World Journal of Advanced Research and Reviews
eISSN: 2581-9615 CODEN (USA): WJARAI
Cross Ref DOI: 10.30574/wjarr
Journal homepage: https://wjarr.com/

(RESEARCH ARTICLE)

Check for updates

EVENTFORGE: OPTIMIZABLE EVENT SCHEMA INDUCTION WITH HYBRID RAG+KG STORAGE

Harshil Lodhiya *

Product and AI Engineering, SlicedHealth, Inc., Woodstock, Georgia, United States.
* Corresponding Author

ORCID Details
Harshil Lodhiya: https://orcid.org/0009-0005-5798-9176

World Journal of Advanced Research and Reviews, 2026, 31(02), 584–591

Publication history: Received on 04 July 2026; revised on 09 August 2026; accepted on 11 August 2026

Article DOI: https://doi.org/10.30574/wjarr.2026.31.2.2105

ABSTRACT

Scientific literature, incident reports and technical documents encode process knowledge that is largely opaque to keyword search and modern retrieval-augmented generation. Event schema induction makes this knowledge available as a queryable graph, but current large-language-model inducers are often organized as hand-crafted prompt cascades that are brittle across models and difficult to improve from data. This paper presents EventForge, an open-source system that reframes schema induction as typed DSPy programs and co-locates the induced knowledge graph with a hierarchical dense-retrieval index in a single PostgreSQL/pgvector substrate. The system reads PDFs, extracts event candidates with DSPy ChainOfThought modules, removes near duplicates with sentence-embedding similarity, stores events in a NetworkX graph, and persists graph and retrieval artifacts together for hybrid RAG+KG querying. On a 12-PDF corpus spanning medical imaging, remote sensing, LLM security, and curriculum learning, EventForge induces 393 events and 286 graph relationships in 497.8 seconds on gpt-4o-mini, with a 6.2% near-duplicate rejection rate. The study reports system behavior, runtime, cost, graph yield, duplicate control, and implementation limitations, while leaving gold-standard event and edge F1 evaluation for future work.

Keywords: Event Schema Induction; DSPy; Retrieval-Augmented Generation; Knowledge Graph; Pgvector; RAPTOR; Large Language Models

1 INTRODUCTION

Event schemas describe the typical structure of scenarios. A disease-outbreak response, for example, can include detection, containment, treatment, and recovery, each of which may decompose into temporally ordered sub-events. Such schemas support event extraction, question answering, narrative understanding, and forecasting (Chambers & Jurafsky, 2008; Schank & Abelson, 1977; Li et al., 2020).

Recent LLM-based inducers such as INCSCHEMA (Li et al., 2023) provide strong open-domain coverage by incrementally expanding events through relation-specific prompts. The approach is powerful, but the software

representation is fragile: prompts are plain strings, model changes require prompt retuning, and the induced graph is usually separated from any dense retrieval index over the source text.

EventForge addresses these issues by migrating the incremental prompting cascade into typed DSPy signatures, storing the induced schema in graph form, and co-locating the graph with a RAPTOR-style dense retrieval substrate in PostgreSQL/pgvector. The paper makes three practical contributions: (1) typed DSPy modules for schema expansion, (2) a shared RAG+KG storage design, and (3) an end-to-end PDF-to-KG pipeline evaluated on a heterogeneous 12-document scientific corpus.

2 RELATED WORK

2.1 Event Schema Induction

Early work generated schemas from predicate-argument structures. Chambers and Jurafsky (2008) introduced unsupervised narrative event chains, and later work extended schemas with participants (Chambers & Jurafsky, 2009). Neural variants used narrative language models (Weber et al., 2018) and path language models for event graph schema induction (Li et al., 2020). INCSCHEMA shifted the task toward LLM-driven incremental expansion with explicit verification (Li et al., 2023).

2.2 Optimizable LLM Programs

DSPy treats LLM applications as programs composed of modules over typed signatures and can compile prompts against metrics using teleprompters such as BootstrapFewShot and MIPRO (Khattab et al., 2024). Chain-of-thought prompting provides a structured reasoning trace for intermediate outputs (Wei et al., 2022). EventForge uses this programmatic representation so relation-specific extraction can be tested, replaced, and eventually optimized without rewriting the surrounding pipeline.

