Hierarchical Lexical Graph for Enhanced Multi-Hop Retrieval

Abdellah Ghassel[*][†]
Queen's University
Kingston, Canada
abdellah.ghassel@queensu.ca

Ian Robinson[*]
Amazon
London, England
ianrob@amazon.co.uk

Gabriel Tanase
Amazon
Seattle, USA
igtanase@amazon.com

Hal Cooper
Amazon
Seattle, USA
halcoope@amazon.com

Bryan Thompson
Amazon
Seattle, USA
bryant@amazon.com

Zhen Han
Amazon
Santa Clara, USA
zhenhz@amazon.com

Vassilis N. Ioannidis
Amazon
Santa Clara, USA
ivasilei@amazon.com

Soji Adeshina
Amazon
Santa Clara, USA
adesojia@amazon.com

Huzefa Rangwala
Amazon
Santa Clara, USA
rhuzefa@amazon.com

Abstract

Retrieval-Augmented Generation (RAG) grounds large language models in external evidence, yet it still falters when answers must be pieced together across semantically distant documents. We close this gap with the Hierarchical Lexical Graph (HLG), a three-tier index that (i) traces every atomic proposition to its source, (ii) clusters propositions into latent topics, and (iii) links entities and relations to expose cross-document paths. On top of HLG we build two complementary, plug-and-play retrievers: StatementGraphRAG, which performs fine-grained entity-aware beam search over propositions for high-precision factoid questions, and TopicGraphRAG, which selects coarse topics before expanding along entity links to supply broad yet relevant context for exploratory queries. Additionally, existing benchmarks lack the complexity required to rigorously evaluate multi-hop summarization systems, often focusing on single-document queries or limited datasets. To address this, we introduce a synthetic dataset generation pipeline that curates realistic, multi-document question-answer pairs, enabling robust evaluation of multi-hop retrieval systems. Extensive experiments across five datasets demonstrate that our methods outperform naive chunk-based RAG, achieving an average relative improvement of 23.1% in retrieval recall and correctness. [1]

CCS Concepts

• **Information systems** → **Novelty in information retrieval**; • **Computing methodologies** → Lexical semantics; *Information extraction.*

Keywords

question answering; graph structures; data generation

ACM Reference Format:
Abdellah Ghassel, Ian Robinson, Gabriel Tanase, Hal Cooper, Bryan Thompson, Zhen Han, Vassilis N. Ioannidis, Soji Adeshina, and Huzefa Rangwala. 2025. Hierarchical Lexical Graph for Enhanced Multi-Hop Retrieval. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.2 (KDD '25), August 3–7, 2025, Toronto, ON, Canada.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3711896.3737233

1 Introduction

Retrieval-Augmented Generation (RAG) systems have gained attention for enhancing large language models (LLMs) with external knowledge, enabling more grounded and accurate responses to complex questions [13, 17]. Despite these advances, current RAG systems face significant challenges with multi-hop reasoning, where answers may be synthesized from multiple, semantically diverse text segments or documents [21]. For example, consider the query: *'How did the FTC lawsuit affect the stock of one of the leading e-commerce companies?'* This requires retrieving facts about the lawsuit, Amazon's financial performance, and market reactions, likely dispersed across multiple, unconnected documents.

Most RAG systems struggle in such scenarios because they rely primarily on vector similarity search (VSS). While VSS excels at finding texts that closely match the query's surface-level semantics, it often fails to bridge related but contextually distant pieces of information [5]. In the previous example, connecting 'company lawsuit details' from one document with 'stock performance data' in another may require structured links or graph-based expansions, as they lack obvious semantic ties. Graph-based retrieval strategies provide a promising solution as they model explicit edges, such as shared entities or discourse links, thereby overcoming the narrowness of vector-based methods [10, 12, 14].

[*] Equal contribution.
[†] Work done during internship at Amazon.
[1] Open-source Python library is available at `github.com/awslabs/graphrag-toolkit`

KDD '25, Toronto, ON, Canada

ACM ISBN 979-8-4007-1454-2/2025/08
https://doi.org/10.1145/3711896.3737233

In addition, we emphasize the importance of retrieval-unit granularity. Conventional systems often utilize large text chunks, consisting of various sentences, leading to the retrieval of extraneous information. To address this, Chen et al. [7] advocate for smaller retrieval units, such as atomic propositions, to enhance precision in information retrieval.

Building on these insights, we propose a Hierarchical Lexical Graph (HLG) framework to address the gap between surface-level similarity and structured multi-hop evidence. While prior work emphasizes the benefits of finer retrieval units [7], HLG extends these ideas by integrating three interconnected tiers: Lineage, Summarization, and Entity-Relationship. Specifically, HLG (1) preserves the lineage of each statement for accurate provenance, (2) clusters statements around topics for flexible retrieval, and (3) links entities and relationships to enable bottom-up graph traversal. This multi-tier design enables more precise retrieval of diverse facts, even when they share minimal semantic overlap.

Using this framework, we propose two complementary RAG methods tailored to different retrieval needs:

- **StatementGraphRAG** focuses on individual propositions in the Summarization Tier. It links them across documents using the Entity-Relationship Tier and preserves provenance via the Lineage Tier. This is ideal for detailed queries needing high-precision evidence (e.g., factoid questions).
- **TopicGraphRAG** retrieves clusters of statements (i.e., topical groups) from the Summarization Tier, using entity relationships to connect thematically related clusters and relying on the Lineage Tier for source traceability. This is efficient for broader, open-ended or higher-level queries.

The choice depends on query type: StatementGraphRAG for specific, single-answer queries, and TopicGraphRAG for exploratory or multi-faceted questions.

Despite the promise of these approaches, a major obstacle remains: most existing QA datasets fail to demand true multi-hop retrieval, either restricting questions to single documents or offering a small set of queries. WikiHowQA [6], for example, focuses on single-document responses, while the SEC10Q [15] dataset contains only 195 queries. To address this gap, we present a synthetic multi-hop summarization pipeline using HLG as a backbone. We generate 674 question-answer pairs from the MultiHop-RAG corpus [21], then apply a second-pass LLM critique and automated retrieval checks [19] to ensure each question genuinely requires multi-hop inference and has sufficient supporting evidence.

Our paper makes the following contributions:

(1) We introduce StatementGraphRAG and TopicGraphRAG, advancing multi-hop QA with fine-grained and structured retrieval capabilities.
(2) We develop a pipeline synthesizing high-quality multi-hop summarization queries, closing gaps in existing benchmarks.
(3) Through extensive experiments across diverse datasets, we show significant performance gains over chunk-based RAG baselines.

2 Related Work

2.1 Multi-hop Reasoning with Graphs

Graph-based approaches have been actively used in modelling relationships required for effective multi-hop reasoning. Fang et al. [12] introduced the Hierarchical Graph Network (HGN) for multi-hop question answering, which integrates nodes of varying granularity including questions, paragraphs, sentences, and entities using graph neural networks. This facilitates joint prediction of answers and supporting facts by enabling multi-hop information propagation across different levels of the graph. However, HGN faces significant limitations that hinder its scalability and generalizability. First, the storage of embeddings across multiple granularities incurs substantial memory and computational overhead, particularly for large datasets. Second, its reliance on explicit relational structures, such as Wikipedia hyperlinks, restricts its applicability to domains where such curated connections are sparse. Our HLG extends HGN's foundational concepts by embedding both fine-grained propositions and higher-level topics within a unified graph structure.

