

Retrieval-Augmented Generation with Retrieval-optimized TRA Structure

JungHyouon Kim¹, SooHyeong Kim², SeokJun Hwang²,
JeongHyeon Park³, Yong Suk Choi^{3*}

¹Department of Intelligence and Convergence, Hanyang University, Seoul, Korea

²Department of Artificial Intelligence, Hanyang University, Seoul, Korea

³Department of Computer Science, Hanyang University, Seoul, Korea

{kkksk1157, ksh970404, cody628, shshjhjh4455, cys}@hanyang.ac.kr

Abstract

Retrieval-Augmented Generation (RAG) has established itself as an innovative technology enabling Large Language Models (LLMs) to answer difficult questions beyond their internal knowledge. Graph-based RAG, which utilizes a triplet-based graph as a database, offers the advantage of structuring relationships between entities through graph connectivity, thereby capturing connections within the corpus that traditional RAG cannot express. However, this paper points out that the process of converting a corpus into a triplet-based graph using LLMs may be unsuitable for RAG due to the inability to represent hierarchical information and structural rigidity. To address this, we propose a novel database structure called the **TRA(Title-Relations-Attributes) structure**. Through experiments on Simple QA and Multi-Hop QA benchmarks, we show that indexing the TRA Structure as Facts provides an effective retrieval-optimized search space. We further demonstrate strong multi-hop reasoning performance when TRA-based retrieval is combined with Query Decomposition (QD), while providing key insights into multi-hop reasoning. The code is available at <https://github.com/Tiabet/TRA-RAG>.

1 Introduction

Retrieval-Augmented Generation (RAG)(Gao et al., 2023; Gupta et al., 2024) is a technique that retrieves relevant information from external knowledge bases to ground the generation process, thereby addressing the limitations of Large Language Models (LLMs)(Hurst et al., 2024; Yang et al., 2025), such as the lack of up-to-date information and the tendency for hallucination(Zhao et al., 2023; Minaee et al., 2024). Recently, RAG has evolved by integrating various techniques; these advancements include generating metadata for filtering and searching(Dadopoulos et al., 2025; Mishra

*Corresponding author.

Corpus

The Wolf of Wall Street is a 2013 American biographical black comedy crime film ...
It recounts Belfort's perspective on his career as a stockbroker ...
Leonardo DiCaprio (who was also a producer) stars as Belfort ...

Triplet

(The Wolf of Wall Street, is a, 2013 American biographical black comedy crime film)
(The Wolf of Wall Street, release year, 2013)
...
(Jordan Belfort, occupation, Stockbroker) → **Contextual Loss**
(Stratton Oakmont, located in, New York City)
...
(Leonardo DiCaprio, plays role, Jordan Belfort) → **Structural Rigidity**
(Leonardo DiCaprio, stars in, The Wolf of Wall Streets)

Search Space for TRA-RAG

"Title": "The Wolf of Wall Street"
"Attributes": {"description": "A 2013 American biographical black comedy crime film",
"plot summary": "It recounts Belfort's perspective on his career as a stockbroker ...",
"cast": [{"actor": "Leonardo DiCaprio",
"role": "Jordan Belfort",
"is_producer": true}, ...]
"Relations": [{"relation": "directed by", "target": "Martin Scorsese"]}

Figure 1: Simple example TRA Structure compared with Triplets.

et al., 2025), or decomposing a single query into multiple sub-queries to retrieve comprehensive information(Lin et al., 2023; Jagerman et al., 2023).

Graph-based RAG(Peng et al., 2024; Han et al., 2026) shifts the retrieval source of RAG from a text corpus to a graph database. Instead of using pre-built Knowledge Graphs (KGs), active research is being conducted on using LLMs to construct KGs directly from a large corpus(Fu et al., 2019) and converting them into a graph database(Edge et al., 2024; Guo et al., 2025b). The advantage of a graph database lies in its ability to discover information through graph connectivity that cannot be found via semantic or keyword matching within the corpus(Han et al., 2024; Zhang et al., 2025). The graph database, with its triplet (Subject, Relation, Object) structure, is recognized as an effective database system(Guo et al., 2025a; Zhu et al., 2025b).

However, graph-based RAG faces a fundamental question: is a graph representation—especially triplets—an optimal information structure for retrieval? For example, even a simple sentence such as "An actor played a specific role in a movie" may be converted into triplets like (Actor, played,

Role) and (Actor, appeared in, Movie). In this example, the semantic connection between the Role and the Movie is fragmented across separate triplets. Many graph-based RAG methods rely on dense-embedding similarity for retrieval(Guo et al., 2025b), yet comparatively fewer studies focus on designing an effective retrieval target (i.e., search space), even though it strongly influences final RAG performance. As a result, triplet-based graph RAG is often structurally rigid and less effective for fine-grained retrieval. These issues can also arise in naive(Gao et al., 2023) and summarization-based(Edge et al., 2024; Sarthi et al., 2024) approaches. When documents or chunks are used directly as the search space, each retrieval unit may become too coarse, making precise retrieval difficult and introducing unnecessary noise. In summarization-based indexing, fine-grained details required by the query may be lost during compression.

Our hypothesis starts from this point: if the search space is explicitly optimized for retrieval, strong performance may be achievable even with relatively simple methods.

Therefore, we aim to overcome the limitations of structure-augmented RAG by reconstructing the corpus into a new data structure: **TRA Structure (Title-Relations-Attributes)**, inspired by metadata and Entity-Relationship (ER) models(Chen, 1976). In particular, ER models motivate us to organize attributes and relations around their central entity rather than representing every association as an independent triplet. Because TRA is represented in a hierarchical JSON format, it supports flexible organization and naturally captures nested information. As a result, the generated TRA Structure can preserve richer contextual information than the triplet format.

Specifically, we propose **TRA-RAG**, a specialized RAG framework that improves retrieval by leveraging a retrieval-optimized search space from TRA Structure, which we refer to as Facts in this paper. TRA-RAG consists of the following three stages: 1) TRA Structure Construction & Indexing on structured Facts, 2) Retrieval based on Reciprocal Rank Fusion (RRF), with Query Decomposition (QD) applied to multi-hop queries, and 3) Reranking subquestion retrieval results for multi-hop question answering. For Simple QA, TRA-RAG performs direct RRF-based retrieval without QD or reranking.

Through extensive experiments, we demonstrate

the superiority of TRA-RAG over a set of powerful structure-augmented RAG methods and multi-step retrieval techniques, achieving the best performance on all reported QA metrics except F1 on LV-Eval, where it ranks second. Furthermore, our ablation studies and analyses provide critical insights into retrieval-structure quality and the essential factors for successful multi-hop reasoning in RAG systems.

2 Related Works

2.1 RAG with Multi-Step Retrieval

RAG alone cannot guarantee retrieval success for questions requiring complex information from diverse fields, such as multi-hop questions(Mavi et al., 2024; Singh et al., 2025). To address this, two main alternatives have been proposed. The first is multi-step retrieval techniques, such as Shao et al. (2023), Press et al. (2023), and Asai et al. (2024), which iteratively retrieve necessary information. These methodologies demonstrate great strength when combined with Chain-of-Thought(Wei et al., 2022; Chen et al., 2025b). The second is QD(Lin et al., 2023; Zhang et al., 2024), which preemptively breaks down a multi-hop question into single-hop questions to specify what needs to be retrieved in advance. More recent work applies this idea in a more efficient manner(Zhuang et al., 2024). In contrast, we intentionally adopt a simple decomposition strategy to focus on the impact of the TRA Structure.

2.2 Data Structure for RAG

Initially, RAG primarily relied on simple embedding vector similarity search(Karpukhin et al., 2020) or keyword matching-based search such as TF-IDF(Sparck Jones, 1988) and BM25(Robertson and Zaragoza, 2009). However, these search methods had the limitation of being unable to leverage the structural advantages of information(Sharma, 2025). To overcome this, various database structures have been proposed(Jiang et al., 2025a; Li et al., 2024), with the triplet-based graph being the most representative. Moving beyond methodologies(Sun et al., 2023) that search pre-built KGs like Wikidata(Vrandečić and Krötzsch, 2014) and Freebase(Bollacker et al., 2008), approaches have been proposed to extract triplets directly from the corpus to construct graphs(Bian, 2025; Edge et al., 2024). Furthermore, to handle diverse and complex queries such as multi-hop questions,

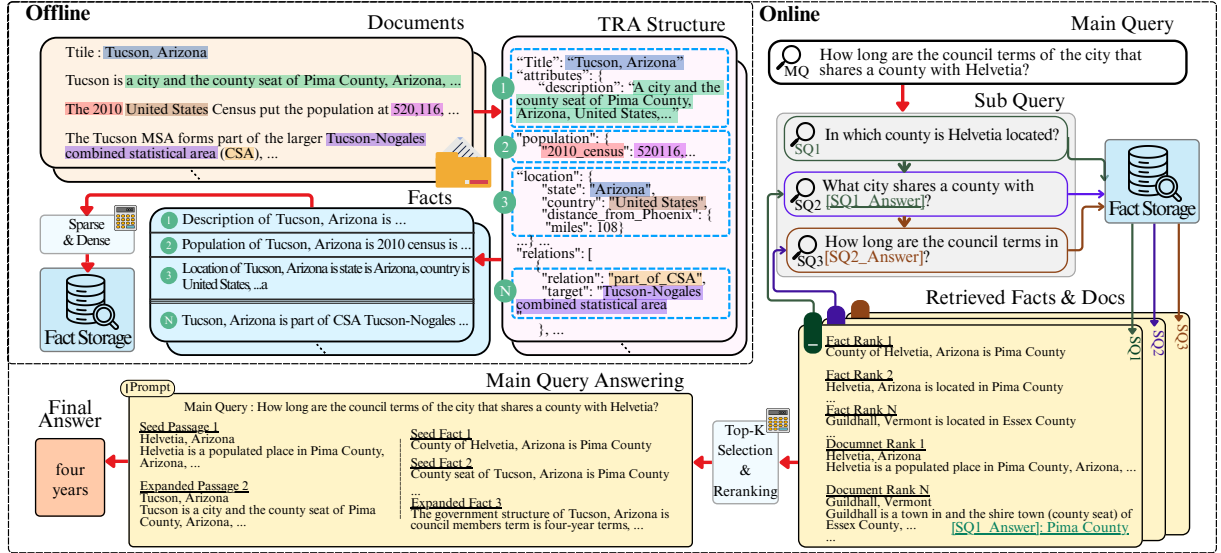


Figure 2: Overview of the TRA-RAG framework. TRA-RAG first reconstructs documents into TRA Structures and indexes Facts into Fact Storage using both dense and sparse indexing. For Multi-Hop QA, an input query is decomposed before relevant Facts and documents are retrieved using RRF-based scoring and reranked for final answering. For Simple QA, retrieval is performed directly without QD or reranking.

numerous methodologies have been introduced, including approaches that use LLMs for graph-hop traversal(Chen et al., 2025a), apply cluster-based summarization(Edge et al., 2024), and employ graph-traversal algorithms over graph structures(Zhu et al., 2025a). However, triplets inherently oversimplify the structure of text, leading to the inability to represent hierarchical information and issues of structural rigidity(Jiang et al., 2025b; Zhou et al., 2025). To replace this, recent studies are exploring various structures such as Luo et al. (2025); Sharma et al. (2025); Zhao et al. (2025); Dong et al. (2025). Among the most effective approaches are methods that insert corpus-level information, such as passages and sentences, into triplet-based graphs (Gutiérrez et al., 2024; Gutiérrez et al., 2025; Zhuang et al., 2025). Sarthi et al. (2024) builds hierarchical tree not using triplet, but makes summarization for retrieval. However, even these approaches do not fully optimize the retrieval targets themselves, as they still suffer from information loss and structural rigidity—the core challenge that our method is designed to address.

