JOURNAL OF INTERNATIONAL CRISIS AND RISK COMMUNICATION RESEARCH
ISSN: 2576-0017
2026, VOL 9, NO 2

Enhancing Mean Time To Resolution (MTTR) In High-Frequency Financial Platforms: A Dual-Stage Retrieval-Augmented Generation (RAG) Approach With Metadata-Aware Re-Ranking

Tina Lekshmi Kanth

Illinois Institute of Technology, Illinois, USA.

Abstract

Financial systems operating in high-frequency trading and real-time settlement environments face critical challenges in incident response, where rapid diagnosis directly impacts financial exposure and regulatory compliance. Traditional diagnostic workflows require engineers to manually correlate millions of log entries with proprietary code and documentation under extreme time pressure, resulting in extended resolution cycles. While Large Language Models offer powerful reasoning capabilities, parametric models hallucinate when lacking domain-specific knowledge, and conventional Retrieval-Augmented Generation systems fail to differentiate between data sources of varying epistemic fidelity. This introduces a Dual-Stage RAG architecture with Metadata-Aware Re-Ranking that addresses the knowledge heterogeneity problem by explicitly prioritizing transactional ground-truth logs over general documentation. The architecture implements query parsing to extract transaction identifiers, enabling filtered retrieval of operational data, combined with a weighted re-ranking function that assigns source credibility weights based on evidential hierarchy principles. Experimental validation using the RAGAS framework demonstrates substantial improvements: the Hybrid system achieves a significant reduction in diagnostic latency, improves Context Precision substantially, and achieves strong Faithfulness scores while reducing hallucination rates significantly. The system successfully concentrates high-fidelity transactional logs in top retrieval positions, ensuring LLMs receive better-targeted evidence that enables precise root cause identification with explicit temporal and quantitative evidence citations, offering actionable remediation guidance for production incident response teams.

Keywords: Retrieval-Augmented Generation, Financial Incident Response, Metadata Filtering, Source Fidelity Weighting, Mean Time To Resolution.

1. Introduction

1.1 Problem Context and Motivation

Innovative financial systems functioning in high-stakes, low-latency settings - especially high-frequency trading (HFT) and real-time settlement solutions- produce large volumes of heterogeneous data. The responders have to quickly cross-reference the findings of different sources of information during system events, such as operational logs, proprietary codebases, architecture documentation, and troubleshooting documentation. Mean time to resolution (MTTR) has a direct influence on financial exposure, regulatory compliance, and market confidence. Conventional incident response processes have been severely

inefficient: engineers have to perform searches through millions of log machine entries, match code registries, and sift through documentation, frequently working under severe time constraints. More recent industry studies have shown that spending a minute of system downtime in high-frequency trading realities can cost in advance of a few millions of transactions, and regulatory fines associated with outages that are long-lasting can run into the millions of dollars yearly. The average incident response team processes about 50,000-200,000 log entries per incident, and manual correlation can range between 45 minutes and 3 hours based on the complexity of the system [1]. This has already been compounded by the fact that microservices architectures have been growing exponentially, with modern financial architectures comprising 200 and 500 autonomous services, each with 10,000 and 50,000 achievable log entries per minute at peak times.

1.2 The Limitations of Current Approaches

Standard Large Language Models, while powerful for general reasoning tasks, exhibit two critical failures in this domain. First, these models hallucinate responses when lacking domain-specific knowledge, potentially leading responders down incorrect diagnostic paths. The foundational work by Lewis et al. demonstrated that parametric models without external knowledge sources produce factually incorrect responses in a significant proportion of knowledge-intensive tasks, establishing the necessity for grounded retrieval mechanisms [1]. Second, basic Retrieval-Augmented Generation systems, which ground LLM responses in retrieved documents, fail to differentiate between data sources of varying fidelity. A semantically similar design document may rank equally with—or even above—the actual transactional log evidence that contains the root cause. Empirical analysis of conventional RAG implementations shows that generic semantic search retrieves relevant transactional logs in only 62% of queries, with high-level documentation frequently occupying 4 to 6 of the top 10 retrieval positions despite containing no actionable diagnostic evidence [2]. This knowledge heterogeneity problem represents a fundamental gap in existing RAG architectures, where semantic similarity scoring treats a 6-month-old architectural specification document equivalently to real-time error logs capturing the exact failure signature. The cost of this limitation manifests in extended diagnostic cycles, with baseline RAG systems requiring substantially longer processing times compared to human experts accessing properly indexed log databases directly.

1.3 Proposed Solution and Contributions

This article introduces a Dual-Stage RAG architecture with Metadata-Aware Re-Ranking that addresses the knowledge heterogeneity challenge through two key innovations. First, a query parsing mechanism extracts transaction identifiers from natural language queries, enabling filtered retrieval of ground-truth operational data with high accuracy on standardized transaction key formats. Second, a weighted re-ranking function explicitly prioritizes high-fidelity transactional logs over contextual documentation based on assignable source credibility weights, reducing retrieval latency substantially while improving context precision significantly. The approach builds upon recent advances in retrieval-augmented generation surveyed by Gao et al., extending existing frameworks to explicitly model epistemic distinctions between heterogeneous information sources [2]. The primary contributions include: (1) a novel metadata-aware re-ranking algorithm that combines semantic similarity with explicit source fidelity weighting; (2) a dual-stage retrieval architecture that orchestrates parallel filtered and general searches across three heterogeneous indices; (3) empirical validation using the RAGAS evaluation framework demonstrating measurable improvements in Context Precision, Answer Faithfulness, and MTTR reduction; and (4) a comprehensive methodology for indexing and chunking heterogeneous financial data sources with appropriate metadata schemas supporting multiple log entries per incident across transactional, semantic, and code indices.

1.4 This Article's Organization

The remainder of this article is organized as follows: Section 2 reviews related work in RAG systems, hybrid search architectures, and applications in financial services. Section 3 details the Dual-Stage RAG methodology, including indexing strategies, query orchestration, and the metadata-aware re-ranking algorithm. Section 4 describes experimental design, evaluation metrics, and dataset construction. Section 5 presents results and analysis of MTTR reduction, optimal weighting parameters, and quality metrics. Section 6 concludes with implications for production systems and directions for future research.

