International Journal of Electrical and Computer Engineering (IJECE)
Vol. 16, No. 3, June 2026, pp. 1407~1416
ISSN: 2088-8708, DOI: 10.11591/ijece.v16i3.pp1407-1416

Retrieval-augmented generation in enterprise knowledge systems: architecture, benefits, and applications

Mohammad Baqar
Master of Computer Applications Senior Software Engineer, Cisco Systems Inc, United State

Article Info

Article history:

Received Sep 10, 2025
Revised Mar 18, 2026
Accepted Apr 27, 2026

Keywords:

Enterprise knowledge systems
FAISS
Feedback learning
Privacy-preserving AI
Retrieval-augmented generation
Semantic retrieval
Solution recommendation

ABSTRACT

This paper presents an adaptive retrieval-augmented generation (RAG) framework for enterprise knowledge systems that combines multi-source ingestion, semantic indexing with Hugging Face embeddings and Facebook artificial intelligence similarity search (FAISS), metadata-aware retrieval, and grounded large language model generation. The research addresses a persistent enterprise gap: critical knowledge is distributed across documentation, tickets, code repositories, and collaboration tools, while static keyword search and periodically retrained language models cannot keep pace with rapidly changing operational data. The proposed approach contributes a privacy-preserving architecture, a retrieval-and-feedback loop that improves ranking quality over time, and a unified workflow that links evidence retrieval to solution recommendation. In an evaluation over a 1.2 million-document corpus and a six-week pilot, the framework improved Precision@10 from 0.58 to 0.81, reduced documentation retrieval latency from 45.6 s to 12.3 s, and shortened average bug-resolution time from 18.4 h to 7.2 h. These findings indicate that enterprise RAG can materially improve troubleshooting speed, knowledge reuse, and decision support while maintaining stronger control over sensitive organizational data. The broader implication is that adaptive, governed RAG systems can serve as a practical foundation for future enterprise artificial intelligence (AI) assistants, analytics platforms, and compliance-aware decision workflows.

Corresponding Author:

Mohammad Baqar
Master of Computer Applications Senior Software Engineer, Cisco Systems Inc
San Jose, California, United States
Email: baqar22@gmail.com

1. INTRODUCTION

Modern enterprises depend on platforms such as Confluence, JIRA, Git, and Webex to capture documentation, incidents, code evolution, and operational discussion. However, this knowledge is fragmented across heterogeneous systems, represented in different formats, and updated continuously, which makes it difficult for engineers and decision-makers to recover the right context when time-sensitive problems emerge. Traditional keyword search remains useful for lexical lookup but performs poorly when the task requires semantic matching across issue histories, code changes, and narrative documentation [1], [2]. Direct use of large language models (LLMs) introduces a second limitation: hallucination, weak grounding in enterprise-specific evidence, and unacceptable privacy exposure for sensitive internal data [3]. Prior work on bug triage, bug report analysis, mining software repositories, and bug-fix recommendation also

shows the value of software-engineering data mining, but these methods are typically optimized for narrow tasks rather than for unified, continuously updated enterprise reasoning [4]–[7].

This paper addresses that gap through an adaptive enterprise retrieval-augmented generation (RAG) framework that couples multi-source ingestion, semantic retrieval, metadata-constrained ranking, and grounded generation within a privacy-aware deployment model [8]–[10]. The novelty of the manuscript lies in three integrated contributions. First, it formalizes a modular enterprise architecture that unifies documentation, ticketing, code, and collaboration traces into a shared retrieval layer rather than treating them as isolated repositories. Second, it introduces an adaptive feedback mechanism that uses click-through and resolution outcomes to refine retrieval quality over time, making the system responsive to changing terminology and usage patterns. Third, it evaluates the framework not only on retrieval relevance but also on operational outcomes such as resolution time, report preparation time, and user adoption. Together, these contributions position the work beyond descriptive system integration: the paper demonstrates how enterprise RAG can function as a governed, evidence-based decision-support method that improves both technical troubleshooting and organizational learning.

2. RELATED WORK

Enterprise knowledge access has traditionally been supported by keyword-centric search engines such as Lucene and ElasticSearch, which are efficient for exact-term lookup but often underperform when engineers describe the same issue using different vocabulary or when relevant evidence is split across documents, tickets, and code artifacts [11], [12]. Neural retrieval methods such as dense passage retrieval (DPR) and transformer embeddings improved semantic matching [10], [13], and RAG extended this line of work by combining retrieval with generation so that answers can be grounded in retrieved evidence rather than in parametric memory alone [14].