3 Method

3.1 TRA Structure Construction

Most graph-based RAG methods begin by using LLMs to convert corpus information into triplets, treating the resulting triplet set as the primary retrieval space and retrieving relevant information based on query–triplet similarity.

However, this approach possesses a critical limitation: not all information in a corpus can be represented in a structure as simple as a triplet. Triplets fundamentally express information in the form of (Subject, Relation, Object). Consider the scenario of structuring the following information into triplets: "Actor A played role C in movie B. This was his first villain role." Without specific instructions, an LLM would typically structure this as (A, played, C), (A, appeared in, B), and (A, played, first villain). In the first and third triplets, information regarding "in which movie" the actor performed the role is missing. This weakens the semantic connection with the second triplet, making it difficult to claim that the text of the corpus has been perfectly encapsulated. Consequently, many graph-based RAGs employ additional methods, which can introduce unnecessary noise into query responses and adversely affect accurate answer generation.

Therefore, drawing inspiration from ER models(Chen, 1976) and metadata, we propose the **TRA Structure**, a new data structure optimized for retrieval tasks that require precise information. TRA is represented as a hierarchical JSON object and consists of the following key components: (1) Title, (2) Attributes, and (3) Relations.

Although it may appear similar to metadata, it does not merely contain empirical information such as keywords or summarization; rather, it reconstructs the corpus itself. We draw on the key–value

organization of metadata and the central principle of ER models, in which an entity’s attributes and relations are structured around the entity itself rather than expressed only as separate triplets. This representation supports nested keys and multi-valued fields, allowing hierarchical and n-ary information to be expressed more flexibly than fixed triplets. Ultimately, the flexibility to include raw or granular text overcomes the structural rigidity and inability to represent hierarchy typical of triplet-based graphs.

Figure 1 illustrates the difference between triplets and TRA. The Title represents the main object that can represent the passage; Attributes organize the characteristics of the main object found within the corpus; and Relations store information when this title is related to other objects. Through this data structure, the corpus can be reconstructed around the main object, minimizing information loss and ensuring an intuitive structure that is easy for LLMs to understand.

3.2 TRA to Facts: Building optimized retrieval space

The limitation of both naive retrieval and summarization-based methods is that important information is either scattered across an excessively broad search space or lost during summarization. Because TRA can form a deeply nested hierarchical JSON structure, indexing the entire structure at once risks losing fine-grained details, similar to summarization, whereas over-fragmentation into triplet-like units causes semantic loss.

To balance granularity and context, we define a Fact as a single indexing unit by grouping the "Title + Top-level Key + entire Value." To address the inherent limitations of embedding models in processing raw JSON structures (Mandlík et al., 2022), we serialize these units into "artificial natural language" using a template-based flattening approach (e.g., "Key of Title is Sub-Key-1 is Sub-Value-1, Sub-Key-2 is Sub-Value-2, ..."). We assume that although these serialized Facts may contain grammatical irregularities, they remain sufficiently intelligible for modern embedding models to capture their underlying semantics. For Relations, where sub-keys are primarily verbs, we apply a simplified "Title + Key + Value" serialization. These Facts are then indexed using both dense embeddings and sparse indexing. For sparse retrieval, stopword removal is specifically applied to optimize BM25 performance. Through this process, we construct

a search space that is well balanced for users seeking specific information, and TRA-RAG performs retrieval over these Facts.

Examples of indexed Facts are shown in Figure 1, with additional detailed examples provided in Appendix E.1.

3.3 QD and Sequential Answering

While most graph-based RAG systems perform multi-hop reasoning through graph traversal or reranking matched entities (Guo et al., 2025b; Luo et al., 2025; Gutiérrez et al., 2025), these approaches can be suboptimal because the semantic similarity between intermediate hops and the main query often decreases as the number of hops increases. To evaluate this issue and demonstrate how retrieval-optimized TRA Facts are, TRA-RAG adopts a simple multi-hop strategy: QD and Sequential Answering.

First, upon receiving a multi-hop query, TRA-RAG decomposes it into single-hop queries. This process involves not only simple decomposition but also the extraction of dependency relations between queries, ensuring that the main component of a subsequent question corresponds to the answer of the preceding question.

In the sequential answering phase, answers are generated sequentially for each decomposed question. Crucially, retrieval is performed at each step, leveraging the history of previous questions and answers as context. This process can be formally expressed as follows: the answer a_i at step i is generated by

$$a_i = \text{LLM}(q_i \mid R_i, \mathcal{H}_{i-1}) \quad (1)$$

where q_i denotes the i -th decomposed question, R_i represents the retrieved context for that question, and $\mathcal{H}_{i-1} = \{(q_1, a_1), \dots, (q_{i-1}, a_{i-1})\}$ signifies the history of preceding query-answer pairs.

While establishing dependency relations risks error propagation, we hypothesize that this is outweighed by the advantage of retrieving hidden information inaccessible via the main query—a hypothesis validated in our experiments.

3.4 RRF and Reranking

To retrieve the indexed Facts, we employ RRF (Rackauckas, 2024) to combine cosine similarity from dense embeddings with BM25 scores based on each Fact’s ranking. Given the variable length of Facts (see detailed statistics in Table 9), the combination of dense and sparse methods provides com-

plementary strengths. For Simple QA, TRA-RAG directly uses the RRF retrieval results without QD or reranking. The remaining procedure in this section describes the Multi-Hop QA setting.

Furthermore, the RRF ranking mechanism mitigates information bias when selecting retrieved context for the final answering stage. Relying solely on raw scores from dense or BM25 retrieval can lead to a "score domination" issue, where results from a specific sub-query (SQ) overwhelm others due to disproportionately high absolute values, potentially causing the omission of critical information from other SQs. RRF normalizes these scores into ranks, ensuring a balanced integration of retrieved information across all sub-queries. The final RRF score for a Fact f is computed as $\text{Score}(f) = w_d \cdot \frac{1}{k+r_d(f)} + w_{bm} \cdot \frac{1}{k+r_{bm}(f)}$, where $r_d(f)$ and $r_{bm}(f)$ denote the rank of Fact f in dense and BM25 retrieval, respectively, and k is a smoothing constant.

Finally, we devise a reranking-based selection mechanism to distill the most relevant information for the Final Answering phase. Given the nature of QD, simply aggregating all retrieved information from each SQ answering phase is computationally inefficient and introduces unnecessary noise. To address this, we define the final input set $\mathcal{R}_{final}$ as a strategic union of two subsets: one capturing global relevance and the other focusing on Sub-Query specificity to ensure the recall of granular details.

1) Global Relevance Selection ($\mathcal{F}_{seed}, \mathcal{P}_{seed}$):

We first aggregate all Facts across all SQs and rank them by their RRF scores. From this list, we select the top F_i Facts to form $\mathcal{F}_{seed}$, along with their top P_i unique source passages to form $\mathcal{P}_{seed}$.

2) Sub-Query Specificity Selection ($\mathcal{F}_{re}, \mathcal{P}_{re}$):

While prior stage identifies broadly relevant facts, this secondary stage captures context-specific evidence that may only be prominent within the scope of a single sub-query. To uncover such hidden information critical for specific reasoning steps, we perform a secondary reranking on the remaining candidates. For a Fact f retrieved by q_{sub} , the reranking score is computed as $\text{Score}_{re}(f, q_{sub}) = w_d \cdot \hat{s}_d(f, q_{sub}) + w_{bm} \cdot \hat{s}_{bm}(f, q_{sub})$, where $\hat{s}$ denotes the Min-Max normalized value of the raw retrieval scores. Based on Score_{re} , we select the top F_j Facts ($\mathcal{F}_{re}$) and their corresponding top P_j source passages ($\mathcal{P}_{re}$).

3) Integration: The final consolidated context

Dataset	# Query	# Docs	Total Tokens
NQ	1,000	9,633	1,298,298
PopQA	1,000	8,603	1,075,152
HotpotQA	1,000	9,811	1,218,300
MuSiQue	1,000	9,838	984,691
2WikiMultiHopQA	1,000	6,119	626,320
LV-Eval	124	22,849	5,025,775

Table 1: Statistics of evaluation datasets. Token counts are computed using the o200k_base tokenizer from OpenAI’s tiktoken.

provided to the LLM is expressed as:

$$\mathcal{R}_{final} = \{\mathcal{F}_{seed} \cup \mathcal{P}_{seed}\} \cup \{\mathcal{F}_{re} \cup \mathcal{P}_{re}\} \quad (2)$$

This dual-track strategy ensures that both globally significant data and query-specific nuances are preserved, providing the necessary context for complex multi-hop reasoning.

Figure 2 illustrates the overall process of TRA-RAG on multi-hop questions.

4 Experiments

We designed a comprehensive set of experiments to address the following research questions: **RQ1**, how effective is TRA-RAG in retrieval and QA across Simple and Multi-Hop QA? **RQ2**, are the TRA Structure and Facts better retrieval structures than existing alternatives? **RQ3**, what is the core component that makes TRA-RAG strong? **RQ4**, are QD and Multi-step Retrieval essential for multi-hop QA?

4.1 Datasets

We evaluated TRA-RAG on six QA benchmarks. To assess its multi-hop reasoning capability, we selected four representative Multi-Hop QA datasets: HotpotQA(Yang et al., 2018), 2WikiMultiHopQA(Ho et al., 2020), MuSiQue(Trivedi et al., 2022), and LV-Eval(Yuan et al., 2024). In addition, to disentangle the effect of the TRA-based retrieval structure from that of QD, we used two Simple QA benchmarks, Natural Questions (NQ)(Kwiatkowski et al., 2019) and PopQA(Mallen et al., 2023), without applying QD. The dataset statistics are presented in Table 1, and additional details are provided in Appendix B.1. Statistics of the TRA Structure collected during experiments are provided in Appendix C.

