KGNode: Path-Aware Subgraph Extraction for KG-Agnostic Knowledge-Grounded Question Answering

Anonymous ACL submission

Abstract

Large-scale knowledge graphs (KGs) use factual grounding for LLMs, but extracting query-relevant subgraphs from knowledge graphs with millions of entities remains challenging. Existing approaches either require expensive KG-specific training (entity linkers, KG embeddings) or lack query-awareness (random walks, fixed-depth BFS). We present KGNode, a training-free framework combining (1) hybrid seed discovery fusing semantic and keyword search with type-aware filtering, and (2) path-aware Markov chain traversal using priority-queue Breath First Search(BFS) with adaptive quality-based stopping. Unlike random walks that follow topology blindly, our approach computes transition probabilities as $P \propto \exp(\cos(\text{path}, \text{template}))$. Evaluation on DBLP-QuAD (252M triples, 400 questions) achieves 60%/72.5% end-to-end accuracy with *gpt-4o-mini*/*gpt-5-mini*, while QALD-10 (Wikidata, 1.65B triples) achieves 53%/66.5%, all without domain-specific training. Cross-domain analysis shows a 7 percentage point performance drop (DBLP→QALD), validating transferability across different knowledge graphs. However, component analysis identifies that label quality and relation extraction remain the primary limiting factors. The framework will be available as an open-source pip package.

Keywords: Knowledge Graphs, Subgraph Extraction, Question Answering, Training-Free Retrieval, Path-Aware Traversal

1 Introduction

Large language models often hallucinate facts (Ji et al., 2023). Knowledge graphs (KGs) can ground LLM responses with verified facts, but large-scale KGs contain billions of triples, far exceeding context windows. Given a query, how do we extract only the relevant subgraph?

For instance, consider answering "Did Kamil Zbikowski and Michal Ostapowicz co-author a paper?" from DBLP (252M triples). A naive 2-hop BFS from both authors retrieves 2,000+ triples including all their publications, co-authors, and venues. We need query-aware retrieval that prioritizes paths:

- Zbikowski → authoredBy → Paper$_i$ → authoredBy → Ostapowicz (relevant)

over irrelevant branches like:

- Zbikowski → authoredBy → Paper$_j$ → publishedIn → Venue (irrelevant)

This requires solving two challenges: (1) *seed discovery*—identifying starting entities from natural language, and (2) *path-aware traversal*—exploring high-relevance paths while pruning irrelevant branches. Existing approaches face a trade-off: training-based methods (entity linkers (Shen et al., 2015), KG embeddings (Wang et al., 2017)) achieve high accuracy but require expensive per-KG retraining; training-free methods (random walks (Perozzi et al., 2014a), fixed-depth BFS) transfer easily but lack query-awareness.

We present KGNode, a training-free framework combining: (1) hybrid seed discovery fusing semantic and keyword search with type-aware filtering (achieving 92-95% accuracy), and (2) path-aware Markov chain traversal using priority-queue BFS with exponential probability scoring ($P \propto \exp(\cos(\text{path}, \text{template}))$) and adaptive quality-based stopping. Our key insight: we convert graph paths into text sequences (e.g., "Kamil Zbikowski authoredBy Paper authoredBy Michal Ostapowicz"), enabling the use of standard pre-trained sentence embeddings without requiring training on the specific knowledge graph.

Evaluation on DBLP-QuAD (252M triples, 400 questions) achieves 72.5% end-to-end accuracy

with *gpt-5-mini* (95.2% seed discovery, 91.9% entity coverage), while QALD-10 (Wikidata, 1.65B triples, 394 questions) achieves 66.5% end-to-end accuracy, all without domain-specific training. Cross-domain performance drops 7 percentage points from DBLP to QALD, consistent across all pipeline stages, attributable to label sparsity rather than method failure. Component analysis identifies label quality, relation extraction and SPARQL generation as the primary bottleneck, suggesting semantic rather than structural limitations.

To summarize our contributions: First, we propose a path-aware Markov chain traversal algorithm using query-derived templates and exponential probability scoring for training-free subgraph extraction, with adaptive stopping that monitors quality distribution. Second, we present a hybrid seed discovery approach combining semantic and keyword search with type-aware filtering, achieving 92-95% accuracy without entity linking training. Third, we provide a comprehensive two-dataset evaluation (DBLP-QuAD, QALD-10) with component-level error analysis identifying failure modes and cross-domain degradation patterns. Finally, we release an open-source implementation demonstrating practical deployment without KG-specific training.

2 Related Work

The following are popular existing approaches for knowledge graph question answering and subgraph retrieval, which can be categorized into four main areas.