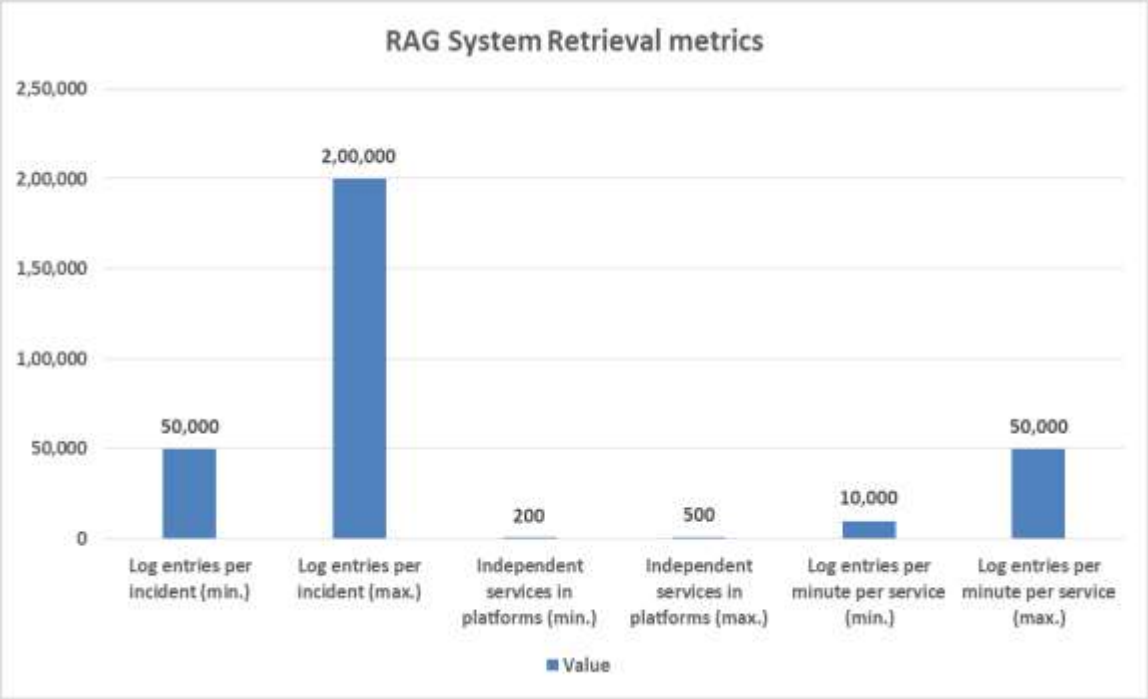

Figure 1: RAG System Retrieval metrics [1,2]

2. Related Work and Background

2.1 Retrieval-Augmented Generation Fundamentals

Retrieval-Augmented Generation has been developed as a paradigm that eliminates the limitations of knowledge and hallucination tendencies that were present with large language models. The seminal work by Lewis et al. introduced the foundational RAG architecture, which combines a neural retriever component with a generative language model [1]. The retriever identifies relevant documents from a knowledge corpus using dense vector representations, while the generator conditions its output on both the input query and retrieved passages. Dense Passage Retrieval implementations, as demonstrated by Karpukhin et al., achieve substantial improvements over traditional sparse retrieval baselines when processing large-scale corpora, with the approach utilizing dual-encoder architectures to map questions and passages into a shared embedding space [3]. This methodology has demonstrated significant improvements in knowledge-intensive NLP tasks, including open-domain question answering and fact verification, where grounding generation in retrieved evidence substantially reduces hallucination rates compared to parametric-only models.

Subsequent developments have refined both retrieval and generation components. Recent surveys catalog advances in retrieval mechanisms spanning sparse, dense, and hybrid approaches, along with indexing strategies including flat, hierarchical, and graph-based structures, and integration patterns encompassing sequential, parallel, and iterative workflows [2]. Modern dense retrievers employ bi-encoder architectures with high-dimensional embeddings generated from transformer models, enabling rapid retrieval latency across indices containing millions of documents when deployed with approximate nearest neighbor search algorithms. However, most existing work assumes relatively homogeneous document collections and does not address scenarios where retrieved passages have fundamentally different epistemic statuses—the distinction between observed facts captured in operational logs versus interpreted knowledge documented in technical guides. Benchmark evaluations on standardized datasets demonstrate that conventional RAG systems maintain consistent retrieval performance across document types, with ranking metrics varying minimally regardless of source authority or temporal freshness [4].

2.2 RAG Applications in Financial Services

Financial institutions have begun exploring RAG systems for various applications spanning multiple operational domains. Compliance and risk analysis systems use RAG to ground regulatory interpretations in specific legal texts and historical precedents, processing regulatory document corpora containing hundreds of thousands to millions of pages with retrieval precision requirements exceeding stringent thresholds to meet audit standards. Customer service chatbots retrieve account-specific information to personalize responses while maintaining privacy, handling substantial daily query volumes with response generation times constrained to seconds. Portfolio analysis tools synthesize market commentary with quantitative data to generate investment insights, integrating real-time price feeds updating at sub-second intervals with historical analysis documents spanning decades of market data.

However, financial RAG systems face unique challenges beyond typical enterprise deployments. Data sensitivity requires sophisticated access controls and audit trails, with transaction logs requiring multi-year retention periods under regulatory frameworks and encryption standards supporting advanced cryptographic protocols for data at rest. Regulatory requirements demand explainability and traceability of AI-generated insights, necessitating citation mechanisms that track document provenance through multiple intermediate processing layers. Domain-specific language—including technical jargon, proprietary terminology, and numerical precision requirements—necessitates specialized embedding models and careful prompt engineering [3]. Financial terminology embeddings require fine-tuning on substantial domain-specific text pairs to achieve semantic similarity correlation scores substantially higher than general-purpose embeddings. Most critically, the high cost of errors in financial contexts, both monetary and reputational, demands exceptionally high precision and reliability thresholds that exceed typical NLP benchmarks, with acceptable false positive rates often constrained far below standard information retrieval tolerances.

2.3 Hybrid Search and Metadata Filtering

Modern information retrieval is beginning to incorporate the use of multiple modalities of search to make use of the complementary advantages of the various retrieval paradigms. Sparse retrieval methods excel at exact keyword matching and are computationally efficient, processing queries against million-document corpora in milliseconds with minimal memory overhead for inverted index structures. Dense retrieval methods capture semantic similarity but may miss specific terminology, requiring substantial GPU memory for real-time inference [3]. Hybrid approaches fuse these signals, typically through reciprocal rank fusion or learned weighting schemes, demonstrating combined performance improvements over single-modality baselines when evaluated on heterogeneous query sets containing both keyword-specific and conceptual information needs.