Edge et al. [11] proposed GraphRAG, a graph-indexing method designed for global summarization tasks. By constructing an entity-relation knowledge graph and hierarchically detecting communities on the graph, GraphRAG enables summarization at indexing time. However, its reliance on pre-defined entities and static Leiden-generated communities [22] poses limitations for dynamically expanding domains. Incorporating additional data requires recalculating the community structure for the entire graph, which can be computationally expensive. In contrast, HLG is designed for continuous data ingestion and supports multiple granularity tiers, facilitating more efficient updates and robust retrieval. Rather than relying on pre-computed communities, HLG dynamically retrieves relevant topics at query time, enabling greater scalability and resource efficiency.

2.2 Dense Retrieval Granularity

Chen et al. [7] analyze how the granularity of retrieval units affects dense retrieval. Their findings indicate that fine-grained propositions outperform larger units (e.g., sentences or passages) in both retrieval precision and downstream task performance. While Chen et al. [7] focus on dense retrieval for standard datasets, we extend this principle by integrating hierarchical graph structures to ensure relevance and connectivity across diverse sources. To enhance the adaptability of statement extraction for structured and tabular data (such as SEC filings), an LLM-modified version of the propositionizer (`propositionizer-wiki-flan-t5-large`) was incorporated.

3 Hierarchical Lexical Graph

The design of our RAG solution is driven by a systematic "working backward" approach that, given a workload's question-answering needs, identifies optimal retrieval and generation strategies and appropriate indexing and storage systems in support of those needs. In this design, consideration is given to the types of end-user or application-specific data needs the workflow is intended to support, the data required to meet these needs, and the indexing structures best suited for efficient retrieval. HLG is constructed once per dataset during initial indexing to optimize relevance for the

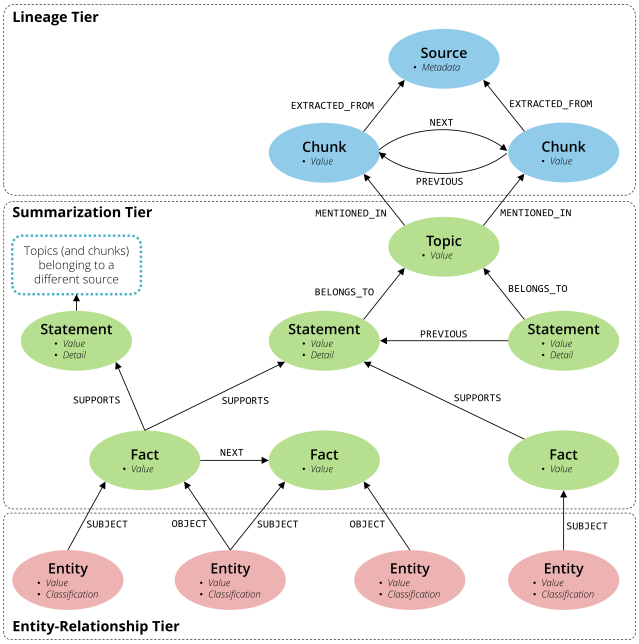

Figure 1: Hierarchical Lexical Graph Model. Statements are atomic propositions *(e.g., "Company X acquired Company Y")*. Topics, thematic groups *(e.g., "Q3 Financial Results")*. Facts, structured S-P-O triplets *(e.g., <Company X, acquired, Company Y>)*. Entities, named concepts *(e.g., "Company X")*. Keywords (not shown but used in retrieval) are salient terms extracted from queries *(e.g., "acquisition", "Company X")*.

target domain. However, HLG supports incremental ingestion of new documents and entity propagation without full reindexing, due to modular updates in its tiers (Section 3.1).

3.1 Statement Retrieval and Graph Structures

Determining the optimal size and structure of retrieval units is critical in search-based workflows. Traditional RAG systems often rely on text "chunks" (segments larger than sentences, smaller than documents). However, such coarse-grained units frequently amalgamate unrelated sentences, reducing precision. In contrast, our system centers on "propositions" (standalone assertions extracted from text) as the primary retrieval unit. This granularity allows for precise, contextually relevant inputs for LLMs.

The retrieval system is built upon a lexical graph model, illustrated in Figure 1, comprising three interconnected tiers that enable fine-grained connectivity across documents.

3.1.1 Lineage Tier. Establishes the foundation of the graph, ensuring traceability and contextual integrity.

- **Source Nodes**: Metadata such as document origin, date, and author information, for provenance.
- **Chunk Nodes**: Sequentially linked text segments, preserving context for further analysis and maintaining lineage.

This tier is useful in compliance-driven scenarios, where maintaining the original context and source of retrieved information is critical for interpretability.

3.1.2 Entity-Relationship Tier. Captures relationships between entities, serves as an entry point for structured, keyword-based searches.

- **Entity Nodes**: Key entities (e.g., "Amazon") classified by category (e.g., "Company").
- **Relationship Edges**: Inter-entity relationships (e.g., "FILED LAWSUIT AGAINST"), for structured retrieval tasks.

This tier supports complex aggregation queries, such as *"Which companies filed lawsuits in Q3 2023?"*, while dynamically incorporating new classifications to enhance adaptability.

3.1.3 Summarization Tier. Links granular facts and statements to broader topics, forming hierarchical semantic units.

- **Facts**: Discrete semantic units (subject-predicate-object triplets) that connect granular and global insights.
- **Statements**: Propositions extracted from source documents, forming the backbone of the retrieval process.
- **Topics**: Thematic summaries grouping related statements, enhancing intra-document connectivity.

By connecting statements to overarching topics, this tier facilitates both local and global reasoning, ensuring efficient retrieval strategies for multi-hop QA tasks.

3.2 Graph Connectivity

HLG optimizes retrieval through a hybrid approach that combines semantic similarity and graph traversal, ensuring flexible and efficient navigation across the graph:

(1) **Local/Global Connectivity**: Topics provide intra-document connectivity by linking statements within a source. In contrast, facts enable inter-document connections, ensuring comprehensive retrieval for complex queries.
(2) **Vector-Based Entry Points**: Both topics and/or statements are embedded for vector-based searches, enabling precise semantic similarity matching. Topic embeddings incorporate associated statements to enhance query alignment.
(3) **Keyword-Based Entry Points**: Query keywords can be matched with entities in the Entity-Relationship Tier, essential for bottom-up lookups.

4 Proposed Retrieval Models

Two methods leverage HLG: StatementGraphRAG (Section 4.1) and TopicGraphRAG (Section 4.2). The choice between them depends on the query's nature. StatementGraphRAG excels at precise, fact-based queries seeking specific information, often with a single expected answer. Its fine-grained statement retrieval and entity linking ensure high precision. TopicGraphRAG is better for broader, exploratory queries or those requiring synthesis across multiple themes. Its topic-level retrieval provides wider context, suitable for open-ended questions or summarization tasks.