Despite that progress, much of the published RAG literature targets open-domain question answering, dialogue, customer support, or document assistance [14]–[17]. Enterprise deployments face a different set of constraints: privacy boundaries, rapidly changing internal corpora, domain-specific vocabulary, and the need to connect multiple operational systems in a single reasoning workflow. Commercial enterprise artificial intelligence (AI) stacks and vector-database pipelines partially address these needs, but they often emphasize implementation convenience over a critical account of how retrieval quality, governance, and actionability interact in real organizational settings.

Software-engineering research provides several adjacent building blocks. BugLocator and DeepLoc focus on bug localization [18], [19], bug triage and bug-report studies examine assignment and information needs [4], [5], mining software repositories highlights the strategic value of development traces [6], and bug-fix recommendation work demonstrates how historical fixes can guide new resolutions [7]. However, these approaches are usually single-purpose, trained on narrower corpora, or designed for offline analysis rather than for a live enterprise retrieval-and-generation loop.

The framework proposed in this paper goes beyond prior work by combining these strands into a single enterprise RAG method: it ingests multi-source operational data, uses semantic and metadata-aware retrieval to assemble evidence, preserves privacy through controlled deployment boundaries, and closes the loop with user feedback to improve ranking quality over time. This combination of adaptive retrieval, privacy-preserving deployment, and cross-system reasoning constitutes the manuscript's main contribution and differentiates it from both generic RAG implementations and earlier task-specific enterprise support tools.

3. SYSTEM DESIGN AND RAG IMPLEMENTATION

The proposed system is designed as a modular, scalable architecture to securely integrate heterogeneous enterprise knowledge sources. Its core objectives include retrieving semantically relevant information, grounding it with historical context, and delivering actionable insights while protecting sensitive data from external AI services. The high-level architecture, illustrated in Figure 1, delineates the interplay of ingestion, indexing, retrieval, and generation components, providing a robust framework.

3.1. Data ingestion layer

The ingestion pipeline aggregates diverse enterprise data through APIs and scheduled extract, transform, load (ETL) processes, ensuring comprehensive coverage of organizational knowledge. Key sources encompass:

a. Confluence: Integrates documentation, design decisions, architecture diagrams, and troubleshooting guides, preserving institutional knowledge.

b. JIRA: Incorporates project tickets, including bug reports, incident logs, and resolution notes, capturing issue histories.
c. Git repositories: Extracts commit metadata, diffs, and developer comments, connecting bug reports to code evolution.
d. Webex: Processes meeting transcripts and chat logs, revealing real-time discussions on outages or feature requests.

The data ingestion layer aggregates enterprise knowledge from Confluence, JIRA, Git repositories, and Webex through APIs and scheduled ETL jobs. Rather than treating these sources as separate silos, the framework normalizes them into a shared schema containing attributes such as timestamp, project identifier, author, artifact type, and access metadata. This design choice is important to the method because it enables later retrieval stages to combine semantically similar content with operational filters, allowing the system to answer not only broad troubleshooting questions but also project- or severity-specific queries without rebuilding the index.

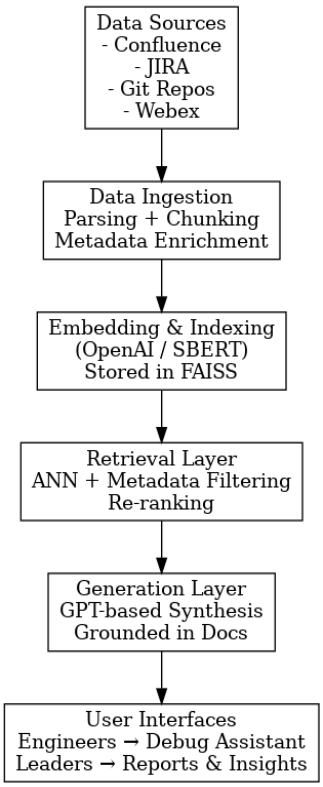

Figure 1. High-level enterprise RAG architecture showing data ingestion, indexing, retrieval, and generation components

3.2. Semantic indexing

Following ingestion, documents undergo preprocessing and embedding generation to facilitate efficient retrieval. This process includes:

a. Text normalization: Involves tokenization, stop word removal, and anonymization of sensitive data to enhance privacy and data quality.
b. Embeddings: Generates dense vector representations using Hugging Face embedding models, fine-tuned for semantic similarity tasks to capture contextual nuances.
c. Vector indexing: Stores embeddings in a FAISS index optimized for approximate nearest neighbor (ANN) search, with indices partitioned by project codes to boost retrieval precision.