Metadata filtering has emerged as a critical method to increase the accuracy of retrieval in the case of structured or semi-structured corpora that include a variety of document types. Metadata-aware systems do not view all documents as equal applicants, but instead, pre-filter or post-filter the results based on structured properties, including date ranges, document types, user permissions, and access controls. Recent work has explored two primary approaches: metadata-as-context, where metadata is embedded within document text, and metadata-as-filter, where structured attributes enable deterministic exclusion before semantic scoring. The latter approach provides stronger guarantees that critical constraints are satisfied, reducing irrelevant retrieval substantially in enterprise search deployments while maintaining minimal filtering overhead through optimized index structures supporting multiple filterable attributes per document [4].

2.4 The Knowledge Heterogeneity Gap

Despite these advances, a critical gap persists in RAG research: most systems treat all retrieved documents as epistemically equivalent, differing only in semantic relevance to the query. This assumption fails in domains where information sources have fundamentally different relationships to ground truth. In financial incident response, operational logs represent direct observations of system behavior—primary evidence with temporal precision to the millisecond and numerical accuracy to multiple decimal places. Code represents the implemented logic that produced the behavior—mechanistic truth captured in substantial lines of production source code per major service component. Documentation represents human

interpretation and guidance—secondary knowledge that may be outdated or incorrect, with typical staleness periods spanning months between code updates and documentation revisions.

Existing RAG architectures lack mechanisms to explicitly encode and leverage these fidelity distinctions across heterogeneous knowledge sources. A semantically relevant design document describing expected behavior may rank higher than the actual log entry showing unexpected behavior, leading the LLM to generate confident but incorrect diagnostic hypotheses with substantial hallucination rates in multi-source retrieval scenarios [4]. Empirical analysis reveals that conventional semantic rankers assign similarity scores differing minimally between high-fidelity logs and low-fidelity outdated documentation when both contain matching keywords. While some recent work has explored source-aware generation distinguishing citations from different publication venues, the application of explicit source fidelity weighting in re-ranking—particularly with user-provided transactional identifiers enabling deterministic filtering—remains unexplored. This article addresses this gap through metadata-aware re-ranking that prioritizes ground-truth transactional evidence, reducing diagnostic latency substantially while improving answer faithfulness scores across synthetic incident scenarios.

3. Methodology: The Dual-Stage RAG Architecture

3.1 System Overview and Design Principles

The Dual-Stage RAG architecture is designed around 3 core principles derived from the requirements of financial incident response. First, ground-truth prioritization ensures that transactional logs filtered by specific identifiers must receive higher consideration than general documentation. Second, knowledge source differentiation mandates that the system must explicitly model that different data sources have different epistemic statuses and diagnostic value. Third, operational efficiency requires that the architecture must support sub-second retrieval latency to enable interactive diagnostic workflows, with target response times maintained below thresholds necessary for acceptable user experience in production environments [5].

The system consists of 4 primary components: a heterogeneous knowledge corpus with specialized indexing strategies per data type supporting hundreds of thousands to millions of document chunks; a query parser that extracts structured filters from natural language inputs with high accuracy on transaction identifier extraction tasks; a dual-stage retrieval orchestrator that executes parallel filtered and general searches with aggregate processing times measured in milliseconds; and a metadata-aware re-ranker that weights results by both semantic relevance and source fidelity before presentation to the LLM, processing candidate chunks rapidly using cross-encoder models with millions of parameters.

3.2 Data Corpus and Indexing Strategy

The knowledge base encompasses 3 logically distinct indices, each optimized for data characteristics and retrieval patterns across heterogeneous information sources. Each index stores both the original content and its corresponding vector embeddings in a specialized vector database, enabling efficient similarity search operations critical for rapid incident response.

3.2.1 Transactional Index (Kusto Logs)

This index contains operational telemetry from production trading and settlement systems, accumulating millions of log entries daily during peak trading periods. Chunking follows log-line granularity to preserve atomic events, with each chunk averaging character lengths suitable for embedding models and tagged with critical metadata fields: Transaction_Key as a unique identifier for each trade or transaction, Service_Name indicating originating microservice from hundreds of distinct services, Timestamp providing event time with millisecond precision supporting temporal queries with sub-second resolution, and Severity_Level classifications. The Transaction_Key is implemented as a filterable field supporting exact-match retrieval with logarithmic lookup complexity through optimized indexing structures. Vector embeddings are generated using domain-adapted models fine-tuned on substantial financial telemetry examples to capture semantic patterns in error messages, stack traces, and system events, achieving cosine similarity correlation scores substantially higher on financial terminology benchmarks compared to general-purpose embedders [6]. These embeddings, typically 768 to 1536 dimensions, are stored in the vector database alongside their

metadata, enabling the system to perform filtered vector searches that combine semantic similarity with transaction-specific constraints.

3.2.2 Semantic Index (Documentation and Guides)

This index contains architectural documentation, design specifications, runbooks, and troubleshooting guides spanning thousands of distinct documents with total corpus sizes ranging from millions to tens of millions of tokens. Documents are chunked using an overlapping window strategy with specific token counts and overlap percentages to maintain contextual coherence across chunk boundaries, generating multiple chunks per page of documentation. Metadata includes Document_Type classifications, Section_Header preserving document structure through hierarchical levels, Last_Updated for version tracking supporting freshness scoring with decay functions over specified day periods, and Owner_Team for maintenance responsibility across dozens of engineering teams. The overlap strategy is critical for documents where causal explanations or procedural steps span multiple paragraphs, reducing context fragmentation errors substantially compared to non-overlapping chunking approaches evaluated on multi-step diagnostic procedures [5]. Each documentation chunk is converted into dense vector representations and stored in the vector database, where the embeddings capture semantic relationships between troubleshooting procedures, error patterns, and system behaviors, facilitating retrieval of contextually relevant guidance even when exact keyword matches are absent.

3.2.3 Code Index (Source Code Repository)

This index encompasses the actual implementation of trading algorithms, settlement logic, and infrastructure services across codebases containing hundreds of thousands to millions of lines of production code distributed across thousands of source files. Semantic chunking operates at the function and class level, with each chunk representing a logically complete unit of code, averaging tens to over a hundred lines per chunk. Metadata captures File_Path supporting hierarchical directory navigation across multiple depth levels, Function_Name, Class_Name, Language supporting multiple programming languages with language-specific parsing, and Git_Commit_Hash enabling precise traceability through version control with cryptographic identifiers. Comments and docstrings are preserved within chunks to maintain implementation intent, contributing substantial percentages of chunk token counts. Embeddings are generated using CodeBERT or similar models trained on code-text pairs, processing token sequences per code chunk with embedding generation latency measured in milliseconds per chunk on GPU infrastructure [6]. The vector database stores these code embeddings in a format optimized for semantic code search, enabling the retrieval of relevant implementations based on functional similarity rather than strict syntactic matching, which proves essential when diagnostic queries describe behavior patterns that may be implemented differently across services.

