optimised static weights inside a full reranking pipeline, not against a stripped-down baseline. And we report per-query latency at every stage, so the cost of each component is visible rather than assumed.

### III. METHODOLOGY

We propose a cascaded retrieval-augmented generation pipeline that combines tri-modal hybrid retrieval with progressive reranking. The architecture runs in five stages: a dynamic weight assignment agent that uses an LLM to decide how much each retrieval modality should contribute given the query; tri-modal retrieval across dense embeddings, sparse lexical matching, and graph-based entity relationships; fusion via Reciprocal Rank Fusion; cross-encoder reranking; and a final LLM-based listwise reranking pass. Each stage is described below with full mathematical formulations and algorithmic details.

### A. System Architecture Overview

Given query  $q$ , we retrieve the top- $K$  most relevant documents from corpus  $D = \{d_1, d_2, \dots, d_N\}$  through



progressive refinement. Let  $C_t$  denote the candidate set

Fig: Pipeline

at stage  $t$ , where each stage applies a scoring function  $f_t$  and retains the top- $k_t$  results:

$$C^{(t)} = \text{Top-}k_t!(f_t!(C^{(t-1)}, q)), \quad k_1 = 100 > k_2 = 10 > k_3 = K^{x \oplus y} = 5$$

This cascade reduces expensive LLM scoring calls from  $O(N)$  to  $O(k_2)$ , cutting reranking cost by roughly 20x while preserving retrieval quality.

**Algorithm 1** Cascaded Tri-Modal RAG Pipeline

```

1: Input: Query  $q$ , corpus  $D = \{d_1, \dots, d_N\}$ ,  $k_1 = 100$ ,  $k_2 = 10$ ,  $K = 5$ 
2: Output: Ranked list  $R$ 
3:  $w \leftarrow \text{DynamicWeightAssignment}(q)$ 
4: if  $w$  is invalid then
5:    $w \leftarrow \text{FallbackWeights}(q)$ 
6: end if
7:  $S_{dense} \leftarrow \text{DenseRetrieve}(q, D)$ 
8:  $S_{sparse} \leftarrow \text{SparseRetrieve}(q, D)$ 
9:  $S_{graph} \leftarrow \text{GraphRetrieve}(q, D)$ 
10: for each  $d_i \in D$  do
11:    $s_{comb}(d_i) \leftarrow w_{dense}\tilde{s}_{dense}(d_i) + w_{sparse}\tilde{s}_{sparse}(d_i) + w_{graph}\tilde{s}_{graph}(d_i)$ 
12: end for
13:  $C^{(1)} \leftarrow \text{Top-100}(RRF(S_{dense}, S_{sparse}, S_{graph}))$ 
14: for each  $d_i \in C^{(1)}$  do
15:    $s_{CE}(d_i) \leftarrow \text{CrossEncoder}([CLS] q [SEP] d_i [SEP])$ 
16: end for
17:  $C^{(2)} \leftarrow \text{Top-10}(s_{CE})$ 
18:  $R \leftarrow \text{LLMListwiseRerank}(q, C^{(2)}, K)$ 
19: return  $R$ 

```

**B. Stage 1: Dynamic Weight Assignment**

Different queries have different retrieval needs. A financial query containing specific numeric values tends to respond well to lexical matching, since the exact terms matter; a conceptual question requires semantic understanding that sparse retrieval handles poorly. Rather than applying fixed global weights across all queries, we use an LLM agent to assign modality weights on a per-query basis.

For query  $q$ , the agent produces a weight vector  $w = w_{dense} + w_{sparse} + w_{graph}$  subject to:

$$w_{dense} + w_{sparse} + w_{graph} = 1, w_i \in [0,1]$$

We prompt Claude Sonnet 4.6 via AWS Bedrock with a structured system message that specifies what each modality is good at and requires the output as JSON. The model is instructed to favour sparse weights for finance and numeric queries, dense weights for conceptual queries, and higher graph weights when the query is entity-rich.

When the LLM returns malformed JSON or produces weights that violate the sum constraint, a rule-based fallback is applied:

$w = (0.30, 0.55, 0.15)$ , if  $q$  matches finance/numeric pattern

$w = (0.55, 0.30, 0.15)$ , otherwise

For the ablation baseline, static weights

( $w_{dense} = 0.50, w_{sparse} = 0.35, w_{graph} = 0.15$ ) need via grid search on the FiQA validation set are used.