4.1 StatementGraphRAG

Overview. The StatementGraphRAG pipeline has four components, designed to progressively refine search results:

(1) *Keyword-Based Retrieval*: Extracts query terms and retrieves statements with those terms via explicit entity matching.
(2) *Vector Similarity Search*: Retrieves statements semantically similar to the query.
(3) *Graph Beam Search*: Explores multi-hop neighbours in HLG by traversing shared entities, and scoring resulting paths.
(4) *Reranking*: Rescores all candidates to finalize top-k results.

Notation. Let $Q \in \mathcal{D}$ be the user query and

$$\mathbf{e}_Q \; := \; \mathrm{Embed}(Q) \in \mathbb{R}^d$$

its d-dimensional embedding. Let $\mathcal{G} = (V, E)$ be the lexical graph and $\mathcal{S}_\mathcal{G} \subseteq V$ the set of statement nodes. Define the set of extracted keywords (and synonyms):

$$K \; = \; \{\, k_i \mid k_i \text{ extracted from } Q\}, \qquad i = 1, \dots, N_k. \tag{1}$$

Step 1: Keyword Retrieval. For $s \in \mathcal{S}_\mathcal{G}$, let $K_s \subseteq K$ be the keywords whose entities appear in s. We rank statements by:

$$\mathrm{Score}_{\mathrm{kw}}(s) \; = \; |K_s|, \tag{2}$$

breaking ties with cosine similarity $\mathrm{sim}(\mathbf{e}_Q, \mathbf{e}_s)$. The k highest-scoring statements form:

$$\mathcal{S}_{\mathrm{kw}} \; = \; \mathrm{Top}_k\big(\mathcal{S}_\mathcal{G}, \, \mathrm{Score}_{\mathrm{kw}}\big). \tag{3}$$

Step 2: Vector Similarity Search. In parallel to Step 1, we retrieve:

$$\mathcal{S}_{\mathrm{vss}} \; = \; \mathrm{Top}_k\big\{\, s \in \mathcal{S}_\mathcal{G} \mid \mathrm{sim}(\mathbf{e}_Q, \mathbf{e}_s)\big\}. \tag{4}$$

The initial candidate set is the union, $\mathcal{S}_{\mathrm{init}} = \mathcal{S}_{\mathrm{kw}} \cup \mathcal{S}_{\mathrm{vss}}$.

Step 3: Graph Beam Search. For a statement s, define its neighbours by shared entities:

$$\mathrm{Nbr}(s) \; = \; \big\{\, s' \in \mathcal{S}_\mathcal{G} \mid \mathrm{Ent}(s) \cap \mathrm{Ent}(s') \neq \varnothing\big\}. \tag{5}$$

Given a path $P = \langle s_1, \dots, s_n \rangle$, we compute an attention-weighted path embedding [4]:

$$\mathbf{e}_{\mathrm{path}}(P) = \sum_{i=1}^{n} \alpha_i \, \mathbf{e}_{s_i}, \qquad \alpha_i = \frac{\exp\big(\mathrm{sim}(\mathbf{e}_Q, \mathbf{e}_{s_i})\big)}{\sum_{j=1}^{n} \exp\big(\mathrm{sim}(\mathbf{e}_Q, \mathbf{e}_{s_j})\big)}. \tag{6}$$

Path relevance is:

$$\mathrm{Score}_{\mathrm{beam}}(P) \; = \; \mathrm{sim}\big(\mathbf{e}_Q, \mathbf{e}_{\mathrm{path}}(P)\big). \tag{7}$$

Starting from each initial state $s \in \mathcal{S}_{\mathrm{init}}$, beam search expands the frontier until either (i) no child improves the score or (ii) the maximum depth $D_{\max}$ is reached. At every depth, it keeps only the B highest-scoring paths, ranked by Eq. (7). This procedure yields:

$$\mathcal{S}_{\mathrm{beam}} \; = \; \bigcup_{s \in \mathcal{S}_{\mathrm{init}}} \textsc{BeamSearch}(s, \mathcal{G}, B, D_{\max}). \tag{8}$$

The candidate pool after graph exploration is: $\mathcal{S}_{\mathrm{final}} = \mathcal{S}_{\mathrm{init}} \cup \mathcal{S}_{\mathrm{beam}}$.

Step 4: Reranking. Each $s \in \mathcal{S}_{\mathrm{final}}$ is rescored by a cross-encoder reranker:

$$\mathrm{Score}_{\mathrm{rank}}(s) \; = \; \mathrm{sim}\big(\mathrm{Rerank}(Q), \mathrm{Rerank}(s)\big). \tag{9}$$

We return the top-k results:

$$\mathcal{S}_{\mathrm{top}} \; = \; \mathrm{Top}_k\big(\mathcal{S}_{\mathrm{final}}, \mathrm{Score}_{\mathrm{rank}}\big). \tag{10}$$

Algorithm 1 StatementGraphRAG

```
Require: Query $Q$, lexical graph $\mathcal{G}$, embedding function $\mathrm{Embed}(\cdot)$,
    reranker Rerank, top-$k$
Ensure: $\mathcal{S}_{\mathrm{top}}$: top-$k$ statements relevant to $Q$
Extract keywords & synonyms $\longrightarrow K$                         (Eq. 1)
Keyword Retrieval: obtain $\mathcal{S}_{\mathrm{kw}}$                   (Eq. 2)
Vector Similarity Search: obtain $\mathcal{S}_{\mathrm{vss}}$
$\mathcal{S}_{\mathrm{init}} \leftarrow \mathcal{S}_{\mathrm{kw}} \cup \mathcal{S}_{\mathrm{vss}}$
Graph Beam Search:
for each $s \in \mathcal{S}_{\mathrm{init}}$ do
    Expand $s$ via $\mathrm{Nbr}(s)$                                    (Eq. 5)
    Compute attention-weighted path embedding                           (Eq. 6)
    Update path score                                                   (Eq. 7)
end for
$\mathcal{S}_{\mathrm{beam}} \leftarrow$ all statements visited during beam search
$\mathcal{S}_{\mathrm{final}} \leftarrow \mathcal{S}_{\mathrm{init}} \cup \mathcal{S}_{\mathrm{beam}}$
Rerank: compute $\mathrm{Score}_{\mathrm{rank}}(s)$ for each $s \in \mathcal{S}_{\mathrm{final}}$
$\mathcal{S}_{\mathrm{top}} \leftarrow \mathrm{Top}_k(\mathcal{S}_{\mathrm{final}}, \mathrm{Score}_{\mathrm{rank}})$
return $\mathcal{S}_{\mathrm{top}}$
```

4.2 TopicGraphRAG

Overview. *TopicGraphRAG* integrates both top-down (topic-driven) and bottom-up (entity-driven) retrieval to identify and expand thematically relevant information through multi-hop reasoning. Topics typically map to a small number of chunks (1:n, where n is small), often from the same document. This makes the approach more storage-efficient than retrieving isolated statements, which tend to have a 10:1 chunk-to-statement ratio. The pipeline consists of four stages, balancing broad thematic coverage with precise detail:

(1) **Top-Down: Topic Discovery.** Embed the query to identify high-level topics aligned with user intent, then retrieve the relevant statements linked to those topics.
(2) **Bottom-Up: Entity Exploration.** Extract query-related keywords and match them with associated entities in the lexical graph. Then, retrieve their associated statements.
(3) **Graph Beam Search.** Topic- and entity-related statements are merged. A beam search explores additional context in multiple hops, guided by a reranker at each step.
(4) **Final Rerank & Truncation.** Rescore all candidate statements using a reranker, and select the top-k relevant ones.