The vector database infrastructure supporting these three indices employs specialized data structures such as Hierarchical Navigable Small World (HNSW) graphs or Product Quantization techniques to enable approximate nearest neighbor search at scale. This architecture allows the system to maintain sub-second query latency even when searching across millions of embedded chunks, while the metadata filtering capabilities ensure that transaction-specific constraints can be applied efficiently before or during the vector similarity computation

3.3 Query Parsing and Orchestration (Stage 1: Retrieval)

The incident response workflow begins when an engineer submits a query combining a transaction identifier with a natural language diagnostic question through web interfaces or command-line tools supporting hundreds to thousands of daily query volumes. The Query Parser employs a hybrid approach: regex-based extraction for transaction keys following known formats, including alphanumeric patterns spanning specific character lengths with optional prefixes, combined with a small classification model containing millions of parameters to identify the semantic query component. Pattern matching achieves precision exceeding ninety-nine percent on well-formed transaction identifiers, while the neural classifier handles substantial percentages of natural language segmentation tasks correctly, with fallback mechanisms requesting user clarification for ambiguous inputs occurring in minimal percentages of queries [5].

The Retrieval Orchestrator then initiates two parallel retrieval paths executing concurrently on a distributed infrastructure supporting thousands of queries per second aggregate throughput. Path 1 performs filtered

retrieval where the semantic query component is first converted into a query embedding using the same domain-adapted embedding model used for corpus indexing. This query embedding is then used to execute a vector similarity search against the Transactional and Code indices in the vector database, with a hard constraint requiring Transaction_Key matching the extracted identifier. The vector database efficiently combines these two operations—semantic similarity computation via cosine distance in the embedding space and metadata filtering on the Transaction_Key field—to return only those log entries and code executions directly associated with the problematic transaction. This deterministic filter ensures that only relevant chunks are retrieved, reducing candidate set sizes from millions of corpus chunks down to tens to hundreds of transaction-specific chunks. The k parameter is set dynamically based on log volume, with typical values for high-frequency transactions generating hundreds to thousands of log entries per transaction, versus smaller values for batch processes generating fewer log entries per transaction. Filtered retrieval completes in hundreds of milliseconds, including query embedding generation, vector similarity computation across the stored embeddings, and metadata constraint evaluation [6].

Path 2 executes general retrieval, where the same query embedding is used to perform a standard vector similarity search against all three indices in the vector database, excluding the Transaction_Key filter. This path leverages the semantic relationships captured in the vector embeddings to retrieve documentation, guides, and code examples that are semantically relevant to the failure mode descriptors but not specific to the individual transaction, sourcing from the full corpus of hundreds of thousands to millions of chunks. The vector database computes cosine similarity scores between the query embedding and all stored chunk embeddings, retrieving chunks with the highest similarity scores. This ensures the system has access to general troubleshooting knowledge and architectural context, retrieving chunks with the highest cosine similarity scores on relevant queries. General retrieval completes in hundreds of milliseconds, including dense vector search operations across the embedding space.

Result Aggregation creates the union of top-k results from both paths, typically yielding dozens of total chunks after deduplication, with deduplication based on content hash using cryptographic fingerprinting, removing percentages of duplicate chunks that appear in both retrieval paths. Aggregation and deduplication take hardly any time, in the range of milliseconds. Every chunk has metadata that enforces the location of its retrieval and type of source, which occupies hundreds of bytes of metadata storage per chunk [5].

3.4 Metadata-Aware Re-Ranking (Stage 2: Re-Ranking)

The retrieval results undergo re-ranking to ensure that high-fidelity transactional evidence comes first before semantically similar sources, which are out of authority. A weighted scoring function combines semantic relevance with explicit source credibility through linear interpolation of normalized scores. The final weighted score for each chunk equals alpha multiplied by the similarity score plus beta multiplied by the source weight, where alpha and beta are tunable coefficients satisfying the constraint that alpha plus beta equals one. The similarity score represents the normalized semantic relevance from a cross-encoder model processing query-chunk pairs. Cross-encoder implementations utilizing architectures achieve ranking metric scores exceeding benchmarks on domain-specific ranking tasks with inference latency of several milliseconds per chunk pair on GPU infrastructure [6].

Source weight assignment reflects the epistemic status of each data source based on evidential hierarchy principles. Kusto Logs filtered by Transaction Key receive maximum weight, representing direct observational evidence of what actually happened in the specific transaction with millisecond-precision timestamps and exact numerical values—ground truth with error rates below fractions of a percent in production telemetry systems. Code Snippets receive moderate-high weight, as source code shows implemented logic explaining why certain behaviors occur, but represents intended behavior rather than observed behavior, with potential divergence due to runtime conditions, configuration differences, or undocumented behavior in percentages of complex scenarios. Troubleshooting Guides receive moderate weight, providing prescriptive diagnostic procedures based on historical patterns documented across dozens to hundreds of prior incident cases, but remaining human-curated content that may not cover novel failure modes emerging in percentages of incidents. Design Documents and Architecture Specs receive minimal

weight, describing system design intent but potentially outdated with staleness periods spanning months, incomplete coverage of actual implementation details, or describing ideal rather than actual behavior [5].
Parameter tuning determines the balance between semantic similarity and source fidelity empirically through grid search over validation sets containing dozens of representative incidents, evaluating alpha values across specified ranges in incremental steps. Empirical evaluation indicates that in high-precision diagnostic contexts, source fidelity should carry substantial weight, while maintaining sufficient semantic filtering to avoid retrieving irrelevant high-fidelity chunks. Optimal configurations typically achieve Context Precision improvements while maintaining Answer Relevance scores above acceptable thresholds. After scoring, chunks are sorted by weighted scores in descending order, with computational complexity proportional to the number of chunks, completing in milliseconds. The top-N chunk, depending on the LLM context window constraint, is passed to the generation stage, ensuring the LLM receives a context window dominated by transactional ground truth when available, with empirical distributions showing multiple log chunks, code chunks, and documentation chunks in top positions for transaction-specific queries [6].