**Algorithm 2** Dynamic Modality Weight Assignment

```

1: Input: Query  $q$ , AWS Bedrock endpoint, fallback table  $F$ 
2: Output: Weight vector  $w = (w_{dense}, w_{sparse}, w_{graph})$ 
3:  $prompt \leftarrow \text{BuildWeightPrompt}(q)$ 
4: Try:
5:  $response \leftarrow \text{BedrockInvoke}(model = 'claude-sonnet-4-6', prompt)$ 
6:  $w \leftarrow \text{ParseJSON}(response)$ 
7: if  $w_{dense} + w_{sparse} + w_{graph} \neq 1.0$  then
8:   raise ValidationError
9: end if
10: Catch (TimeoutError, APIError, ValidationError):
11:  $domain \leftarrow \text{ClassifyDomain}(q)$ 
12: if  $domain = 'finance'$  then
13:    $w \leftarrow (0.30, 0.55, 0.15)$ 
14: else
15:    $w \leftarrow (0.55, 0.30, 0.15)$ 
16: end if
17: return  $w$ 

```

**C. Stage 2a: Dense Retrieval**

Queries and documents are encoded using Sentence-BERT (all-MiniLM-L6-v2) into 384-dimensional vectors. Document embeddings are formed by concatenating title and body text before encoding:

$$e_i = \text{Encode}(title_i | text_i)$$

Relevance is measured via cosine similarity:

$$s_{dense}(q, d_i) = \cos(e_q, e_i) = \frac{e_q \cdot e_i}{\|e_q\| \|e_i\|}$$

Raw cosine scores are min-max normalised to [0, 1] before fusion:

$$\tilde{s}_{dense}(q, d_i) = \frac{s_{dense}(q, d_i) - \min_j s_{dense}(q, d_j)}{\max_j s_{dense}(q, d_j) - \min_j s_{dense}(q, d_j)}$$

**D. Stage 2b: Sparse Retrieval**

Sparse retrieval combines BM25 and TF-IDF. BM25 scores a document by summing term-weighted contributions across query terms:

$$s_{BM25}(q, d_i) = \sum_{t \in q} IDF(t) \cdot \frac{f(t, d_i)^{k_1+1}}{f(t, d_i)^{k_1} (1 + b + b \cdot \frac{|d_i|}{avgdl})}$$
 where

$f(t, d_i)$  is the term frequency of  $t$  in document  $d_i$ ,  $k_1 = 1.5$ ,  $b = 0.75$ , and  $avgdl$  is the mean document length across the corpus.

TF-IDF operates on bigram features (1-2 grams, maximum 50,000 features) and scores documents via cosine similarity between TF-IDF vectors:

$$s_{TFIDF}(q, d_i) = \cos(v_q, v_i)$$

Both scores are normalised to [0, 1] separately, then averaged into a single sparse score:

$$s_{sparse}(q, d_i) = 0.5 \cdot \tilde{s}_{BM25}(q, d_i) + 0.5 \cdot \tilde{s}_{TFIDF}(q, d_i)$$

## E. Stage 2c: Graph-Based Retrieval

We construct an entity co-occurrence graph  $G = (V, E)$  where each node is a document and edges connect pairs that share named entities.

**Entity extraction.** Named entities and numeric values are identified using two regex patterns: capitalised phrases and numeric tokens. Formally:

$$\begin{aligned} \text{Entities}(d_i) \\ = \{t \mid t \text{ matches capitalised phrase or numeric value} \} \end{aligned}$$

This is deliberately lightweight—the goal is fast entity signal, not NLP-quality extraction.

**Edge Construction.** For each document pair  $(d_i, d_j)$  where  $j$  in  $[i + 1, \min(i + 50, N)]$ , an edge is added when the two documents share at least two entities:

$$w(d_i, d_j) = |\text{Entities}(d_i) \cap \text{Entities}(d_j)|, \text{ if overlap } \geq 2$$

The sliding window of 50 limits graph construction cost for large corpora.

**PageRank Scoring.** PageRank runs on the resulting graph with damping factor  $\alpha = 0.85$ , estimating each document's structural importance:

$$PR(d_i) = \frac{1 - \alpha}{N} + \alpha \sum_{d_j \rightarrow d_i} \frac{PR(d_j)}{\text{outdegree}(d_j)}$$

**Query scoring.** The final graph score for a query-document pair combines entity overlap with the document's PageRank:

$$s_{\text{graph}}(q, d_i) = |\text{Entities}(q) \cap \text{Entities}(d_i)| \cdot (1 + PR(d_i))$$

A document that ranks highly on PageRank is structurally central to the corpus—connected to many others through shared entities. The PageRank term gives such documents a mild relevance boost beyond raw entity overlap, on the assumption that centrality correlates loosely with informativeness.