2.1 Knowledge Graph Question Answering

Traditional KGQA systems (Berant et al., 2013; Lan et al., 2021; Omar et al., 2023) directly translate natural language queries into executable SPARQL queries through semantic parsing. In contrast, our approach extracts relevant subgraphs for LLM-based question answering, providing flexibility for queries difficult to express in SPARQL.

2.2 Entity Linking

Entity linking methods (Mendes et al., 2011; Ferragina and Scaiella, 2010; Kolitsas et al., 2018; Wu et al., 2020) map text mentions to knowledge graph entities using either hand-crafted features (Mendes et al., 2011; Ferragina and Scaiella, 2010) or neural bi-encoders (Kolitsas et al., 2018; Wu et al., 2020) trained on millions of entity mentions. These approaches require extensive KG-specific training data, making them expensive to deploy on new domains. Our hybrid seed discovery combines semantic and keyword search without entity-specific training, achieving comparable accuracy while remaining transferable.

2.3 Knowledge Graph Embeddings

KG embedding methods (Bordes et al., 2013; Sun et al., 2019; Trouillon et al., 2016; Yang et al., 2015) learn low-dimensional vector representations by training on graph structure, using techniques like translations (Bordes et al., 2013), rotations (Sun et al., 2019), or graph neural networks (Schlichtkrull et al., 2018; Vashishth et al., 2020). Some systems use supervised fine-tuning (SFT) with golden seeds, training on manually annotated entity and relation pairs to improve accuracy. While effective, these methods require expensive retraining for each new knowledge graph. Our method deliberately avoids KG-specific training, instead leveraging pre-trained sentence embeddings that work across different knowledge graphs without modification.

2.4 Subgraph Retrieval

Subgraph retrieval methods extract relevant portions of knowledge graphs for downstream tasks. Random walk approaches (Perozzi et al., 2014b; Grover and Leskovec, 2016) rely purely on graph topology without query-awareness, while trainable retrieval systems (Zhang et al., 2022; Ding et al., 2024) use graph neural networks or reinforcement learning but require large amounts of query-subgraph training pairs. In contrast, our path-aware Markov chain traversal computes query-aware transition probabilities using pre-trained embeddings, achieving query-awareness without KG-specific training.

3 Methodology

Given knowledge graph $G = (V, E, L)$ and natural language query q, the goal is to extract a relevant subgraph $S \subseteq G$ containing triples that help answer the query. KGNode (Figure 1) operates in three stages: (1) *Index Compilation*—one-time preprocessing to build vector databases for entities and relations, (2) *Seed Discovery*—identify starting entities via hybrid search, and (3) *Path-Aware Traversal*—explore semantically aligned paths using Markov chain BFS. This section describes each component, followed by complexity analysis and implementation details.

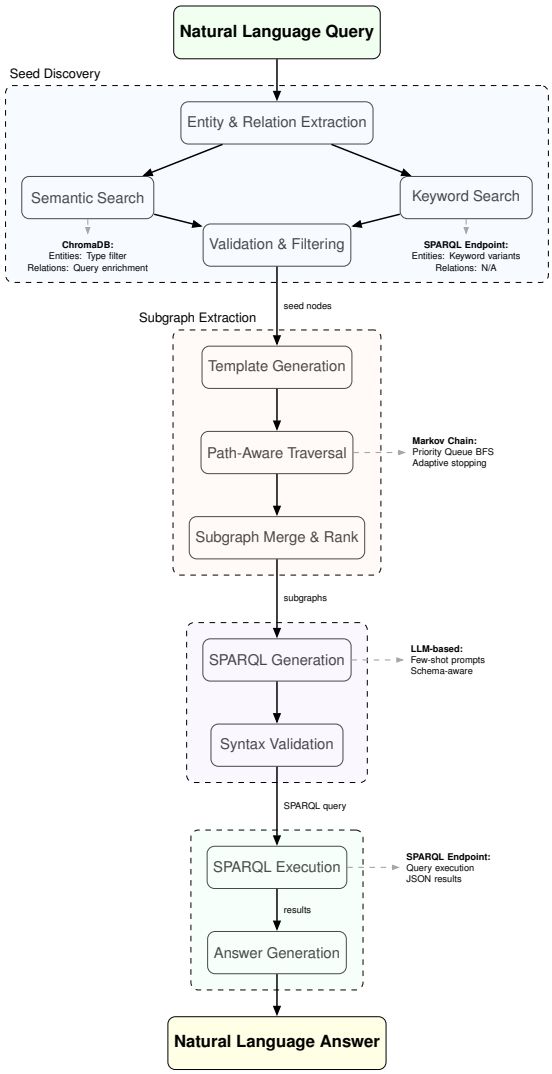

Figure 1: KGNode architecture: seed node discovery via hybrid search, followed by path-aware subgraph extraction.