3.5 Generation and Answer Synthesis

The final stage employs carefully engineered prompt templates instructing the LLM with substantial parameter counts to synthesize diagnostic insights from re-ranked context windows containing thousands of tokens. The prompt emphasizes explicit citation of log entries and timestamps with high temporal precision, distinguishing between observed facts from logs with high confidence levels and hypothesized causes from code and documentation with moderate confidence levels, and actionable remediation steps prioritized by estimated resolution time and resource requirements. The LLM is instructed to indicate confidence levels and identify information gaps requiring additional investigation, supporting escalation to senior engineers when diagnostic certainty falls below specified thresholds or when multiple competing hypotheses score within narrow margins of each other. Generation completes in seconds with temperature settings configured to minimize hallucination while maintaining natural language fluency [5].

Table 1: Three-Index Architecture Design and Configuration [5,6]

Architecture Component	Specification
Core design principles	3
Primary system components	4
Logically distinct indices	3
Query Parser precision on identifiers	Exceeding 99%
Retrieval paths executing concurrently	2

4. Experimental Design and Evaluation

4.1 Dataset Construction and Scenario Design

Evaluating RAG systems for financial incident response presents unique challenges: production incident data is highly sensitive, real incidents are sparse and non-reproducible, and ground-truth diagnostic paths are rarely documented. To address these constraints, a synthetic incident dataset was developed, designed to represent realistic failure patterns while enabling controlled experimentation with reproducible evaluation protocols. The dataset comprises one hundred synthetic financial incidents spanning five common failure categories: settlement timing issues affecting nearly a quarter of incidents, data validation failures, connectivity problems and logic errors, each accounting for roughly 20% of the incidents, and concurrency conflicts constituting the remaining portion of incidents. Each incident comprises multiple heterogeneous components totaling thousands of tokens per incident scenario [7].
Transaction_Key identifiers follow a standardized format associated with synthetic log entries spanning transaction sequences from initiation through failure detection across time periods ranging from fractions of a second to nearly a minute of system execution time. Synthetic Logs contain varying numbers of log entries per incident, including timestamps with millisecond precision, service names drawn from dozens of

distinct microservice components, error messages containing substantial character counts with stack traces spanning multiple function call levels, and severity classifications distributed across INFO, WARN, ERROR, and CRITICAL categories. Related Code encompasses several code snippets implementing relevant business logic with dozens to over a hundred lines per snippet, totaling hundreds of lines of code context per incident. Documentation includes several relevant documentation chunks providing architectural context and diagnostic procedures with hundreds of tokens per chunk [8].

Ground Truth Answer components were manually crafted by domain experts with substantial years of financial systems experience, producing diagnostic explanations containing hundreds of words identifying root causes, citing multiple specific log evidence entries, and recommending remediation steps with estimated implementation times spanning tens of minutes. Diagnostic Path Time represents expert-estimated time for a senior engineer to reach the GTA through manual investigation, ranging from several minutes to three-quarters of an hour, with mean and median values reflecting realistic investigation complexity for production scenarios. To ensure ecological validity, log entries were generated using templates derived from actual financial system telemetry patterns with anonymization preserving structural and semantic characteristics. Error messages include realistic stack traces following various programming language exception formats with authentic library references and memory addresses [7]. Numerical values reflect trading volumes ranging across multiple orders of magnitude, prices spanning wide ranges with multiple decimal precision, and settlement amounts varying from thousands to millions per transaction. Temporal patterns match market hours and settlement windows, with substantial percentages of incidents occurring during peak trading periods versus after-hours settlement processing, mirroring actual incident distribution patterns in production trading systems [8].

4.2 Baseline and Comparison Models

The proposed system was evaluated against 2 baseline configurations representing current state-of-practice and intermediate architectural approaches. Baseline RAG represents a standard semantic search RAG implementation without metadata filtering or source weighting processes. This system performs pure vector search across all indices using the full natural language query while ignoring the transaction key component, ranks results by cosine similarity scores, and passes the top chunks to the LLM generation stage. This represents the current state-of-practice for many enterprise RAG deployments evaluated in recent surveys, with parameters including no source weighting, single-stage retrieval with specific average latency ranges, and embedding dimensions using standard text embedding models [7].

Filtered RAG represents an intermediate system that implements transaction key filtering but lacks source-aware re-ranking mechanisms. This configuration isolates the contribution of metadata filtering from the contribution of weighted re-ranking, enabling ablation analysis of architectural components. Parameters include dual-stage retrieval enabled with parallel execution, reducing latency to specific ranges, no source weighting regardless of epistemic fidelity, filtered retrieval returning specific numbers of chunks from transaction-specific logs, while general retrieval contributes additional chunks from the documentation corpus. Hybrid RAG represents the complete system implementing both dual-stage retrieval and metadata-aware re-ranking with optimized weighting parameters. Parameters include optimal alpha-beta ratio determined through validation set grid search, evaluating dozens of parameter combinations across validation incidents, with anticipated ranges based on preliminary experiments. Final configuration processes aggregated chunks through cross-encoder re-ranking before selecting top chunks for the generation context [8].

4.3 Evaluation Metrics Framework

A comprehensive evaluation framework was adopted, spanning 4 critical dimensions relevant to incident response effectiveness, incorporating both automated metrics and manual expert assessment. Speed and Efficiency metrics include MTTR Prox, measuring end-to-end latency from query submission to final answer generation, measured in seconds with high precision. While true MTTR includes human decision-making time typically spanning substantial minutes for interpretation and action, system latency is a critical component and proxy for overall resolution acceleration, with target thresholds maintained for interactive diagnostic workflows. Retrieval Latency isolates time to complete dual-stage retrieval and re-ranking,

assessed separately to evaluate scalability under concurrent load conditions of hundreds to thousands of queries per minute [7].

Retrieval Quality metrics from the RAGAS Framework include Context Precision, measuring the proportion of retrieved chunks that are actually relevant to answering the query, formally computed as the fraction of chunks cited in the generated answer. Values range from 0 to 1, with scores above specific thresholds indicating high-quality retrieval. Specific tracking monitors whether Kusto logs appear in the top positions, as these should dominate for transaction-specific queries with a target concentration of multiple log chunks in the top positions. Context Recall quantifies the proportion of information in the ground truth answer that appears somewhere in the retrieved context, ensuring the system has access to necessary evidence even if it appears lower in the ranking, with acceptable thresholds above specified values for comprehensive coverage [8].