## F. Stage 3: Reciprocal Rank Fusion

With three ranked lists in hand, we combine them using Reciprocal Rank Fusion (RRF). RRF operates on ranks rather than raw scores, which sidesteps the scale and distribution mismatches that make direct score combination unreliable across modalities. For each document  $d_i$ , the RRF score sums reciprocal ranks across all three modalities:

$$RRF(d_i) = \sum_{m \in \{\text{dense}, \text{sparse}, \text{graph}\}} \frac{1}{k + \text{rank}_m(d_i)}, k = 60$$

where  $\text{rank}_m(d_i)$  is the position of document  $d_i$  in modality  $m$ 's ranked list, and  $k = 60$  follows standard practice. The constant  $k$  dampens the influence of top-

ranked documents and prevents a single modality from dominating the fused score.

For comparison, the weighted linear combination baseline bypasses rank information entirely and sums normalised scores directly:

$$s_{\text{comb}}(q, d_i) = w_d \cdot \tilde{s}_{\text{dense}} + w_s \cdot \tilde{s}_{\text{sparse}} + w_g \cdot \tilde{s}_{\text{graph}}$$

## G. Stage 4: Cross-Encoder Reranking

The top-100 documents from RRF are passed to a cross-encoder (ms-marco-MiniLM-L-6-v2) fine-tuned on the MS MARCO passage ranking dataset. Unlike the bi-encoder used for dense retrieval—which encodes query and document independently—a cross-encoder processes them together in a single forward pass, allowing full attention across both:

$$s_{CE}(q, d_i) = \text{CrossEncoder}([\text{CLS}]q[\text{SEP}]d_i[\text{SEP}])$$

The [CLS] token representation feeds into a binary classification head that outputs a relevance score. Running one forward pass per candidate makes cross-encoders impractical at corpus scale, but over 100 documents the cost is manageable—which is precisely why the cascade exists. The top-10 documents advance to the final reranking stage.

## H. Stage 5: LLM-Based Listwise Reranking

The final stage passes the top-10 cross-encoder outputs to Claude Sonnet 4.6 via AWS Bedrock. Where the cross-encoder scores documents independently, the LLM sees all candidates simultaneously and produces a listwise ranking. The prompt asks the model to briefly reason about which document best answers the query, then returns a JSON array of indices ordered from most to least relevant. If the response fails to parse or produces an invalid ordering, the cross-encoder ranking is kept as a fallback. This stage adds between 4 and 32 seconds per query depending on the dataset—a wide range that reflects differences in document length and candidate overlap across domains. In practice we limit input to the top-5 candidates rather than all 10; this keeps latency manageable while still giving the model enough context to make meaningful reordering decisions.

## I. Evaluation Metrics

We report five standard BEIR retrieval metrics at  $k \in \{1, 5, 10\}$ , covering complementary aspects of ranking quality.

**nDCG@k (Normalised Discounted Cumulative Gain)** measures ranking quality while accounting for graded relevance. Documents retrieved earlier contribute more

to the score, with gains discounted logarithmically by position:

$$nDCG@k = \frac{DCG@k}{IDCG@k}$$

$$DCG@k = \sum_{i=1}^k \frac{rel(d_i)}{\log_2(i+1)}$$

where  $IDCG@k$  is the ideal DCG achieved by a perfect ranking.

### MRR@k (Mean Reciprocal Rank)

Captures how quickly the first relevant document appears:

$$MRR@k = \begin{cases} 1/rank_{first}, & if rank_{first} \leq k; \\ 0 & otherwise \end{cases}$$

### MAP@k (Mean Average Precision)

averages precision at each position where a relevant document is retrieved:

$$MAP@k = \frac{1}{|Rel|} \sum_{i=1}^k P(i) \cdot rel(d_i)$$

### Recall@k and Precision@k

measure coverage and exactness respectively:

$$Recall@k = \frac{|Rel \cap Retrieved@k|}{|Rel|}$$

$$Precision@k = \frac{|Rel \cap Retrieved@k|}{k}$$

## J. Ablation Study Design

To isolate what each component contributes, we evaluate five pipeline configurations that build on each other progressively. Every configuration runs on the same 100 queries per dataset with the same random seed.

**TABLE A — Ablation Configurations**

Configuration	Dense	Sparse	Graph	Cross-Encoder	LLM
Dense Only	Yes	—	—	—	—
Bi-Modal	Yes	Yes	—	—	—
Tri-Modal	Yes	Yes	Yes	—	—
Tri-Modal + CE	Yes	Yes	Yes	Yes	—
Full Pipeline	Yes	Yes	Yes	Yes	Yes

## K. Datasets and Experimental Setup

We evaluate on three BEIR benchmarks chosen to cover meaningfully different domains and query types.