4.2 Metrics

For answer generation quality, we utilized the standard **F1 score** and **LLM-Acc**, an accuracy score

Method	Simple QA				Multi-Hop QA								Avg.	
	NQ		PopQA		HotpotQA		2Wiki		MuSiQue		LV-Eval		F1	LLM-Acc
	F1	Acc	F1	Acc	F1	Acc	F1	Acc	F1	Acc	F1	Acc		
Non-Retrieval														
Pure-LLM	51.2	57.9	28.2	35.0	36.8	45.9	29.9	35.0	17.1	17.8	8.0	7.3	28.5	33.1
Oracle	68.5	70.4	57.9	64.4	75.3	85.7	60.5	66.1	52.0	56.3	-	-	62.8	68.6
Naive-Retrieval														
Cosine-Sim	58.7	60.1	59.7	68.1	56.2	65.1	39.3	42.7	24.5	26.2	2.9	0.8	40.2	43.8
BM25	56.9	58.6	54.7	62.9	51.5	59.6	34.0	35.8	16.8	17.5	5.0	3.2	36.5	39.6
Hybrid	59.5	61.7	60.3	69.3	56.4	66.0	37.3	40.4	20.8	21.7	2.9	0.8	39.5	43.3
Structure-augmented Retrieval														
GraphRAG	39.0	42.0	39.5	48.8	60.8	72.0	43.5	47.4	24.7	27.3	14.0	12.1	36.9	41.6
LightRAG	37.0	40.0	33.5	42.8	42.1	50.8	40.4	44.3	21.7	26.5	12.4	11.3	31.2	35.9
PathRAG	43.5	46.5	39.5	48.8	52.0	62.3	41.1	46.3	24.8	31.0	14.2	12.9	35.9	41.3
HippoRAG	44.5	47.5	45.0	54.3	60.2	71.6	50.9	56.0	26.3	29.0	7.2	4.8	39.0	43.9
RAPTOR	47.0	50.0	52.0	61.3	64.2	74.1	46.6	52.3	27.8	30.9	9.3	7.3	41.1	46.0
LinearRAG	55.7	58.4	57.4	67.2	68.9	79.1	56.1	64.7	34.2	39.6	14.8	13.0	47.9	53.7
HyperGraphRAG	46.5	49.5	34.0	43.3	66.1	77.6	49.9	56.6	36.0	45.9	17.1	16.1	41.6	48.2
HippoRAG2	60.1	63.8	61.0	70.2	66.3	78.1	55.9	64.6	38.5	43.1	8.6	5.6	48.4	54.2
Multi-Step Retrieval														
Iter-RetGen	59.8	62.6	59.6	67.7	57.7	67.0	38.5	42.2	22.1	23.9	2.9	0.8	40.1	44.0
Self-Ask	57.3	60.7	59.6	68.8	64.7	74.4	52.8	58.7	29.3	31.2	3.7	1.6	44.6	49.2
Self-RAG	57.1	61.5	58.4	69.3	62.6	72.1	49.2	55.3	27.5	30.6	7.8	5.6	43.8	49.1
TRA-RAG	65.7	66.6	65.1	73.4	71.2	81.8	70.7	78.2	48.8	53.4	15.1	16.1	56.1	61.6

Table 2: **Main Results.** **Bold** and underlined values indicate the best and second-best performance, respectively. Exact ties receive the same formatting. *Italicized* values represent the *Oracle* (ideal scenario with 100% recall). For LV-Eval, there are no gold-evidence annotations, so we could not measure its *Oracle* score.

Method	Simple QA				Multi-Hop QA				Avg.	
	NQ		PopQA		HotpotQA		2Wiki		MuSiQue	
	Rec@5	Rec@5	Rec@5	Hit-All@5	Rec@5	Hit-All@5	Rec@5	Hit-All@5	Rec@5	Hit-All@5
<i>Naive-Retrieval</i>										
Cosine-Sim	61.8	49.9	81.6	65.3	70.7	41.5	55.4	21.1	63.9	42.6
BM25	56.2	38.3	74.9	53.3	65.8	32.5	44.9	10.6	56.0	32.1
Hybrid	63.9	52.1	81.9	65.7	69.7	38.4	52.1	16.3	63.9	40.1
<i>Structure-augmented Retrieval</i>										
HippoRAG	44.2	53.5	82.3	70.5	82.9	64.4	53.7	21.0	63.3	52.0
RAPTOR	68.3	49.6	85.8	72.3	77.2	61.8	55.1	24.2	67.2	52.8
LinearRAG	70.1	54.8	88.3	76.2	87.4	72.3	<u>62.9</u>	32.1	<u>72.7</u>	60.2
HippoRAG2	<u>70.5</u>	51.9	87.3	75.9	86.5	67.1	<u>62.0</u>	31.7	71.6	58.2
<i>Multi-Step Retrieval</i>										
Iter-RetGen	67.7	51.9	84.6	71.4	71.1	41.2	54.1	19.7	65.9	44.1
Self-Ask	60.7	50.4	86.8	75.9	89.3	76.8	62.8	32.0	69.9	61.6
Self-RAG	60.3	48.9	86.3	74.2	84.7	67.1	60.6	28.8	68.2	56.7
TRA-RAG	75.4	58.5	89.8	81.6	97.5	94.5	72.2	43.5	78.7	73.2

Table 3: **Main Retrieval Results.** **Bold** and underlined values indicate the best and second-best performance, respectively. Exact ties receive the same formatting. Methods without passage-level retrieval outputs are omitted. Rec@5 denotes Recall@5. Reported averages are computed from unrounded scores.

derived from the *LLM-as-a-judge* (Li et al., 2025) paradigm. Regarding retrieval performance, we adopted **Recall@5**. Additionally, to assess whether the system successfully retrieves *all* necessary supporting facts for a given query, we employed the **Hit-All@5** metric. Further details on these metrics are provided in Appendix B.3.

4.3 Baselines

To evaluate the robustness of TRA-RAG and the impact of TRA Structure and multi-step retrieval, we compared baselines across four categories: (1) **Non-Retrieval**: We define the lower and upper bounds using *PureLLM* (0% recall) and *Oracle* (100% recall with gold passages). (2) **Naive Retrieval**: Represents general RAG approaches. We tested three retriever types: Cosine similarity, BM25, and Hybrid using RRF.

(3) **Multi-step Retrieval:** We employed widely used methods, Iter-RetGen(Shao et al., 2023), Self-Ask(Press et al., 2023), and Self-RAG(Asai et al., 2024). (4) **Structure-augmented Retrieval:** We selected promising structure-augmented RAG methods from three groups: graph-based methods (GraphRAG(Edge et al., 2024), LightRAG(Guo et al., 2025b), PathRAG(Chen et al., 2025a), HyperGraphRAG(Luo et al., 2025)), summarization-based method (RAPTOR(Sarathi et al., 2024)), and passage-enriched graph methods (LinearRAG(Zhuang et al., 2025), HippoRAG(Gutiérrez et al., 2024), HippoRAG2(Gutiérrez et al., 2025)).

Additional details about baselines are provided in Appendix B.2.

4.4 Implementation Details

The RRF setting was configured with $k = 60$. After retrieving 300 Facts each from dense and BM25 retrieval, RRF was performed with weights set to 1.0 for BM25 and 1.3 for dense. From the RRF results, we selected 3 passages and 20 facts as *seed*, and during the reranking process, an additional 2 passages and 10 facts were selected. For Simple QA, where QD is not applied, the reranking process is omitted. For all methods, we standardized the passage input used for final answering to five passages; structure-augmented methods additionally retained their indexing-time structural information. Additional implementation details are provided in Appendix B.4, and all prompts are provided in Appendix A.

5 Results

5.1 Results on Simple and Multi-Hop QA (RQ1)

As shown in Table 2, TRA-RAG achieved the best performance on every reported QA metric except F1 on LV-Eval, where it ranked second. On the Simple QA benchmarks, it outperformed the strongest baseline, HippoRAG2, by 5.6 F1 and 2.8 LLM-Acc points on NQ, and by 4.1 F1 and 3.2 LLM-Acc points on PopQA. Retrieval performance also reached peak figures across all five datasets with retrieval annotations (Table 3).

We highlight three key findings from these results. First, TRA-RAG’s gains over the strongest baselines are modest on HotpotQA (2.3 F1 and 2.7 LLM-Acc) but substantially larger on 2WikiMultiHopQA (14.6 and 13.5 points) and MuSiQue (10.3 F1 and 7.5 LLM-Acc). This suggests that the ben-

Metric	Baseline	TRA (Ours)
<i>Exp 1: vs. Triplets</i>		
Completeness	4.185	4.869
Faithfulness	4.594	4.990
Structural Clarity	3.890	4.916
Atomic Granularity	4.124	4.854
Overall Average	4.198	4.907
Win Rate	-	87.25%
<i>Exp 2: vs. Hyperedge</i>		
Completeness	3.737	4.969
Faithfulness	4.350	4.997
Structural Clarity	3.683	4.961
Atomic Granularity	3.804	4.958
Overall Average	3.894	4.971
Win Rate	-	98.55%

Table 4: Structural suitability comparison of the TRA Structure against Triplets and Hyperedges.

efit of TRA-RAG becomes more pronounced on more demanding multi-hop tasks.

Second, TRA-RAG notably outperformed the *Oracle* on the 2WikiMultiHopQA dataset. Its 97.5% retrieval recall provides near-Oracle evidence coverage, while the retrieved context additionally includes structured Facts derived from the same source passages. This result suggests that explicitly structured evidence can help the LLM identify key reasoning hops more effectively than raw passages alone.

Third, multi-step retrieval methods, especially Self-Ask, demonstrated performance comparable to or exceeding triplet-based methods and RAPTOR. This suggests that merely representing a corpus as a graph to link entities or summarize relevant information does not provide as much inherent advantage for retrieval as previously anticipated, unless paired with an effective reasoning framework. Further analysis of this result is provided in RQ4.

5.2 Evaluation on TRA Structure (RQ2)

To evaluate whether the TRA Structure is genuinely retrieval-optimized compared with existing alternatives, we conducted an *LLM-as-a-judge* comparison against Triplets and HyperEdges. We evaluated all structures on four criteria—Completeness, Faithfulness, Structural Clarity, and Atomic Granularity—using a five-point Likert scale, followed by overall winner selection. The evaluation was performed on 1,000 passages from HotpotQA, with candidate order randomized for each comparison.

As shown in Table 4, the TRA Structure outperforms both Triplets and HyperEdges overall, with

# Facts	# Passages			
	0	3	5	10
0	-	57.3	59.8	66.5
10	48.5	61.5	67.5	68.6
30	58.5	66.2	70.7	70.5
50	57.2	68.1	70.1	69.3

Table 5: Ablation results on 2WikiMultiHopQA (F1). #Facts and #Passages denote the numbers used at answer generation. Seed counts were set by the target size: for #Facts of 10/30/50, the seed Facts were 4/10/20; for #Passages of 3/5/10, the seed Passages were 1/2/4 (and 0 means no seed). Across repeated runs, F1 varied by approximately ± 0.1 points.

the largest gains in *Structural Clarity* and *Atomic Granularity*. This result suggests that TRA provides a better retrieval target for similarity matching by preserving key information in finer, more coherent units. This structural advantage is also reflected in the retrieval performance reported in Table 3. In contrast, HyperEdges appear less effective for single-document detail representation, likely because n-ary structures are better suited to aggregating related information across multiple documents. A similar trade-off is often observed in summarization-based representations, where compression can obscure query-critical details. By preserving contextual cues while keeping evidence units structurally explicit, TRA Facts provide a stronger balance between granularity and coherence for retrieval.

Additional examples of TRA Structures and Facts are provided in Appendix E.

5.3 Ablation studies (RQ3)

We conducted ablation studies to quantify the contribution of each component in TRA-RAG.

First, we varied the amount of information used in the final answering stage. Table 5 shows performance changes as the numbers of passages and Facts are adjusted, while keeping retrieval fixed. Performance generally improves as more evidence is provided, but drops after a threshold (approximately near 10 passages or 50 Facts), indicating an information-overload effect. In other words, simply increasing context does not guarantee better reasoning once irrelevant or redundant information starts to dominate. This highlights the importance of selecting only the most relevant evidence for RAG and supports the need for appropriately sized information units and a search space aligned with them.

Method	F1	LLM-Acc	Rec@5	Hit-All@5
Ours (Full)	70.7	78.2	97.5	94.5
<i>Retriever Variants</i>				
w/ BM25 (Sparse)	69.5	77.0	95.7	90.4
w/ Cosine (Dense)	70.3	77.6	96.9	94.2
<i>Component Ablation</i>				
w/o Reranker	69.3	76.8	96.5	92.6
w/o Prev. Context	66.2	73.6	97.1	93.9

Table 6: Ablation study on the 2WikiMultiHopQA dataset.