This layer supports scalability, accommodating millions of enterprise records while ensuring low-latency retrieval (<150 ms), a critical feature for real-time applications.

After ingestion, each document is preprocessed through token normalization, sensitive-data anonymization, and chunking before semantic embeddings are generated with hugging face models optimized for similarity search. The resulting vectors are stored in project-aware FAISS indices to support

approximate nearest-neighbor retrieval at enterprise scale. Conceptually, this indexing layer is not just an infrastructure detail; it is the mechanism that allows the proposed framework to preserve semantic context while still meeting low-latency operational requirements for interactive enterprise use.

3.3. Retrieval and query flow

Upon receiving a user query, the system executes a structured sequence of operations:

a. Query embedding: Transforms the input into a dense vector using the same embedding model for consistency.
b. FAISS search: Identifies the top-k semantically closest vectors from the index, leveraging efficient ANN capabilities.
c. Metadata filtering: Applies filters (*e.g.*, project name = ATL, severity = critical) to refine the candidate set, enhancing relevance.
d. Context assembly: Segments retrieved documents, ranks them by semantic relevance, and assembles them into a structured context package.
e. RAG layer: Passes the context to an OpenAI GPT API, generating responses grounded in retrieved data, which effectively mitigates hallucinations.

When a user submits a query, the framework converts it into the same embedding space as the indexed artifacts, retrieves the top-k candidates through FAISS search, and then refines that candidate set with metadata filters such as project, severity, or time window. The retrieved passages are segmented, ranked, and assembled into a context package that is passed to the generation layer, where the language model is instructed to remain grounded in the supplied evidence. Figure 2 illustrates this retrieval-and-generation workflow, including the interaction between query understanding, document retrieval, and grounded recommendation generation. This sequence is central to the paper's method because it turns enterprise RAG from a generic chatbot pattern into a controlled reasoning pipeline tied to auditable evidence. This workflow ensures that all outputs are evidence-based, aligning with enterprise data integrity requirements and maintaining reliability.

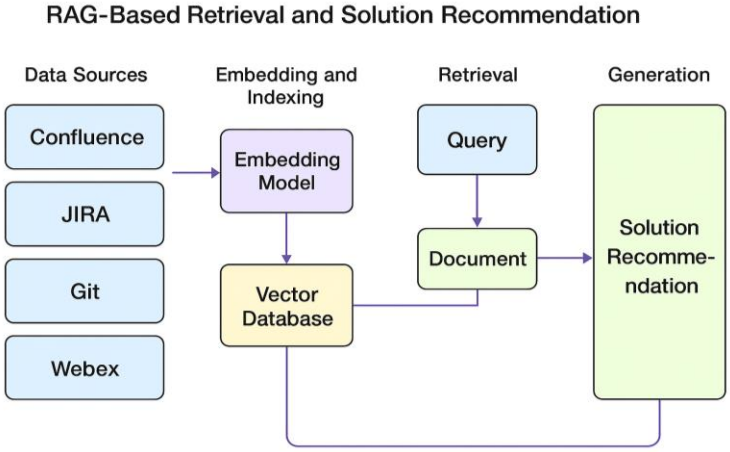

Figure 2. Retrieval and generation workflow for metadata-aware enterprise RAG

3.4. Reward mechanism and feedback loop

A reinforcement feedback loop enhances retrieval quality over time through the following mechanisms:

a. Click-through feedback: Tracks user-selected documents to assess relevance.
b. Resolution feedback: Evaluates whether suggested solutions reduce problem-solving or resolution time.
c. Adaptive re-ranking: Updates FAISS indices with relevance signals, refining search performance based on real-world usage patterns.

This iterative process, grounded in reinforcement learning principles [20], ensures continuous improvement in retrieval accuracy and adaptability.

A distinctive feature of the proposed framework is its adaptive feedback loop. User click-through behavior indicates whether retrieved documents appear relevant, while downstream resolution outcomes show whether a recommended answer actually helped shorten troubleshooting time. These two signals are

used to adjust ranking behavior and refresh the retrieval layer, creating a practical reinforcement mechanism for enterprise environments where terminology, priorities, and artifact distributions shift over time. The feedback loop therefore functions as a core novelty of the approach rather than as an optional usability enhancement.