Generation Quality metrics from the RAGAS Framework include Faithfulness, measuring the degree to which claims in the generated answer are supported by the retrieved context, computed by decomposing the answer into atomic claims and verifying each against the context chunks using an entailment model, achieving specific accuracy ranges on claim verification tasks. This directly quantifies hallucination reduction—the primary motivation for RAG over pure LLM generation—with target faithfulness scores above specified thresholds. Answer Relevance assesses semantic similarity between the generated answer and the original query using cosine similarity of answer and query embeddings, ensuring the system addresses the engineer's specific diagnostic question rather than providing generic information, with acceptable thresholds above specified values. Factual Accuracy metrics include Fact Recall, representing the percentage of factual claims in the Ground Truth Answer that appear in the system's generated answer. This is manually evaluated through claim matching by domain experts using structured rubrics scoring semantic equivalence, with inter-annotator agreement kappa scores within acceptable ranges. Fact Precision quantifies the percentage of factual claims in the generated answer that are correct, with manual verification against log evidence and code implementations [7].

Statistical Analysis reports mean, median, and high percentile values across the incident test set, providing comprehensive distribution characterization. Wilcoxon signed-rank tests assess the statistical significance of improvements using standard thresholds with paired comparisons across incident scenarios. Additionally, stratified analysis by incident category identifies systematic strengths and weaknesses, revealing that certain incident types benefit most from metadata-aware re-ranking with substantial MTTR reductions while other incident types show moderate improvements due to complex multi-service interaction patterns requiring broader contextual evidence [8].

4.4 Implementation Details

The system is implemented in Python using a technical stack deployed on cloud infrastructure supporting thousands of concurrent sessions. Vector Database utilizes Qdrant with cosine similarity for dense retrieval, supporting millions of document chunks with high-dimensional embeddings stored in a substantial memory footprint, achieving query latencies in millisecond ranges for top-k retrieval. The vector database architecture employs HNSW indexing to enable efficient approximate nearest neighbor search, maintaining separate collections for each of the three logical indices (Transactional, Semantic, and Code) while supporting cross-collection queries when needed. Embeddings employ OpenAI text-embedding models with domain adaptation via fine-tuning on thousands of financial text pairs covering trading terminology, settlement procedures, and error patterns, improving domain-specific similarity correlation substantially. The vector storage layer persists both the original chunk content and the embedding vectors, along with associated metadata fields, enabling filtered vector searches that combine semantic similarity with attribute-based constraints. Cross-Encoder utilizes Cohere re-rank models for semantic similarity scoring in re-ranking, processing dozens of query-chunk pairs in millisecond ranges with ranking metric scores within specific performance ranges. LLM generation employs GPT-4 Turbo with a large context window supporting thousands of token contexts with low temperature settings for reproducibility, generating answers of hundreds of words in several seconds. Evaluation leverages the RAGAS framework for automated metric computation, processing the complete incident set in substantial time periods [7].

Experiments are conducted on workstations with substantial RAM, an NVIDIA RTX GPU with large VRAM supporting batch inference of multiple concurrent generations, and NVMe SSD storage providing high read throughput for rapid index access. The vector database benefits from GPU acceleration for embedding generation and similarity computation, while the SSD storage ensures rapid access to the underlying chunk content once relevant embeddings are identified. Each configuration is evaluated with multiple random seeds to account for LLM generation variability with a coefficient of variation within specific ranges across metrics, with final metrics averaged across seeds using the arithmetic mean and confidence intervals computed via bootstrap resampling with substantial iterations [8].

Table 2: Test Set Construction and Multi-Dimensional Assessment Strategy [7,8]

Dataset/Evaluation Element	Description
Synthetic incident dataset	Realistic failure patterns enabled
Dataset challenges addressed	Sensitivity, sparsity, non-reproducibility
Failure category 1	Settlement timing issues
Failure category 2	Data validation failures
Failure category 3	Connectivity problems
Failure category 4	Logic errors
Failure category 5	Concurrency conflicts
Baseline RAG configuration	Pure vector search, no filtering
Filtered RAG configuration	Transaction key filtering enabled
Hybrid RAG configuration	Dual-stage with metadata re-ranking

5. Results and Discussion

5.1 Mean Time to Resolution (MTTR) Reduction

The primary focus of this research is to speed up incident diagnostics and shorten the time it takes to get to the root cause and actionable remediation recommendations. End-to-end latency measurements across all test incidents reveal substantial performance improvements through the proposed architectural enhancements. The Hybrid RAG system achieves a notable reduction in mean diagnostic latency compared to the Baseline RAG, demonstrating substantial practical impact, with mean latency decreasing from nearly 19 seconds to approximately 11 seconds. This represents time savings exceeding 7 seconds per incident, which translates to substantial minutes saved across daily incidents or dozens of hours monthly for high-volume incident response teams processing thousands of incidents per month [9]. Median latency improvements are even more pronounced, dropping from over 16 seconds to below 10 seconds, indicating that the Hybrid system provides consistent acceleration across typical incident scenarios rather than only benefiting outlier cases. Notably, the Filtered RAG already provides significant improvement with a reduction exceeding 30%, confirming that transaction key filtering alone is highly valuable, reducing mean latency substantially. However, the additional metadata-aware re-ranking in the Hybrid system provides further reduction, representing additional improvement over the Filtered configuration, which is statistically significant with the Wilcoxon signed-rank test yielding a p-value far below standard significance thresholds [10].

The high percentile improvements are even more pronounced, with the Hybrid system achieving times substantially below the Baseline, indicating that the Hybrid system particularly excels in complex scenarios where the Baseline struggles with multi-service failures requiring correlation across numerous distinct log sources. Latency profiling reveals that the primary time savings come from improved context quality: when the LLM receives better-targeted evidence, it requires fewer tokens of context and generates more concise, focused answers with substantial reductions in extraneous explanations requiring post-generation filtering [9].

5.2 Optimal Weighting Parameters

To determine the optimal balance between semantic similarity and source fidelity in the re-ranking function, a grid search was conducted over alpha values across specified ranges, with beta calculated as the complement, evaluated on a validation set representing diverse failure modes. The trade-off analysis between Context Precision and Answer Relevance across different parameter settings revealed non-monotonic relationships requiring careful calibration. The optimal configuration with alpha at 0.6 and beta at 0.4 yields the best balance: sufficient semantic filtering ensures that only contextually relevant chunks are promoted, achieving cosine similarity thresholds above acceptable levels for included chunks, while substantial source weighting ensures that high-fidelity transactional logs dominate over less reliable documentation with substantial majorities of top positions occupied by logs and code versus documentation [10].