Rationale. Using *topics* (top-down) and *entities* (bottom-up), TopicGraphRAG broadens coverage while preserving precision. Multi-hop traversal explores cross-document links, and the final reranking yields a set of statements well-aligned with the user query.

4.3 Post-Retrieval Processing

After performing retrieval using StatementGraphRAG or TopicGraphRAG, post-processing can enhance diversity and clarity.

4.3.1 Statement Diversity: Reducing Redundancy. In many real-world scenarios, multiple sources may report overlapping information, which can constrain the context window. To address this, a diversity filtering procedure identifies similar statements and preserves the highest-scoring representative. Empirical evaluations

(Table 2) show a diversity threshold (τ) as low as 0.5% can yield modest gains with negligible overhead.

(1) **Preprocessing:** Numeric tokens are standardized, stop words are removed, and text is lemmatized. This ensures consistent representation of numeric and textual data.
(2) **Similarity Detection:** Vectorize statements via TF-IDF [20], and compute cosine similarity to detect redundancies.

$$\mathrm{Sim}(S_i, S_j) = \frac{\sum_{k=1}^{n} \text{TF-IDF}_{S_i,k} \cdot \text{TF-IDF}_{S_j,k}}{\sqrt{\sum_{k=1}^{n} (\text{TF-IDF}_{S_i,k})^2} \cdot \sqrt{\sum_{k=1}^{n} (\text{TF-IDF}_{S_j,k})^2}},$$

where $\text{TF-IDF}_{S_i,k}$ represents the k-th term in statement S_i, and n is the total number of terms.
(3) **Filtering:** Define a *diversity threshold* τ, which corresponds to $1-$(similarity threshold). Discard the lower-scoring statement and retain the more informative version:

$$\text{Retain } S_i \iff \forall S_j : \mathrm{Sim}(S_i, S_j) > (1-\tau) \Rightarrow \mathrm{Score}(S_i) > \mathrm{Score}(S_j)$$

4.3.2 Statement Enhancement: Tabular Data Processing. For financial reports or documents with tabular data, this step enriches statements by incorporating context from their text chunks. This is important for sources like SEC-10Q filings, where numeric propositions may lack clarity without additional details. Relevant contextual information, such as column headers and row labels, are appended to clarify the meaning of the data, improving the readability and interpretability of numeric propositions. This reduces ambiguity (e.g., clarifying units or time periods), ensures completeness, and makes statements more accurate.

5 Synthetic Multi-Hop Summarization

Motivation. Existing multi-hop summarization datasets, such as SEC-10Q or WikiHow, lack the complexity or diversity required to evaluate RAG systems. SEC-10Q has a limited set of 195 comprehensive queries, whereas WikiHow is not multi-hop since the answer is derived from one text segment. To address these shortcomings, we introduce a synthetic dataset generation pipeline that assembles multi-hop queries spanning multiple documents, thereby testing robust retrieval and summarization capabilities.

5.1 Dataset Characteristics

- **Complexity.** Each question needs information from two to four documents ("hops").
- **Realism.** Queries mirror real-world inquiries that span multiple themes, needing deeper reasoning.
- **Structure.** Each entry includes a multi-hop question, a ground truth answer (from multiple chunks) and document snippets. Ground truth answers are synthesized from multiple document snippets and validated for factual accuracy.

5.2 Pipeline Architecture

Our dataset creation process uses a four-stage pipeline:

(1) **Topic Collection.** From a seed topic, we retrieve semantically related topics from different documents.
(2) **Chunk Selection.** Collect chunks from each relevant topic (3-5 distinct articles) for cohesive context.
(3) **Query Generation.** Prompt an LLM with diverse chunks to generate a multi-hop question. These questions require synthesizing information across articles *(e.g., cause-effect, contrasting perspectives, complementary insights).*
(4) **Critiquing and Refinement.** Queries are refined and validated by a second LLM pass to ensure clarity, coherence, and multi-article coverage. Queries that fail validation, for instance due to insufficient evidence, are discarded.

Quality Control and Filtering. In Step 4, we utilize a superior LLM (`Claude-3.5 Sonnet v2` [2]) than the one used for query-generation (`Claude-3 Sonnet` [1]), to align better with human judgement [16]. We incorporate a two-stage critique:

- *Query Refinement:* Each initial query-answer pair is examined to enhance clarity, address ambiguities, and enrich the multi-hop nature (ensuring at least three distinct sources).
- *Query Validation:* Refined queries are paired with their associated document snippets and evaluated against the ground truth. An internal retriever pipeline is used to simulate the reasoning path. A query is accepted only if the system can reconstruct the ground truth from the provided snippets, indicating sufficient evidence coverage and reasoning fidelity.

5.3 Dataset Statistics

Using the MultiHop-RAG Corpus [21], we initially generated 1,173 questions and retained 674 high-quality queries after filtering. On average, each query spans 4.1 chunks and 3.4 documents (4 entities per question and 9 entities per answer).

Sample Synthetic Query.

> *How has Manchester United's on-field performance under Erik ten Hag evolved amid the impending Ratcliffe ownership transition, considering their pressing statistics, Champions League failure, and potential managerial succession plans?*

Ground-truth answer: *"Manchester United's current situation reflects a complex intersection of tactical evolution and organizational transition. While Ten Hag has shown some statistical improvements in certain areas—notably with United ranking at the top of the Premier League in high ball regains and middle-third possession win..."*

This demands the system's ability to integrate performance metrics (pressing and possession statistics), organizational context (ownership transition), and multi-document references (Champions League details, managerial prospects).

6 Experimental Setup

We describe our experimental design, including datasets, indexing procedures, retrieval approaches, and evaluation metrics.

6.1 Datasets

Our experiments span five diverse datasets (Table 1), selected to evaluate multi-hop reasoning, domain-specific language, and tabular/context integration:

MultiHop-RAG [21]. Multi-document QA dataset with queries that require synthesizing evidence from multiple sources, covering inference, temporal reasoning, comparisons, and null cases.

SEC-10Q [15]. Quarterly financial reports (Form 10-Q) from publicly traded companies, filed with the U.S. Securities and Exchange Commission. These reports include financial statements, management discussions, and disclosures of market risks.

ConcurrentQA [3]. A multi-hop QA benchmark requiring reasoning across both public (Wikipedia) and private (emails) data sources. It builds on HotpotQA [23] and evaluates the ability to handle complex cross-domain queries.

NTSB [15]. Corpus of aviation accident and incident reports from the National Transportation Safety Board. Each report details the event date, location, aircraft involved, and probable causal factors.

WikiHowQA [6]. Derived from WikiHow articles, with community-generated questions, along with step-by-step procedures.

