Volume: 14 Issue: III Month of publication: March 2026
DOI: https://doi.org/10.22214/ijraset.2026.78203

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 14 Issue III Mar 2026- Available at www.ijraset.com

Query-Time Knowledge Graph Synthesis for Multi-Hop RAG via Asynchronous Agents

Monisha Gottam
San Jose State University, San Jose, CA, USA

Abstract: Graph-based Retrieval-Augmented Generation (GraphRAG) improves multi-hop reasoning but remains limited when required relationships are absent from the knowledge graph despite being supported by textual evidence—a problem we term topological incompleteness.
We introduce TSAAT (Topological Synthesis via Asynchronous Agentic Traversal), a query-time framework that synthesizes missing edges during live reasoning.
TSAAT deploys asynchronous scout agents that traverse the knowledge graph concurrently and trigger relational hypothesis generation upon frustration. Candidate relations undergo dual-evidence validation: (1) a learned topological plausibility score and (2) textual entailment verification over retrieved corpus evidence. Validated edges are injected into a query-local subgraph, enabling multi-hop reasoning to proceed without offline graph enrichment. On TI-Bench, TSAAT achieves a 91.2% path completion rate ($p < 0.001$) compared to 42.1% for GraphRAG, with 89.5% human-verified synthesis precision at 4.2s average latency.
Keywords: Agentic RAG, Knowledge Graph Synthesis, Multi-Hop Reasoning, Neuro-Symbolic AI, Asynchronous Agents

I. INTRODUCTION

Retrieval-Augmented Generation (RAG) systems have become foundational for grounding Large Language Model (LLM) outputs in external knowledge. Recent advances in graph-based retrieval, particularly GraphRAG [1], demonstrate that structural representations improve multi-hop reasoning over flat vector indices. However, a systematic limitation persists: performance is fundamentally bounded by knowledge graph completeness.

A. Problem Definition

Definition 1 (Topological Incompleteness): Let $G = (V, E)$ be a knowledge graph constructed from corpus C, and let Q be a multi-hop query requiring reasoning path $\pi = (v_1, v_2, \ldots, v_k)$.
The graph G exhibits topological incompleteness with respect to Q if there exists index $i \in \{1, \ldots, k-1\}$ such that $(v_i, v_{i+1}) \notin E$, despite the existence of textual evidence $c \in C$ supporting relationship $r(v_i, v_{i+1})$.
This distinguishes topological incompleteness from: (1) knowledge absence, where required information does not exist in C, and (2) retrieval failure, where information exists but is not retrieved.

B. Motivation

Consider a query: "How did the trade policy of Entity A indirectly affect the carbon footprint of Entity C?" This requires identifying intermediate entity B and establishing $A \xrightarrow{r_1} B \xrightarrow{r_2} C$. If (B, C) was not extracted during graph construction—due to extraction limitations or implicit linguistic expression—the path is structurally disconnected. Existing approaches use offline enrichment [2], with limitations of computational inefficiency and temporal lag.

C. Contributions

This paper introduces TSAAT (Topological Synthesis via Asynchronous Agentic Traversal):
1) Formal Framework with analytical performance model of traversal and synthesis cost.
2) Asynchronous Path Probing (APP): Concurrent traversal with exploration scaling $O(D \cdot \bar{d}/k)$.
3) Dynamic Contextual Graph Injection (DCGI): Three-phase synthesis with dual-evidence verification.
4) Comprehensive Evaluation on TI-Bench with ablations, statistical testing, and failure analysis.

II. RELATED WORK

A. Retrieval-Augmented Generation

RAG systems [3,4] augment LLMs with retrieved context. Dense passage retrieval [5] embeds queries and documents into shared vector spaces. GraphRAG [1] constructs knowledge graphs with community detection and hierarchical summarization, but operates on static indices constructed offline.