Provider-independent DSPy configuration also makes EventForge compatible with routing and model-selection systems. Lodhiya (2026a) studies feedback control for adaptive language model routing under non-stationary workloads, and Lodhiya (2026b) studies dynamic model selection in heterogeneous multi-LLM architectures. EventForge can use similar infrastructure by changing the dspy.LM configuration rather than changing schema-induction logic.

2.3 Retrieval Augmentation and Hybrid Graph Retrieval

Retrieval-augmented generation grounds generation in retrieved passages for knowledge-intensive tasks (Lewis et al., 2020). RAPTOR builds recursive summaries and retrieves at multiple abstraction levels (Sarthi et al., 2024). GraphRAG combines graph structure with dense retrieval for multi-hop query-focused summarization (Edge et al., 2024). EventForge contributes an event-schema-centered implementation using PostgreSQL and pgvector as the shared storage substrate (Kane & pgvector contributors, 2021).

3 PROPOSED WORK

3.1 System Overview

Given a scenario name and a document collection, EventForge induces a directed schema graph $G = (V, E)$, where each node is an event with a natural-language description and each edge represents a hierarchy, temporal, or causal relation. The current evaluation stores parent-child expansions as hierarchy edges; relation-specific provenance is preserved at generation time but not yet propagated into every stored edge label.

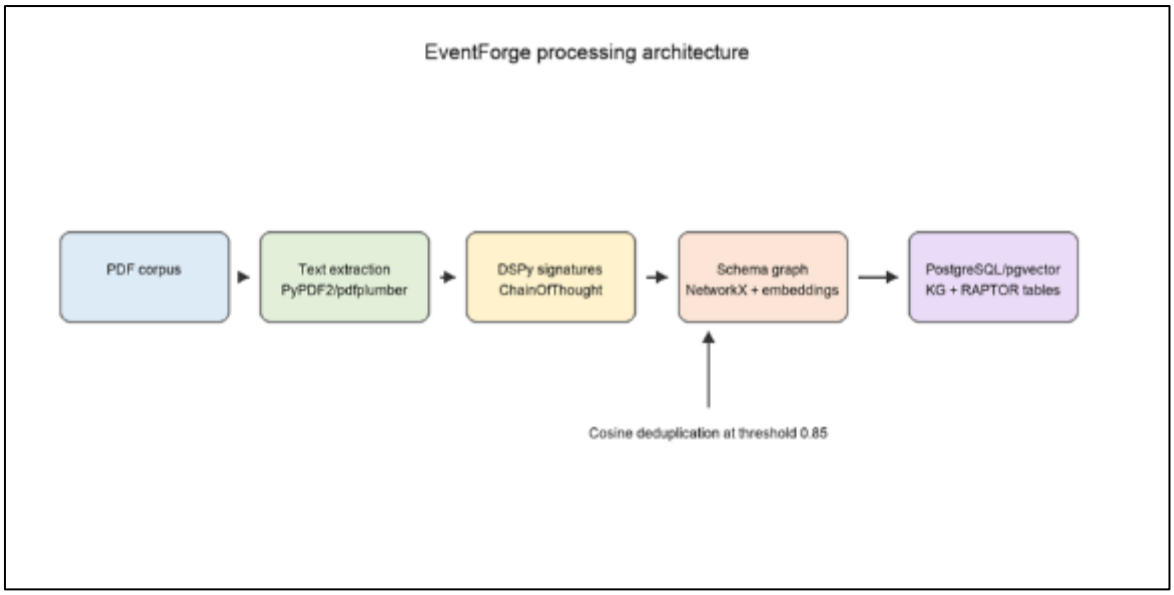

Figure 1: EventForge system architecture: PDF documents flow through extraction, DSPy-based induction, embedding deduplication, and unified PostgreSQL/pgvector storage.