6.2 Indexing Steps

All datasets undergo a four-step indexing procedure. Steps 2–3 rely on an LLM (`Claude-3 Sonnet` [1] used here) to relate domain-relevant entities and topics, with fine-grained propositions. We use AWS Neptune Analytics for the graph store and AWS OpenSearch for vector storage. At the time of writing, the framework also supports PostgreSQL with pgVector, FalkorDB, and MosaicML.

1. **Chunking.** Each document is segmented into 300-token chunks, with a 20% overlap to preserve context.
2. **Domain-Adaptive Refinement.** We sample five chunks from each dataset to infer domain-specific entity types and topics (e.g., financial terminology in SEC-10Q). This ensures that HLG reflects relevant domain concepts.
3. **Proposition & Graph Construction.** From each chunk, we extract fine-grained propositions, and then link them with topics, entities, and relationships in HLG.
4. **Embedding Generation.** Embeddings for topics, statements, or both, per retrieval method. We use *Cohere-Embed-English-v3* to compute 1024-dimensional vectors [8].

6.3 Retrieval Approaches

We evaluate three baseline methods and four HLG-based retrieval approaches. For all methods, the context window is fixed at 10 chunks, which corresponds to approximately 3,000 tokens. Following retrieval, (`BAAI/bge-reranker-v2-minicpm-layerwise` [18]) selects the final set of passages (chunks/statements). This reranker, based on a streamlined 2B-LLM, processes queries and retrieved statements jointly, enabling deeper context understanding compared to embedding-based methods that rely on fixed vector representations. All methods, including baselines, use the same LLM (`Claude-3 Sonnet` [1]) and identical prompting for answer generation. We avoid techniques like Chain-of-Thought (CoT) prompting, as they can conflate retrieval effectiveness with reasoning ability. Our focus is strictly on retrieval augmentation. Importantly, `Claude-3 Sonnet`'s training data was frozen in August 2023 [1], predating the MultiHop-RAG articles (Sept-Dec 2023) [21]. Thus, its performance on this dataset cannot be attributed to memorization.

6.3.1 Baselines.

- **Naive RAG** (B0). Retrieves 10 chunks via VSS. VSS uses a dense retriever, applying k-nearest neighbour (kNN) algorithms to rank chunks based on the similarity between the query and chunk embedding [9].
- **Naive RAG (w/ reranking)** (B1). Retrieves 50 chunks initially, applies the reranking model, and prunes the output to retain the top 10 chunks for the final set.
- **Entity-Linking Expansion** (E1). A simpler graph-like baseline. Retrieves 5 chunks via VSS, expands with one-hop entity links from HLG, merges up to 50 nodes (chunks associated with linked entities), then reranks to top 10 chunks. This is similar to the triplet extraction idea in some related works.

6.3.2 Our Approaches.

- **StatementRAG** (SRAG). Performs VSS on individual statements (rather than entire chunks). Retrieve 100 statements, then reranks and truncates the top results.
- **StatementGraphRAG** (SGRAG). Extends SRAG with graph-based expansions. Starting from 100 initial statements, we run a beam search over graph neighbours (width=50, depth=3). The expanded pool is reranked and truncated.
- **TopicRAG** (TRAG). Retrieves 50 topics based on VSS. The associated statements for each topic are retrieved. The statements are then reranked and truncated.
- **TopicGraphRAG** (TGRAG). Combines topic-level retrieval with graph expansions. First, retrieves 50 topics via VSS, expands using beam search (width=50, depth=3), and reranks.

6.4 Chunks as the Generation Unit

In certain domains, compliance regulations mandate the use of original text blocks rather than LLM-generated statements. To address this, we evaluate two HLG methods (SGRAG and TGRAG) in a "chunk-based" variant. This approach leverages the detailed connections between individual statements during graph expansion, while ensuring the generation prompt is composed solely of original text blocks. In the final step, the top-ranked statements are traced back to their source chunks. We select up to 10 unique chunks and apply a diversity filter to remove near-duplicates, further promoting chunk-level diversity in the final output.

6.5 Evaluation Metrics

Correctness is used for single-answer datasets (MultiHop-RAG, SEC-10Q, ConcurrentQA) and **Answer Recall** for multi-answer datasets (NTSB, WikiHowQA). *Correctness* assesses if the generated answer semantically contains all necessary information from the gold answer, even if phrasing differs. For single-answer datasets, correctness is akin to accuracy, requiring a semantically correct response (e.g., a "true" response may be conceptually correct for a "yes" ground truth, even if not an exact string match). This is preferred over exact match, which can be overly rigid for free-form, multi-part responses. *Answer Recall* measures the proportion of gold answer facts correctly included in the generated answer. For instance, listing 4 of 5 steps scores 0% on strict correctness but 80% on recall. This captures partial coverage. For the synthetic dataset, we evaluate the retrievers using claim recall (CR) and context precision (CP) from RAGChecker [19].

Table 1: Overview of the datasets used for evaluation.

Dataset	Domain	# Queries	# Docs	Sample Query
MultiHop-RAG	News Articles	2,556	609	Who is the individual associated with OpenAI, recognized for both his vision of AI agents and his generosity and has made headlines in both Fortune and TechCrunch for his controversial departure?
SEC-10Q	Finance	195	20	How has Apple's revenue from iPhone sales fluctuated across quarters?
NTSB	Aviation	197	20	Which operators have been involved in fatal accidents with amateur-built aircraft?
WikiHowQA	General	300	5,000	How to grow orchids in a greenhouse?
ConcurrentQA	Email	400	13,500	Which OpenTable.com investor also invested in Acta?

7 Results

We present retrieval evaluations across multiple datasets and metrics. We compare (a) chunk- vs. statement-level retrieval, (b) correctness vs. recall, and (c) pairwise comparisons against baselines.

7.1 Overall Retrieval Performance

Table 2 reports results on three datasets with a single gold answer (MultiHop-RAG, SEC-10Q, and ConcurrentQA) and two datasets allowing multiple valid answers (NTSB, WikiHowQA).

Discussion. Among the baselines, B1 achieves 66.1% correctness averaged over the three single-answer datasets, surpassing B0 by 4.1 absolute percentage points. E1, which performs simple entity linking, shows minimal improvement over B1 on average (65.6% correctness) and underperforms on SEC-10Q, suggesting that naive entity expansion alone is not consistently beneficial. In contrast, every HLG-based method (SRAG, SGRAG, TRAG, TGRAG, and their variants) outperforms B1 in at least one dimension (correctness or recall). SGRAG-0.5% attains the highest average correctness (73.6%), an overall improvement of 7.5 points over B1. This method also improves average answer recall (52.4%), up from 50.8% with B1.