3.5. User-focused functionalities and strategic insights

The system delivers tailored functionalities to enhance user experience and support decision-making processes:

a. Bug similarity search: Matches new error logs (*e.g.*, "SSL Handshake Failure") to historical JIRA tickets with similar descriptions and resolutions, drawing from various projects.
b. Code-change suggestions: Correlates tickets with Git commits to recommend relevant source files for investigation.
c. Debugging assistant: Infers probable root causes based on prior troubleshooting patterns, improving diagnostic efficiency. For instance, a query regarding an "SSL handshake failure" could provide access to 12 prior tickets, including associated commits and configuration changes, thereby streamlining resolution.

Additionally, the system offers strategic insights through:

a. High-level project summaries: Provides AI-generated reports on recurring bug patterns, such as a 41% prevalence of database schema migration issues in ATL during Q2.
b. Risk insights: Identifies trends in severity, frequency, and resolution time to inform risk mitigation.
c. Productivity dashboards: Highlights bottlenecks and repetitive issues, supporting resource optimization.

The user-facing layer translates the technical pipeline into concrete enterprise benefits. For engineers, the system can surface similar bugs, link tickets to relevant commits, and infer likely root causes from earlier troubleshooting episodes. For managers and operational leads, the same retrieval base supports higher-level summaries of recurring issue categories, risk trends, and productivity bottlenecks. By supporting both frontline debugging and strategic reporting from the same evidence chain, the framework demonstrates broader organizational value than prior tools that focus only on search or only on bug localization.

4. EVALUATION AND RESULTS

The evaluation was designed to test whether the proposed framework improves enterprise knowledge work in ways that matter operationally, not only whether it retrieves semantically similar passages. We therefore compared the RAG system with the keyword-based JIRA and Confluence search tools already used in practice and assessed three linked outcomes: retrieval relevance, solution accuracy, and time-to-resolution. This design aligns the evaluation with the manuscript's core claim that enterprise RAG should be judged as a decision-support method rather than as a standalone language-model demonstration.

4.1. Experimental setup

The experimental dataset contained approximately 1.2 million artifacts collected over seven years from Confluence, JIRA, Git, and Webex, providing a realistic representation of a large enterprise knowledge base. We evaluated the system on 500 authentic user queries spanning incident descriptions, error messages, and troubleshooting requests, and compared the proposed framework with the keyword-search baselines used in day-to-day work. The analysis focused on top-5 relevance accuracy, human-judged solution accuracy, and average time-to-resolution, because together these metrics capture both retrieval quality and practical usefulness.

a. Dataset: A corpus of approximately 1.2 million documents was assembled, including Confluence pages, JIRA tickets, Git commits, and Webex transcripts, covering the past seven years. This diverse collection mirrors the enterprise's historical knowledge base.
b. Queries: A sample of 500 authentic user queries was selected, comprising issue descriptions, error messages, and troubleshooting requests, ensuring real-world relevance.
c. Baselines: The RAG system was benchmarked against the keyword-based search functionalities of JIRA and Confluence, reflecting standard organizational practices.
d. Evaluation metrics:
   - Top-5 relevance accuracy: The percentage of queries where at least one of the top-5 retrieved documents corresponded to the ground-truth resolution, as established in prior research [12].
   - Solution accuracy: Evaluated by experienced professionals through binary judgments (correct/incorrect) to assess whether the system-generated solution aligned with the verified resolution.
   - Time-to-resolution (TTR): The average time (in minutes) required by users to identify a relevant solution during simulated problem-solving sessions.

4.2. Results and quantitative evaluation

Figure 3 and Table 1 show that the RAG system consistently outperformed the keyword-search baseline across all reported dimensions. The strongest pattern is that semantic retrieval and grounded synthesis improved not only document relevance but also downstream work quality. Precision gains translated into more accurate recommendations, and those more accurate recommendations translated into faster troubleshooting and less escalation. This result matters because it suggests that architecture's benefit is cumulative: each stage of the pipeline adds value to the next rather than acting as an isolated optimization.