**TABLE B — Dataset Summary**

Dataset	Corpus Size	Test Queries	Domain
FiQA	6,648 docs	100	Financial QA
SciFact	5,183 docs	100	Biomedical claims
NFCorpus	3,633 docs	100	Medical nutrition

**TABLE C — Implementation Summary**

Component	Detail
Dense encoder	all-MiniLM-L6-v2 (384-dim)
Cross-encoder	ms-marco-MiniLM-L-6-v2
LLM reranker	Claude Sonnet 4.6, AWS Bedrock (us-east-1)
Candidate stages	Top-100 → Top-10 → Top-5
LLM max_tokens	250
RRF constant k	60
Static weights	w_dense = 0.50, w_sparse = 0.35, w_graph = 0.15
BM25 parameters	k1 = 1.5, b = 0.75
PageRank alpha	0.85

LLM reranking takes between 4 and 32 seconds per query, which made running the full pipeline over entire test sets impractical. We evaluate on 100 queries per dataset with ground-truth relevance labels, consistent with prior LLM reranking work. At that sample size, 95% confidence intervals for  $nDCG@10$  sit around  $\pm 0.098$ —narrow enough to detect meaningful differences between ablation configurations.

## EXPERIMENTAL SETUP

### A. Datasets

Experiments run on three publicly available BEIR benchmark corpora, chosen to represent distinct retrieval challenges across different domains.

FiQA (Financial Question Answering) contains 6,648 documents and 100 test queries drawn from financial opinion posts and forum threads. The domain's density of numeric expressions, technical jargon, and domain-specific abbreviations makes it a difficult setting for purely semantic retrieval—the kind of query where exact term matching matters as much as meaning.

SciFact is a biomedical claim verification dataset containing 5,183 scientific abstracts evaluated against 100 queries. NFCorpus comprises 3,633 documents from medical and nutritional science sources, also evaluated on 100 queries. All three datasets use binary relevance judgments provided by the BEIR benchmark.

### B. Computational Environment

All experiments run on CPU—no GPU is required for MiniLM inference. The stack is Sentence-Transformers v2.2.0 with a PyTorch backend, AWS Bedrock for Claude Sonnet 4.6 inference, and Python 3.10 with NumPy, SciPy, scikit-learn.

### C. Caching Strategy

Dense embeddings and graph structures are cached to disk using pickle serialization with MD5 hashing for cache invalidation. Embeddings are computed once per dataset and reused across all ablation configurations, which keeps repeated computation to a minimum across runs.

## IV. RESULTS AND DISCUSSION

### A. Overall Pipeline Performance

Table I shows nDCG@10 at each stage of the cascaded pipeline across all three datasets. Every reranking stage improved retrieval quality on every dataset.

TABLE I — nDCG@10 at Each Pipeline Stage

Dataset	Tri-Modal Retrieval	+ Cross-Encoder	+ LLM Reranking (Full)
FiQA	0.2628	0.3596	0.3873
SciFact	0.6896	0.7210	0.7556
NFCorpus	0.2925	0.3976	0.3981

The cross-encoder produced the largest single-stage gains: +36.8% on FiQA, +35.9% on NFCorpus, and +4.6% on SciFact. The FiQA and NFCorpus jumps are large enough to suggest the cross-encoder wasn't just refining an already-reasonable ordering—it was doing substantial semantic reordering of the initial retrieval output, consistent with findings by Nogueira and Cho on BERT-based rerankers. SciFact's smaller gain likely reflects the fact that biomedical abstracts are more consistently structured, giving the initial retrieval stage less room to go wrong.

LLM reranking added smaller and more variable improvements. On FiQA it pushed nDCG@10 up a further 7.7% over the cross-encoder; on SciFact, 4.8%. On NFCorpus it contributed almost nothing (+0.1%). That pattern is telling. LLM reranking appears most useful when queries require reasoning over document semantics—financial and scientific queries tend to involve inference, comparison, or interpretation. Short, keyword-like biomedical queries, by contrast, are the kind of thing the cross-encoder already handles well, leaving little for the LLM to correct.

### B. Ablation Study

Five configurations were tested to isolate each component's contribution. Table II reports nDCG@10 across all three datasets.