Second, Table 6 reports results after modifying key components: (1) replacing the RRF-based hybrid retriever with BM25-only or cosine-only retrieval, (2) removing reranking, and (3) excluding the previous QA history during sub-question answering. Changing the retriever and removing reranking caused only modest differences in aggregate metrics, although reranking still improved specific complex cases at minimal additional cost because it requires no extra LLM calls. This makes the reranking step a practical design choice even when average gains are moderate. In contrast, removing the previous QA history led to a clear drop in answer quality despite similar retrieval performance. This indicates that retrieving relevant evidence alone is insufficient; context chaining across steps is crucial for accurate multi-hop answering.

5.4 Impact of Multi-step Retrieval (RQ4)

Finally, we examine whether the performance of TRA-RAG is driven primarily by QD or also by the quality of its retrieval structure. For a fair comparison, we combined the same QD module with selected single-step retrieval methods and compared their Base and +QD variants in Table 7.

QD improves all tested methods, confirming its importance for multi-hop reasoning. However, the results also show that QD alone does not account for the performance of TRA-RAG. Without QD, TRA-RAG achieves 48.0 F1 and 51.5 LLM-Acc, outperforming both Hybrid and LightRAG, although remaining below HyperGraphRAG. Once the same QD strategy is applied, TRA-RAG reaches 70.7 F1 and 78.2 LLM-Acc, exceeding HyperGraphRAG +QD by 12.4 and 13.3 points, respectively. This indicates that QD and the TRA Fact search space provide complementary benefits rather than either component being solely responsible for the final performance.

The Simple QA results provide further evidence for the contribution of the retrieval structure be-

Method	F1 Score		LLM-Acc	
	Base	+QD	Base	+QD
<i>Single-step Baselines</i>				
Hybrid	37.3	58.7	40.4	65.3
LightRAG	40.4	48.1	44.3	55.3
HyperGraphRAG	49.9	58.3	56.6	64.9
<i>Multi-step Baselines</i>				
Iter-RetGen	38.5	-	42.2	-
Self-Ask	52.8	-	58.7	-
<i>Ours</i>				
TRA-RAG	48.0	70.7	51.5	78.2

Table 7: Comparing performance with and without QD reveals consistent improvements. “Base” denotes the original single-step method. For TRA-RAG, removing QD also removes sequential sub-question answering and reranking. “+QD” denotes the method augmented with the same QD strategy. Experiments on 2WikiMultiHopQA.

cause QD and reranking are not used in this setting. As shown in Table 2, TRA-RAG still outperforms all baselines on both NQ and PopQA. These results support our central claim that the TRA Structure and its Fact-based indexing provide an inherently effective retrieval target, independent of the additional gains obtained from multi-step reasoning.

The comparison also raises questions about the practical contribution of graph connectivity to multi-hop reasoning. Hybrid +QD slightly outperforms both LightRAG +QD and HyperGraphRAG +QD despite using no graph connectivity, while TRA-RAG substantially surpasses them using a retrieval-optimized structure and a simple QD strategy. Although connectivity may remain useful in some settings, these results suggest that it does not necessarily provide a decisive advantage when the retrieval target and reasoning procedure are well designed. Overall, the results emphasize that TRA-RAG’s strength arises from combining an effective Fact search space with QD, SQ answering, and reranking rather than relying on graph traversal or connectivity.

5.5 Efficiency Analysis

TRA-RAG applies QD to Multi-Hop QA but not to Simple QA. Without QD, retrieval requires no LLM calls and relies only on similarity-based search; therefore, the retrieval cost and efficiency of TRA-RAG are expected to be comparable to those of naive RAG.

Table 8 reports the retrieval time of each method on the Simple QA benchmarks. TRA-RAG

Method	NQ (s)	PopQA (s)
NaiveRAG (Hybrid)	0.672	0.613
HippoRAG2	1.526	1.432
LinearRAG	0.888	1.812
Self-RAG	8.452	7.822
TRA-RAG	0.753	0.721

Table 8: Retrieval time in seconds per query on Simple QA benchmarks, averaged over all questions in each benchmark. Lower is better.

achieves retrieval times close to those of NaiveRAG (Hybrid), requiring 0.753 seconds on NQ and 0.721 seconds on PopQA. In contrast, methods that rely on additional retrieval mechanisms incur substantially higher latency, including Personalized PageRank in HippoRAG2 and iterative retrieval processes in LinearRAG and Self-RAG.

These results demonstrate that the retrieval-optimized TRA Structure and Facts provide advantages not only in effectiveness but also in efficiency, without relying on costly retrieval-time mechanisms. More broadly, they suggest that designing a strong retrieval structure can yield meaningful gains in both the performance and efficiency of RAG systems. Additional cost analysis is provided in Appendix D.2.

Failure cases are presented in Appendix E.4, and future work on the graph adaptability of TRA is discussed in Appendix F.

6 Conclusion

In this paper, we proposed the **TRA Structure** to overcome the limitations of existing retrieval structures, including triplets, and to enable more structured and flexible information modeling.

TRA-RAG converts text into retrieval-optimized Facts and executes a robust retrieval pipeline. Experiments across Simple QA and Multi-Hop QA show the broader applicability of these Facts, while the multi-hop experiments provide practical insights into effectively combining QD with RAG.

Prior studies have introduced a variety of retrieval structures, such as triplets, hyperedges, and summarization-based representations, but many focus primarily on retrieval procedures while under-emphasizing the retrieval target itself. In contrast, this paper shows that strong performance can be achieved with a simple retrieval process when the underlying structure is well optimized for retrieval.

Limitations

The limitations of paper can be summarized into three main aspects. First, we did not perform TRA construction on long passages. As passages become longer, the appearing objects become more diverse, making the boundaries for selecting which object to construct as Title ambiguous. Therefore, following prior research(Gutiérrez et al., 2025), our experiments were conducted using datasets where these objects are clearly defined. To apply this to a longer passage, more sophisticated design prompts or logical chunking methods would be required.

Second, we did not propose a method for tasks that encompass the entire corpus through global understanding, such as summarization. This is because the main purpose of this paper lies in multi-hop reasoning, and achieving global understanding would require additional methods. For instance, techniques such as accessing information based on the TRA title to summarize, or utilizing the TRA field as metadata, could be employed.

Third, the datasets used are all Wikipedia-based. In other words, while the capability in the general domain has been sufficiently verified, its robustness in specific domains has not been tested. However, we expect the method to demonstrate strength in multi-hop reasoning provided it can be executed correctly. We will address these areas as future work.

Acknowledgements

This work was supported by the Institute of Information and communications Technology Planning and evaluation (IITP) grant (No.RS-2025-25422680, No. RS-2020-II201373), and the National Research Foundation of Korea (NRF) grant (No. RS-2025-00520618) funded by the Korean Government (MSIT).

References

- Akari Asai, Zeqiu Wu, Yizhong Wang, Avi Sil, and Hannaneh Hajishirzi. 2024. [Self-rag: Learning to retrieve, generate, and critique through self-reflection](#). In *International Conference on Learning Representations*, volume 2024, pages 9112–9141.
- Haonan Bian. 2025. Llm-empowered knowledge graph construction: A survey. *arXiv preprint arXiv:2510.20345*.
- Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. [Freebase: a collaboratively created graph database for structuring human knowledge](#). In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, SIGMOD '08, pages 1247–1250, New York, NY, USA. Association for Computing Machinery.
- Boyu Chen, Zirui Guo, Zidan Yang, Yuluo Chen, Junze Chen, Zhenghao Liu, Chuan Shi, and Cheng Yang. 2025a. Pathrag: Pruning graph-based retrieval augmented generation with relational paths. *arXiv preprint arXiv:2502.14902*.
- Peter Pin-Shan Chen. 1976. [The entity-relationship model—toward a unified view of data](#). *ACM Trans. Database Syst.*, 1(1):9–36.
- Qiguang Chen, Libo Qin, Jinhao Liu, Dengyun Peng, Jiannan Guan, Peng Wang, Mengkang Hu, Yuhang Zhou, Te Gao, and Wanxiang Che. 2025b. Towards reasoning era: A survey of long chain-of-thought for reasoning large language models. *arXiv preprint arXiv:2503.09567*.
- Michail Dadopoulos, Anestis Ladas, Stratos Moschidis, and Ioannis Negkakis. 2025. Metadata-driven retrieval-augmented generation for financial question answering. *arXiv preprint arXiv:2510.24402*.
- Junnan Dong, Siyu An, Yifei Yu, Qian-Wen Zhang, Linhao Luo, Xiao Huang, Yunsheng Wu, Di Yin, and Xing Sun. 2025. Youtu-graphrag: Vertically unified agents for graph retrieval-augmented complex reasoning. *arXiv preprint arXiv:2508.19855*.
- Darren Edge, Ha Trinh, Newman Cheng, Joshua Bradley, Alex Chao, Apurva Mody, Steven Truitt, Dasha Metropolitan, Robert Osazuwa Ness, and Jonathan Larson. 2024. From local to global: A graph rag approach to query-focused summarization. *arXiv preprint arXiv:2404.16130*.
- Tsu-Jui Fu, Peng-Hsuan Li, and Wei-Yun Ma. 2019. [GraphRel: Modeling text as relational graphs for joint entity and relation extraction](#). In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1409–1418, Florence, Italy. Association for Computational Linguistics.
- Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, Meng Wang, and Haofen Wang. 2023. Retrieval-augmented generation for large language models: A survey. *arXiv preprint arXiv:2312.10997*.
- Kai Guo, Xinnan Dai, Shenglai Zeng, Harry Shomer, Haoyu Han, Yu Wang, and Jiliang Tang. 2025a. Beyond static retrieval: Opportunities and pitfalls of iterative retrieval in graphrag. *arXiv preprint arXiv:2509.25530*.
- Zirui Guo, Lianghao Xia, Yanhua Yu, Tu Ao, and Chao Huang. 2025b. [LightRAG: Simple and fast retrieval-augmented generation](#). In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 10746–10761, Suzhou, China. Association for Computational Linguistics.