3.2 DSPy Signatures and Modules

EventExpansionSignature accepts event_description and relation_type and returns a reasoning field plus a list of related_events. The supported relation types are subevent_during, subevent_step, temporal_after, temporal_before, causal_after, and causal_before. SkeletonExtractionSignature accepts a scenario and returns five to ten major events. EventExpander wraps EventExpansionSignature with dspy.ChainOfThought and exposes both single-relation expansion and expansion across all six relation types.

3.3 Schema Graph and Deduplication

Events are inserted into a NetworkX directed graph (Hagberg et al., 2008). A parallel sentence-transformer embedding dictionary uses all-MiniLM-L6-v2 embeddings (Reimers & Gurevych, 2019). A candidate event is rejected when its maximum cosine similarity to an existing event exceeds tau_desc = 0.85 or when its normalized name is already present.

3.4 PDF-to-KG Pipeline and Storage

The pipeline has four stages: PDFProcessor extracts text using PyPDF2 with pdfplumber fallback; EventExtractionPipeline chunks each document and calls EventExpander; PostgreSQLStorage writes kg_schemas, kg_events, and kg_relationships using batched inserts; and the orchestrator returns one Schema per document. The PostgreSQL schema includes kg_events, kg_relationships, kg_schemas, cluster_event_mapping, and raptor_embeddings with a vector(384) IVFFlat cosine index.

4 RESULTS

4.1 Experimental Setup

The evaluation uses 12 publicly available scientific PDFs from medical imaging and lung cancer detection, land-cover classification and remote sensing, LLM security, and curriculum learning. The corpus yields 945,020 extracted characters, with document-level text ranging from 23,700 to 251,706 characters. The LLM backend is OpenAI gpt-4o-mini with temperature 0.3 and max_tokens 1,024 (OpenAI, 2024). The pipeline uses 4,000-character chunks, the first three chunks per document, at most eight level-1 events per document, at most three children per parent, and tau_desc = 0.85.

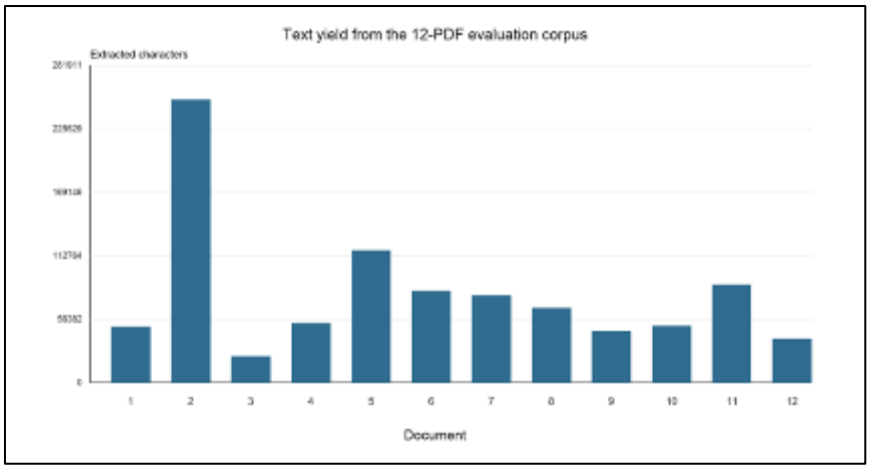

Figure 2: Extracted text yield from the 12-PDF evaluation corpus

4.2 Per-Document Results

Table 1: Per-document induction results

#	Scenario	Chars	L1	L2	Causal	Dedup	LLM s
1	Lung Nodule Detection Using C...	49,914	8	21	3	3	37.0
2	Deep Learning Applications in...	251,706	8	22	3	2	37.9
3	Dynamic Curriculum Learning f...	23,700	7	19	0	3	32.2
4	Security Enhancements for Lar...	52,851	8	21	2	3	38.3
5	AI in Medical Imaging for Lun...	117,910	8	21	2	3	36.1
6	Land Cover Classification in ...	81,630	8	23	2	1	36.9
7	Machine Learning for Lung Can...	77,625	8	21	1	3	40.5
8	Deep Learning for Lung Cancer...	66,486	8	24	3	0	46.5
9	Land-Cover Classification Usi...	46,411	8	21	3	3	38.8
10	Advancements in Lung Tumor Se...	50,785	8	22	3	2	34.6
11	Satellite Imagery Analysis fo...	87,053	8	21	3	3	36.6
12	Deep Learning for Early Lung ...	38,949	8	24	1	0	40.2