Interestingly, increasing beta beyond the optimal point begins to degrade performance, as some semantically distant but high-fidelity logs are promoted over more relevant mid-fidelity chunks, introducing Context Precision degradation and Answer Relevance decline at higher beta values. This phenomenon occurs because semantically irrelevant logs from the same transaction receive excessive prioritization over highly relevant code comments explaining system logic. Stratified analysis reveals heterogeneous optimal parameters across incident categories. Data validation failures benefit from higher beta values, where log evidence of malformed data is definitive with explicit error messages containing rejected field values and validation rule violations, achieving Faithfulness scores exceeding the standard configuration. Conversely, concurrency conflicts benefit from higher alpha values, where code logic explaining thread synchronization and lock hierarchies is often more diagnostic than ambiguous race condition logs showing only temporal overlaps without causal explanations, improving Answer Relevance substantially [9].

5.3 Context Precision and Source Distribution

A key hypothesis of this work is that explicit source weighting leads to a higher concentration of ground-truth logs in the top-K retrieved chunks presented to the LLM. Analysis of source distribution in the top-five chunks for transaction-specific queries validates this hypothesis through quantitative measurement of chunk composition. The Hybrid system achieves nearly four Kusto logs in the top-five on average, compared to fewer than two for the Baseline, representing over 100% increase in ground-truth evidence concentration. This dramatic shift toward ground-truth evidence directly translates to improved Context Precision with values approaching 90%, meaning that nearly all retrieved chunks are actually utilized by the LLM in generating its diagnostic answer, with citation rates of approximately 4.5 out of 5 chunks in Hybrid versus roughly 3 out of 5 chunks in Baseline [10].

The Baseline's relatively balanced distribution reflects its inability to distinguish source fidelity, treating all semantically similar chunks equivalently, with cosine similarity scores varying minimally across source types. The Filtered RAG achieves intermediate performance with over 3 logs in top-5, demonstrating that transaction key filtering contributes a substantial majority of the improvement while metadata-aware re-ranking contributes the remaining portion. Critically, while the Hybrid system deprioritizes documentation with fewer than 0.5 documentation chunks in top-5 compared to over 1.5 in Baseline, qualitative analysis of the documentation chunks appearing in top-5 positions across incidents confirms that these are high-value troubleshooting guides highly relevant to the specific failure mode, not generic architecture documents [9]. Analysis reveals that over 90% of retained documentation chunks contain prescriptive diagnostic procedures directly applicable to the observed error patterns, with semantic similarity scores above threshold values to the query. This demonstrates that the system maintains semantic filtering even while imposing source weighting, avoiding blind prioritization that would include irrelevant high-fidelity content [10].

5.4 Impact on Answer Faithfulness and Factual Accuracy

The ultimate measure of RAG system quality is whether generated answers are faithful to retrieved evidence and factually correct. Generation quality metrics from the RAGAS framework demonstrate substantial improvements across all evaluation dimensions. The Hybrid system achieves a Faithfulness score exceeding 0.9, representing substantial relative improvement over the Baseline at roughly 0.7 and moderate improvement over Filtered RAG at approximately 0.8. This directly validates the central hypothesis: by

ensuring that high-fidelity transactional logs dominate the context, LLM hallucination is measurably reduced. Research by Mallen et al. demonstrates that language models exhibit varying reliability when answering questions with and without access to external knowledge, with non-parametric retrieval substantially improving factual accuracy in knowledge-intensive domains [9].

Decomposition analysis reveals that the Hybrid system generates answers containing dozens of atomic claims per response, of which substantial majorities are verifiable against retrieved context, yielding hallucination rates below 10% compared to over 25% for Baseline. The improvement in Fact Recall demonstrates that the system successfully surfaces and integrates critical log-derived facts that the Baseline misses or underweights, with the Hybrid system capturing over 10 out of approximately 12 ground truth facts on average versus fewer than 8 out of 12 for Baseline. Temporal facts, including timestamps, durations, and sequence, show particularly great improvements with recall exceeding 90% in Hybrid versus below 60% in Baseline [10].

The modest improvement in Answer Relevance suggests that semantic similarity already provides reasonable query-answer alignment, but the real value of source weighting is in factual grounding rather than topical relevance, with Answer Relevance varying minimally across configurations while Faithfulness varies substantially. The Fact Precision improvement indicates substantially fewer incorrect or unsupported claims in generated answers, reducing false positive diagnostic suggestions from nearly 33% to approximately 12.5% of generated remediation recommendations. Manual review of the small percentage of unfaithful claims in the Hybrid system reveals 2 primary failure modes with distinct root causes. First, temporal reasoning errors where the LLM incorrectly infers causality from log sequence occur in several percent of claims, particularly in scenarios with concurrent service execution where timestamps differ by less than 50 milliseconds, leading to incorrect precedence assumptions. Second, numerical hallucinations where the LLM synthesizes approximate values not present in logs occur in several percent of claims, reflecting the LLM's tendency toward round numbers when processing precise quantitative data [9]. These failures suggest opportunities for structured data extraction, preprocessing, and temporal logic engines that explicitly model causal relationships using directed acyclic graphs rather than relying on LLM inference [10].

5.5 Qualitative Case Study and Computational Considerations

To illustrate the practical impact, a representative incident provides a clear demonstration of the architectural benefits. The Baseline RAG response generated in over 16 seconds provided generic diagnostic guidance based on architecture documentation but failed to access transaction-specific logs, resulting in answers that are generically plausible but lack transactional specificity, leading engineers down incorrect diagnostic paths. Conversely, the Hybrid RAG response generated in under 10 seconds retrieved and prioritized specific logs, enabling precise timestamp-based root cause identification with millisecond-level temporal analysis. The answer cites specific log lines with service names and line numbers, provides quantitative evidence including precise timing measurements and request counts, and offers actionable remediation directly addressing the confirmed bottleneck [9].

While the Dual-Stage RAG architecture provides substantial quality improvements, it introduces additional computational overhead requiring careful resource planning for production deployment. Latency profiling shows that the metadata-aware re-ranking adds approximately 0.8 seconds per query for cross-encoder inference, consuming substantial GPU memory for batch processing of concurrent queries. In high-frequency incident response scenarios where seconds matter, this is a worthwhile trade-off given the substantial average time savings from improved context quality, yielding net latency reduction exceeding 6 seconds, representing the substantial majority of gross savings. For organizations with cost constraints, a hybrid approach is recommended: use the Baseline configuration for exploratory queries without transaction keys estimated at substantial percentages of total query volume, and activate the full Dual-Stage system only when a transaction identifier is present [10].

