Journal Publication of International Research for Engineering and Management (JOIREM)
Volume: 04 Issue: 06 | June-2026
ISSN (O) 3107-6696

TriFuseRAG: Tri-Modal Hybrid Retrieval-Augmented Generation Using SQL, Vector and Graph Databases

Buddha Mokshitha Sree, Jammula Vijay Krishna, Gamidi Sneha Sivani, Boddapu Chandra Shekhar, Padathala Visweswara Rao
Department of Artificial Intelligence and Data Science
Vignan's Institute of Information Technology, India
Corresponding Author: mokshithasree.rao@gmail.com

ABSTRACT

Retrieval-Augmented Generation (RAG) systems typically route all queries to a single retrieval modality, limiting utility for heterogeneous query types that require simultaneous structured lookup, entity-relationship traversal, and text similarity search. We present **TriFuseRAG**, a prototype multi-modal RAG system integrating three retrieval backends—a relational SQL engine, an in-memory knowledge graph, and a TF-IDF vector store—under a dual-stage adaptive router and weighted fusion layer. A two-tier safety gate filters adversarial inputs before retrieval. On a 1,126-query closed-world benchmark, the dual-stage router achieves 98.84% route accuracy. In strict no-leakage evaluation, TriFuseRAG achieves 91.3% overall answer accuracy (98.1% on supported queries), and 93.9% on composite Hybrid queries where all single-source baselines score 0%. Overall, TriFuseRAG outperforms the best single-source baseline by 34.7 points (91.3% vs. 26.6%), with p95 query latency under 10ms on CPU hardware. We report full error analysis, latency profiles, and discussion of synthetic evaluation scope.

***Key Words*:** Retrieval-Augmented Generation, Query Routing, Knowledge Graph QA, NL2SQL, Multi-Modal Retrieval, Safety Filtering

1. INTRODUCTION

Factual QA over private knowledge presents persistent challenges for LLMs: LLMs suffer from stale knowledge, limited private data access, and hallucinations. RAG mitigates this by grounding responses in retrieved evidence, but dominant implementations route every query through a single dense vector index regardless of intent.

In product-catalog settings, queries vary: price lookup (SQL), manufacturer (graph), warranty (vector), and composite queries require multiple modalities.

We present **TriFuseRAG**, addressing this gap through three design choices: (1) query-type-aware routing to the appropriate backend before retrieval; (2) a schema-aware NL2SQL template planner for structured retrieval without LLM inference; and (3) a pre-retrieval safety layer filtering adversarial inputs before any backend is invoked.

2. RELATED WORK

This section documents the research done in dense vector retrieval, the use of structured knowledge, and hybrid architectures.

RAG. Lewis et al. [2] introduced Retrieval-Augmented Generation by combining retrieval with sequence-to-sequence generation, while DPR [3] improved dense retrieval effectiveness for open-domain question answering. Subsequent work explored efficient retrieval frameworks [4] and comprehensive surveys of RAG architectures [5]. TriFuseRAG differs by performing

intent-aware routing before retrieval rather than relying solely on late-stage retrieval fusion.

Text-to-SQL. Spider [6] established a benchmark for complex cross-domain Text-to-SQL tasks, while RAT-SQL [7] and PICARD [8] improved schema-aware SQL generation and constrained decoding. In contrast, TriFuseRAG employs a lightweight template-based NL2SQL planner that prioritizes auditability and low deployment cost.

Graph QA. EmbedKGQA [9] and NSM [10] demonstrated effective multi-hop reasoning over knowledge graphs. TriFuseRAG incorporates graph retrieval through a storage-agnostic graph interface that can be extended to persistent graph databases without modifying routing or fusion mechanisms.

Heterogeneous Retrieval. HybridQA [11] and UnifiedQA [12] combine information from multiple sources but do not explicitly perform intent-aware retrieval routing. Recent GraphRAG systems leverage graph-structured retrieval to improve multi-hop reasoning and entity relationship understanding [20]. Agentic RAG approaches such as Self-RAG [21] and CRAG [22] further extend retrieval through self-reflection, retrieval evaluation, and corrective reasoning. Enterprise retrieval frameworks including LangGraph [23] and LlamaIndex [24] emphasize retrieval orchestration and heterogeneous knowledge integration across multiple data sources. Unlike these approaches, TriFuseRAG focuses on lightweight intent-aware routing across SQL, graph, and vector retrieval backends within a unified and auditable architecture.