On multi-answer tasks (NTSB and WikiHowQA), TGRAG yields the best average recall (53.8%), outperforming SGRAG-0.5% by 1.4 absolute points. Meanwhile, statement-based methods such as SRAG and SGRAG demonstrate consistently higher correctness on single-answer datasets, with SRAG reaching 87.0% on MultiHop-RAG and SGRAG-0.5% yielding 74.4% on SEC-10Q. These results suggest that statement-level retrieval is adept at pinpointing a single correct piece of evidence, whereas topic-based retrieval may capture a broader set of relevant facts. The WikiHowQA dataset predominantly comprises single-hop, procedural questions (further discussed in Appendix C.1.2). In such cases, multi-hop graph expansions (as in SGRAG/TGRAG) can introduce noise or irrelevant hops, potentially degrading performance compared to simpler methods like B1, which performs best on this dataset. While HLG-based methods underperform slightly on WikiHowQA, they consistently outperform baselines on all true multi-hop datasets (MultiHop-RAG, ConcurrentQA). This dataset-specific behaviour shows that HLG is optimized for multi-hop retrieval, and single-hop cases may benefit from reduced complexity or early-stopping heuristics.

7.2 HLG Chunk-Based Variants

Because some applications require preserving the original text at the chunk level, we also evaluate Chunk-SGRAG and Chunk-TGRAG. These methods internally retrieve statements using graph traversal, but for the final output, they map the selected top statements back to their parent chunks. Table 3 shows how these chunk-level variants compare to the baselines.

Discussion. Relative to naive chunk-only retrieval (B0 or B1), the hybrid chunk-based variants consistently yield higher correctness. On ConcurrentQA, for example, Chunk-SGRAG-0.5% reaches 67.0% correctness, a 23.7-point increase over B0 and 16.0 points over B1. Despite initially retrieving content at the finer statement granularity, the final output remains chunked. This suggests that graph-based expansions are beneficial even if the generation context window must be text segments (chunks).

7.3 Synthetic Dataset Evaluation

7.3.1 RAGChecker Evaluation. We further investigated retrieval quality on our synthetic MultiHop-RAG subset using RAGChecker [19], which provides fine-grained retriever metrics such as claim recall (CR) and context precision (CP). Table 4 reports these metrics for the baselines (B0, B1) and our primary HLG-based methods (SGRAG, TGRAG). Generator metrics are omitted because all systems use the same LLM for answer generation.

Discussion. Although the baselines (B0, B1) show slightly higher context precision (78.8%-81.5%), both SGRAG and TGRAG surpass them in recall and F1. In particular, TGRAG achieves 49.3% recall and 47.9% F1, exceeding B1 by 3.0 and 1.6 absolute points, respectively. The stronger claim recall (67.6% vs. 62.2% for B1) also showcases TGRAG's multi-hop capabilities, as it is more likely to gather all relevant facts for each query.

7.3.2 Pairwise Evaluation: Win Rates. We also conducted a head-to-head comparison of SGRAG and TGRAG against the baselines. In each comparison, an LLM evaluator was shown the two retrieved answers in random order to mitigate any position bias. Table 5 presents the percentage of times each method won, lost, or tied.

Discussion. Both SGRAG and TGRAG significantly outperform the chunk-based baselines, winning over 74% of head-to-head comparisons. TGRAG achieves a slightly higher win rate against B0 (78.3%) compared to SGRAG (74.8%), reflecting its stronger coverage of multiple answers. These consistent pairwise improvements highlight that graph-based retrieval methods (whether statement- or topic-centered) tend to produce more relevant and higher-quality evidence than VSS retrieval.

Table 2: Correctness and answer recall results. Bold = highest in column; underlined = second highest.

	Correctness			Correctness Average	Answer Recall		Answer Recall Average
	MultiHop-RAG	SEC-10Q	ConcurrentQA		NTSB	WikiHowQA	
B0 (Naive RAG)	78.7%	64.1%	43.3%	62.0%	24.2%	69.6%	46.9%
B1 (Naive RAG + rerank)	82.8%	64.6%	51.0%	66.1%	31.1%	**70.5%**	50.8%
E1 (Entity-Linking RAG)	82.6%	61.5%	52.8%	65.6%	31.2%	70.0%	50.6%
SRAG	**87.0%**	69.2%	51.3%	69.2%	33.4%	65.0%	49.2%
SGRAG	86.9%	73.9%	57.3%	72.7%	34.8%	67.1%	50.9%
SGRAG-0.5%	86.9%	**74.4%**	59.5%	**73.6%**	36.1%	68.7%	52.4%
TRAG	84.2%	70.3%	53.8%	69.4%	35.7%	68.3%	52.0%
TGRAG	84.5%	72.3%	**59.8%**	72.2%	**39.3%**	68.2%	**53.8%**

Table 3: Chunk-based variants of StatementGraphRAG and TopicGraphRAG.

	Correctness			Correctness Average	Answer Recall		Answer Recall Average
	MultiHop-RAG	SEC-10Q	ConcurrentQA		NTSB	WikiHowQA	
Chunk-SGRAG	86.1%	65.6%	66.8%	72.8%	25.8%	67.2%	46.5%
Chunk-SGRAG-0.5%	86.1%	66.2%	67.0%	73.1%	25.8%	67.7%	46.7%
Chunk-TGRAG	86.3%	72.8%	61.8%	73.6%	27.7%	67.0%	47.3%
Chunk-TGRAG-0.5%	86.0%	73.3%	64.3%	74.5%	26.7%	67.5%	47.1%

Table 4: RAGChecker metrics on a synthetic MultiHop-RAG subset. Retriever is assessed using claim recall (CR) and context precision (CP).

	Overall			Retriever	
	Precision (↑)	Recall (↑)	F1 (↑)	CR (↑)	CP (↑)
B0	**51.2**	46.0	46.1	58.3	78.8
B1	51.1	46.3	46.3	62.2	**81.5**
SGRAG	50.8	48.6	47.5	59.5	50.9
TGRAG	50.8	**49.3**	**47.9**	**67.6**	66.1

Table 5: Pairwise comparisons of StatementGraphRAG and TopicGraphRAG vs. chunk-based baselines (B0 and B1).

	Baseline	Win (%)	Loss (%)	Tie (%)	Win/Loss Ratio
SGRAG	B0	74.8	21.5	3.7	3.5
	B1	74.8	22.9	2.8	3.3
TGRAG	B0	78.3	17.4	4.3	4.5
	B1	76.3	18.4	5.3	4.2

7.4 Summary of Findings

For single-answer tasks, statement-based retrieval methods (SRAG, SGRAG) consistently achieve higher correctness, averaging 72.7%-73.6%, compared to 62.0%-66.1% for naive baselines. In contrast, topic-based approaches (TRAG, TGRAG) perform best on multi-answer tasks. Graph expansions yield significant improvements: SGRAG-0.5% exceeds the naive baseline (B0) by 8-12 points in correctness across single-answer datasets, while TGRAG improves answer recall by more than 10 points over B0 in multi-answer settings. Introducing a modest diversity threshold (0.5%) further enhances coverage by filtering near-duplicate statements, resulting in an additional 1-point gain in correctness. Even when chunk-based outputs are used, leveraging statement-level graph traversals internally enables retrieval of more relevant content, demonstrating that fine-grained graph expansions enhance overall chunk selection.