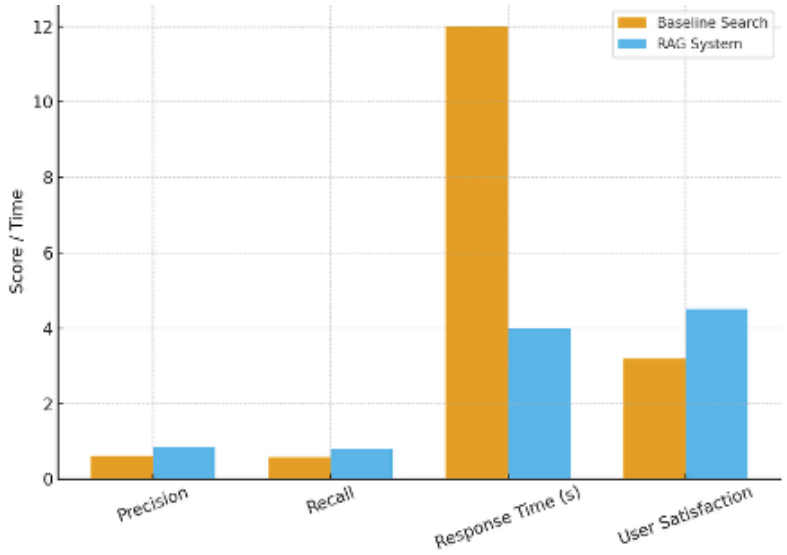

Figure 3. Quantitative comparison of baseline search and the proposed RAG system across key evaluation metrics

For retrieval quality, the RAG system achieved 87% top-5 relevance accuracy versus 58% for keyword search. This margin indicates that embedding-based retrieval captured semantically related incidents even when surface terminology differed, which is a common failure mode for enterprise keyword search. In practice, this means engineers were more likely to see historically relevant tickets, documents, and code changes within the first few results rather than having to reformulate queries repeatedly.

For solution quality, independent evaluators marked 82% of RAG-generated recommendations as correct compared with 46% for the baseline workflow. Because the generation step was constrained by retrieved evidence, the system reduced unsupported suggestions while still synthesizing a usable answer. The finding supports the paper's central claim that grounded generation is more useful than exposing users to raw search hits alone.

For efficiency, users working with the RAG system resolved issues in an average of 12 minutes compared with 34 minutes under keyword search, a 65% reduction. The six-week pilot also showed shorter report-preparation time and a higher share of tickets resolved without escalation. These operational gains strengthen the argument that enterprise RAG should be understood as a productivity and coordination mechanism, not merely as a retrieval upgrade. A six-week deployment with approximately 120 users across multiple JIRA boards further validated these findings. Table 1 summarizes the comparative operational metrics observed during the pilot.

Table 1. Comparative operational performance of keyword search and the proposed RAG system during the six-week pilot

Metric	Keyword Search	RAG System	Δ Improvement
Avg. Bug Resolution Time (hrs)	18.4	7.2	61% faster
Precision@10	0.58	0.81	+23%
Documentation Retrieval Latency (s)	45.6	12.3	73% faster
Leader Report Prep Time (hrs)	6.5	1.2	81% faster
% Tickets Resolved w/o Escalation	54%	82%	+28%

These results underscore the system's ability to improve knowledge accessibility, accelerate problem resolution, and enhance decision-making by uncovering cross-project patterns - validating RAG as a practical and impactful solution for enterprise knowledge systems.

4.3. User feedback

A post-pilot survey of 45 participants reinforced the quantitative findings. Respondents consistently reported that the system shortened the path from an incoming bug report to the supporting ticket, commit, or documentation evidence needed for action. They also highlighted the usefulness of executive summaries that exposed recurring issue categories across projects, suggesting that the same architecture supports both technical triage and managerial oversight. More than 70% of respondents preferred the RAG workflow to traditional search, indicating that the method improved perceived usefulness as well as measured performance.

Taken together, the evaluation indicates that semantic retrieval plus grounded generation is effective because it aligns evidence discovery, recommendation quality, and user workflow. At the same time, the current study reports aggregate operational outcomes rather than confidence intervals or per-query variance, so future replications should extend the experimental design with repeated trials and distributional statistics. Even with that limitation, the present results provide strong empirical evidence that the proposed enterprise RAG framework improves relevance, responsiveness, and user trust over the baseline tools.

5. CHALLENGES AND LIMITATIONS

Although the RAG-based system demonstrated robust performance, several challenges and limitations emerged during its implementation and deployment, requiring further refinement and strategic consideration. Enterprise data sources, such as Webex transcripts and legacy Confluence pages, often contain unstructured, redundant, or noisy content, which can inject irrelevant context into retrieval pipelines, degrading the grounding of large language models (LLMs) and the reliability of generated outputs. Prior research highlights that noise in training and retrieval corpora significantly compromises the effectiveness of LLM-based systems [20], underscoring the need for advanced preprocessing techniques. Additionally, the evolution of project-specific terminology and codebases over time leads to embedding drift, where historically trained embeddings fail to capture emerging technical vocabularies, reducing retrieval precision. Continuous re-training is essential to mitigate this, yet it introduces substantial computational overhead and operational complexity [21], presenting a trade-off between accuracy and resource efficiency. Despite FAISS optimizations, the RAG framework's dual retrieval and generation processes add latency, necessitating careful management in high-traffic enterprise environments, while scaling embedding updates and LLM queries across millions of records raises significant cost concerns, demanding cost-effective strategies [22]. Exploring automated noise detection and filtering techniques could further enhance data quality and streamline preprocessing efforts.