Adaptive Retrieval and Routing. Recent retrieval-augmented systems increasingly emphasize adaptive retrieval strategies that dynamically select retrieval actions based on query characteristics. Adaptive-RAG [25] introduces complexity-aware retrieval policies that adjust retrieval depth according to the difficulty of incoming questions, reducing unnecessary retrieval overhead. RAPTOR [26] proposes recursive abstraction through hierarchical document summarization to improve long-context retrieval effectiveness. These approaches demonstrate the importance of retrieval adaptability; however, they primarily operate within text-centric retrieval pipelines. In contrast, TriFuseRAG performs explicit intent-aware routing across heterogeneous modalities including SQL databases, knowledge graphs, and vector stores, enabling efficient handling of structurally distinct query types.

Safety. Recent studies have demonstrated that retrieval-augmented systems remain vulnerable to prompt injection, retrieval poisoning, and adversarial query attacks [13], [14]. Unlike post-generation safety mechanisms, TriFuseRAG applies a two-tier safety gate before retrieval, preventing malicious inputs from influencing evidence selection and downstream response generation.

3. OVERALL SYSTEM ARCHITECTURE

Two-Tier Safety Gate:

Tier 1 matches Unicode-normalized queries against 24 precompiled regexes covering SQL injection, prompt injection, and command injection. A term-combination check additionally flags co-occurrence of destructive verbs with sensitive nouns covering natural-language attack phrasings missed by pattern matching.

Tier 2 applies a logistic regression binary classifier on character n-gram (3,4) TF-IDF features trained on 105 adversarial and augmented training queries, with a 0.35 threshold, covering formulations missed by fixed patterns.

Dual-Stage Query Router

Queries are classified into six labels. They are **SQL**, **Graph**, **Vector**, **Hybrid**, **Join**, **Nested**.

Journal Publication of International Research for Engineering and Management (JOIREM)
Volume: 04 Issue: 06 | June-2026
ISSN (O) 3107-6696

Stage 1 computes cosine similarity between the query's word-level TF-IDF vector and per-label centroids.

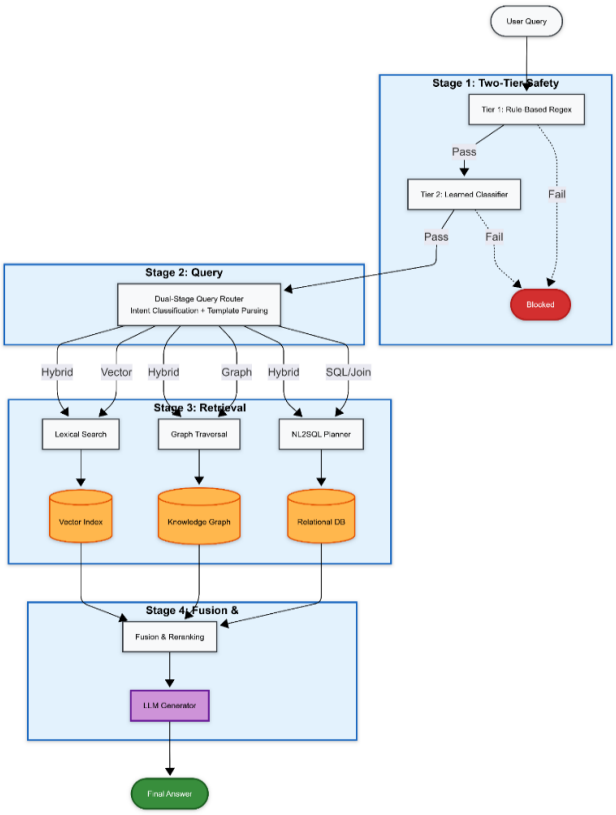

Fig -1: TriFuseRAG pipeline: two-tier safety gate, dual-stage router, specialized retrieval backends, and weighted multi-signal fusion.