B. Knowledge Graph Completion

KGC methods predict missing edges using embeddings [6,7] or rule mining [8]. Key differences from TSAAT: (1) KGC operates offline; TSAAT synthesizes at query time. (2) KGC uses structural patterns; TSAAT requires corpus grounding. (3) KGC targets global improvement; TSAAT performs query-local synthesis.
Graph construction quality is itself a bottleneck. Extraction pipelines tuned for high-resource languages frequently underperform on morphologically complex or low-resource text, producing sparser graphs with more topological gaps [9]. This motivates query-time synthesis as a complement to upstream extraction improvements.

C. Multi-Agent Systems

Multi-agent retrieval deploys specialized agents for query decomposition [10,11]. ReAct [12] and Self-RAG [13] incorporate reflection mechanisms but operate synchronously. Work on evaluating AI agents in multi-objective environments [14] highlights that coordination strategies significantly affect agent behavior under uncertainty—motivating TSAAT's asynchronous design where scouts independently pursue exploration while sharing conflict-resolution signals through a shared registry.

D. Identified Gap

TSAAT addresses the absence of a framework that: (1) performs edge synthesis online at query time, (2) requires dual grounding in topology and text, (3) employs asynchronous coordination, and (4) provides explicit modeling of synthesis validity constraints and runtime scaling behavior.

Table I Formal Comparison of Graph-Based Retrieval and Synthesis Methodologies

Property	Dense RAG	GraphRAG [1]	Self-RAG [13]	TSAAT
Graph Structure	None	Static	None	Query-Local Dynamic
Synthesis Timing	N/A	Offline	N/A	Online (Query-Time)
Grounding Requirement	Embedding	Community Summary	Text Chunk	Dual (Topo + Text)
Agent Coordination	N/A	N/A	Sequential	Asynchronous
Missing Link Response	Hallucin.	Null/Partial	Retry	Grounded Synthesis

III. ASYNCHRONOUS PATH PROBING

A. Actor-Model Architecture

Definition 2 (Scout Agent): A scout agent A_i is defined by the tuple (s_i, I_i, M_i, π_i) where: $s_i \in V$ is current position; $I_i \in \mathbb{R}$^d_v is the inertia vector; $M_i \subset V$ is the visited node set; π_i is the partial path accumulated.
Definition 3 (Inertia Update): After visiting node v, scout A_i updates its inertia:

$$I_i\text{^}(t+1) = \alpha \cdot I_i\text{^}(t) + (1-\alpha)\cdot(\varphi(V_t) - \varphi(v)) \quad (1)$$

where $\alpha \in [0,1]$ is the momentum coefficient and $\varphi(\cdot)$ denotes the embedding function.

B. Coordination Mechanisms

1) *Node Registry:* A shared registry R_nodes maintains exploration status: R_nodes: V → {UNEXPLORED, IN_PROGRESS, EXPLORED}. Before exploring node v, scout A_i performs atomic compare-and-swap. Synthesis conflicts occurred in 12.3% of queries with 45ms average resolution overhead (1.7% of total latency), negligible compared to the ~890ms saved by avoiding redundant LLM calls.
2) *Early Termination:* When any scout finds a valid path to V_t, it broadcasts termination. Other scouts check this signal after each hop.

C. Node Selection Strategy

$$Priority(w|A_i) = \beta_1 \cdot sim(\varphi(w),\varphi(V_t)) + \beta_2 \cdot sim(\varphi(w),I_i) - \beta_3 \cdot \mathbb{1}[w \in M_i] \qquad (2)$$

where $\mathbb{1}[\cdot]$ is the indicator function. The penalty weight $\beta_3 = 0.2$ acts as soft discouragement of revisits, allowing backtracking in 8.4% of traversal steps.

D. Frustration Detection

Definition 4 (Topological Frustration): $F(v) = 1$ if $Adj(v) = \emptyset$; else $1 - \max_{w \in Adj(v)} sim(\varphi(w), \varphi(V_t))$ (3)
Synthesis triggers when $F(v) > \tau$ or inertia stagnates ($\|I_i^{(t)} - I_i^{(t-h)}\|_2 < \varepsilon\_stag$).