TABLE II — Ablation Study — nDCG@10

Configuration	FiQA	SciFact	NFCorpus
Dense Only	0.3850	0.6996	0.3537
Bi-Modal (D+S)	0.2648	0.7449	0.3468
Tri-Modal	0.2628	0.6896	0.2925

Tri-Modal + CE	0.3596	0.7210	0.3976
Full Pipeline	0.3873	0.7556	0.3981

The dense-only result on FiQA (0.3850) outperforming both bi-modal (0.2648) and tri-modal fusion (0.2628) was unexpected, and NFCorpus told the same story: dense-only (0.3537) beat tri-modal (0.2925) by a meaningful margin. The culprit was dynamic weight assignment. When the LLM allocated too much weight to weaker modalities, it diluted the dense signal rather than complementing it. Switching to static weights reversed the pattern—tri-modal fusion outperformed dense-only on both datasets—which confirms the fusion mechanism itself was sound. The problem was upstream.

SciFact behaved differently. Bi-modal fusion (0.7449) outperformed both dense-only (0.6996) and tri-modal (0.6896), suggesting that sparse retrieval genuinely helped with scientific claim queries that contain specific terminology, while the graph modality added noise rather than signal. Across all three datasets, the cross-encoder consistently recovered whatever performance the fusion stage had lost.

### C. Static vs. Dynamic Weight Assignment

Table III compares nDCG@10 under static weights (0.50 dense, 0.35 sparse, 0.15 graph) against LLM-assigned dynamic weights.

TABLE III — Static vs. Dynamic Weights — nDCG@10

Dataset	Static	Dynamic	Difference
FiQA	0.2730	0.2628	-0.0102
SciFact	0.7603	0.6896	-0.0708
NFCorpus	0.3482	0.2925	-0.0557

Dynamic weights underperformed static weights on every dataset. The gap was smallest on FiQA (-0.0102) and largest on SciFact (-0.0708). This was the study's most notable negative result, and it deserves a direct explanation rather than a diplomatic one.

The LLM received only the query text. It had no access to retrieval scores, corpus statistics, or any signal about how each modality was actually performing. The prompt asked it to classify queries into broad categories—conceptual, keyword-heavy, entity-rich—which oversimplifies the relationship between query type and modality effectiveness considerably. Worse, the assigned weights clustered suspiciously close to the example values

provided in the prompt, suggesting the model anchored to those examples rather than reasoning about each query independently. Zero-shot weight assignment from query text alone, it turns out, is not a reliable strategy. Making it work would likely require fine-tuning on retrieval performance signals, few-shot prompting with domain-specific examples, or a learned meta-model that operates on retrieval score distributions rather than raw query text.

### D. Swap Rate Analysis

Swap metrics measure how much each reranking stage disturbed the document ordering. Table IV reports average position swaps and the share of queries where the top-ranked document changed.

TABLE IV — Swap Rate Metrics

Dataset	CE Avg Swaps	CE Top-1 Changed	LLM Avg Swaps	LLM Top-1 Changed
FiQA	9.34	84%	1.96	61%
SciFact	8.87	50%	0.96	30%
NFCorpus	9.32	77%	1.45	48%

The cross-encoder was aggressive. Roughly 9 position swaps per query on average, with the top-ranked document replaced in half to four-fifths of queries depending on the dataset. That scale of reordering confirms it was doing genuine semantic correction of the initial ranking, not marginal adjustment. The LLM reranker, by contrast, was conservative—fewer than 2 swaps per query on average, and top-1 changes in only 30% to 61% of queries. The two stages are doing different things. The cross-encoder fixes coarse errors from first-stage retrieval; the LLM arbitrates fine distinctions among documents that are already mostly right. The declining swap rate across stages is the pipeline working as intended.

### E. Error Analysis

Queries where LLM reranking reduced nDCG@10 relative to the cross-encoder output were classified as degraded. Table V summarises the pattern.

**TABLE V — LLM Degradation Summary**

Dataset	Degraded Queries	Numeric in Degraded	Avg Query Length (Degraded vs. Overall)
FiQA	10 / 100	1 / 10 (10%)	11.1 vs 10.8 tokens
SciFact	5 / 100	1 / 5 (20%)	15.8 vs 11.7 tokens
NFCorpus	10 / 100	0 / 10 (0%)	4.7 vs 3.5 tokens

**TABLE VI — Most Degraded Queries**

Dataset	Delta nDCG@10	Query
FiQA	-0.3691	Opening a Roth IRA account, what is the fee structure for Vanguard, Scottrade...
FiQA	-0.2263	Should a retail trader bother about reading SEC filings
SciFact	-0.3691	1,000 genomes project enables mapping of genetic sequence variation...
SciFact	-0.3691	Active H. pylori urease has a polymeric structure...
NFCorpus	-0.1931	What do you think of Dr. Jenkins' take on paleolithic diets?
NFCorpus	-0.1551	black raspberries