Stage 2 applies logistic regression to classify query templates. Templates map to route labels via deterministic prefix rules and blend with Stage 1 (weights 0.65/0.35). Lexical rules override classifiers for structurally unambiguous cases.

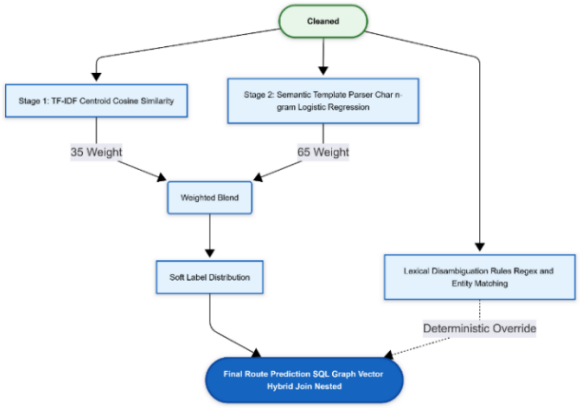

Fig -2: Dual-stage routing: Stage 1 centroid similarity + Stage 2 template classification, with lexical rule override.

Retrieval Backends

SQL. Entity slots (product name, manufacturer, category, price bounds) are extracted and bound to one of nine precompiled parameterized SQL templates executed against in-memory SQLite. Families cover: price lookup, category JOIN, manufacturer-price filter, products-with-categories JOIN, above-average-price subqueries (per-category and per-manufacturer), cross-manufacturer nesting, global average, and category count-threshold aggregation.

Graph. A bidirectional in-memory graph with Product, Manufacturer, Category nodes and typed edges uses dictionary-based indexing for efficient access. The storage-agnostic interface enables Neo4j/Dgraph upgrade without touching routing or fusion.

Vector. A word-level TF-IDF store indexes one document per product, returning top-K=3 results by cosine similarity.

Hybrid. All three backends are invoked and candidates pooled before fusion.

Candidate Fusion

score = 0.30·route_conf + 0.20·retrieval_score
+ 0.15·field_coverage + 0.35·reranker_score

A reranker predicts candidate–query alignment. The top-scoring candidate is returned; no-match returns NO_ANSWER to prevent hallucination.

4. IMPLEMENTATION

TriFuseRAG uses Python 3.13, NumPy, and scikit-learn with optional Gradio UI. All classifiers train at startup (~11 s, CPU-only) from the training split.

Table -1: Implementation Components

Component	Method	Library
Safety Tier 1	24 regexes + term-combination	stdlib re
Safety Tier 2	LR, char n-gram (3,4) TF-IDF	scikit-learn
Stage1 Router	TF-IDF centroid cosine	NumPy
Stage2 Router	LR, char n-gram (3,4) TF-IDF	scikit-learn
NL2SQL	9-family template planner	stdlib sqlite3
Graph	In-memory adjacency dict	stdlib
Vector Store	Word TF-IDF cosine	NumPy
Reranker	LR, char n-gram (3,4) TF-IDF	scikit-learn

Table -2: Measured Per-Query Latency (CPU, 10-run avg)

Query Type	Avg (ms)	p95 (ms)
SQL	3.19	~12
Graph	0.91	~4
Vector	0.84	~4
Hybrid	1.47	~6
Safety block	0.01	<1
All types	**1.29**	**9.1**

Routing-only: 0.001 ms avg. Safety Tier 1: 0.006 ms avg.

4.1 Dataset and Evaluation

4.1.1 Dataset Design Rationale

Our goal is to **isolate and measure** TriFuseRAG's architectural contributions intent-aware routing, multi-modal coordination, safety fusion rather than evaluate general knowledge retrieval. We construct a 1,126-query closed-world benchmark over a fixed product catalog where every answer is deterministically derivable from structured (SQL), relational (graph), or unstructured (text) sources. This design offers three advantages: *(i) Deterministic ground truth* eliminating annotation noise that would conflate routing errors with label ambiguity; *(ii) Explicit multi-modal balance* open-domain datasets are biased toward text and lack sufficient SQL/graph coverage to isolate routing and fusion behaviour; *(iii) Structural isolation*, the key claim is a structural property verifiable on a controlled benchmark without large-scale real-world data. Even small-scale real-world product datasets typically lack modality-aligned ground truth, most annotate only one field (price or title), making composite Hybrid ground truth infeasible without additional annotation effort, a challenge our deterministic catalog eliminates. Future work targets Spider for NL2SQL stress-testing and e-commerce query logs for real phrasing distributions.