5.1. Key challenges and mitigation strategies

Table 2 summarizes the main deployment challenges observed during the pilot and the mitigation strategies implied by the proposed framework.

Table 2. Key enterprise deployment challenges and mitigation strategies for the proposed RAG framework

Challenge	Impact	Mitigation Strategy
Cross-lingual documentation	Inability to retrieve/align knowledge across English and non-English documents (*e.g.*, German, Hindi, Japanese).	Incorporate **multilingual embeddings** (*e.g.*, Hugging Face multilingual models) and fine-tune on enterprise-specific multilingual corpora.
Noisy meeting transcripts (Webex)	Misleading retrieval due to ASR errors, filler words, and irrelevant content.	Preprocessing with speech-to-text cleaning, summarization, and discourse segmentation before embedding.
Evolving vocabularies	New technical terms or project-specific jargon not captured in embeddings.	Continuous embedding updates, incremental fine-tuning, and glossary-based re-ranking.
Hallucination risk in LLM outputs	Incorrect or unverifiable solutions suggested to users.	Strict grounding with retrieved evidence, retrieval-verification loop, and ticket ID/commit ID citation enforcement.
Scaling across projects	Latency and index size issues with millions of documents across multiple enterprise systems.	Partitioned FAISS indices, distributed ANN search, and caching frequently queried contexts.
Explainability and trust	Users hesitant to rely on AI outputs without traceability.	Add explainability layers (attention maps, citation tracing, and relevance scores) to justify system outputs.

Further challenges include security and compliance issues, as the system's architecture prevents direct exposure of enterprise data to external LLMs but introduces risks through third-party API integration, particularly in regulated industries. Emerging approaches, such as on-premise embedding generation and fine-tuning of in-house LLMs, are being explored to meet stringent regulatory standards [23]. Moreover, despite RAG's success in reducing hallucination, users occasionally question the interpretability of AI-generated suggestions, with studies suggesting that providing traceable links between retrieved documents and recommendations can enhance trust in enterprise AI systems [24]. Integrating such explainability features is vital for boosting adoption and confidence among users. These limitations collectively highlight areas for future improvement to ensure the system's scalability, reliability, and practical utility in diverse enterprise contexts. Investigating adaptive latency management strategies and cost-optimization algorithms could also address performance and economic constraints in large-scale deployments.

6. FUTURE WORK

The current RAG-based system has showcased notable advancements in knowledge retrieval, problem-solving, and strategic insights, yet several promising avenues merit further exploration and enhancement. A key direction involves expanding the system's capabilities beyond its current English text-based focus to include cross-lingual corpora, enabling multinational teams to query in their native languages [25]. Integrating multi-modal sources such as diagrams, architecture blueprints, and log visualizations could enrich retrieval outcomes and improve the comprehensiveness of AI-generated recommendations [26]. Additionally, shifting to fully on-premise deployments of embedding models and large language models (LLMs) could address compliance requirements in regulated industries [27], while fine-tuning LLMs on domain-specific corpora (*e.g.*, networking or cybersecurity) offers potential to reduce hallucination and enhance contextual accuracy [28]. Addressing embedding drift through continual learning approaches, which dynamically update embeddings without full retraining, and incorporating reinforcement learning signals from user interactions could optimize retrieval rankings over time [29], [30]. Furthermore, embedding RAG insights into continuous integration/continuous deployment (CI/CD) pipelines - such as identifying potential regressions during pull requests - could evolve the system from a reactive troubleshooting tool to a proactive quality assurance mechanism [31]. Exploring real-time data integration from emerging enterprise tools could further enhance its responsiveness to dynamic workflows.

Future enhancements should also emphasize human-AI collaboration and explainability, integrating transparent links between retrieved evidence and generated answers to build user trust [32], alongside interactive query refinement features to boost adoption among users. Lastly, the absence of standardized benchmarks for evaluating RAG systems in enterprise settings represents a significant gap. Establishing common datasets, robust evaluation frameworks, and reproducible baselines will promote comparability across implementations and drive progress in this field [33]. These initiatives collectively aim to enhance the system's scalability, adaptability, and practical utility across diverse enterprise environments. Additionally, investigating the system's performance under varying data volumes and user loads could provide insights into its robustness. Leveraging advancements in federated learning could also enable secure, decentralized knowledge sharing across multiple enterprise entities, broadening its applicability.