- Shailja Gupta, Rajesh Ranjan, and Surya Narayan Singh. 2024. [A comprehensive survey of retrieval-augmented generation \(rag\): Evolution, current landscape and future directions](#). *ArXiv*, abs/2410.12837.
- Bernal Jiménez Gutiérrez, Yiheng Shu, Yu Gu, Michihiro Yasunaga, and Yu Su. 2024. [Hipporag: Neurobiologically inspired long-term memory for large language models](#). In *Advances in Neural Information Processing Systems*, volume 37, pages 59532–59569. Curran Associates, Inc.
- Bernal Jiménez Gutiérrez, Yiheng Shu, Weijian Qi, Sizhe Zhou, and Yu Su. 2025. From rag to memory: Non-parametric continual learning for large language models. In *International Conference on Machine Learning*, pages 21497–21515. PMLR.
- Haoyu Han, Li Ma, Yu Wang, Harry Shomer, Kai Guo, Yongjia Lei, Zhisheng Qi, Zhigang Hua, Bo Long, Hui Liu, et al. 2026. Rag vs. graphrag: A systematic evaluation and key insights. In *Proceedings of the 32nd ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 2*, pages 8966–8977.
- Haoyu Han, Yu Wang, Harry Shomer, Kai Guo, Jiayuan Ding, Yongjia Lei, Mahantesh Halappanavar, Ryan A Rossi, Subhabrata Mukherjee, Xianfeng Tang, et al. 2024. Retrieval-augmented generation with graphs (graphrag). *arXiv preprint arXiv:2501.00309*.
- Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. 2020. Constructing a multi-hop qa dataset for comprehensive evaluation of reasoning steps. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 6609–6625.
- Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, et al. 2024. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*.
- Rolf Jagerman, Honglei Zhuang, Zhen Qin, Xuanhui Wang, and Michael Bendersky. 2023. Query expansion by prompting large language models. *arXiv preprint arXiv:2305.03653*.
- Pengcheng Jiang, Lang Cao, Ruikun Zhu, Minhao Jiang, Yunyi Zhang, Jimeng Sun, and Jiawei Han. 2025a. Ras: Retrieval-and-structuring for knowledge-intensive llm generation. *arXiv preprint arXiv:2502.10996*.
- Pengcheng Jiang, Siru Ouyang, Yizhu Jiao, Ming Zhong, Runchu Tian, and Jiawei Han. 2025b. Retrieval and structuring augmented generation with large language models. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 2*, pages 6032–6042.
- Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 conference on empirical methods in natural language processing (EMNLP)*, pages 6769–6781.
- Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, et al. 2019. Natural questions: a benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:453–466.
- Dawei Li, Bohan Jiang, Liangjie Huang, Alimohammad Beigi, Chengshuai Zhao, Zhen Tan, Amrita Bhat-tacharjee, Yuxuan Jiang, Canyu Chen, Tianhao Wu, et al. 2025. From generation to judgment: Opportunities and challenges of llm-as-a-judge. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 2757–2791.
- Zhuoqun Li, Xuanang Chen, Haiyang Yu, Hongyu Lin, Yaojie Lu, Qiaoyu Tang, Fei Huang, Xianpei Han, Le Sun, and Yongbin Li. 2024. Structrag: Boosting knowledge intensive reasoning of llms via inference-time hybrid information structurization. *arXiv preprint arXiv:2410.08815*.
- Kevin Lin, Kyle Lo, Joseph Gonzalez, and Dan Klein. 2023. Decomposing complex queries for tip-of-the-tongue retrieval. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 5521–5533.
- Haoran Luo, Haihong E, Guanting Chen, Yandan Zheng, Xiaobao Wu, Yikai Guo, Qika Lin, Yu Feng, Zemin Kuang, Meina Song, Yifan Zhu, and Anh Tuan Luu. 2025. [Hypergraphrag: Retrieval-augmented generation via hypergraph-structured knowledge representation](#). In *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pages 152206–152234. Curran Associates, Inc.
- Alex Mallen, Akari Asai, Victor Zhong, Rajarshi Das, Daniel Khashabi, and Hannaneh Hajishirzi. 2023. When not to trust language models: Investigating effectiveness of parametric and non-parametric memories. In *Proceedings of the 61st annual meeting of the association for computational linguistics (volume 1: Long papers)*, pages 9802–9822.
- Šimon Mandlík, Matěj Račinský, Viliam Lisý, and Tomáš Pevný. 2022. JsonGrinder.jl: Automated differentiable neural architecture for embedding arbitrary JSON data. *Journal of Machine Learning Research*, 23(298):1–5.
- Vaibhav Mavi, Anubhav Jangra, and Adam Jatowt. 2024. Multi-hop question answering. *Foundations and Trends® in Information Retrieval*, 17(5):457–586.
- Shervin Minaee, Tomas Mikolov, Narjes Nikzad, Meysam Chenaghlu, Richard Socher, Xavier Amatriain, and Jianfeng Gao. 2024. Large language models: A survey. *arXiv preprint arXiv:2402.06196*.

- Pranav Pushkar Mishra, Kranti Prakash Yeole, Ramyashree Keshavamurthy, Mokshit Bharat Surana, and Fatemeh Sarayloo. 2025. A systematic framework for enterprise knowledge retrieval: Leveraging llm-generated metadata to enhance rag systems. *arXiv preprint arXiv:2512.05411*.
- Boci Peng, Yun Zhu, Yongchao Liu, Xiaohe Bo, Haizhou Shi, Chuntao Hong, Yan Zhang, and Siliang Tang. 2024. Graph retrieval-augmented generation: A survey. *arXiv preprint arXiv:2408.08921*.
- Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah A Smith, and Mike Lewis. 2023. Measuring and narrowing the compositionality gap in language models. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 5687–5711.
- Zackary Rackauckas. 2024. Rag-fusion: a new take on retrieval-augmented generation. *arXiv preprint arXiv:2402.03367*.
- Stephen Robertson and Hugo Zaragoza. 2009. [The probabilistic relevance framework: Bm25 and beyond](#). *Found. Trends Inf. Retr.*, 3(4):333–389.
- Parth Sarthi, Salman Abdullah, Aditi Tuli, Shubh Khanna, Anna Goldie, and Christopher Manning. 2024. [Raptor: Recursive abstractive processing for tree-organized retrieval](#). In *International Conference on Learning Representations*, volume 2024, pages 32628–32649.
- Zhihong Shao, Yeyun Gong, Yelong Shen, Minlie Huang, Nan Duan, and Weizhu Chen. 2023. Enhancing retrieval-augmented large language models with iterative retrieval-generation synergy. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 9248–9274.
- Chaitanya Sharma. 2025. Retrieval-augmented generation: A comprehensive survey of architectures, enhancements, and robustness frontiers. *arXiv preprint arXiv:2506.00054*.
- Kartik Sharma, Peeyush Kumar, and Yunqing Li. 2025. [OG-RAG: Ontology-grounded retrieval-augmented generation for large language models](#). In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 32962–32981, Suzhou, China. Association for Computational Linguistics.
- Aditi Singh, Abul Ehtesham, Saket Kumar, and Tala Talaei Khoei. 2025. Agentic retrieval-augmented generation: A survey on agentic rag. *arXiv preprint arXiv:2501.09136*.
- Karen Sparck Jones. 1988. *A statistical interpretation of term specificity and its application in retrieval*, pages 132–142. Taylor Graham Publishing, GBR.
- Jiashuo Sun, Chengjin Xu, Lumingyuan Tang, Saizhuo Wang, Chen Lin, Yeyun Gong, Lionel M Ni, Heung-Yeung Shum, and Jian Guo. 2023. Think-on-graph: Deep and responsible reasoning of large language model on knowledge graph. *arXiv preprint arXiv:2307.07697*.
- Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2022. Musique: Multi-hop questions via single-hop question composition. *Transactions of the Association for Computational Linguistics*, 10:539–554.
- Denny Vrandečić and Markus Krötzsch. 2014. [Wiki-data: a free collaborative knowledgebase](#). *Commun. ACM*, 57(10):78–85.
- Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed Chi, Quoc V Le, and Denny Zhou. 2022. [Chain-of-thought prompting elicits reasoning in large language models](#). In *Advances in Neural Information Processing Systems*, volume 35, pages 24824–24837. Curran Associates, Inc.
- An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.
- Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D Manning. 2018. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 conference on empirical methods in natural language processing*, pages 2369–2380.
- Tao Yuan, Xuefei Ning, Dong Zhou, Zhijie Yang, Shiyao Li, Minghui Zhuang, Zheyue Tan, Zhuayu Yao, Dahua Lin, Boxun Li, et al. 2024. Lv-eval: A balanced long-context benchmark with 5 length levels up to 256k. *arXiv preprint arXiv:2402.05136*.
- Le Zhang, Yihong Wu, Qian Yang, and Jian-Yun Nie. 2024. Exploring the best practices of query expansion with large language models. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 1872–1883.
- Qinggang Zhang, Shengyuan Chen, Yuanchen Bei, Zheng Yuan, Huachi Zhou, Zijin Hong, Hao Chen, Yilin Xiao, Chuang Zhou, Junnan Dong, et al. 2025. A survey of graph retrieval-augmented generation for customized large language models. *arXiv preprint arXiv:2501.13958*.
- Mengyue Zhao, Matthew Nokleby, Bo Shen, Wenbo Dong, Deepti Pachauri, and Andrew Yang. 2025. [Ontology-guided knowledge graph retrieval for multi-hop and cross-granularity store fulfillment queries](#). In *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR ’25*, pages 4360–4364, New York, NY, USA. Association for Computing Machinery.
- Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, et al. 2023. A survey of large language models. *arXiv preprint arXiv:2303.18223*.

- Yingli Zhou, Yaodong Su, Youran Sun, Shu Wang, Tao-tao Wang, Runyuan He, Yongwei Zhang, Sicong Liang, Xilin Liu, Yuchi Ma, et al. 2025. In-depth analysis of graph-based rag in a unified framework. *arXiv preprint arXiv:2503.04338*.
- Xiangrong Zhu, Yuexiang Xie, Yi Liu, Yaliang Li, and Wei Hu. 2025a. Knowledge graph-guided retrieval augmented generation. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 8912–8924.
- Zulun Zhu, Tiancheng Huang, Kai Wang, Junda Ye, Xinghe Chen, and Siqiang Luo. 2025b. Graph-based approaches and functionalities in retrieval-augmented generation: A comprehensive survey. *arXiv preprint arXiv:2504.10499*.
- Luyao Zhuang, Shengyuan Chen, Yilin Xiao, Huachi Zhou, Yujing Zhang, Hao Chen, Qinggang Zhang, and Xiao Huang. 2025. Linearrag: Linear graph retrieval augmented generation on large-scale corpora. *arXiv preprint arXiv:2510.10114*.
- Ziyuan Zhuang, Zhiyang Zhang, Sitao Cheng, Fangkai Yang, Jia Liu, Shujian Huang, Qingwei Lin, Saravan Rajmohan, Dongmei Zhang, and Qi Zhang. 2024. Efficientrag: Efficient retriever for multi-hop question answering. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 3392–3411.

A Prompts

A.1 TRA Structure Construction Prompt

```
metadata_construction_prompt = """
You are an expert metadata extraction engine specialized in transforming natural
language passages into fully structured JSON metadata objects.

You will be given a passage that describes one or more entities. Your task is to
extract ALL information without ANY omission** and convert it into a
structured metadata JSON.

### CRITICAL REQUIREMENTS - NO INFORMATION LOSS
- MANDATORY: The output MUST include EVERY SINGLE factual detail from the
  passage.
- NO omissions: Every name, date, number, location, relationship, description,
  attribute MUST be captured.
- NO summarizations: Use exact quotes and full details from the original text.
- Metadata must be hierarchical, semantically precise, and fully structured JSON
  .
- All relationships must be explicitly captured under a "relations" key.

### JSON SCHEMA TEMPLATE
{
  "title": <string>,
  "attributes": {
    <attribute_name>: <value_or_nested_object>
  },
  "relations": [
    {
      "relation": <string>,
      "target": <entity_name_or_object>
    }
  ]
}

### Instruction:
- Transform the following passage into structured JSON metadata using the schema
  above.
- Output only the final JSON object -- no explanations.
- The output must include every factual detail**.

### Passage:
{{input}}
"""
```

Figure 3: The system prompt used for TRA Structure construction. It instructs the model to extract all factual details into a hierarchical JSON format without omission, using a provided schema.