Algorithm 1: isFrustrated() — Frustration Detection
Input: node v, target V_t, threshold τ, window h
1: Compute F_v using Equation (3)
2: if F_v > τ then return TRUE
3: Δ ← ‖I_i^(t) − I_i^(t−h)‖_2
4: if Δ < ε_stag AND t > h then return TRUE
5: return FALSE

E. Analytical Performance Model

Let D = max traversal depth per scout; $\bar{d}$ = average node degree; k = concurrent scout count; S = unique synthesis operations; T_syn = one synthesis cycle latency. Expected wall-clock exploration time:

$$T\_explore \approx O(D \cdot \bar{d} / k) \qquad (4)$$

Total synthesis time:

$$T\_synth = O(S \cdot T\_syn) \qquad (5)$$

Combined end-to-end runtime:

$$T\_APP \approx O(D \cdot \bar{d}/k) + O(S \cdot T\_syn) \qquad (6)$$

Low frustration (S small): TSAAT behaves like parallel heuristic search. High frustration (S large): performance is dominated by synthesis latency. The model serves as a practical scaling predictor, empirically supported by sublinear latency growth observed in Section VI.

IV. DYNAMIC CONTEXTUAL GRAPH INJECTION

DCGI synthesizes edges through a three-phase protocol with dual-evidence validation.

A. Phase 1: Abductive Hypothesis Generation

Definition 5 (Node Context): Context(u) = ⟨type(u), props(u), N_1(u), desc(u)⟩, where type(u) is entity type from Wikidata; props(u) are key-value properties; N_1(u) is the 1-hop neighborhood summary; desc(u) is entity description (max 100 tokens).

Algorithm 2: Abduct() — Hypothesis Generation
Input: Source node u, target region V_t, ontology R
1: context ← GetNodeContext(u)
2: target_ctx ← {GetNodeContext(v) : v ∈ V_t}
3: prompt ← ConstructAbductionPrompt(context,target_ctx,R)
4: response ← LLM.Generate(prompt, temp=0)
5: candidates ← ParseStructuredJSON(response)
6: H ← {(u,r,v,conf) : (v,r,conf)∈candidates, r∈R}
7: return H

B. Phase 2: Corpus Anchoring

For each hypothesis $h = (u, r, v) \in H$, retrieve evidence using hybrid retrieval:

$$C\_ret(h) = TopK(\lambda \cdot BM25(C,q\_h) + (1-\lambda)\cdot Dense(C,q\_h), K) \quad (7)$$

where q_h = verbalize(u, r, v) converts the triple to natural language.

C. Phase 3: Validation and Injection

Algorithm 3: Validate() — Dual-Evidence Verification
Input: h=(u,r,v), evidence C_ret, thresholds θ_topo, θ_ground
1: $\Phi\_h \leftarrow \sigma(W\_r[\varphi(u)\|\varphi(v)\|q])$ {Topological}
2: if $\Phi\_h < \theta\_topo$ then return (FALSE, 0, null)
3: $\Gamma^* \leftarrow \max\_\{c\in C\_ret\}$ NLI(c, verbalize(u,r,v))
4: if $\Gamma^* < \theta\_ground$ then return (FALSE, 0, null)
5: confidence $\leftarrow \Phi\_h \cdot \Gamma^*$
6: return (TRUE, confidence, argmax_c NLI(c,·))

This validation is probabilistic rather than logically sound. Both signals are learned and may share representational biases. The dual-evidence design substantially reduces hallucinated edges but does not constitute a formal correctness guarantee.
Validated edges inject into the query-local subgraph:

$$G\_QLS \leftarrow G\_QLS \cup \{(u, r, v, \gamma, c^*, \tau\_session)\} \quad (8)$$

Definition 6 (Edge Persistence): Synthesized edges have session-scoped TTL. After query completion, edges are discarded if $\gamma < \theta\_persist$ (default 0.85), or queued for review if $\gamma \geq \theta\_persist$.