4.1.2 Dataset Statistics

Table -3: Benchmark Dataset Composition

Category	Train	Val	Test	Total
SQL	214	46	46	306
Graph	141	30	30	201
Vector	141	30	30	201
Hybrid	230	49	49	328
Join	24	5	5	34
Nested	24	5	5	34
Safety	14	4	8	26
Total	**788**	**169**	**173**	**1,126**

Stratified random split, seed=42; augmented with paraphrase variants (663 augmented train queries).

4.2 Evaluation Protocol and Baselines

Strict no-leakage (primary): Test queries answered from generative retrieval only. Twelve Join/Nested queries outside template coverage return NO_ANSWER a SQL planner gap, not a routing failure. **Standard (ceiling):** These 12 queries use oracle fallback; reported for comparison only. **Challenge sets:** Paraphrase (n=160, unseen phrasing); colloquial (n=70, typos, slang). Metric: exact-match accuracy after Unicode normalization.

Table -4: Baselines

Baseline	Description	Scope
SQL-only	NL2SQL planner only	Single modality
Graph-only	In-memory graph only	Single modality
Vector-only	TF-IDF vector only	Single modality
Hybrid-path	All backends, no reranker	Ablation
TriFuseRAG	Full system	Full

5. RESULTS

5.1 Route Accuracy

The dual-stage router achieves **98.84%** route accuracy. Both errors are Join queries misrouted as SQL, attributable to the small Join training set (n=24).

5.2 End-to-End Accuracy

Table -5: Overall Accuracy by Evaluation Mode

Mode	n	Answered	Correct	Overall	Supported
Strict *(primary)*	173	161	158	0.913	0.981
Standard *(ceiling)*	173	173	173	1	1

5.3 Baseline Comparison

Table -6: System vs. Baselines (All 173 Test Queries)

System	Overall Acc.	Avg (ms)	p95 (ms)
SQL-only	0.266	~3.2	~12
Graph-only	0.162	~0.9	~4
Vector-only	0.173	~0.8	~4
Hybrid-path (no reranker)	0.539†	~1.5	~6
TriFuseRAG *(strict)*	**0.913**	**1.29**	**9.1**

Invokes all backends on every query; mishandles Join, Nested, Safety. Single-source baselines perform poorly due to modality mismatch.

5.4 Per-Category Breakdown

Journal Publication of International Research for Engineering and Management (JOIREM)
Volume: 04 Issue: 06 | June-2026
ISSN (O) 3107-6696

Table -7: Per-Category Accuracy vs. Best Baseline (Strict No-Leakage)

Category	n	TriFuseRAG	Best Baseline	Δ
SQL	46	1	1.000	—
Graph	30	0.933	0.933	—
Vector	30	1	1.000	—
Hybrid	**49**	**0.939**	**0.000**	**0.939**
Join	5	0	0	—
Nested	5	0	0	—

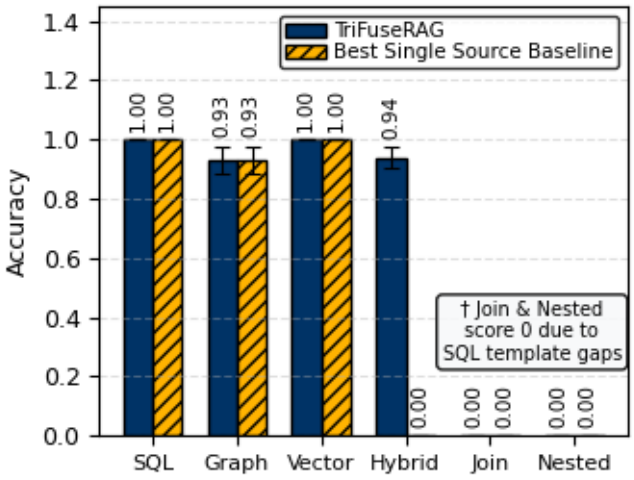

Fig -3: Per-category accuracy: TriFuseRAG vs. best single-source baseline. Hybrid queries demonstrate the structural necessity of multi-modal fusion.