3.1 Index Compilation

KGNode requires one-time preprocessing to enable fast semantic search over millions of entities and relations. This compilation step builds two vector databases that are reused across all queries.

Entity Index: To enable fast semantic search, we pre-compile a vector index of high-connectivity nodes. We select top-N entities by out-degree (e.g., 1000K most connected nodes), convert each to a rich text description ("Person: John Smith, MIT, ML, 42 pubs"), embed using `all-MiniLM-L6-v2` (384-dim), and store in ChromaDB vector database. This filters the entity space to high-value nodes while keeping index size manageable. Compilation time depends on N but is a one-time cost.

Relations and entity type Index: To reduce LLM hallucination of non-existent predicates, we ground queries in actual KG vocabulary. We extract entity types and unique relations from the KG ontology (RDF/OWL files) or via SPARQL queries. Each type is embedded using the same `all-MiniLM-L6-v2` model and stored in a separate ChromaDB collection. At query time, we retrieve top-k relevant entity types and relations to provide schema context to the LLM, enabling generation of grounded templates (e.g., "John Smith authoredBy publications publishedIn ICML" using actual predicates vs generic "John Smith papers ICML"). Like entity indexing, relations and entity type compilation is performed once and cached for reuse.

3.2 Seed Node Discovery

Seed discovery identifies starting entities and relations mentioned in the query, serving as entry points for graph exploration (Figure 2). First, an LLM identifies entity names and relation names from the query text, along with their expected types (e.g., "John Smith" as Creator, "publish" as authoredBy predicate). For entity search, we use two complementary approaches in parallel: (1) semantic search finds similar entities in the vector database using embedding similarity, filtered by the predicted entity types, and (2) keyword search finds entities through SPARQL queries using name variations. For relation search, we perform semantic search in the relation vector database using the extracted relation names. Since incorrect seed nodes lead to exponential growth in subgraph exploration time, rigorous filtering is essential to remove irrelevant candidates. The entity candidates are filtered in two steps: first, we remove any literal values (non-entity nodes), then we rank by label similarity using fuzzy string matching and keep only top-scoring candidates based on adaptive thresholds (Table 2). Relation candidates are filtered by sorting by embedding distance and applying adaptive thresholds. The output is a list of validated entity URIs and relation URIs with their similarity scores.

3.3 Path-Aware Markov Chain Traversal

Path-aware traversal explores from seeds while prioritizing semantically relevant paths. Unlike random walks (topology-based) or trainable retrievers (KG-specific training), we use query-guided probability scoring with pre-trained embeddings. An LLM generates template $\mathbf{t}$ from query + seeds (Table 1). Paths $P = [v_0, r_1, v_1, \ldots, v_n]$ become text ("Robert Schober authorOf Survey publishedIn

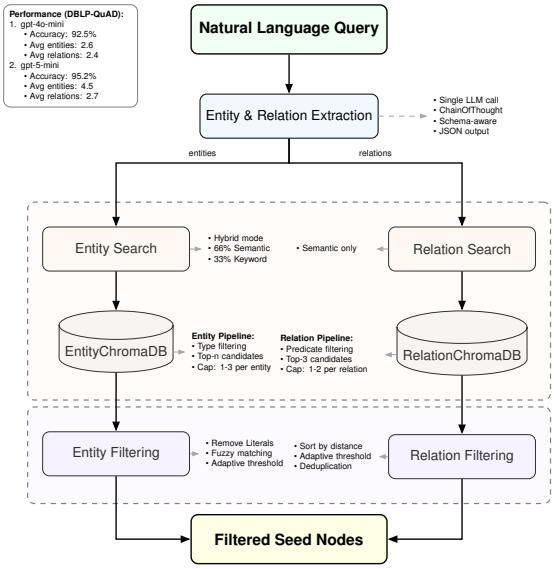

Figure 2: Seed node discovery: hybrid search combining semantic similarity and keyword matching.

Query	Template
Show the Wikidata ID of the person Robert Schober.	Robert Schober Wikidata ID
What is the average number of publications by Anthony G. Constantinides per year?	Anthony G. Constantinides publications average
Did A. Hertz publish in the last 5 years?	A. Hertz publications recent

Table 1: Example query-to-template transformations from DBLP-QuAD. Templates capture the semantic structure of expected answer paths.

ICML") and are embedded. Transition probability:

$$P(v_n \to w \mid P, q) \propto \exp\left(\cos(\mathbf{p}', \mathbf{t})\right) \quad (1)$$

where $\mathbf{p}'$ is the embedding of the extended path $P' = P \oplus [r, w]$ (the current path P plus the new edge consisting of relation r and neighbor node w). The exponential function transforms cosine similarity values from the range $[-1, 1]$ to $[0.37, 2.72]$, which amplifies the difference between relevant and irrelevant paths by a factor of $7\times$ (ratio between $\exp(1)$ and $\exp(-1)$), making the priority queue more effective at selecting promising paths first.