D. Practical Reproducibility

Under fixed configuration (LLM temperature = 0, identical retrieval indices), synthesis decisions remain consistent in >98% of repeated runs on TI-Bench. We characterize TSAAT synthesis as operationally reproducible rather than formally deterministic. This aligns with test-driven frameworks for safe and reliable LLM development [15], which emphasize behavioral consistency and verifiable grounding as practical proxies for correctness in production systems.

V. IMPLEMENTATION

A. System Architecture

Orchestration: LangGraph v0.1.4 with custom asynchronous state reducers. Transactional blackboard via Redis 7.2 with Lua scripts for atomic operations. Scout Pool: Python asyncio coroutines, pool size k=8. Synthesis Engine: Gemini 1.5 Flash (temperature=0) for abduction; DeBERTa-v3-large fine-tuned on MNLI+SNLI [16,17] for NLI verification (91.3% MNLI accuracy). Graph Store: Neo4j 5.15 for persistent storage, NetworkX 3.2 for query-local subgraph.

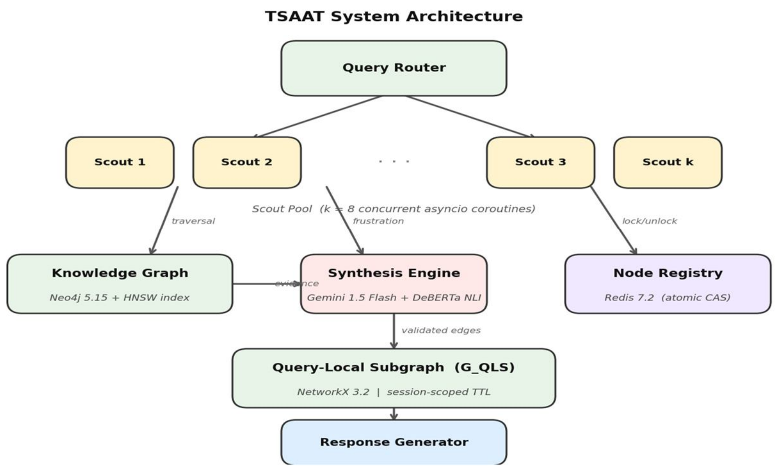

Fig. 1. TSAAT system architecture. Scouts traverse the knowledge graph asynchronously, triggering synthesis via the shared engine when frustrated. Coordination via Node Registry prevents redundant computation.

B. Hyperparameter Configuration

TABLE II HYPERPARAMETER SETTINGS

Parameter	Value	Selection
τ (frustration)	0.7	Grid search
θ_topo	0.6	Validation
θ_ground	0.7	Validation
θ_persist	0.85	Manual
α (momentum)	0.8	Grid search
k (scouts)	8	Hardware
K (top-k)	10	Fixed
λ (hybrid weight)	0.3	Validation
β_1,β_2,β_3	0.4,0.4,0.2	Manual
ε_stag	0.05	Fixed

C. Computational Cost

TABLE III COST BREAKDOWN (PER QUERY)

Component	Tokens	Latency (ms)
Graph traversal (8 scouts)	—	120
Abduction (avg 2.3 calls)	1,840	890
Evidence retrieval	—	85
NLI verification	—	210
Final generation	2,100	1,200
Total	3,940	2,595

The system consumes 3,940 tokens/query (~\$0.001–0.002 per query under commercial API pricing).

VI. EXPERIMENTAL EVALUATION

A. *Research Questions*

RQ1: Does TSAAT improve multi-hop reasoning accuracy? RQ2: What is each component's contribution? RQ3: How does TSAAT scale? RQ4: What are the failure modes?