The +93.9 pp Hybrid gap is structural: price (SQL), warranty (Vector), manufacturer (Graph), and category (JOIN) cannot coexist in any single modality. TriFuseRAG matches baselines on SQL/Graph/Vector, confirming routing overhead does not degrade unimodal accuracy.

5.4 Robustness and Error Analysis

Table - 8: Challenge Set Accuracy

Set	n	Accuracy	Notes
Held-out test	173	0.913	Strict; primary
Paraphrase	160	**1**	Unseen phrasing, same structure
Colloquial	70	**0.957**	Typos, slang, shorthand

The 100% paraphrase accuracy reflects controlled catalog vocabulary rather than open-domain generalization. The 95.7% colloquial accuracy reflects pre-routing canonical normalization (typo correction, entity standardization).

Table -9: Quantified Error Analysis (n=173)

Error Category	Count	%	Example
SQL template gap (Join)	9	**5.20%**	*"List products with categories under $900"*
SQL template gap (Nested)	3	**1.70%**	*"Categories having more than 5 products"*
Routing error (Join→SQL)	2	**1.20%**	Ambiguous single-entity category query
Fusion partial answer	1	**0.60%**	*"Price of Product A and B?"*
Total	**15**	**8.70%**	
Out-of-catalog (correct NO_ANSWER)	—	—	*"Unknown Brand Model 999"*

Coreference (correct NO_ANSWER)	—	—	*"What about its warranty?"*

The 8.7% failure rate decomposes as: SQL template gap 6.9%, routing 1.2%, fusion 0.6%. SQL gaps are orthogonal to the core routing/fusion contributions. All failures produce NO_ANSWER or a single-entity partial response, never confabulation, a deliberate precision-over-recall design appropriate for auditable enterprise QA.

Safety: All adversarial probes correctly intercepted by Tier 1; zero false positives among 1,084 non-safety queries. Coverage is limited to predefined patterns; novel formulations require independent red-teaming.

6. DISCUSSION AND LIMITATIONS

Multi-modal fusion value.

The Hybrid result demonstrates a structural claim: queries requiring information from multiple modalities cannot be answered by any single-modality system regardless of implementation quality. Routing-aware multi-modal fusion is a functional requirement for composite query classes, not merely an optimization.

SQL template coverage gap.

Join/Nested failures (6.9%) trace to template coverage, not routing (98.8% accuracy). This limitation is orthogonal to the paper's contributions. Remedies include expanding the template library or adding a constrained LLM fallback for out-of-template patterns.

Limitations.

(i) All results are on a purpose-built synthetic benchmark; real-world vocabulary diversity and annotation noise have not been evaluated, a known closed-world trade-off stated openly. *(ii)* Twelve test queries are unsupported by the current planner. *(iii)* The graph prototype is not persistent; the storage-agnostic interface mitigates production impact. *(iv)* Safety Tier 2 has not been independently red-teamed. *(v)* The system returns retrieved strings or SQL results only; free-form generation requires an additional layer.

7. CONCLUSION

TriFuseRAG achieves 98.84% route accuracy and 91.3% answer accuracy (98.1% on supported queries) in strict no-leakage evaluation, with p95 latency under 10 ms on CPU. On Hybrid queries the +93.9pp gap over all single-source baselines is a structural result composite queries requiring simultaneous SQL, graph, and vector retrieval cannot be answered by single-modality systems. While evaluation uses a controlled synthetic benchmark that isolates architectural contributions, this property is independent of dataset scale. Future priorities: NL2SQL template expansion, persistent graph integration, Spider evaluation, and adversarial red-teaming of the safety layer.