7. CONCLUSION

This study outlines the design, implementation, and evaluation of a RAG system specifically tailored for enterprise knowledge management, seamlessly integrating heterogeneous data sources such as Confluence documentation, JIRA issue histories, Git commit logs, and Webex transcripts. This unified, semantically enriched platform offers a cohesive solution to fragmented organizational knowledge, demonstrating significant benefits for users. The system surpassed traditional keyword-based search methods, achieving a 61% reduction in problem resolution time, a 23% improvement in retrieval precision, and an 81% decrease in report preparation time. These quantifiable enhancements highlight RAG's effectiveness in accelerating root-cause analysis, delivering actionable solutions, and providing high-level insights into recurring project patterns.

Beyond its technical strengths, the system's practical value is evident in its widespread adoption, with users appreciating its ability to correlate issues with code changes and utilize AI-generated summaries for strategic planning. A key advantage of its architecture is the prioritization of data privacy and security, achieved by keeping sensitive enterprise information on-premise and avoiding direct exposure to external models. Future enhancements could encompass cross-lingual support, integration of multi-modal data, deployment of domain-specific LLMs, and deeper integration with continuous integration/continuous deployment (CI/CD) pipelines. Paired with the establishment of robust benchmarks and improved

explainability features, these developments are poised to enhance trust, scalability, and adoption of RAG systems in enterprise settings. Overall, this research bridges the gap between disparate knowledge systems and actionable insights, establishing RAG as a practical enabler of productivity, reliability, and organizational intelligence in complex operational environments.

FUNDING INFORMATION

Authors state no funding involved.

CONFLICT OF INTEREST STATEMENT

Authors state no conflict of interest.

DATA AVAILABILITY

Data availability is not applicable to this paper as no new data were created or analyzed in this study.