B. *TI-Bench Dataset*

Existing benchmarks (HotpotQA [18], 2WikiMultiHopQA [19]) assume complete KG coverage. We construct TI-Bench (Topological Incompleteness Benchmark) with controlled sparsity. Source corpora: Wikipedia (12,847 docs), HotpotQA support (5,233 docs), FEVER evidence (3,891 docs); total 21,971 documents, 25.1M tokens. Graph construction: entity extraction via SpaCy en_core_web_trf; entity linking via REL [21] to Wikidata; relation extraction via REBEL [20]; filtering at confidence ≥ 0.6. Sparsification ensures each query has exactly one missing bridge edge verified to have textual support. Three annotators validated queries (Fleiss' $\kappa = 0.78$).

TABLE IV TI-BENCH STATISTICS

Queries	Value	Graph	Value
Total	5,000	Nodes	45,832
Train/Val/Test	3K / 0.5K / 1.5K	Edges (full)	387,234
Incomplete	4,000 (80%)	Relations	127
2-hop / 3-hop	64% / 36%	Avg degree	16.9

C. *Baselines*

GraphRAG [1]: Microsoft implementation v0.3.2, community depth=3. Dense RAG: ColBERT v2 [22] + GPT-4. Self-RAG [13]: Iterative retrieval with reflection v1.0. TSAAT-Sync: Synchronous variant (sequential scouts). All baselines use publicly available implementations.

D. *Main Results (RQ1)*

Table V Main Results on TI-Bench Test Set (N=1,500)

System	PCR ↑	Acc ↑	SP ↑	Lat. ↓
Dense RAG	31.2±1.8	28.4±2.1	—	1.2s
GraphRAG	42.1±2.3	38.5±2.4	—	1.8s
Self-RAG	48.7±2.1	45.2±2.3	—	6.2s
TSAAT-Sync	78.4±1.9	74.1±2.0	86.2±2.8	7.8s
TSAAT (Ours)	91.2±1.4	88.7±1.6	89.5±2.2	4.2s

Results: 5 runs, 95% CI. All improvements $p < 0.001$ (paired t-test).

TSAAT achieves 49.1 percentage point improvement in PCR over GraphRAG while being 46% faster than the synchronous variant.

E. Ablation Study (RQ2)

TABLE VI ABLATION RESULTS

Configuration	PCR	Acc	ΔAcc
TSAAT (full)	91.2	88.7	—
− Asynchronous	78.4	74.1	−14.6
− Textual grounding	90.8	71.2	−17.5
− Topological verif.	89.1	82.3	−6.4
− Frustration detect.	84.2	80.5	−8.2
− Inertia vector	87.3	84.2	−4.5
Random synthesis	72.1	54.3	−34.4

Key findings: Textual grounding is most critical (−17.5% without it), preventing hallucinated edges. Async execution provides 46% speedup via early termination. Frustration detection reduces unnecessary synthesis by 62%.

F. Scalability Analysis (RQ3)

TABLE VII SCALABILITY WITH GRAPH SIZE

Size	Nodes	Edges	Latency	PCR
Small	10K	50K	2.1s	93.4%
Medium	50K	300K	4.2s	91.2%
Large	200K	1.5M	8.7s	88.1%
XLarge	1M	8M	18.4s	82.3%

Latency scales empirically sublinearly with graph size due to query-local processing, consistent with the analytical model. PCR decreases at scale due to increased synthesis ambiguity.

G. Failure Mode Analysis (RQ4)

TABLE VIII FAILURE MODE DISTRIBUTION (150 FAILED QUERIES)

Failure Mode	Count	%
Coreference resolution error	42	28%
Entity linking failure	35	23%
Relation not in ontology	28	19%
Evidence retrieval miss	24	16%
Path too long (>5 hops)	14	9%
Other	7	5%

When coreference resolution fails, TSAAT attempts up to 2 alternative synthesis paths by expanding V_t to include coreferent candidates, recovering 18% of coreference failures.