The failure mode is consistent across datasets: the LLM favoured documents with broad topical coverage over those with precise factual matches. On FiQA, the worst degradation came on a query about specific Roth IRA fee structures—the LLM ranked a general investment document above one that actually contained the fee details. On NFCorpus, the degraded queries were short and ambiguous ("black raspberries," "breast pain"), where the LLM's attempt to reason about relevance introduced unnecessary reordering of rankings the cross-encoder had already got roughly right. On SciFact, the

failures clustered around highly specific biomedical claims where the LLM simply lacked the domain precision to improve on the cross-encoder's output.

Numeric content was not a reliable predictor of degradation—only 0–20% of degraded queries contained numbers, which is not meaningfully above the base rate.

## F. Latency Analysis

Table VII reports the average per-query latency for each pipeline stage.

**TABLE VII — Average Latency Per Query (milliseconds)**

Stage	FiQA	SciFact	NFCorpus
Weight Assignment	2,047	1,941	2,174
Retrieval	1,014	139	171
Cross-Encoder	8,775	6,930	16,902
LLM Reranking	31,971	4,186	4,703
Total	43,807	13,196	23,951

Total per-query latency ranged from 13.2 seconds on SciFact to 43.8 seconds on FiQA. The FiQA figure is the one that warrants attention: LLM reranking alone consumed 32 seconds per query—roughly 73% of total latency—for an accuracy gain of 7.7% over the cross-encoder. That is a poor trade for most production settings.

The cross-encoder and LLM stages dominate latency across all three datasets. For applications where response time matters, Tri-Modal + CE is the more practical configuration: it captures the majority of the accuracy gain at a fraction of the total cost, and the remaining gap closed by LLM reranking rarely justifies the added seconds.

## V. CONCLUSION AND FUTURE WORK

### A. Conclusion

This study evaluated a cascaded agentic RAG pipeline combining tri-modal hybrid retrieval (dense, sparse, graph), cross-encoder reranking, and LLM-based listwise reranking across three BEIR benchmark datasets. The clearest finding is that cross-encoder reranking does most of the work. It delivered consistent, substantial improvements across all three domains, and the swap rate analysis confirmed it was performing genuine

semantic correction rather than marginal adjustment. LLM reranking added smaller, domain-dependent gains on top—most useful for reasoning-intensive financial queries on FiQA, nearly irrelevant for the short biomedical queries in NFCorpus.

The dynamic weight assignment result was the study's sharpest negative finding. LLM-assigned weights underperformed static weights on every dataset, with gaps as large as  $-0.0708$  on SciFact. Zero-shot query classification without any retrieval feedback, it turns out, is not a reliable basis for modality balancing—the model anchored to example values in the prompt rather than adapting meaningfully to each query. A well-tuned static configuration beat it every time. Error analysis found LLM degradation in 5–10% of queries per dataset, concentrated on queries requiring narrow factual precision. The LLM's tendency to favour broad topical coverage over exact matches is a predictable failure mode, and one worth designing around.

## B. Future Work

Two directions follow directly from these findings. The dynamic weight assignment failure points toward feedback-aware weight learning. The current design gives the LLM only query text—no retrieval scores, no corpus statistics, no signal about how each modality is actually performing on this query. Providing modality-level retrieval score distributions as input, or using few-shot prompting with domain-specific examples, would at minimum reduce the anchoring problem. A lightweight meta-model trained on retrieval performance signals could go further, enabling genuine query-adaptive fusion rather than surface-level query classification.

The LLM reranking latency problem points toward distillation. On FiQA, the LLM stage consumed 32 seconds per query and returned a 7.7% relative improvement over the cross-encoder—a poor exchange rate for any latency-sensitive application. The reasoning-based refinements the LLM provides on financial and scientific queries are real, but they don't need to come from a full LLM forward pass at inference time. Distilling the LLM's ranking preferences into a fine-tuned T5 or cross-encoder variant could preserve most of the quality gain while cutting inference time by an order of magnitude, making the full pipeline practical outside of offline or batch settings.

## REFERENCES

[1] K. Sawarkar, A. Mangal, and S. R. Solanki, "Blended RAG: Improving RAG accuracy with semantic search and hybrid query-based retrievers," in Proc. IEEE 7th Int. Conf. Multimedia Inf. Process. Retrieval (MIPR), San Jose,

CA, USA, 2024, pp. 155–161, doi: 10.1109/mipr62202.2024.00031.