8 Conclusion

We presented a Hierarchical Lexical Graph framework that supports fine-grained, multi-hop retrieval for QA tasks. Our experiments demonstrated that statement-level and topic-level retrieval consistently outperform baselines across correctness, answer recall, and pairwise judgements, with TopicGraphRAG especially strong in multi-hop scenarios demanding broader coverage. The graph expansion strategies capture inter-document relationships more effectively, while a lightweight reranking model balances performance with reduced latency. Despite higher indexing costs and domain constraints, experimental results, including strong statement-to-source alignment, show viability for structured and unstructured data. Future work will focus on refining proposition extraction with smaller models, incorporating adaptive multi-hop detection, and expanding domain specialization, thereby advancing the accuracy and efficiency of multi-hop RAG systems in real-world applications.

References

[1] Anthropic. 2024. Claude 3 Model Card. https://www.anthropic.com/model_cards/claude_3.pdf

[2] Anthropic. 2024. Model Card Addendum: Claude 3.5 Haiku and Upgraded Claude 3.5 Sonnet. https://assets.anthropic.com/m/1cd9d098ac3e6467/original/Claude-3-Model-Card-October-Addendum.pdf

[3] Simran Arora, Patrick Lewis, Angela Fan, Jacob Kahn, and Christopher Ré. 2023. Reasoning over Public and Private Data in Retrieval-Based Systems. *Transactions of the Association for Computational Linguistics* 11 (2023), 902–921. https://doi.org/10.1162/tacl_a_00580

[4] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2016. Neural Machine Translation by Jointly Learning to Align and Translate. arXiv:1409.0473 [cs.CL] https://arxiv.org/abs/1409.0473

[5] Maciej Besta, Ales Kubicek, Roman Niggli, Robert Gerstenberger, Lucas Weitzendorf, Mingyuan Chi, Patrick Iff, Joanna Gajda, Piotr Nyczyk, Jürgen Müller, Hubert Niewiadomski, Marcin Chrapek, Michał Podstawski, and Torsten Hoefler. 2024. Multi-Head RAG: Solving Multi-Aspect Problems with LLMs. arXiv:2406.05085 [cs.CL] https://arxiv.org/abs/2406.05085

[6] Valeriia Bolotova-Baranova, Vladislav Blinov, Sofya Filippova, Falk Scholer, and Mark Sanderson. 2023. WikiHowQA: A Comprehensive Benchmark for Multi-Document Non-Factoid Question Answering. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki (Eds.). Association for Computational Linguistics, Toronto, Canada, 5291–5314. https://doi.org/10.18653/v1/2023.acl-long.290

[7] Tong Chen, Hongwei Wang, Sihao Chen, Wenhao Yu, Kaixin Ma, Xinran Zhao, Hongming Zhang, and Dong Yu. 2024. Dense X Retrieval: What Retrieval Granularity Should We Use?. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (Eds.). Association for Computational Linguistics, Miami, Florida, USA, 15159–15177. https://doi.org/10.18653/v1/2024.emnlp-main.845

[8] Cohere. 2023. Introducing Embed v3. https://cohere.com/blog/introducing-embed-v3. Accessed: 2025-05-27.

[9] T. Cover and P. Hart. 1967. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory* 13, 1 (1967), 21–27. https://doi.org/10.1109/TIT.1967.1053964

[10] Nicola De Cao, Wilker Aziz, and Ivan Titov. 2019. Question Answering by Reasoning Across Documents with Graph Convolutional Networks. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, Jill Burstein, Christy Doran, and Thamar Solorio (Eds.). Association for Computational Linguistics, Minneapolis, Minnesota, 2306–2317. https://doi.org/10.18653/v1/N19-1240

[11] Darren Edge, Ha Trinh, Newman Cheng, Joshua Bradley, Alex Chao, Apurva Mody, Steven Truitt, and Jonathan Larson. 2024. From Local to Global: A Graph RAG Approach to Query-Focused Summarization. arXiv:2404.16130 [cs.CL] https://arxiv.org/abs/2404.16130

[12] Yuwei Fang, Siqi Sun, Zhe Gan, Rohit Pillai, Shuohang Wang, and Jingjing Liu. 2020. Hierarchical Graph Network for Multi-hop Question Answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Bonnie Webber, Trevor Cohn, Yulan He, and Yang Liu (Eds.). Association for Computational Linguistics, Online, 8823–8838. https://doi.org/10.18653/v1/2020.emnlp-main.710

[13] Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, Qianyu Guo, Meng Wang, and Haofen Wang. 2023. Retrieval-Augmented Generation for Large Language Models: A Survey. *CoRR* abs/2312.10997 (2023). https://doi.org/10.48550/ARXIV.2312.10997 arXiv:2312.10997

[14] Yunjie He, Philip John Gorinski, Ieva Staliunaite, and Pontus Stenetorp. 2023. Graph Attention with Hierarchies for Multi-hop Question Answering. arXiv:2301.11792 [cs.CL] https://arxiv.org/abs/2301.11792

[15] Taqi Jaffri. 2023. Announcing Docugami Knowledge Graph Retrieval Augmented Generation (KG-RAG) Datasets in the LlamaHub. https://www.docugami.com/blog/kg-rag-datasets-llama-index

[16] Seungone Kim, Juyoung Suk, Shayne Longpre, Bill Yuchen Lin, Jamin Shin, Sean Welleck, Graham Neubig, Moontae Lee, Kyungjae Lee, and Minjoon Seo. 2024. Prometheus 2: An Open Source Language Model Specialized in Evaluating Other Language Models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (Eds.). Association for Computational Linguistics, Miami, Florida, USA, 4334–4353. https://doi.org/10.18653/v1/2024.emnlp-main.248

[17] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Proceedings of the 34th International Conference on Neural Information Processing Systems* (Vancouver, BC, Canada) *(NIPS '20)*. Curran Associates Inc., Red Hook, NY, USA, Article 793, 16 pages.

[18] Chaofan Li, Zheng Liu, Shitao Xiao, and Yingxia Shao. 2023. Making Large Language Models A Better Foundation For Dense Retrieval. *CoRR* abs/2312.15503 (2023). https://doi.org/10.48550/arXiv.2312.15503

[19] Dongyu Ru, Lin Qiu, Xiangkun Hu, Tianhang Zhang, Peng Shi, Shuaichen Chang, Cheng Jiayang, Cunxiang Wang, Shichao Sun, Huanyu Li, Zizhao Zhang, Binjie Wang, Jiarong Jiang, Tong He, Zhiguo Wang, Pengfei Liu, Yue Zhang, and Zheng Zhang. 2024. RAGChecker: A Fine-grained Framework for Diagnosing Retrieval-Augmented Generation. In *Advances in Neural Information Processing Systems*, A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (Eds.), Vol. 37. Curran Associates, Inc., 21999–22027. https://proceedings.neurips.cc/paper_files/paper/2024/file/27245589131d17368cccdfa990cbf16e-Paper-Datasets_and_Benchmarks_Track.pdf

[20] K. Sparck Jones. 1972. A Statistical Interpretation of Term Specificity and Its Application in Retrieval. *Journal of Documentation* 28, 1 (1972), 11–21. https://doi.org/10.1108/eb026526

[21] Yixuan Tang and Yi Yang. 2024. MultiHop-RAG: Benchmarking Retrieval-Augmented Generation for Multi-Hop Queries. *CoRR* abs/2401.15391 (2024). https://doi.org/10.48550/ARXIV.2401.15391 arXiv:2401.15391

[22] V. A. Traag, L. Waltman, and N. J. van Eck. 2019. From Louvain to Leiden: guaranteeing well-connected communities. *Scientific Reports* 9, 1 (March 2019). https://doi.org/10.1038/s41598-019-41695-z

[23] Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018. HotpotQA: A Dataset for Diverse, Explainable Multi-hop Question Answering. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

A Validation of Generated Statements

We evaluated statement extraction faithfulness by sampling 1,000 statements per dataset and comparing each one to the original text block. Table 6 reports the proportion of accurate statements for general and tabular domains. A statement is considered accurate if it preserves the intended meaning of the original chunk without introducing errors such as hallucinations.