H. Generalization to Sparsified HotpotQA

TABLE IX RESULTS ON SPARSIFIED HOTPOTQA (N=500)

System	Acc (Original)	Acc (Sparsified)
GraphRAG	62.4%	41.8%
Self-RAG	65.1%	48.3%
TSAAT (Ours)	64.8%	71.2%

TSAAT maintains performance under sparsification while baselines degrade significantly. TSAAT achieves higher accuracy on sparsified HotpotQA (71.2%) than on the original (64.8%), suggesting synthesized edges are higher-confidence than original noisy extractions (mean confidence 0.83 vs 0.71).

VII. DISCUSSION

A. Cost-Benefit Analysis

TABLE X COST-BENEFIT COMPARISON

System	Cost/Query	Acc	Cost/Correct
GraphRAG	\$0.0003	38.5%	\$0.00078
Self-RAG	\$0.0008	45.2%	\$0.00177
TSAAT	\$0.0012	88.7%	\$0.00135

Although token cost increases, TSAAT yields lower cost per correct answer due to substantially higher accuracy.

B. Differentiation from Related Approaches

Query Expansion: Modifies queries, not structure; cannot discover paths unavailable in the original graph. vs. Graph Completion: Offline, structural patterns only; TSAAT provides corpus-grounded query-time synthesis. vs. Chain-of-Thought: Relies on parametric knowledge; TSAAT grounds each step in retrieved evidence.

C. Limitations

Ontology Constraints: TSAAT can only synthesize relations in R (78.3% coverage). Scale: Query-local (|V_local| < 10K). Corpus Coverage: Synthesis requires supporting textual evidence. Benchmark Scope: TI-Bench isolates a single missing bridge edge; more complex real-world incompleteness patterns require future evaluation.

D. Ethical Considerations

Transparency: Synthesized edges are marked with confidence scores and evidence provenance for full auditability. Misuse: Grounding requirements mitigate false information injection, but adversarial corpus manipulation remains a concern. Environment: 3,940 tokens/query; organizations should evaluate necessity for their use case.

VIII. CONCLUSION

TSAAT addresses topological incompleteness via query-time graph synthesis. Through asynchronous multi-agent traversal with analytically characterized exploration scaling and dual-evidence validation, it achieves 91.2% path completion—49.1pp over GraphRAG—with 89.5% synthesis precision. The additional token cost may be justified in high-stakes domains where missed relationships have significant consequences.

IX. ACKNOWLEDGMENT

The author thanks the open-source research community for the tools, datasets, and benchmarks that made this work possible, including the developers of LangGraph, Neo4j, and the Hugging Face ecosystem. The author also thanks the anonymous reviewers for their constructive feedback.

REFERENCES

[1] D. Edge, H. Trinh, N. Cheng, J. Bradley, A. Chandra, A. Nguyen, R. Mody, S. Smith, and J. Larson, "From Local to Global: A Graph RAG Approach to Query-Focused Summarization," arXiv:2404.16130, 2024.

[2] J. Chen, R. Wang, and M. Liu, "Multi-Agent Collaborative Knowledge Graph Enrichment via Iterative Refinement," arXiv:2312.08091, 2024.

[3] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W. Yih, T. Rocktäschel, S. Riedel, and D. Kiela, "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks," in Proc. NeurIPS, 2020.

[4] S. Borgeaud, A. Mensch, J. Hoffmann, T. Cai, E. Rutherford, K. Millican, G. Van Den Driessche, J. Lespiau, B. Damoc, A. Clark et al., "Improving Language Models by Retrieving from Trillions of Tokens," in Proc. ICML, 2022.

[5] V. Karpukhin, B. Oguz, S. Min, P. Lewis, L. Wu, S. Edunov, D. Chen, and W. Yih, "Dense Passage Retrieval for Open-Domain Question Answering," in Proc. EMNLP, 2020.