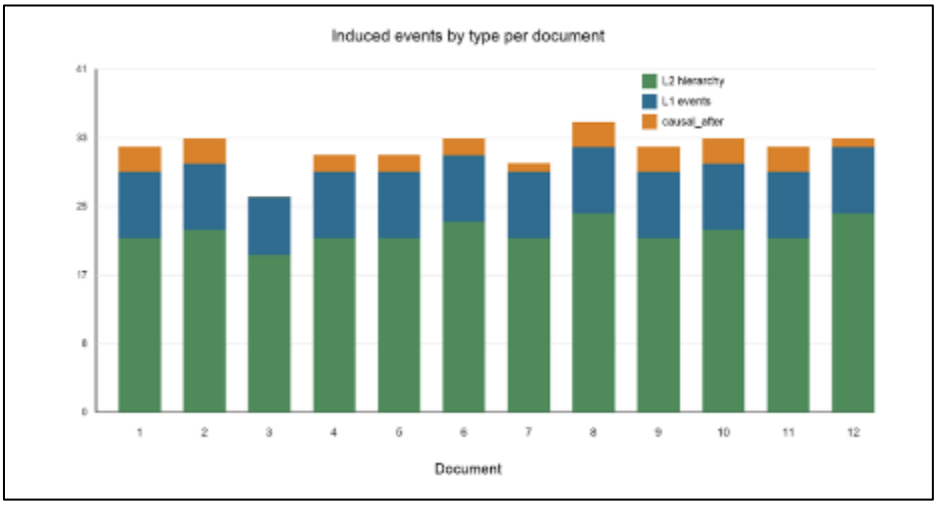

Figure 3: Induced events per document, stacked by type.

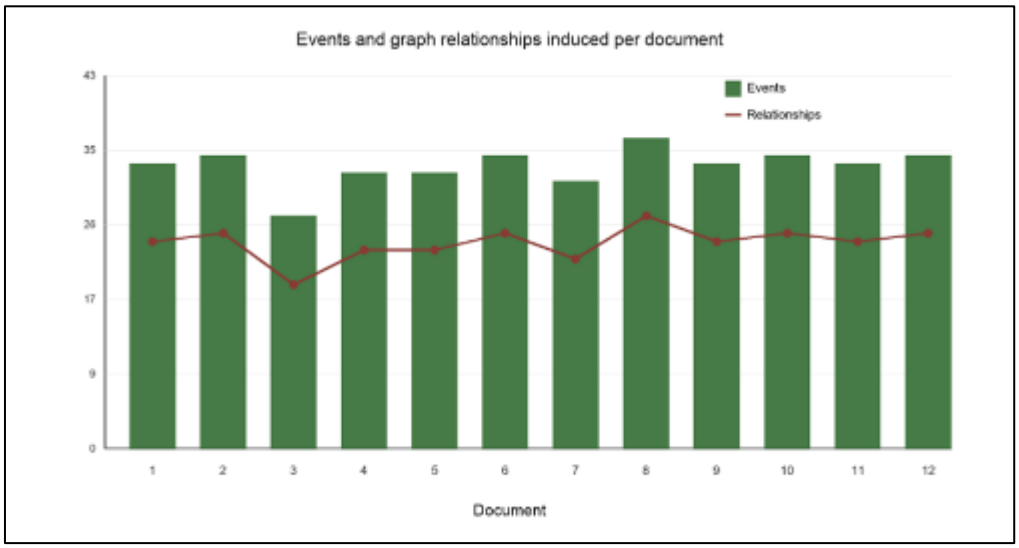

Figure 4: Total induced events and graph relationships per document.