Algorithm: We use priority queue breadth-first search (BFS) to explore the graph, where each state in the queue is represented as $(v, P, P_{\max}, \text{hop})$: v is the current node being explored, P is the list of nodes and relations traversed so far (the path), $P_{\max}$ is the maximum probability seen along this path, and hop is the path depth (Figure 3). The

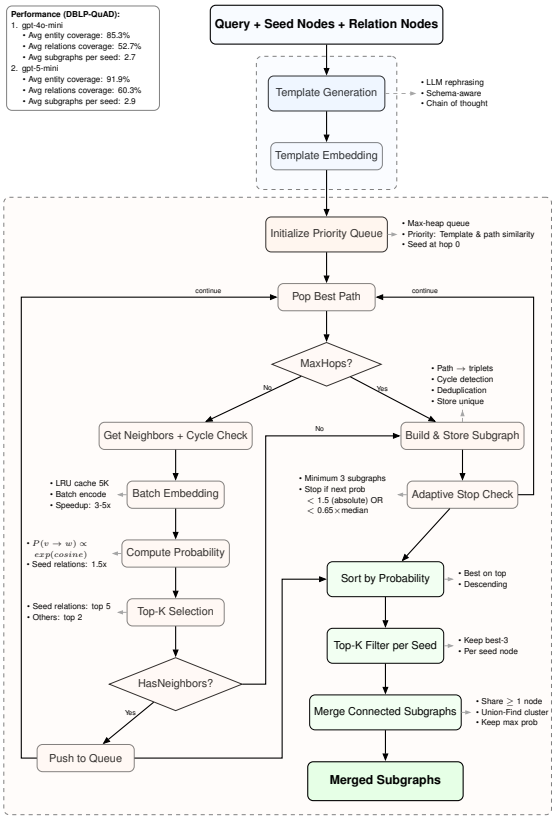

Figure 3: Path-aware traversal with probability-based pruning guided by template similarity.

queue prioritizes states with higher probabilities, ensuring we explore the most promising paths first. For each node, we query its neighbors in both directions (outgoing and incoming edges) using parallel SPARQL queries, then compute embeddings for all candidate paths in batches for efficiency. We apply a monotonicity constraint to prune the search space: a neighbor is added to the queue only if its probability P exceeds the parent's maximum probability $P_{\max}$, filtering out paths that become less relevant. Paths using relations identified during seed discovery receive a probability boost ($1.5\times$) and can select more neighbors (top-5 instead of top-2), prioritizing query-relevant predicates. The algorithm stops adaptively when it has collected sufficient subgraphs (minimum 3) and the next best path has significantly lower probability than collected paths (Table 2).

3.4 Computational Complexity

Time Complexity (per query):

- Entity/relation extraction: 1 LLM call to extract M entities and R relations from query.
- Seed discovery: $O(M \cdot (\log k + K) + R \cdot \log k)$

for M entities (approximate nearest neighbor (ANN) search in vector DB + K keyword SPARQL variants) and R relations (ANN search), yielding S filtered seeds.
- Template generation: 1 LLM call using extracted entities and relations.
- Path traversal: $O(S \cdot N \cdot D \cdot F)$ where S = filtered seed count, N = avg nodes visited per seed, D = avg node degree, F = embedding cost. Adaptive stopping reduces N by $\sim$70%.
- SPARQL generation: 1 LLM call with top-G subgraphs as context.

Space Complexity:
Preprocessing (one-time storage):
- Entity vector DB: $O(N \cdot d)$ for N entity embeddings with dimension $d = 384$ (ChromaDB).
- Relation vector DB: $O(R \cdot d)$ for R relation embeddings.

Runtime (per query):
- Path embedding cache: $O(C \cdot d)$ for LRU cache with C entries (default: 5000).
- Label cache: $O(V)$ for V node labels retrieved via SPARQL during traversal.
- Subgraphs: $O(S \cdot P)$ for S collected subgraphs with avg path length P.

3.5 Default Hyperparameters

Table 2 shows the default parameter values used in KGNode. These values are selected based on empirical performance and computational constraints.

Seed Discovery Thresholds: The semantic similarity threshold (0.6) balances precision and recall in vector database retrieval, filtering out weakly similar entities while maintaining coverage. The fuzzy matching threshold (0.8) ensures strict label matching to avoid false positives during seed filtering, as incorrect seeds cause exponential growth in traversal time.