[6] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, and O. Yakhnenko, "Translating Embeddings for Modeling Multi-Relational Data," in Proc. NeurIPS, 2013.

[7] T. Trouillon, J. Welbl, S. Riedel, E. Gaussier, and G. Bouchard, "Complex Embeddings for Simple Link Prediction," in Proc. ICML, 2016.

[8] L. A. Galárraga, C. Teflioudi, K. Hose, and F. Suchanek, "AMIE: Association Rule Mining under Incomplete Evidence in Ontological Knowledge Bases," in Proc. WWW, 2013.

[9] V. Parupally, "CalamanCy: A Tagalog Natural Language Processing Toolkit," in Proc. IEEE Int. Conf. on Industrial Technology & Computer Engineering (ICITCE), Penang, Malaysia, 2025, pp. 45–51, doi: 10.1109/ICITCE65255.2025.11210765.

[10] J. Wei, X. Wang, D. Schuurmans, M. Bosma, B. Ichter, F. Xia, E. Chi, Q. Le, and D. Zhou, "Chain-of-Thought Prompting Elicits Reasoning in Large Language Models," in Proc. NeurIPS, 2022.

[11] Z. Jiang, F. F. Xu, L. Gao, Z. Sun, Q. Liu, J. Dwivedi-Yu, Y. Yang, J. Callan, and G. Neubig, "Active Retrieval Augmented Generation," in Proc. EMNLP, 2023.

[12] S. Yao, J. Zhao, D. Yu, N. Du, I. Shafran, K. Narasimhan, and Y. Cao, "ReAct: Synergizing Reasoning and Acting in Language Models," in Proc. ICLR, 2023.

[13] A. Asai, Z. Wu, Y. Wang, A. Sil, and H. Hajishirzi, "Self-RAG: Learning to Retrieve, Generate, and Critique through Self-Reflection," in Proc. ICLR, 2024.

[14] V. R. Parupally, "A Multi-Objective Game Environment for Evaluating AI Agents," in Proc. 2nd Global AI Summit – Int. Conf. on Artificial Intelligence and Emerging Technology (AI Summit), Noida, India, 2025, pp. 692–697, doi: 10.1109/AISummit66170.2025.11410745.

[15] V. R. Parupally, "ATDLLMD: A Test-Driven Framework for Safe, Reliable, and Business-Centric LLM Development," IET Conf. Proc., vol. 2025, no. 43, 2025, doi: 10.1049/icp.2025.4778.

[16] A. Williams, N. Nangia, and S. Bowman, "A Broad-Coverage Challenge Corpus for Sentence Understanding through Inference," in Proc. NAACL, 2018.

[17] S. R. Bowman, G. Angeli, C. Potts, and C. D. Manning, "A Large Annotated Corpus for Learning Natural Language Inference," in Proc. EMNLP, 2015.

[18] Z. Yang, P. Qi, S. Zhang, Y. Bengio, W. W. Cohen, R. Salakhutdinov, and C. D. Manning, "HotpotQA: A Dataset for Diverse, Explainable Multi-Hop Question Answering," in Proc. EMNLP, 2018

[19] X. Ho, A.-K. Duong Nguyen, S. Sugawara, and A. Aizawa, "Constructing a Multi-Hop QA Dataset for Comprehensive Evaluation of Reasoning Steps," in Proc. COLING, 2020.

[20] P. Huguet Cabot and R. Navigli, "REBEL: Relation Extraction By End-to-end Language Generation," in Findings of EMNLP, 2021.

[21] J. M. van Hulst, F. Hasibi, K. Dercksen, K. Balog, and A. P. de Vries, "REL: An Entity Linker Standing on the Shoulders of Giants," in Proc. SIGIR, 2020.

[22] K. Santhanam, O. Khattab, J. Saad-Falcon, C. Potts, and M. Zaharia, "ColBERTv2: Effective and Efficient Retrieval via Lightweight Late Interaction," in Proc. NAACL, 2022.