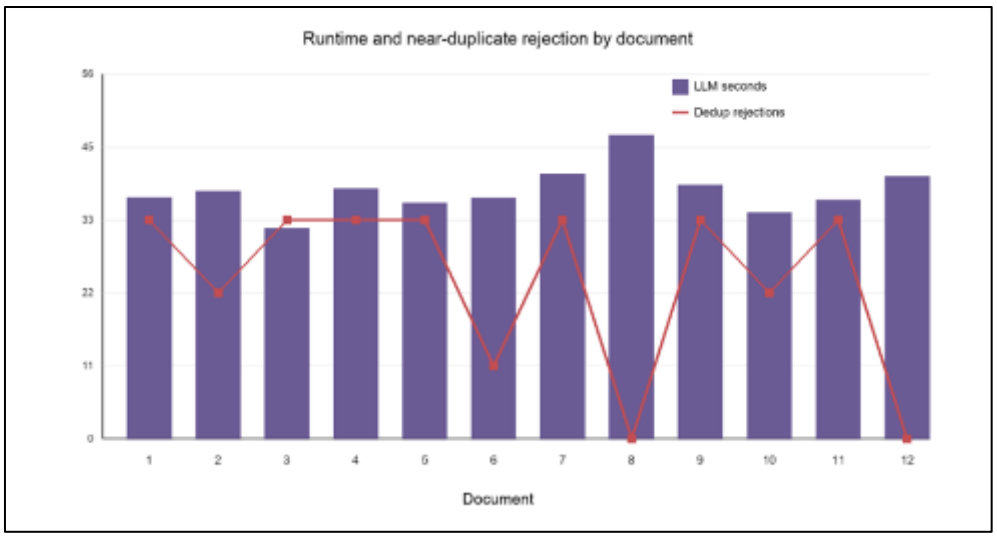

Figure 5: Runtime and near-duplicate rejection by document.

4.3 Aggregate Statistics and Additional Analysis

Table 2: Corpus-level aggregate statistics.

Metric	Value
PDFs processed	12 / 12 (100%)
Extracted characters	945,020
Level-1 events retained	95
Level-2 hierarchical children	260
causal_after children	26
Total induced events	393
Graph relationships	286
Dedup rejections	26 (6.2%)

Total wall-clock	497.8 s
Estimated cost	approximately US $0.03 total

Table 3: Additional descriptive analysis from the 12-document run.

Measure	Observed value
Events per document	32.75 +/- 2.13
Relationships per document	23.83 +/- 1.91
LLM seconds per document	37.96 +/- 3.38
Wall-clock seconds per event	1.27
LLM seconds per event	1.16
Events per 10,000 extracted characters	5.67
Correlation: text length vs events	0.27
Correlation: text length vs LLM time	0.05
Correlation: events vs relationships	1.00

The low correlation between extracted text length and LLM time indicates that runtime was dominated by the fixed chunk cap rather than by full document length. The weak correlation between text length and event count should not be interpreted as evidence that short and long documents contain equivalent schema content; it reflects the bounded evaluation setting. The near-perfect event-relationship correlation follows from the current graph-building policy, where most retained non-root events are attached through parent-child edges.

4.4 Deduplication Effectiveness

Across the 12 documents, DSPy generated 419 candidate event descriptions. The deduplication filter accepted 393 candidates and rejected 26 near-duplicates at tau_desc = 0.85, corresponding to a 6.2% rejection rate. This rate is best interpreted as an engineering diagnostic rather than an accuracy score: it suggests that the threshold removes close paraphrases without collapsing the graph too aggressively.