Subgraph Extraction Limits: We collect minimum 3 subgraphs to ensure sufficient coverage for diverse query interpretations, while capping at 15 to prevent excessive computation. The quality threshold ratio (0.65) implements adaptive stopping by comparing the next path probability against 65% of the median collected probability. The absolute probability threshold (1.5 = $\exp(0.4)$) provides a hard minimum cutoff, discarding paths with cosine similarity below 0.4. For seed relations (relations mentioned in the query), we explore more neighbors (max-k = 5) with 1.5$\times$ probability boost, prioritizing query-relevant predicates over general exploration (max-k = 2).

Parameter	Default
Semantic similarity threshold	0.6
Fuzzy matching threshold	0.8
Min subgraphs collected	3
Max subgraphs per query	15
Quality threshold ratio	0.65
Absolute probability threshold	1.5
Seed relation boost factor	1.5$\times$
Seed relation neighbors (max-k)	5
Other relation neighbors (max-k)	2
Schema top-k entity types	10
Schema top-k relation types	15
Subgraphs for SPARQL generation	5
Few-shot SPARQL examples	9

Table 2: Key algorithmic parameters with default values.

Schema Configuration: During entities and relations extraction from query, we provide the LLM with top-10 most relevant entity types and top-15 relations from the knowledge graph schema. This grounds the extraction in actual KG vocabulary, reducing hallucination of non-existent predicates while keeping prompt size manageable.

SPARQL Generation: We include top-5 highest-scoring subgraphs as context for SPARQL generation, balancing answer coverage with prompt length. Nine carefully selected few-shot examples cover diverse query patterns (Boolean, Count, Superlative, Comparative, Negation, Union) to guide the LLM toward syntactically correct and semantically appropriate SPARQL queries.

4 Evaluation

We evaluate KGNode on large-scale knowledge graph question answering to assess the effectiveness of our KG-agnostic approach.

4.1 Experimental Setup

Datasets: We evaluate on two benchmarks: (1) DBLP-QuAD (Banerjee et al., 2023) (252M triples, 92M entities), a question answering benchmark over the DBLP scholarly knowledge graph, using an equally distributed question type subset containing 400 questions across 10 query types: Boolean, Count, Disambiguation, Double Intent, Double Negation, Multi-Fact, Negation, Single Fact, Superlative+Comparative, and Union. (2) QALD-10 (Usbeck et al., 2024) (1.65B triples, 120M entities), a question answering benchmark over Wikidata, using the test set containing 394 questions

#	Golden	Generated	State
1	Error/Timeout	Error/Timeout	Definitely Wrong
2	Error/Timeout	Success + Results	Likely Correct
3	Error/Timeout	Success + Empty	Likely Correct
4	Success + Results	Error/Timeout	Definitely Wrong
5	Success + Results	Success + Results	Likely Correct
6	Success + Results	Success + Empty	Likely Wrong
7	Success + Empty	Error/Timeout	Definitely Wrong
8	Success + Empty	Success + Results	Definitely Correct
9	Success + Empty	Success + Empty	Definitely Correct

Table 3: Nine-case SPARQL evaluation logic.

designed to evaluate complex SPARQL query patterns including COUNT, FILTER, ASK, GROUP BY, and OFFSET operators. Compared to DBLP-QuAD, QALD-10 is particularly challenging because Wikidata is highly heterogeneous and multilingual, with many alternative labels, aliases, symbols, and dense neighborhoods per entity, which increases ambiguity during seed discovery and subgraph extraction.

Models: We use GPT-4o-mini and GPT-5-mini as LLM backends, with `all-MiniLM-L6-v2` for text embeddings (384 dimensions). Algorithmic parameters follow Table 2. The SPARQL endpoint is served by Oxigraph.

Metrics: For seed discovery, we measure what percentage of golden entities and relations are successfully discovered. For subgraph extraction, we measure what percentage of gold entities appear in the extracted subgraphs (entity coverage), what percentage of gold relations are captured (relation coverage), and the average number of subgraphs extracted per seed node. For SPARQL generation, we measure whether the generated query runs without errors (executable rate), how well query results match the gold answers (F1 score), and classify query correctness using the nine-case evaluation logic into four categories: Definitely Correct, Definitely Wrong, Likely Correct, or Likely Wrong based on execution outcomes. This provides a more nuanced evaluation than binary correctness, distinguishing definitely correct/wrong cases from likely correct/wrong cases. Finally, we measure end-to-end accuracy for the complete pipeline from question to final answer. For error analysis of Likely Correct and Likely Wrong cases, an LLM judge assigns failure categories by comparing both golden and generated SPARQL queries along with their execution results. Table 5 presents detailed SPARQL evaluation metrics.