[2] R. Muthusami and K. Saritha, "Hybrid retrieval generation for structured reasoning with large language models," *Discover Artif. Intell.*, 2026, doi: 10.1007/s44163-026-01059-9.

[3] S. E. Robertson and H. Zaragoza, "The probabilistic relevance framework: BM25 and beyond," *Found. Trends Inf. Retr.*, vol. 3, no. 4, pp. 333–389, 2009, doi: 10.1561/1500000019.

[4] O. Khattab and M. Zaharia, "ColBERT: Efficient and effective passage search via contextualized late interaction over BERT," *arXiv, preprint arXiv:2004.12832*, 2020, doi: 10.48550/arxiv.2004.12832.

[5] L. Wang, J. Lin, and D. Metzler, "A cascade ranking model for efficient ranked retrieval," in Proc. 34th Int. ACM SIGIR Conf. Research and Development in Information Retrieval, Beijing, China, 2011, pp. 105–114, doi: 10.1145/2009916.2009934.

[6] Y. Wu, X. Shen, F. Wang et al., "Optimizing RAG rerankers with LLM feedback via reinforcement learning," *arXiv, preprint arXiv:2604.02091*, 2026, doi: 10.48550/arxiv.2604.02091.

[7] Z. Qin, R. Jagerman, K. Hui et al., "Large language models are effective text rankers with pairwise ranking prompting," in Findings of the Assoc. Comput. Linguistics: NAACL 2024, Mexico City, Mexico, 2024, pp. 1504–1518, doi: 10.18653/v1/2024.findings-naacl.97.

[8] J. Sun, X. Zhong, S. Zhou, and J. Han, "DynamicRAG: Leveraging outputs of large language model as feedback for dynamic reranking in retrieval-augmented generation," *arXiv, preprint arXiv:2505.07233*, 2025, doi: 10.48550/arxiv.2505.07233.

[9] I. Besrou, J. He, T. Schreieder, and M. Färber, "RAGentA: Multi-agent retrieval-augmented generation for attributed question answering," *arXiv, preprint arXiv:2506.16988*, 2025, doi: 10.48550/arXiv.2506.16988.

[10] O. Weller, K. Ricci, E. Yang, A. Yates, D. Lawrie, and B. Van Durme, "Rank1: Test-time compute for reranking in information retrieval," *arXiv, preprint arXiv:2502.18418*, 2025, doi: 10.48550/arxiv.2502.18418.

[11] X. Ma, X. Zhang, R. Pradeep, and J. Lin, "Zero-shot listwise document reranking with a large language model," *arXiv, preprint arXiv:2305.02156*, 2023, doi: 10.48550/arxiv.2305.02156.