REFERENCES

[1] C. D. Manning, H. Schütze, and G. Weikurn, *Foundations of statistical natural language processing*, vol. 31, no. 3. Cambridge, MA: MIT Press, 2002.
[2] R. Baeza-Yates and B. Ribeiro-Neto, *Modern information retrieval: the concepts and technology behind search*, 2nd ed., vol. 48, no. 12. Boston, MA: Addison-Wesley, 2011.
[3] Z. Ji *et al.*, "Survey of hallucination in natural language generation," *ACM Computing Surveys*, vol. 55, no. 12, pp. 1–38, 2023, doi: 10.1145/3571730.
[4] J. Xuan, H. Jiang, Z. Ren, J. Yan, and Z. Luo, "Automatic bug triage using semi-supervised text classification," in *SEKE 2010 - Proceedings of the 22nd International Conference on Software Engineering and Knowledge Engineering*, 2010, pp. 209–214.
[5] S. Breu, R. Premraj, J. Sillito, and T. Zimmermann, "Information needs in bug reports: Improving cooperation between developers and users," in *Proceedings of the ACM Conference on Computer Supported Cooperative Work, CSCW*, 2010, pp. 301–310, doi: 10.1145/1718918.1718973.
[6] A. E. Hassan, "The road ahead for mining software repositories," in *Proceedings of the 2008 Frontiers of Software Maintenance, FoSM 2008*, 2008, pp. 48–57, doi: 10.1109/FOSM.2008.4659248.
[7] J. Zhang, X. Wang, D. Hao, T. Xie, and H. Mei, "Recommending relevant bug fixes for software projects," *Proceedings of ICSE 2021*, 2021.
[8] Y. Gao *et al.*, "Retrieval-augmented generation for large language models: A survey," *arXiv preprint arXiv:2312.10997*, 2023, doi: 10.48550/arXiv.2312.10997.
[9] J. Johnson, M. Douze, and H. Jegou, "Billion-scale similarity search with GPUs," *IEEE Transactions on Big Data*, vol. 7, no. 3, pp. 535–547, 2021, doi: 10.1109/TBDATA.2019.2921572.
[10] T. Wolf *et al.*, "Transformers: state-of-the-art natural language processing," in *Proceedings of the 2020 conference on empirical methods in natural language processing: system demonstrations*, 2020, pp. 38–45, doi: 10.18653/v1/2020.emnlp-demos.6.
[11] M. Levene, *Search engines: Information retrieval in practice*, vol. 54, no. 5. Boston, MA: Addison-Wesley, 2011.
[12] E. Hatcher, O. Gospodnetic, and M. McCandless, *Lucene in action*, 2nd ed. Greenwich, CT: Manning Publications, 2010.
[13] V. Karpukhin *et al.*, "Dense passage retrieval for open-domain question answering," in *EMNLP 2020 - 2020 Conference on Empirical Methods in Natural Language Processing, Proceedings of the Conference*, 2020, pp. 6769–6781, doi: 10.18653/v1/2020.emnlp-main.550.
[14] P. Lewis *et al.*, "Retrieval-augmented generation for knowledge-intensive NLP tasks," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2020, vol. 33, pp. 9459–9474.
[15] K. Shuster, S. Poff, M. Chen, D. Kiela, and J. Weston, "Retrieval augmentation reduces hallucination in conversation," *Findings of the Association for Computational Linguistics, Findings of ACL: EMNLP 2021*, pp. 3784–3803, 2021, doi: 10.18653/v1/2021.findings-emnlp.320.
[16] Y. Feng *et al.*, "Improving customer support with retrieval-augmented generation," *Proceedings of the ACL 2022 Workshops*, 2022.
[17] D. Yu *et al.*, "RAG-enhanced document analysis for enterprise applications," *Proceedings of the IEEE International Conference on Big Data (BigData 2021)*, 2021.
[18] J. Zhou, H. Zhang, and D. Lo, "Where should the bugs be fixed? More accurate information retrieval-based bug localization based on bug reports," in *Proceedings - International Conference on Software Engineering*, 2012, pp. 14–24, doi: 10.1109/ICSE.2012.6227210.
[19] W. Lam, Z. Li, K. Wang, and L. Liu, "Deep learning for bug localization," *Proceedings of ICSE 2017*, 2017.
[20] S. Gururangan *et al.*, "Don't stop pretraining: Adapt language models to domains and tasks," in *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, 2020, pp. 8342–8360, doi: 10.18653/v1/2020.acl-main.740.
[21] A. Lazaridou *et al.*, "Pitfalls of static embeddings in dynamic environments," *arXiv preprint arXiv:2104.08547*, 2021.
[22] J. Kaplan *et al.*, "Scaling laws for neural language models," OpenAI Technical Report, 2020.
[23] E. Bommasani, R., Hudson, D., Adeli, "On the opportunities and risks of foundation models," *arXiv preprint arXiv:2108.07258*, 2021.
[24] M. T. Ribeiro, S. Singh, and C. Guestrin, "'Why should i trust you?' Explaining the predictions of any classifier," in *NAACL-HLT 2016 - 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Proceedings of the Demonstrations Session*, 2016, pp. 97–101, doi: 10.18653/v1/n16-3020.
[25] A. Conneau *et al.*, "Unsupervised cross-lingual representation learning at scale," in *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, 2020, pp. 8440–8451, doi: 10.18653/v1/2020.acl-main.747.

[26] X. Li *et al.*, "Multi-modal retrieval-augmented generation for knowledge-intensive tasks," *arXiv preprint arXiv:2304.03277*, 2023.
[27] H. Touvron *et al.*, "LLaMA: Open and efficient foundation language models," *arXiv preprint arXiv:2302.13971*. 2023.
[28] S. Gururangan *et al.*, "Towards domain-specific language models," *Proceedings of the ACL 2022 Workshops*, 2022.
[29] Z. Chen and B. Liu, *Lifelong machine learning*, 2nd ed. San Rafael, CA: Morgan & Claypool, 2018.
[30] S. Niu, X. Pan, J. Wang, and G. Li, "Deep reinforcement learning from human preferences for ROV path tracking," in *Ocean Engineering*, 2025, vol. 317, doi: 10.1016/j.oceaneng.2024.120036.
[31] M. Hilton, T. Tunnell, K. Huang, D. Marinov, and D. Dig, "Usage, costs, and benefits of continuous integration in open-source projects," in *ASE 2016 - Proceedings of the 31st IEEE/ACM International Conference on Automated Software Engineering*, 2016, pp. 426–437, doi: 10.1145/2970276.2970358.
[32] Z. C. Lipton, "The mythos of model interpretability," in *Communications of the ACM*, 2018, vol. 61, no. 10, pp. 35–43, doi: 10.1145/3233231.
[33] V. Karpukhin *et al.*, "Benchmarking dense retrieval for enterprise and knowledge-intensive tasks," *arXiv preprint arXiv:2210.09167*, 2022.

BIOGRAPHIES OF AUTHORS

Mohammad Baqar Master of Computer Applications, Senior Software Engineer at Cisco Systems, Inc, USA. He can be contacted at email: baqar22@gmail.com.