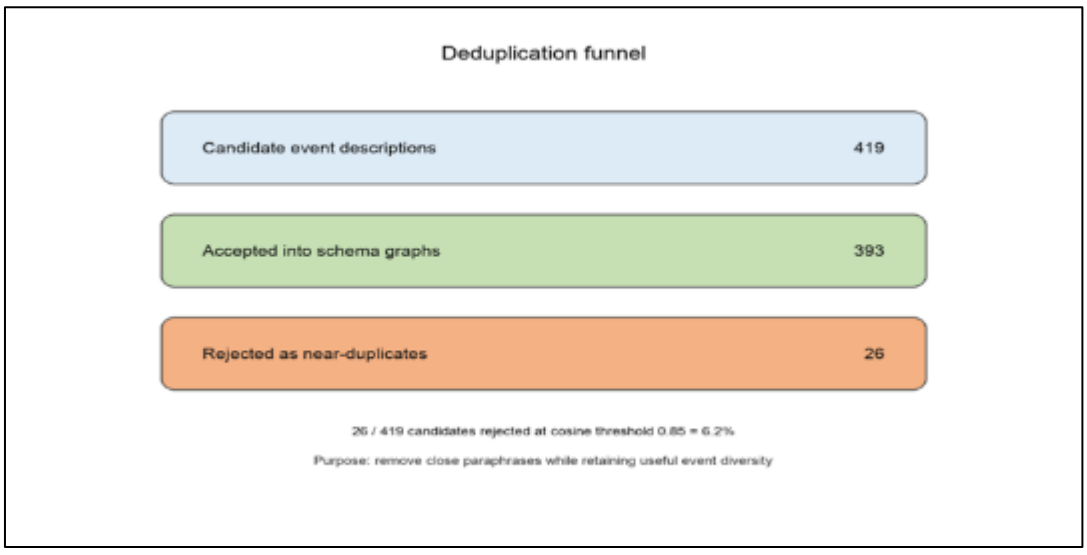

Figure 6: Deduplication funnel across the full evaluation run.

4.5 Qualitative Traces

Illustrative EventExpander outputs show domain coherence across corpus clusters. In lung nodule detection, extracted events include capsule-network segmentation, CT feature extraction, and nodule classification. In land-cover classification, extracted events include preprocessing, algorithm application, confusion-matrix analysis, and kappa

reporting. In LLM security, extracted events include adversarial prompt injection detection, guardrail application, unsafe-request refusal, and attempted-jailbreak logging.

5 DISCUSSION

Storage co-location is the primary engineering result. Placing the induced knowledge graph and RAPTOR-compatible pgvector index in one PostgreSQL instance makes nearest-neighbor retrieval followed by graph traversal expressible as one SQL workflow. The design reduces operational complexity compared with systems that keep graph and vector stores separate.

The current system is intentionally conservative. It demonstrates typed modularity and storage integration, but it does not yet prove accuracy superiority over INCSCHEMA. The next evaluation should add gold schemas and compare raw prompts, uncompiled DSPy signatures, and compiled DSPy signatures across relation types.

Limitations

Two of the original nine INCSCHEMA-style modules are DSPy-native in the current release. Naming, grounding, verification, specificity, and relevance modules remain future work. The Schema.add_child_event method currently labels parent-child edges as hierarchy even when generated by temporal_after or causal_after calls. RAPTOR storage is provisioned, but downstream RAPTOR retrieval was not exercised in the reported run. The evaluation corpus is unlabeled, so event-node F1 and edge F1 are not reported.

6 CONCLUSION

EventForge shows that open-domain event schema induction can be expressed as typed DSPy modules and persisted with a dense retrieval substrate in PostgreSQL/pgvector. On a 12-document scientific corpus, the system induces 393 events and 286 graph relationships in 497.8 seconds at approximately US $0.03 on gpt-4o-mini. The architecture is open-source, provider-agnostic, and ready for stronger accuracy-facing evaluation with gold schemas, relation-specific edge labels, and DSPy compilation ablations.

STATEMENTS ON COMPLIANCE WITH ETHICAL STANDARDS

Acknowledgments

OpenAI gpt-4o-mini was used as the induction LLM backend in the empirical study. All experimental design, results, analysis, and interpretation are the author's own. The source code, evaluation scripts, and machine-readable results are published at https://github.com/harshillodhiya/EventForge under the MIT license.

Funding Source

This research received no specific grant from any funding agency in the public, commercial, or not-for-profit sectors.

Disclosure of conflict of interest

The author is employed by SlicedHealth. The work uses publicly available tools and a self-assembled corpus of publicly available scientific PDFs, and does not involve proprietary or customer data. The author declares no other conflict of interest.