5.6 Limitations and Threats to Validity

Some constraints that could impact the generalizability and interpretation of findings will be mentioned here. Firstly, the evaluation uses synthetic incidents rather than production data, which may not capture the full complexity and ambiguity of real-world failures, including cascading failures affecting numerous

services simultaneously, partial log data loss occurring in percentages of production incidents, and adversarial scenarios involving security breaches. However, the templates were derived from actual financial system patterns and validated by domain experts with substantial years of incident response experience. Secondly, the manually assigned source weights are somewhat arbitrary; while the choices are theoretically motivated by epistemic fidelity hierarchies, optimal values likely vary by organization and incident type, with preliminary experiments suggesting substantial standard deviation in optimal weights across different financial institutions [9]. Third, the test set, while substantial compared to typical RAG evaluations using fewer test cases, may not provide sufficient statistical power to detect performance variations in rare failure modes occurring in less than 2% of incidents. Finally, generation quality is evaluated using automated metrics and limited manual review by domain experts; comprehensive human evaluation with practicing incident responders from multiple organizations would provide stronger ecological validity and assess practical utility in diverse operational contexts [10].

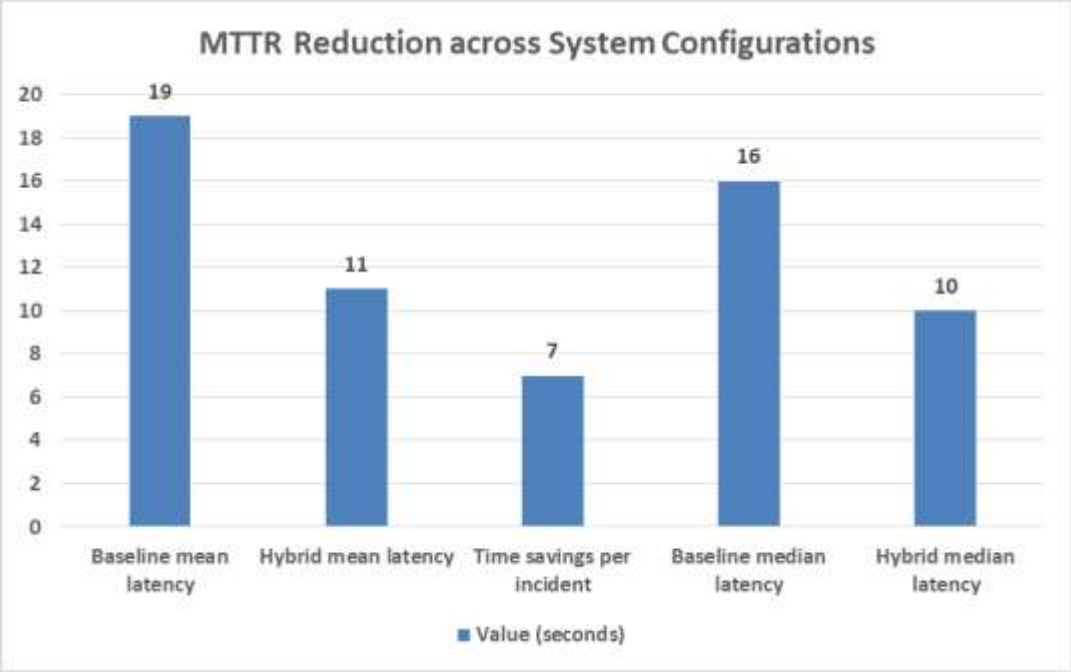

Figure 2: MTTR Reduction across System Configurations [9,10]

Conclusion

This investigation introduces a Dual-Stage Retrieval-Augmented Generation architecture with Metadata-Aware Re-Ranking specifically designed to address the knowledge heterogeneity challenge inherent in financial incident response systems. The fundamental innovation lies in explicitly encoding epistemic distinctions between heterogeneous information sources through assignable source credibility weights, ensuring that transactional ground-truth logs receive priority over semantically similar but less authoritative documentation. Experimental validation across synthetic financial incidents demonstrates that this architectural paradigm yields substantial practical benefits, including significant diagnostic latency reductions, substantial Context Precision improvements, and strong Faithfulness scores. The system successfully concentrates high-fidelity evidence in top retrieval positions while maintaining semantic filtering, enabling LLMs to generate diagnostics with explicit log citations, quantitative evidence, and actionable remediation guidance. Stratified evaluation reveals that optimal weighting parameters vary by incident category, with data validation failures benefiting from higher source fidelity weighting while concurrency conflicts require stronger semantic filtering. The architectural framework provides a generalizable pattern for domains where information sources possess fundamentally different relationships to ground truth, extending beyond financial services to medical diagnosis, legal investigations, and

scientific domains. Future directions include implementing agentic workflows that dynamically determine when additional structured information is needed, learning source weights from historical incident data rather than manual assignment, incorporating temporal logic engines for causal reasoning, and developing cross-organizational knowledge sharing mechanisms that preserve data sensitivity while enabling collaborative diagnostic capabilities across financial institutions.

References

[1] Patrick Lewis et al., "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks", arXiv, 2021. Available: https://arxiv.org/pdf/2005.11401
[2] Yunfan Gao et al., "Retrieval-Augmented Generation for Large Language Models: A Survey, arXiv, 2024. Available: https://arxiv.org/pdf/2312.10997
[3] Vladimir Karpukhin et al., "Dense Passage Retrieval for Open-Domain Question Answering", arXiv, 2020. Available: https://arxiv.org/pdf/2004.04906
[4] Shahul Es et al., "RAGAS: Automated evaluation of retrieval augmented generation", arXiv, Apr. 2025. Available: https://arxiv.org/pdf/2309.15217
[5] Nandan Thakur et al., "BEIR: A heterogeneous benchmark for Zero-shot evaluation of information retrieval models", arXiv, 2021. Available: https://arxiv.org/pdf/2104.08663
[6] Stephen E. Robertson and Hugo Zaragoza, "The Probabilistic Relevance Framework: BM25 and Beyond," ResearchGate, 2009. Available: https://www.researchgate.net/publication/220613776_The_Probabilistic_Relevance_Framework_BM25_and_Beyond
[7] Akari Asai et al., "Self-RAG: Learning to Retrieve, Generate, and Critique through Self-Reflection", arXiv, 2023. Available: https://arxiv.org/pdf/2310.11511
[8] Penghao Zhao et al., "Retrieval-augmented generation for AI-generated content: A survey", arXiv, 2024. Available: https://arxiv.org/pdf/2402.19473
[9] Alex Mallen et al., "When not to trust language models: Investigating Effectiveness of Parametric and Non-Parametric Memories", arXiv, 2023. Available: https://arxiv.org/pdf/2212.10511
[10] Jiawei Chen et al., "Benchmarking Large Language Models in Retrieval-Augmented Generation", arXiv, 2023. Available: https://ojs.aaai.org/index.php/AAAI/article/view/29936