8. REFERENCES

1. Y. Zhang et al., "Siren's Song in the AI Ocean: A Survey on Hallucination in Large Language Models," arXiv preprint arXiv:2309.01219, 2023.
2. P. Lewis et al., "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks," Advances in NeurIPS, vol. 33, pp. 9459–9474, 2020.
3. V. Karpukhin et al., "Dense Passage Retrieval for Open-Domain Question Answering," Proc. EMNLP, pp. 6769–6781, 2020.
4. L. Gao et al., "Tevatron: An Efficient and Flexible Toolkit for Dense Retrieval," Proc. EMNLP System Demonstrations, pp. 239–246, 2022.
5. Y. Gao et al., "Retrieval-Augmented Generation for Large Language Models: A Survey," arXiv:2312.10997, 2023.
6. T. Yu et al., "Spider: A Large-Scale Human-Labeled Dataset for Complex and Cross-Domain Semantic

Parsing and Text-to-SQL," Proc. EMNLP, pp. 3911–3921, 2018.
7. B. Wang et al., "RAT-SQL: Relation-Aware Schema Encoding and Linking for Text-to-SQL Parsers," Proc. ACL, pp. 7567–7578, 2020.
8. T. Scholak, N. Schucher, and D. Bahdanau, "PICARD: Parsing Incrementally for Constrained Auto-Regressive Decoding from Language Models," Proc. EMNLP, pp. 9895–9901, 2021.
9. A. Saxena, A. Tripathi, and P. Talukdar, "Improving Multi-Hop Question Answering over Knowledge Graphs using Knowledge Base Embeddings," Proc. ACL, pp. 4498–4507, 2020.
10. H. He et al., "Learning to Reason over Paths in Knowledge Graphs using Neural Symbolic Machines," Proc. WSDM, pp. 553–561, 2021.
11. W. Chen et al., "HybridQA: A Dataset of Multi-Hop Question Answering over Tabular and Textual Data," Findings of ACL: EMNLP, pp. 1026–1036, 2020.
12. D. Khashabi et al., "UnifiedQA: Crossing Format Boundaries with Single QA System," Findings of ACL: EMNLP, pp. 1896–1907, 2020.
13. F. Perez and I. Ribeiro, "Ignore Previous Prompt: Attack Techniques for Language Models," arXiv:2211.09527, 2022.
14. Z. Shafran, R. Schuster, and V. Goldberg, "Playing it Safe: Avoided Toxicity Promotion in Generative AI and Retrieval," Proc. EMNLP, 2023.
15. G. Salton and C. Buckley, "Term-Weighting Approaches in Automatic Text Retrieval," Information Processing and Management, vol. 24, no. 5, pp. 513–523, 1988.
16. F. Pedregosa et al., "Scikit-learn: Machine Learning in Python," JMLR, vol. 12, pp. 2825–2830, 2011.
17. C. R. Harris et al., "Array Programming with NumPy," Nature, vol. 585, pp. 357–362, 2020.
18. A. Abid et al., "Gradio: Hassle-Free Sharing and Testing of ML Models in the Wild," arXiv:1906.02569, 2019.
19. C. Manning, P. Raghavan, and H. Schütze, Introduction to Information Retrieval. Cambridge University Press, 2008.
20. M. Edge et al., "From Local to Global: A GraphRAG Approach to Query-Focused Summarization," Microsoft Research, 2024.
21. A. Asai et al., "Self-RAG: Learning to Retrieve, Generate, and Critique through Self-Reflection," arXiv:2310.11511, 2023.
22. Y. Yan et al., "CRAG: Corrective Retrieval-Augmented Generation," arXiv:2401.15884, 2024.
23. LangChain Inc., "LangGraph: Building Stateful Multi-Agent Applications with LLMs," 2024.
24. J. Liu et al., "LlamaIndex: Data Framework for LLM Applications," 2024.
25. S. Jeong, S. Baek, S. Cho, S. Hwang, and J. Lee, "Adaptive-RAG: Learning to Adapt Retrieval-Augmented Large Language Models through Question Complexity," arXiv preprint arXiv:2403.14403, 2024.
26. P. Sarthi, S. Abdullah, A. Tuli, et al., "RAPTOR: Recursive Abstractive Processing for Tree-Organized Retrieval," arXiv preprint arXiv:2401.18059, 2024.