Statement of ethical approval

This article does not contain studies with human participants or animals performed by the author. All experiments used publicly available scientific PDF documents.

REFERENCES

[1] Chambers, N., & Jurafsky, D. (2008). Unsupervised learning of narrative event chains. In Proceedings of ACL-HLT (pp. 789-797).

[2] Schank, R. C., & Abelson, R. P. (1977). Scripts, plans, goals, and understanding: An inquiry into human knowledge structures. Erlbaum.

[3] Li, M., Zeng, Q., Lin, Y., Cho, K., Ji, H., May, J., Chambers, N., & Voss, C. (2020). Connecting the dots: Event graph schema induction with path language modeling. In Proceedings of EMNLP (pp. 684-695).

[4] Li, S., Zhao, R., Li, M., Ji, H., Callison-Burch, C., & Han, J. (2023). Open-domain hierarchical event schema induction by incremental prompting and verification. In Proceedings of ACL.

[5] Khattab, O., Singhvi, A., Maheshwari, P., Zhang, Z., Santhanam, K., Vardhamanan, S., Haq, S., Sharma, A., Joshi, T. T., Moazam, H., Miller, H., Zaharia, M., & Potts, C. (2024). DSPy: Compiling declarative language model calls into self-improving pipelines. In Proceedings of ICLR. https://arxiv.org/abs/2310.03714

[6] Sarthi, P., Abdullah, S., Tuli, A., Khanna, S., Goldie, A., & Manning, C. D. (2024). RAPTOR: Recursive abstractive processing for tree-organized retrieval. In Proceedings of ICLR. https://arxiv.org/abs/2401.18059

[7] Chambers, N., & Jurafsky, D. (2009). Unsupervised learning of narrative schemas and their participants. In Proceedings of ACL-IJCNLP (pp. 602-610).

[8] Weber, N., Balasubramanian, N., & Chambers, N. (2018). Event representations with tensor-based compositions. In Proceedings of AAAI, 32(1).

[9] Edge, D., Trinh, H., Cheng, N., Bradley, J., Chao, A., Mody, A., Truitt, S., Metropolitansky, D., Ness, R. O., & Larson, J. (2024). From local to global: A GraphRAG approach to query-focused summarization. arXiv:2404.16130.

[10] Kane, A., & pgvector contributors. (2021). pgvector: Open-source vector similarity search for Postgres. https://github.com/pgvector/pgvector

[11] Reimers, N., & Gurevych, I. (2019). Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In Proceedings of EMNLP-IJCNLP (pp. 3982-3992).

[12] Wei, J., Wang, X., Schuurmans, D., Bosma, M., Ichter, B., Xia, F., Chi, E. H., Le, Q. V., & Zhou, D. (2022). Chain-of-thought prompting elicits reasoning in large language models. In Advances in Neural Information Processing Systems.

[13] DataRobot Research. (2025). syftr: Pareto-optimized agentic flows. arXiv:2505.20266. https://arxiv.org/abs/2505.20266

[14] OpenAI. (2024). GPT-4o mini: Advancing cost-efficient intelligence. https://openai.com/index/gpt-4o-mini-advancing-cost-efficient-intelligence/

[15] Lodhiya, H. (2026a). Feedback control for adaptive language model routing under non-stationary workloads. Journal of Computer Science and Technology Studies, 8(8), 238-243. https://doi.org/10.32996/jcsts.2026.8.8.17

[16] Lodhiya, H. (2026b). LLM-Advisor: Dynamic model selection and query routing in heterogeneous multi-LLM architectures. International Journal of Research in Engineering, Science and Management, 9(7), 84-86. https://doi.org/10.65138/ijresm.v9i7.3488

[17] Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Kuttler, H., Lewis, M., Yih, W., Rocktaschel, T., Riedel, S., & Kiela, D. (2020). Retrieval-augmented generation for knowledge-intensive NLP tasks. In Advances in Neural Information Processing Systems.

[18] Hagberg, A. A., Schult, D. A., & Swart, P. J. (2008). Exploring network structure, dynamics, and function using NetworkX. In Proceedings of the 7th Python in Science Conference (pp. 11-15).