Approach	gpt-4o-mini		gpt-5-mini	
	DBLP-QuAD	QALD-10	DBLP-QuAD	QALD-10
Schema	10%	14.3%	11%	27%
KG Name + GS	22%	20%	43%	30%
Schema + ES	27%	21.4%	46%	34.7%
Schema + GS	32%	24.3%	53%	35%
KGNode	**60%**	**53%**	**72.5%**	**66.5%**

Table 4: Baseline comparison across models and datasets. GS: Golden Seed, ES: Extracted Seed.

Metric	gpt-4o-mini		gpt-5-mini	
	DBLP-QuAD	QALD-10	DBLP-QuAD	QALD-10
Executable	98.2%	87.6%	99.2%	94.4%
F1 Score	0.638	0.602	0.749	0.670
Definitely Wrong	1.8%	12.4%	0.8%	5.6%
Definitely Correct	14.0%	21.1%	15.0%	24.6%
Likely Wrong	15.2%	14.0%	8.0%	16.5%
Likely Correct	69.0%	52.5%	76.2%	53.3%

Table 5: SPARQL evaluation metrics (%).

4.2 Results

We compare KGNode against several baselines across two datasets (DBLP-QuAD and QALD-10) and two model configurations (GPT-4o-mini and GPT-5-mini). Table 4 shows the baseline comparison, where GS denotes Golden Seed (oracle seed nodes), ES denotes Extracted Seed (our seed finding). KGNode demonstrates competitive performance without requiring KG-specific training or golden seeds.

On DBLP-QuAD with GPT-4o-mini, KGNode achieves 92.5% accuracy in seed discovery with an average of 2.6 entities and 2.4 relations extracted per query. Subgraph extraction attains 85.3% entity coverage and 52.7% relation coverage with an average of 2.7 subgraphs per seed, validating the path-aware traversal approach. The end-to-end pipeline achieves 60% accuracy, with SPARQL generation identified as the primary bottleneck (with 98.2% executable rate, F1 score of 0.638). With GPT-5-mini, performance improves to 95.2% seed discovery accuracy (4.5 entities, 2.7 relations per query), 91.9% entity coverage, and 60.3% relation coverage in subgraph extraction (2.9 subgraphs per seed), and 72.5% end-to-end accuracy (with 99.2% executable, F1 score of 0.749).

4.3 Error Analysis

The training-free framework achieves competitive performance without domain-specific customization, a significant advantage over supervised approaches. We analyze failures at each pipeline

Category	gpt-4o-mini		gpt-5-mini	
	DBLP-QuAD	QALD-10	DBLP-QuAD	QALD-10
Correct Equivalent	46.0%	31.97%	57.5%	41.9%
Execution Failed	1.8%	12.4%	0.8%	5.6%
Over-Constrained	2.2%	0.8%	1.0%	1.3%
Under-Constrained	3.8%	3.0%	3.0%	4.1%
Missing Negation	5.0%	0.3%	0.8%	0.0%
Missing OR Logic	3.0%	0.3%	5.8%	0.0%
Wrong Entity URI	7.0%	6.1%	5.0%	12.7%
Wrong Predicate	16.8%	23.4%	11.2%	9.4%
Wrong Aggregation	0.4%	0.8%	0%	0%

Table 6: Error distribution (%) across models and datasets.

stage using gpt-4o-mini on 400 DBLP-QuAD questions to identify improvement opportunities.

Seed Discovery: The system successfully finds at least one golden entity for 370 out of 400 questions (92.5%), with 30 questions (7.5%) having no correct seed entities discovered. Common failure patterns include type mismatch (e.g., "MIT" detected as Person not Organization), keyword ambiguity (e.g., "Wang" matches 200+ entities), and low out-degree entities missing from the index.

Subgraph Extraction: Starting from 1045 filtered seeds (average 2.6 seeds per question), the system extracts 5200 subgraphs (average 5.0 subgraphs per seed), which are then filtered down to 2786 final subgraphs. This achieves 85.3% coverage of golden entities (492 out of 577 total) and 52.7% coverage of golden relations (337 out of 639 total). The 14.7% missing entities and 47.3% missing relations indicate opportunities for improving template generation and path-aware traversal to capture more relevant graph structure.

SPARQL Generation: With 160 out of 400 questions (40%) failing to produce correct queries using gpt-4o-mini. However, most generated queries are syntactically valid (98-99% executable rate), indicating semantic rather than structural problems. Table 6 shows detailed error breakdown, where an LLM judge categorizes failures for Likely Correct and Likely Wrong cases by comparing golden and generated queries. The most common errors are wrong predicate selection, missing UNION logic, and incorrect entity URIs.

Query Type Analysis: Performance varies significantly across query types. SPARQL generation struggles particularly with disambiguation, double intent, union, and multi-fact queries, where identifying the correct entities and relations from the query becomes challenging due to ambiguous references or complex constraints. These patterns suggest that improving entity and relation extraction (currently 52-60% relation coverage) could address semantic correctness issues without fundamental method changes.