A.2 Query Decomposition (QD) Prompt

```
query_decomposition_prompt = """
You are an expert query decomposition engine. Your goal is to break down a
complex multi-hop question into a series of simple, reasoning-based sub-
questions.

### INSTRUCTION
1. Analyze the input question and identify the necessary steps to answer it.
2. Decompose the question into a dependency graph of sub-questions.
3. Use placeholders like "[SQ_ID_Answer]" to refer to the answers of previous
   sub-questions.
4. Ensure the output is a VALID JSON list.

### JSON OUTPUT FORMAT
[
  {
    "id": "SQ1",
    "question": "The first step question needed to solve the problem?",
    "depends_on": [],
    "reasoning": "Explain why this step is needed."
  },
  {
    "id": "SQ2",
    "question": "The next question that uses the [SQ1_Answer]?",
    "depends_on": ["SQ1"],
    "reasoning": "Explain the dependency."
  }
]

NOW DECOMPOSE THIS QUESTION

Question:
{{input_question}}

Return ONLY valid JSON.
"""
```

Figure 4: The system prompt for QD. The model is instructed to break down a complex query into a dependency-aware sequence of sub-questions, enabling multi-hop reasoning.

A.3 SQ Answering Prompt

```
subquestion_answering_prompt = """
---Role---
You are a multi-hop retrieval-augmented assistant.

---Goal---
Read the provided Information passages and generate the correct answer to the
Sub-Question.

---Constraints & Format---
- Output ONLY the answer text (max 5 words).
- Do NOT include emojis, bullet points, or extra commentary.
- If the Information does not contain the answer, respond with "Insufficient
  information."

---Input Data---
Previous Context: {{previous_context}}
Information: {{passages}}
Sub-Question: {{subquestion}}

---Instruction---
You should think step-by-step to answer the question, but output ONLY the final
answer text without any prefix.
"""
```

Figure 5: The prompt for Sub-Question Answering. It constrains the model to generate concise, factual answers based solely on the retrieved passages and previous context.

A.4 Final Answering Prompt

```
final_answer_synthesis_prompt = """
---Role & Goal---
You are a multi-hop retrieval-augmented assistant. Your goal is to generate the
final answer by synthesizing the Sub-Question Chain and All Retrieved
Passages.

---Critical Strategies---
1. **Verify**: Cross-check sub-question answers against original passages.
2. **Fill Gaps**: If a sub-question returned "Insufficient information",
  aggressively re-examine passage titles and metadata.
3. **Reasoning**: Perform arithmetic or logical inference (e.g., temporal logic)
  to combine partial clues.

---Constraints---
- Output ONLY the final answer text (max 5 words).
- If the answer is absent from both sub-answers and passages, return "
  Insufficient information."

---Input Data---
Sub-Question Chain: {{subquestion_chain}}
All Passages: {{passages}}
Main Query: {{main_question}}

---Instruction---
Think step-by-step to synthesize the information, but output ONLY the final
answer.
"""
```

Figure 6: The Final Answer Synthesis prompt. It instructs the model to aggregate findings from the sub-question chain and original passages, applying reasoning to deduce the final answer even if individual steps were incomplete.

A.5 Answer Evaluation Prompt

```
evaluation_prompt = """
You are an expert evaluator for QA systems. Your task is to verify if a
    Predicted Answer is factually consistent with the Gold Answer.

---Input Data---
Question: {question}
Gold Answer: {gold_answer}
Predicted Answer: {predicted_answer}

---Evaluation Criteria---
1. CORRECT if it contains the essential information, even if worded differently.
2. CORRECT if it adds context without contradiction.
3. INCORRECT if it contradicts the gold answer or claims "Insufficient
    information" when an answer exists.

---Output Format---
Respond with ONLY a JSON object:
{
    "verdict": "CORRECT" | "INCORRECT",
    "confidence": "HIGH" | "MEDIUM" | "LOW",
    "reason": "Brief explanation"
}
"""
```

Figure 7: The Evaluation Prompt used for automated scoring. It instructs the LLM to perform semantic verification rather than exact string matching, ensuring that predictions with correct facts but different wording are judged accurately.

A.6 Comparison Evaluation Prompt

```
comparison_evaluation_prompt = """
### Role
You are an expert in Knowledge Graph construction. Evaluate and compare the
quality of two structured extractions (Candidate 1 vs. Candidate 2) derived
from the same Corpus.

### Evaluation Rubrics (Score 1-5)
1. **Completeness**: Does the set capture all key facts, entities, and relations
?
2. **Faithfulness**: Is the structure strictly supported by the corpus without
hallucination?
3. **Structural Clarity**: Are entities/predicates well-defined and pronouns
resolved?
4. **Atomic Granularity**: Is each fact irreducible and properly decomposed?

### Input Data
Original Corpus: {{original_corpus}}
Candidate 1: {{candidate_1}}
Candidate 2: {{candidate_2}}

### Output Format (Strict JSON)
Return a single JSON object. "final_decision" must be "Candidate 1" or "
Candidate 2".
{
  "scores": {
    "completeness": {"candidate1": <int>, "candidate2": <int>},
    "faithfulness": {"candidate1": <int>, "candidate2": <int>},
    "structural_clarity": {"candidate1": <int>, "candidate2": <int>},
    "atomic_granularity": {"candidate1": <int>, "candidate2": <int>}
  },
  "average": {"candidate1": <float>, "candidate2": <float>},
  "final_decision": "Candidate 1" | "Candidate 2",
  "brief_justification": {
    "candidate1": "<string>",
    "candidate2": "<string>"
  }
}
"""
```

Figure 8: The Comparative Evaluation Prompt used for the ablation study. LLM acts as a judge to score the quality of the extracted structures (Triplets/Hyperedges vs. TRA Structure) based on completeness, faithfulness, clarity, and granularity.

B Experimental Setup

B.1 Dataset

Following the evaluation setup of HippoRAG2, we used six benchmarks comprising two Simple QA datasets and four Multi-Hop QA datasets to evaluate both single-hop retrieval and multi-hop reasoning capabilities of **TRA-RAG**.

- **Natural Questions (NQ)** (Kwiatkowski et al., 2019): A large-scale open-domain QA benchmark constructed from real user queries submitted to Google Search. We use the repaired version adopted in the HippoRAG2 evaluation setup.
- **PopQA** (Mallen et al., 2023): An open-domain QA benchmark designed to evaluate factual knowledge across entities with varying levels of popularity. We use the canonical, deduplicated version adopted in the HippoRAG2 evaluation setup.
- **HotpotQA** (Yang et al., 2018): The most widely adopted benchmark serving as the archetype for multi-hop QA, despite known limitations regarding reasoning shortcuts.
- **2WikiMultiHopQA** (Ho et al., 2020): Designed to mitigate the shortcut issues of HotpotQA, this dataset explicitly requires multi-step reasoning over structured logical patterns.
- **MuSiQue** (Trivedi et al., 2022): A highly challenging dataset that enforces connected reasoning, where successfully retrieving evidence for one hop is strictly necessary to proceed to the next.
- **LV-Eval** (Yuan et al., 2024): A dataset known for its high difficulty, which minimizes knowledge leakage and overfitting by replacing key entities, thereby forcing the model to rely solely on the provided context.

The original NQ and PopQA releases do not provide gold-passages annotations in the form required for our retrieval evaluation. We therefore use the gold passages constructed through preprocessing by the HippoRAG2 authors. Because these annotations are not part of the official benchmark releases, they may contain preprocessing or matching errors, and the corresponding retrieval results should be interpreted with caution.

B.2 Baselines

To validate the effectiveness of the TRA Structure and TRA-RAG, we compared our method against a comprehensive set of baselines organized into four groups. Unless otherwise specified, all baselines use the publicly available default settings from their official implementations.

Non-Retrieval We defined two settings to estimate the lower and upper bounds of RAG performance. **PureLLM** generates answers without external information (i.e., 0% recall), serving as the lower bound. **Oracle** is given all gold passages (i.e., 100% recall), serving as the upper bound. In the Oracle setting, to match our method’s 5-passage input, we padded with noise passages when fewer than five gold passages were available.

Naive Retrieval We included standard single-step retrieval methods to represent conventional RAG approaches. To evaluate retriever effectiveness, we tested three types: **Cosine similarity** with dense embeddings, **BM25** with sparse retrieval, and a **Hybrid** approach. The Hybrid method uses RRF, consistent with the retriever setting of ours. For fairness, all naive baselines use the top-5 retrieved documents for answer generation.

Multi-Step Retrieval We employed **Iter-RetGen** (Shao et al., 2023), **Self-Ask** (Press et al., 2023), and **Self-RAG** (Asai et al., 2024), which are representative multi-step retrieval frameworks. The *Max-Iteration* for Iter-RetGen and the *Max-Hop* for Self-Ask were both fixed at 3, following prior study (Zhuang et al., 2024), while the maximum depth and beam width of Self-RAG were both fixed at 2. All three methods select the top-5 passages at each step based on retrieval scores, and their final evidence is consolidated into five unique passages for answer generation. To isolate the effect of the framework itself, we used RRF as the underlying retriever for all three methods, ensuring a fair comparison with TRA-RAG. These baselines are used to compare whether multi-step retrieval is more effective for multi-hop reasoning than graph-connectivity-based retrieval.

Structure-augmented Retrieval Structure-augmented baselines are divided into three groups: graph-based, summarization-based, and enriched-graph methods. The graph-based methods are **GraphRAG** (Edge et al., 2024), **LightRAG** (Guo et al., 2025b), **PathRAG** (Chen

et al., 2025a), and **HyperGraphRAG** (Luo et al., 2025). In particular, HyperGraphRAG uses hyperedges instead of triplets, allowing us to examine whether triplets are structurally optimal for retrieval. GraphRAG and LightRAG were run in *local search* mode, which is effective for retrieving specific details. PathRAG was included for its specialization in reasoning-path retrieval within graph structures. The summarization-based method is **RAPTOR** (Sarathi et al., 2024), which helps assess whether key contextual information is preserved or lost during summarization. The enriched-graph methods are **HippoRAG** (Gutiérrez et al., 2024), **HippoRAG2** (Gutiérrez et al., 2025), and **LinearRAG** (Zhuang et al., 2025). These methods enrich triplet-based graphs with sentence- or passage-level information and retrieve evidence through graph traversal. Together, these baselines provide strong comparisons for validating that the TRA Structure is retrieval-optimized and that a simple query-decomposition strategy can be highly effective for multi-hop reasoning.

While structure-augmented retrieval methods use a limited number of chunks together with additional metadata generated during indexing (e.g., community reports in GraphRAG and entity/relation descriptions in LightRAG and HyperGraphRAG), the Non-Retrieval and Naive Retrieval methods rely only on raw document information. Through these comparisons, we aim to show the importance of designing retrieval-optimized structures.

B.3 Metric

We employed token-level F1 and LLM-Acc as answer evaluation metrics for all six datasets. While F1 evaluates word-level overlap with the gold answer, LLM-Acc was adopted to capture semantic equivalence between the generated answer and the gold answer. For retrieval evaluation, Recall@5 indicates the extent to which the top-5 retrieved documents contain the dataset’s supporting documents. Hit-All@5 represents the proportion of questions for which all supporting documents were successfully retrieved and is reported only for the Multi-Hop QA datasets, because it is not meaningful for the Simple QA setting.

B.4 Implementation Details

For fairness, all methods were configured to use five passages for final answer generation. Structure-augmented retrieval methods and TRA-RAG addi-

tionally use indexing-time structural information (e.g., graph-derived metadata). Aside from this, all settings followed the original papers and official codebases to preserve each method’s default behavior. Across all models, we standardized the answer prompt by including the instruction: "if provided information is insufficient, answer with Insufficient Information," and set the temperature to 0 for all processes. In the PureLLM setting, this instruction was omitted, and the model answered using only its internal parametric knowledge. Most methods were implemented with their original configurations. Accordingly, all graph-based retrieval methods except HippoRAG2 used a chunk size of 1,200 tokens with a 100-token overlap. In contrast, HippoRAG2 used the same dataset-defined document (passage) units as TRA-RAG, Non-Retrieval, and Naive Retrieval to ensure consistent comparison. For fairness, we used *gpt-4o-mini* for all generations and *text-embedding-3-small* for embeddings.