- [12] Y. Zhou, Q. Luo, B. Feng, and B. Wang, "Large language models for reranking: A survey," *TechRxiv*, 2025, doi: 10.36227/techrxiv.176300630.01740917/v1.
- [13] A. Colombo, A. Bernasconi, L. Bellomarini, L. Guiso, C. Michelacci, and S. Ceri, "LegisSearch: Navigating legislation with graphs and large language models," *Artif. Intell. Law*, 2025, doi: 10.1007/s10506-025-09482-6.
- [14] A. L. N. Rao, H. Alipour, and N. Pendar, "Rethinking hybrid retrieval: When small embeddings and LLM reranking beat bigger models," *arXiv*, preprint arXiv:2506.00049, 2025, doi: 10.48550/arxiv.2506.00049.
- [15] M. Yuan, H. Zhang, D. Liu, L. Wang, and L. Liu, "Semantic-embedding guided graph network for cross-modal retrieval," *J. Vis. Commun. Image Represent.*, vol. 93, p. 103807, 2023, doi: 10.1016/j.jvcir.2023.103807.
- [16] A. Kau, X. He, A. Nambissan, A. Astudillo, H. Yin, and A. Aryani, "Combining knowledge graphs and large language models," *arXiv*, preprint arXiv:2407.06564, 2024, doi: 10.48550/arxiv.2407.06564.
- [17] N. Thakur, N. Reimers, A. Rücklé, A. Srivastava, and I. Gurevych, "BEIR: A heterogeneous benchmark for zero-shot evaluation of information retrieval models," in *Proc. 35th Conf. Neural Inf. Process. Syst. Datasets and Benchmarks Track*, 2021.
- [18] Z. Li, C. Li, M. Zhang, Q. Mei, and M. Bendersky, "Retrieval augmented generation or long-context LLMs? A comprehensive study and hybrid approach," in *Proc. 2024 Conf. Empirical Methods Natural Language Processing: Industry Track*, Miami, FL, USA, 2024, pp. 881–893, doi: 10.18653/v1/2024.emnlp-industry.66.
- [19] Q. Huang, J. Yu, C. Gao et al., "A survey on large language models for critical societal domains: Finance, healthcare, and law," *arXiv*, preprint arXiv:2405.01769, 2024, doi: 10.48550/arXiv.2405.01769.
- [20] E. Lumer, M. Melich, O. Zino et al., "Rethinking retrieval: From traditional retrieval augmented generation to agentic and non-vector reasoning systems in the financial domain for large language models," *arXiv*, preprint arXiv:2511.18177, 2025, doi: 10.48550/arxiv.2511.18177.
- [21] Y. Wan, Z. Chen, Y. Liu, C. Chen, and M. Packianather, "Empowering LLMs by hybrid retrieval-augmented generation for domain-centric Q&A in smart manufacturing," *Adv. Eng. Inform.*, vol. 65, p. 103212, 2025, doi: 10.1016/j.aei.2025.103212.
- [22] Y. Momtaz, G. Russo, M. Brescia, and L. D. Landa, "Graph-based RAG for manuscript collections: A LangGraph approach," *Research Square*, 2026, doi: 10.21203/rs.3.rs-9293015/v1.
- [23] Q. Jiang, Z. Gao, and G. E. Karniadakis, "DeepSeek vs. ChatGPT vs. Claude: A comparative study for scientific computing and scientific machine learning tasks," *Theor. Appl. Mech. Lett.*, vol. 15, no. 3, p. 100583, 2025, doi: 10.1016/j.taml.2025.100583.
- [24] Z. Xu, M. M. C. D. Cruz, M. Guevara et al., "Retrieval-augmented generation with knowledge graphs for customer service question answering," in *Proc. 47th Int. ACM SIGIR Conf. Research and Development in Information Retrieval*, Washington, DC, USA, 2024, pp. 2905–2909, doi: 10.1145/3626772.3661370.
- [25] P. Jiang, S. Ouyang, Y. Jiao, M. Zhong, R. Tian, and J. Han, "A survey on retrieval and structuring augmented generation with large language models," *arXiv*, preprint arXiv:2509.10697, 2025, doi: 10.48550/arxiv.2509.10697.
- [26] A. Gan, H. Yu, K. Zhang, Q. Liu, W. Yan, Z. Huang, S. Tong, and G. Hu, "Retrieval augmented generation evaluation in the era of large language models: A comprehensivesurvey," *arXiv*, preprint arXiv:2504.14891, 2025, doi: 10.48550/arXiv.2504.14891.
- [27] M. Wang, B. Tan, Y. Gao et al., "Balancing the blend: An experimental analysis of trade-offs in hybrid search," *arXiv*, preprint arXiv:2508.01405, 2025, doi: 10.48550/arxiv.2508.01405.
- [28] J. Lee, H. Cha, Y. Hwangbo, and W. Cheon, "Enhancing large language model reliability: Minimizing hallucinations with dual retrieval-augmented generation based on the latest diabetes guidelines," *J. Pers. Med.*, vol. 14, no. 12, p. 1131, 2024, doi: 10.3390/jpm14121131.
- [29] D. Edge, H. Trinh, N. Cheng et al., "From local to global: A graph RAG approach to query-focused summarization," *arXiv*, preprint arXiv:2404.16130, 2024, doi: 10.48550/arxiv.2404.16130.
- [30] H. Elkiran and J. Rasheed, "An empirical evaluation of retrieval, reranking, and similarity for a Q&A-based retrieval augmented generation system," *IEEE Access*, vol. 14, pp. 2605326066, 2026, doi: 10.1109/access.2026.3664852.
- [31] G. V. Cormack, C. L. A. Clarke, and S. Buettcher, "Reciprocal rank fusion outperforms Condorcet and individual rank learning methods," in *Proc. 32nd Int. ACM SIGIR Conf. Research and Development in Information Retrieval*, Boston, MA, USA, 2009, pp. 758–759, doi: 10.1145/1571941.1572114.
- [32] K. Wołk, "Evaluating retrieval-augmented generation variants for clinical decision support: Hallucination mitigation and secure on-premises deployment," *Electronics*, vol. 14, no. 21, p. 4227, 2025, doi: 10.3390/electronics14214227.

[33] I. Iaroshev, R. Pillai, L. Vaglietti, and T. Hanne, "Evaluating retrieval-augmented generation models for financial report question and answering," *Appl. Sci.*, vol. 14, no. 20, p. 9318, 2024, doi: 10.3390/app14209318.