Metric	DBLP-QuAD	QALD-10	Δ
At Least One Entity	92.5%	87.3%	-5.2%
Entity Coverage	85.3%	78.1%	-7.2%
Relation Coverage	52.7%	44.9%	-7.8%
End-to-End Acc	60.0%	53.0%	-7.0%

Table 7: Cross-domain performance comparison (gpt-4o-mini). Δ = absolute difference in percentage points (QALD-10 minus DBLP-QuAD). Consistent degradation indicates systematic challenge (label sparsity) rather than method overfitting.

4.4 Cross-Domain Generalization (QALD-10)

To validate training-free transferability, we evaluate on QALD-10 (Wikidata, general knowledge domain) without modifying hyperparameters or retraining any components or even changing the prompts, QALD-10 achieves 53% (gpt-4o-mini) and 66.5% (gpt-5-mini) end-to-end accuracy (Table 4), representing a 7 percentage point drop from DBLP-QuAD. Table 7 compares performance across datasets. Notably, degradation is consistent across all stages:

Consistent degradation across stages suggests a systematic challenge rather than method failure. A primary cause is the noisier nature of Wikidata compared to DBLP. Wikidata is multilingual and highly connected, so seed neighborhoods often contain labels from multiple languages and semantically weak or ambiguous surface forms. In addition, many entities have sparse or low-quality `rdfs:label` annotations (35% entities lack human-readable labels vs 5% in DBLP). These characteristics make both seed discovery and path serialization more difficult.

Despite this, the framework achieves majority accuracy (>50%) on both datasets without domain-specific training, or even prompt changing, validating the training-free claim.

5 Conclusion

This paper presents KGNode, a training-free framework for query-aware subgraph retrieval from large knowledge graphs. The core contribution is a path-aware Markov chain traversal algorithm using exponential probability scoring ($P \propto \exp(\cos(\text{path}, \text{template}))$) and adaptive stopping,

combined with hybrid seed discovery achieving 92-95% accuracy. Evaluation on two benchmarks (DBLP-QuAD, QALD-10) demonstrates competitive performance without KG-specific training, achieving 60-72.5% accuracy. Cross-domain analysis shows a 7 percentage point degradation attributable to label sparsity, validating transferability. Component-level error analysis identifies SPARQL generation as the primary bottleneck (with 0.638 - 0.749 F1 Score and 98-99% syntax validity), suggesting prompt engineering as one of the impactful improvement directions. The framework will be available as open-source.

6 Limitations

KGNode has several key limitations. First, entity and schema indexing requires one-time compilation that may be non-trivial for extremely large knowledge graphs with millions of high-degree entities. Second, the framework relies on meaningful entity and relation descriptors for vector database indexing, specifically `rdfs:label` annotations, and knowledge graphs with sparse or low-quality labels (e.g., auto-generated identifiers like "entity_12345") will experience degraded seed discovery performance. Third, the framework makes multiple LLM API calls per query (entity extraction, template generation, SPARQL generation), introducing latency and financial cost that may be prohibitive for high-throughput applications. Fourth, relation extraction remains challenging, with 52-60% relation coverage leading to suboptimal SPARQL generation. Finally, while we evaluate on two complementary benchmarks (DBLP-QuAD and QALD-10 covering scholarly and general knowledge domains), additional domain evaluation (biomedical, e-commerce) would strengthen transferability claims.

7 Future Work

Two promising future directions could address these limitations. First, one-time prompt optimization using frameworks like DSPy could systematically improve entity extraction, template generation, and SPARQL generation prompts without per-query overhead, potentially addressing the relation extraction and SPARQL correctness bottlenecks. Second, an agentic framework implementation using LangGraph with automatic hyperparameter tuning could adapt traversal parameters (quality thresholds, neighbor limits, stopping criteria) based on knowledge graph characteristics and query types, improving performance across diverse domains while reducing manual configuration effort.

8 Ethical Considerations

KGNode operates on existing knowledge graphs and inherits any biases or inaccuracies present in the source data, which may perpetuate historical inequities in knowledge representation. The framework relies on commercial LLM APIs with non-trivial environmental costs, and we recommend users consider local models or carbon offset programs for large-scale deployments. While grounding LLM responses in knowledge graph triples improves answer transparency and enables source attribution, users should verify critical information against original sources as subgraph extraction errors may lead to misleading responses. Finally, deployers should implement appropriate access controls and usage monitoring to prevent misuse for scraping or extracting sensitive information from knowledge graphs containing personal or proprietary data.