C Statistics of TRA

The statistics for the Facts, Attributes, and Relations derived from the TRA Structures constructed for all six evaluation datasets are presented in Table 9.

Dataset	General Counts		TRA Facts					Attributes		Relations	
	# QA	# Docs	N Facts	U Keys	Avg. Tok.	Min Tok.	Max Tok.	N Values	U Keys	N Values	U Keys
NQ	1,000	9,633	90,299	34,244	23.72	4	404	155,296	78,519	76,687	1,042
PopQA	1,000	8,603	85,405	31,164	26.18	4	664	172,859	41,966	49,507	530
HotpotQA	1,000	9,811	107,254	30,051	20.11	5	733	158,746	60,597	81,217	538
MuSiQue	1,000	9,838	108,442	32,966	20.56	4	705	283,138	71,378	147,832	799
2Wiki	1,000	6,119	62,780	13,269	19.50	5	556	122,196	24,932	51,412	274
LVEVAL	124	22,849	229,722	73,780	28.70	4	1,326	525,126	209,430	193,167	1,624

Table 9: Detailed statistics of the TRA Structure across all datasets. We report the scale of extracted Facts, Attributes, and Relations. N denotes the total number of items, and U denotes the number of unique items. In the TRA Facts columns, “Avg. Tok.”, “Min Tok.”, and “Max Tok.” denote the average, minimum, and maximum token lengths of indexed Facts, computed using the o200k_base tokenizer from OpenAI’s tiktoken. Attribute and Relation values are counted before their integration into Facts. Together, these metrics reflect the diversity and granularity of information captured by the TRA Structure.

D Additional Analysis

While our primary objective was to secure strong retrieval and multi-hop reasoning capabilities, practical RAG frameworks also require **large-scale robustness, cost-efficiency, serialization stability, and low hyperparameter sensitivity**. This section examines TRA-RAG across these four aspects.

D.1 Large Scale Robustness Analysis

First, we verify the robustness of our model in a large-scale setting. Since TRA-RAG constructs the corpus into a retrieval-friendly TRA structure and utilizes a retrieval mechanism similar to naive approaches, we selected **Hybrid Search** as the comparative baseline. To artificially scale up the corpus, we collected 50,000 unique passages from the 2WikiMultiHopQA dataset and performed TRA indexing. For evaluation, we used the same 1,000 QA pairs as in the main results.

Experimental results shown in Table 10 indicate that both the version with and without QD exhibited no significant performance degradation at this scale. In contrast, while the Hybrid baseline showed a distinct drop in answer quality, TRA-RAG demonstrated a much smaller decline in both retrieval and answering performance. This confirms that the TRA-RAG retrieval mechanism remains effective even as the corpus size increases.

Method	Original (~6k)		Large Scale (50k)	
	F1	Rec@5	F1	Rec@5
Hybrid Retrieval	39.3	67.8	33.7	63.8
Hybrid + QD	58.7	97.0	56.9	94.9
TRA-RAG (w/o QD)	48.0	71.2	46.6	70.7
TRA-RAG (Full)	70.7	97.5	70.3	95.9

Table 10: Robustness analysis on corpus size scaling. We compare the performance stability when scaling the corpus from the original size (about 6k) to a large-scale setting (50k) in 2WikiMultiHopQA.

D.2 Cost-Efficiency Analysis

To ensure a fair assessment of construction costs, we calculated the **database indexing cost** and **retrieval process cost** compared to several baselines (Table 11).

Indexing Cost We used the default settings from the official codebases of three graph-augmented baselines: LightRAG, HyperGraphRAG, and HippoRAG2. LightRAG and HyperGraphRAG employ a “gleaning” process, iteratively extracting entities and relationships. Since the output is fed back as input for the next step, this incurs additional token costs (α). Specifically, HyperGraphRAG performs gleaning twice, resulting in a total cost of approximately $3 \times \text{Tokens} + \alpha$. HippoRAG2 requires a two-step process (NER followed by relationship extraction), costing $2 \times \text{Tokens} + \alpha$. In contrast, TRA-RAG completes TRA construction in a **single pass** ($1 \times \text{Tokens}$), significantly reducing indexing token costs and maintaining high efficiency.

Retrieval Cost We measured the resources required for a single retrieval session. “Time” records only the pure retrieval latency, while “LLM Calls”

Method	Indexing		Retrieval	
	Tokens	Calls	Time (s)	LLM Calls
Hybrid	-	-	0.55	1 (QA)
LightRAG	$2 \times \text{Tokens} + \alpha$	$2 \times \text{Docs}$	2.32	2 (Ext. + QA)
HyperGraphRAG	$3 \times \text{Tokens} + \alpha$	$3 \times \text{Docs}$	5.30	2 (Ext. + QA)
HippoRAG2	$2 \times \text{Tokens} + \alpha$	$2 \times \text{Docs}$	2.67	2 (Rerank + QA)
Ours (w/o QD)	$1 \times \text{Tokens}$	$1 \times \text{Docs}$	0.74	1 (QA)

Table 11: Efficiency Analysis. We compare indexing costs (input tokens, API calls) and retrieval costs (retrieval time and LLM calls per query). ‘Tokens’ refers to the total corpus tokens, and ‘Docs’ refers to the number of documents. ‘Ext.’ denotes Entity Extraction.

includes the final QA step. For a fair comparison of the retrieval mechanism itself, we recorded the cost of TRA-RAG *without* QD; this setting also omits sequential sub-question answering and reranking. While graph-augmented baselines incur additional LLM calls for tasks such as entity extraction or triplet reranking during retrieval, TRA-RAG uses dense and sparse retrieval with RRF and selects source passages based on the resulting Fact scores, requiring only a single LLM call for final answer generation. It therefore remains significantly faster than graph-based RAG approaches. We note that the full TRA-RAG framework applies QD in the Multi-Hop QA setting, which naturally increases the cost during actual multi-hop QA.

Across the following sensitivity analyses, the model, QD and sub-question-answering outputs, final answer prompt, and top-5 passage budget are held fixed; only the variable under evaluation is changed.

D.3 Serialization Sensitivity

We examine whether the template used to serialize TRA Facts affects retrieval and downstream answering performance. The Natural setting uses the natural-sentence serialization described in Section 3.2, which is also used in the main experiments. As shown in Table 12, retrieval is largely insensitive to the serialization format, with Recall@5 varying by only 0.1 points across the tested settings. Key-value achieves the highest retrieval scores, while Natural achieves the best F1 and LLM-Acc.

Overall, TRA-RAG shows low sensitivity to Fact serialization, with the Natural representation providing the strongest downstream answering performance among the tested formats.

Serialization	Recall@5	Hit-All@5	F1	Acc
Natural	97.7	95.0	71.0	72.9
Key-value	97.8	95.1	69.9	71.7
JSON	97.7	94.8	70.3	72.3
Triple	97.7	94.9	70.3	72.4

Table 12: Sensitivity to fact serialization on 2WikiMultiHopQA.

Variable	Setting	Recall@5	Hit-All@5	F1	Acc
RRF k	10	97.6	94.7	71.5	73.6
	60 (default)	97.7	95.0	71.0	72.9
	100	97.8	95.2	70.7	72.6
Path pool	20	97.8	95.1	70.8	72.9
	50 (default)	97.7	95.0	71.0	72.9
	100	97.5	94.4	70.8	72.7
Final rerank	0 rerank	97.4	94.4	70.7	72.6
	+ 5 seed				
	2 rerank	97.7	95.0	71.0	72.9
	+ 3 seed				
	4 rerank	97.5	94.2	70.6	72.8
	+ 1 seed				

Table 13: Sensitivity to RRF and retrieval budget settings on 2WikiMultiHopQA.

D.4 RRF and Reranking-Budget Sensitivity

We further evaluate sensitivity to the RRF constant, SQ path-pool size, and final seed–reranked passage allocation. The default configuration uses $k = 60$, an SQ path pool of 50, and a final passage mix of three seed passages and two reranked passages.

Table 13 shows that TRA-RAG is relatively robust to these hyperparameters, with only small variations in retrieval and downstream QA performance. While $k = 10$ achieves the highest F1 and Acc, $k = 100$ slightly improves retrieval. Changes to the SQ path pool and final reranking budget also produce only minor differences.

Overall, TRA-RAG shows low sensitivity to the tested RRF and reranking-budget settings.

E Case Study

E.1 TRA Examples

Input Document

“The Goat and Her Three Kids” or “The Goat with Three Kids” (Romanian: “Capra cu trei iezi”) is an 1875 short story, fable and fairy tale by Romanian author Ion Creangă. Figuratively illustrating for the notions of motherly love and childish disobedience, it recounts how a family of goats is ravaged by the Big Bad Wolf, allowed inside the secured home by the oldest, most ill-behaved and least prudent of the kids. The only one of the children to survive is the youngest and most obedient, who then helps his mother plan her revenge on the predator, leading to a denouement in which the wolf is tricked, burned alive and stoned to death.

Output: TRA Structure

```
{
  "title": "The Goat and Her Three Kids",
  "attributes": {
    "alternative_title": "The Goat with Three Kids",
    "original_title": {
      "language": "Romanian",
      "text": "Capra cu trei iezi"
    },
    "publication_year": 1875,
    "genre": ["short story", "fable", "fairy tale"],
    "author": {
      "name": "Ion Creangă",
      "nationality": "Romanian"
    },
    "themes": ["motherly love", "childish disobedience"],
    "plot_summary": {
      "description": "It recounts how a family of goats is ravaged by the Big Bad Wolf, allowed inside the  

        ↪ secured home by the oldest, most ill-behaved and least prudent of the kids.",
      "survivor": {
        "child": "youngest",
        "characteristics": ["most obedient"]
      },
      "revenge": {
        "action": "helps his mother plan her revenge on the predator",
        "denouement": {
          "events": [
            "the wolf is tricked",
            "burned alive",
            "stoned to death"
          ]
        }
      }
    },
    "relations": [
      {
        "relation": "written_by", "target": "Ion Creangă"
      },
      {
        "relation": "illustrates", "target": ["motherly love", "childish disobedience"]
      },
      {
        "relation": "features", "target": "Big Bad Wolf"
      },
      {
        "relation": "includes", "target": ["family of goats", "three kids"]
      }
    ]
  }
}
```

Figure 9: Full Example of TRA Structure Construction. The unstructured input text (Top) is transformed into a comprehensive JSON object (Bottom). All narrative details, including the complex ‘plot_summary’ and ‘denouement’, are fully preserved in the structured output.