Table 6: Alignment of generated statements against the original chunked text.

Domain	Dataset	Statement Accuracy
General	MultiHop-RAG	96.5%
	ConcurrentQA	97.4%
	WikiHowQA	97.9%
	Average	*97.3%*
Tabular	SEC-10Q	94.3%
	NTSB	97.7%
	Average	*96.0%*

Statement generation attains $\geq$ 96% accuracy, slipping only slightly on table-centric sources. This confirms that our LLM-proposition extractor still delivers high precision. SEC-10Q tables pose extra hurdles, such as scattered units/periods and repetitive layouts that split related details and blur entity links. The tabular enhancement step (Section 4.3.2) restores context by attaching headers, units, and other metadata to each extracted proposition.

B HLG Structure Analysis

Table 7 provides an overview of the node distribution within HLG across multiple datasets.

B.1 Scalability and Performance

B.1.1 Indexing Performance (MultiHop-RAG dataset): The dataset can be indexed and ingested in under 1 hour using AWS Neptune

Table 7: Number of nodes in each tier of HLG for each dataset.

Dataset	Source	Chunk	Topic	Statement	Fact	Entity
MultiHop-RAG	609	6,272	7,067	53,927	274,890	18,291
SEC-10Q	20	5,054	5,219	47,284	57,443	6,903
NTSB	20	6,574	7,633	86,837	50,462	8,867
WikiHowQA	5000	15,255	15,455	163,456	174,262	36,997
ConcurrentQA	13,500	55,328	57,213	495,835	558,897	106,859

Analytics for graph storage and AWS OpenSearch for vector storage. Token consumption for LLM-based extraction (`Claude-3 Sonnet`):

Table 8: Token statistics during indexing

Stage	Input Tokens	Output Tokens
Statement Extraction	3,098,745	1,827,175
Topics/Entities/Relationships	10,812,763	5,080,192
Total	13,911,508	6,907,367

Under AWS Bedrock prices (December 2024), end-to-end indexing with `Claude-3 Sonnet` for LLM inference and `Cohere Embed-English-v3` for embeddings costs $145.34. Running the same workload in batch mode halves the LLM expenditure to $72.67 without affecting quality, and the embedding stage adds only $0.22. Since boilerplate text (e.g., legal disclaimers) recurs across articles, enabling caching lowers token counts and total spend even further.

B.1.2 Retrieval Performance. On a standard AWS `g5.xlarge` instance, an uncached query, covering passage retrieval, embedding creation, and LLM call(s), completes in 20–30 s and costs $0.032. Once the query's vectors and LLM outputs are cached, the same request is answered in well under one second with no additional charge, meeting real-time requirements while keeping operating costs minimal. Future work includes methods to recognize single-hop queries to avoid unnecessary multi-hop expansions, further reducing costs and latency (Section C).

C Limitations and Future Work

C.1 Limitations

C.1.1 LLM Invocation and Indexing Costs. HLG delivers high-fidelity extractions, but at the cost of multiple LLM calls. Each chunk is processed twice; first to extract statements, then to refine topics and entities, so corpora with hundreds of millions of tokens accumulate substantial compute. These costs arise from building a fine-grained knowledge graph that explicitly encodes topics, entities, and relations. Expenses can be reduced by delegating the statement extraction step to a lighter model such as *wiki-flan-t5-large* (783 M parameters) with little loss in retrieval quality. Because extraction is performed once offline during index construction, online queries add no further LLM latency.

C.1.2 Single-Hop Domains and Early Stopping. Our evaluation of WikiHowQA highlights a minor regression for multi-hop graph methods in single-hop contexts. When the query addresses a single chunk of text, graph expansions can introduce extraneous statements or prolong retrieval. An early-stopping heuristic could detect queries dominated by a single source document and return top-k statements directly from that source, improving efficiency and reducing noise.

C.1.3 LLM Beam Search vs. Lightweight Reranking. An early alternative to our BGE-based reranker was an LLM-driven beam search, where each expansion step invoked a language model to rank expansions. This approach showed strong semantic alignment but was prohibitively slow for practical use. Swapping to a lighter reranker maintained strong performance gains while lowering latency, making graph-based methods viable in production environments.

C.2 Error Analysis

We observe several recurring error modes:

C.2.1 Over-Expansion. Even with beam size limited to 3 expansions, queries with highly connected entities (i.e., supernodes) can cause the pipeline to retrieve loosely related statements. Our entity-overlap ranking mitigates this by surfacing neighbours that share more than one entity, but some out-of-scope expansions remain.

C.2.2 Numeric/Tables Misalignment. In SEC-10Q, numeric values sometimes appear in multiple contexts. If two statements are semantically similar but reference different quarters, the top-k filtering can inadvertently retrieve the wrong time period. This is especially important for statement-level tabular enhancements (Section 4.3.2).

C.2.3 Duplicate or Near-Duplicate Statements. The same or nearly identical facts can appear across different documents or versions (especially in historical filings). Without diversity filtering, these duplicates can dominate the top-k and reduce coverage. Our 0.5% threshold helps alleviate this but does not fully eliminate overlapping data for heavily repeated claims.

C.3 Future Work

C.3.1 Propositionizer Pipelines. Subsequent research will prioritize the development of more efficient propositionization, reducing indexing tokens and LLM calls. Aligning the propositionizer's output with our knowledge-graph schema (topics/entities/relationships) remains key to maintaining high-quality links.

C.3.2 Hybrid Retrieval of Chunks and Statements. We plan to explore a layered pipeline that first employs chunk-level searches to localize relevant segments, followed by statement-level expansions or reranking for multi-hop clarity. Such a hybrid approach might preserve the speed of chunk-based retrieval while leveraging more precise graph-based statements where necessary.

C.3.3 Domain Specialization and Fine-Tuning. Enhanced domain adaptation, particularly within finance, healthcare, and legal corpora, may sharpen the extraction of domain-specific topics, entities, and relations. Targeted fine-tuning of the reranker and propositionizer components in these specialized settings is expected to improve precision while keeping inference costs manageable. Although we deliberately avoided highly specialized models in this work to ensure broad applicability with general-purpose LLMs, we are internally testing fine-tuned approaches for structured numeric data (e.g., unit conversion tools, temporal linking improvements). These are ongoing efforts to refine proposition extraction in complex, table-based contexts like legal or financial search.