References

Debayan Banerjee, Sushil Awale, Ritam Biswas, Ricardo Usbeck, and Chris Biemann. 2023. Dblp-quad: A question answering dataset over the dblp scholarly knowledge graph. In *Advances in Information Retrieval - 45th European Conference on Information Retrieval, ECIR 2023, Dublin, Ireland, April 2-6, 2023, Proceedings, Part II*, volume 13981 of *Lecture Notes in Computer Science*, pages 121–136. Springer.

J. Berant, A. Chou, R. Frostig, and P. Liang. 2013. Semantic parsing on freebase from question-answer pairs. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1533–1544.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Advances in Neural Information Processing Systems*, volume 26, pages 2787–2795. Curran Associates, Inc.

Wentao Ding, Jinmao Li, Liangchuan Luo, and Yuzhong Qu. 2024. Enhancing complex question answering over knowledge graphs through evidence pattern retrieval. In *Proceedings of the ACM Web Conference 2024*.

P. Ferragina and U. Scaiella. 2010. Tagme: On-the-fly annotation of short text fragments (by wikipedia

entities). In *Proceedings of the 19th ACM International Conference on Information and Knowledge Management*, pages 1625–1628.

A. Grover and J. Leskovec. 2016. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 855–864.

Z. Ji, N. Lee, R. Frieske, T. Yu, D. Su, Y. Xu, E. Ishii, Y. J. Bang, A. Madotto, and P. Fung. 2023. Survey of hallucination in natural language generation. *ACM Computing Surveys*, 55(12):1–38.

N. Kolitsas, O. E. Ganea, and T. Hofmann. 2018. End-to-end neural entity linking. In *Proceedings of the 22nd Conference on Computational Natural Language Learning*, pages 519–529.

Y. Lan, G. He, J. Jiang, J. Jiang, W. X. Zhao, and J. R. Wen. 2021. A survey on complex knowledge base question answering: Methods, challenges and solutions. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence*, pages 4483–4491.

P. N. Mendes, M. Jakob, A. García-Silva, and C. Bizer. 2011. Dbpedia spotlight: Shedding light on the web of documents. In *Proceedings of the 7th International Conference on Semantic Systems*, pages 1–8.

R. Omar, I. Dhall, P. Kalnis, and E. Mansour. 2023. A universal question-answering platform for knowledge graphs. In *Proceedings of the 2023 ACM SIGMOD International Conference on Management of Data*, pages 1–28.

B. Perozzi, R. Al-Rfou, and S. Skiena. 2014a. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 701–710.

B. Perozzi, R. Al-Rfou, and S. Skiena. 2014b. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 701–710.

M. Schlichtkrull, T. N. Kipf, P. Bloem, R. Van Den Berg, I. Titov, and M. Welling. 2018. Modeling relational data with graph convolutional networks. In *Proceedings of the 15th Extended Semantic Web Conference*, pages 593–607.

W. Shen, J. Wang, and J. Han. 2015. Entity linking with a knowledge base: Issues, techniques, and solutions. *IEEE Transactions on Knowledge and Data Engineering*, 27(2):443–460.

Z. Sun, Z. H. Deng, J. Y. Nie, and J. Tang. 2019. Rotate: Knowledge graph embedding by relational rotation in complex space. In *Proceedings of the 7th International Conference on Learning Representations*.

Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. 2016. Complex embeddings for simple link prediction. In *International conference on machine learning*, pages 2071–2080. PMLR.

Ricardo Usbeck, Xiaoyu Yan, Alexey Perevalov, Mathieu d'Aquin, and Jeff Z. Pan. 2024. Qald-10 – the 10th challenge on question answering over linked data: Shifting from dbpedia to wikidata as a kg for kgqa. *Semantic Web*, 15(6):2193–2207.

S. Vashishth, S. Sanyal, V. Niber, and P. Talukdar. 2020. Composition-based multi-relational graph convolutional networks. In *Proceedings of the 8th International Conference on Learning Representations*.

Q. Wang, Z. Mao, B. Wang, and L. Guo. 2017. Knowledge graph embedding: A survey of approaches and applications. *IEEE Transactions on Knowledge and Data Engineering*, 29(12):2724–2743.

L. Wu, F. Petroni, M. Josifoski, S. Riedel, and L. Zettlemoyer. 2020. Scalable zero-shot entity linking with dense entity retrieval. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*, pages 6397–6407.

B. Yang, W. T. Yih, X. He, J. Gao, and L. Deng. 2015. Embedding entities and relations for learning and inference in knowledge bases. In *Proceedings of the 3rd International Conference on Learning Representations*.

Jing Zhang, Xiaokang Zhang, Jifan Yu, Jian Tang, Jie Tang, Cuiping Li, and Hong Chen. 2022. Subgraph retrieval enhanced model for multi-hop knowledge base question answering. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics*, pages 5773–5784.