Output: Indexed Facts (for Retrieval)

```
{
  "fact_count": 11,
  "facts": [
    {
      "text": "The alternative title of The Goat and Her Three Kids is The Goat with Three Kids",
      "key_path": "alternative_title",
      "value": "The Goat with Three Kids",
      "title": "The Goat and Her Three Kids"
    },
    {
      "text": "The original title of The Goat and Her Three Kids is language is Romanian, text is Capra cu  
    ↪ trei iezi",
      "key_path": "original_title",
      "value": "{\language\": \"Romanian\", \"text\": \"Capra cu trei iezi\"}",
      "title": "The Goat and Her Three Kids"
    },
    {
      "text": "The publication year of The Goat and Her Three Kids is 1875",
      "key_path": "publication_year",
      "value": "1875",
      "title": "The Goat and Her Three Kids"
    },
    {
      "text": "The genre of The Goat and Her Three Kids is short story, fable, and fairy tale",
      "key_path": "genre",
      "value": "[\"short story\", \"fable\", \"fairy tale\"]",
      "title": "The Goat and Her Three Kids"
    },
    {
      "text": "The author of The Goat and Her Three Kids is name is Ion Creangă, nationality is Romanian",
      "key_path": "author",
      "value": "{\name\": \"Ion Creangă\", \"nationality\": \"Romanian\"}",
      "title": "The Goat and Her Three Kids"
    },
    {
      "text": "The themes of The Goat and Her Three Kids is motherly love and childish disobedience",
      "key_path": "themes",
      "value": "[\"motherly love\", \"childish disobedience\"]",
      "title": "The Goat and Her Three Kids"
    },
    {
      "text": "The plot summary of The Goat and Her Three Kids is description is It recounts how a family  
    ↪ of goats is ravaged by the Big Bad Wolf, allowed inside the secured home by the oldest, most  
    ↪ ill-behaved and least prudent of the kids., survivor is child is youngest, characteristics  
    ↪ is most obedient, revenge is action is helps his mother plan her revenge on the predator,  
    ↪ denouement is events is the wolf is tricked, burned alive, and stoned to death",
      "key_path": "plot_summary",
      "value": "{\description\": \"It recounts how a family of goats is ravaged by the Big Bad Wolf,  
    ↪ allowed inside the secured home by the oldest, most ill-behaved and least prudent of the  
    ↪ kids.\", \"survivor\": {\child\": \"youngest\", \"characteristics\": [\"most obedient\"]},  
    ↪ \"revenge\": {\action\": \"helps his mother plan her revenge on the predator\", \"  
    ↪ denouement\": {\events\": [\"the wolf is tricked\", \"burned alive\", \"stoned to death  
    ↪ \"]}}",
      "title": "The Goat and Her Three Kids"
    },
    {
      "text": "The Goat and Her Three Kids is written by Ion Creangă",
      "key_path": "written_by",
      "value": "{\target\": \"Ion Creangă\"}",
      "title": "The Goat and Her Three Kids"
    },
    {
      "text": "The Goat and Her Three Kids is illustrates motherly love and childish disobedience",
      "key_path": "illustrates",
      "value": "{\target\": [\"motherly love\", \"childish disobedience\"]}",
      "title": "The Goat and Her Three Kids"
    },
    {
      "text": "The Goat and Her Three Kids is features Big Bad Wolf",
      "key_path": "features",
      "value": "{\target\": \"Big Bad Wolf\"}",
      "title": "The Goat and Her Three Kids"
    },
    {
      "text": "The Goat and Her Three Kids is includes family of goats and three kids",
      "key_path": "includes",
      "value": "{\target\": [\"family of goats\", \"three kids\"]}",
      "title": "The Goat and Her Three Kids"
    }
  ]
}
```

Figure 10: Example of retrieval-ready Fact indexing derived from the TRA structure in Figure 9. We show the full set of indexed facts, including ‘text’, ‘key_path’, ‘value’, and ‘title’ fields. In TRA-RAG, the actual retrieval target is the ‘text’ field.

E.2 Compact Structure Comparison Examples

Example 1: Colin Reader

1. Input Corpus

Colin Reader suggested, based on weathering evidence, that the Sphinx is older than its commonly accepted 4th dynasty date and instead dates to the early dynastic or late predynastic period.

2. Baseline: Triplets

```
(Sphinx, has_date, 4th dynastic date)
(Sphinx, is_older_than, 4th dynastic date)
(Sphinx, has_date, early dynastic period)
(Sphinx, has_date, late predynastic period)
```

3. TRA Structure

```
"studies": {
  "subject": "Sphinx",
  "suggestion": {
    "basis": "weathering evidence",
    "claim": "older than its commonly accepted 4th dynasty date",
    "alternative_date": "early dynastic or late predynastic period",
    "related_structures": ["Sphinx", "two other buildings on the Giza Plateau"]
  }
}
```

4. Indexed Fact

The studies of Colin Reader is subject is Sphinx, contribution is controversial debate regarding a possible older dating of the monument, suggestion is basis is weathering evidence, claim is the Sphinx is older than its commonly accepted 4th dynastic date, alternative date is early dynastic or late predynastic period, related structures is Sphinx and two other buildings on the Giza Plateau

Example 2: Banco de Ponce

1. Input Corpus

It headquartered in San Juan in 1970 and, 20 years later, merged with Banco Popular, forming BancPonce Corporation.

2. Baseline: Triplets

```
(Banco de Ponce, headquartered_in, San Juan)
(Banco de Ponce, has_date, 1970)
(Banco de Ponce, merged_with, Banco Popular)
(Banco de Ponce, formed, BancPonce Corporation)
(Banco de Ponce, has_date, 20 years after 1970)
```

3. TRA Structure

```
"headquarters": {
  "location": "San Juan, Puerto Rico",
  "year": 1970
},
"merger": {
  "with": "Banco Popular",
  "year": 1990,
  "new_entity": "BancPonce Corporation"
}
```

4. Indexed Facts

The headquarters of Banco de Ponce is location is San Juan, Puerto Rico, year is 1970
The merger of Banco de Ponce is with is Banco Popular, year is 1990, new entity is BancPonce Corporation

Figure 11: Compact examples comparing Triplet and TRA representations of the same source text.

E.3 Detailed Structure Comparison Example

1. Input Corpus

Gundobad (452 – 516 AD) was King of the Burgundians (473 – 516), succeeding his father Gundioc of Burgundy. Previous to this, he had been a Patrician of the moribund Western Roman Empire in 472 – 473, three years before its collapse, succeeding his uncle Ricimer. He is perhaps best known today as the probable issuer of the “Lex Burgundionum” legal codes, which synthesized Roman law with ancient Germanic customs. He was the husband of Caretene.

2. Baseline A: Triplets

```
[
  ["Gundobad", "has_date", "452 - 516 AD"],
  ["Gundobad", "served_as", "King of the Burgundians"],
  ["Gundobad", "has_date", "473 - 516"],
  ["Gundobad", "succeeded", "Gundioc of Burgundy"],
  ["Gundobad", "served_as", "Patrician of the Western Roman Empire"],
  ["Gundobad", "has_date", "472 - 473"],
  ["Gundobad", "succeeded", "Ricimer"],
  ["Gundobad", "issued", "Lex Burgundionum"],
  ["Lex Burgundionum", "synthesized", "Roman law"],
  ["Lex Burgundionum", "synthesized", "ancient Germanic customs"],
  ["Gundobad", "has_role", "husband"],
  ["Gundobad", "has_spouse", "Caretene"]
]
```

3. Baseline B: HyperEdges

```
[
  {
    "edge_id": 1,
    "relation_type": "Succession",
    "participants": { "actor": "Gundobad", "object": "Gundioc", "recipient": "Burgundians" },
    "attributes": { "temporal": "516 AD", "spatial": "Burgundy", "modality": "succeeded" } },
  {
    "edge_id": 2,
    "relation_type": "Patrician Appointment",
    "participants": { "actor": "Gundobad", "object": "Western Roman Empire", "recipient": "Ricimer" },
    "attributes": { "temporal": "472 - 473", "spatial": "Western Roman Empire", "modality": "succeeded" } },
  {
    "edge_id": 3,
    "relation_type": "Issuance of Legal Codes",
    "participants": { "actor": "Gundobad", "object": "Lex Burgundionum", "recipient": "Burgundians" },
    "attributes": { "temporal": "Unknown", "spatial": "Burgundy", "modality": "issuer" } }
]
```

4. TRA Structure

```
{
  "title": "Gundobad",
  "attributes": {
    "full_name": "Gundobad",
    "birth_year": 452,
    "death_year": 516,
    "reign": { "title": "King of the Burgundians", "years": "473 - 516" },
    "predecessor": { "title": "King of the Burgundians", "name": "Gundioc of Burgundy" },
    "previous_position": {
      "title": "Patrician",
      "organization": "Western Roman Empire",
      "years": "472 - 473"
    },
    "successor": { "title": "Patrician", "name": "Ricimer" },
    "notable_contribution": {
      "title": "Lex Burgundionum",
      "description": "Legal codes that synthesized Roman law with ancient Germanic customs"
    },
    "spouse": { "name": "Caretene" }
  },
  "relations": [
    { "relation": "succeeded", "target": "Gundioc of Burgundy" },
    { "relation": "preceeded_by", "target": "Ricimer" },
    { "relation": "husband_of", "target": "Caretene" },
    { "relation": "issuer_of", "target": "Lex Burgundionum" }
  ]
}
```

Figure 12: Qualitative Comparison of extracted structures from the same corpus. We present the outputs of Baseline A (Triplets), Baseline B (HyperEdges), and TRA Structure.

Criteria	Candidate 1 (TRA)	Candidate 2 (HyperEdge)
<i>Quantitative Scores (1-5 Scale)</i>		
Completeness	5	4
Faithfulness	5	4
Structural Clarity	5	4
Atomic Granularity	5	4
Average Score	5.0	4.0
<i>Qualitative Justification (Generated by Judge)</i>		
Reasoning	"Candidate 1 captures all key facts and relations accurately, with clear structure and atomic facts."	"Candidate 2 misses some details, such as Gundobad's birth and death years, and has less clarity in its relations."
Final Verdict	WINNER: Candidate 1 (TRA)	

Figure 13: Example of the LLM-based Evaluation Output. The Judge (GPT-4o-mini) assesses both extracted structures based on four criteria. In this case, the TRA structure (Candidate 1) achieves a perfect score due to its ability to capture atomic details like birth/death years, which the HyperEdge extraction (Candidate 2) missed or represented less clearly.

E.4 Failure Cases

The failure cases of TRA-RAG can be summarized into two main categories: 1) **QD Failure** and 2) **Failure Chain from Inappropriate TRA Construction**.

The first case occurs when the LLM misinterprets the context of the question, resulting in a QD structure from which the correct answer cannot be derived. This issue can be attributed less to a problem with the TRA structure and more to the fact that TRA-RAG opted for a very traditional decomposition method. Therefore, it can be resolved by using a more sophisticated approach. The second case arises when the TRA structure inadequately represents the essential hop information required for multi-hop reasoning, leading to a failure in retrieving the corresponding evidence during the retrieval process. This case occurs particularly often in passages with very ambiguous content, and it appears that it can be resolved through more sophisticated prompt engineering or by using a better LLM.

We provide specific examples for each of these two scenarios in Figure 14 and Figure 15, respectively.

F Future Work: Expandability as a graph

Although we proposed TRA structure primarily as a method utilizing a naive retrieval strategy, this structure possesses significant potential to be extended into a graph-based representation.

We explored two potential scenarios for this expansion: 1) a configuration where only **Titles** serve as nodes, with edges established based on **shared**

values, and 2) a configuration where both **Titles and values from Relations and Attributes** serve as nodes, with edges connecting a Title (Entity) node to a Value node if the value belongs to the corresponding TRA.

Such graph structures enable the application of retrieval techniques from Graph-based RAGs, offering the possibility to leverage the unique advantages of the TRA Structure while mitigating the structural limitations of traditional triplets. We present the visualization of these proposed structures in Figure 16.

Complex Query: <i>"What is the manufacturer of the Kentucky Tavern headquarters location saints symbol called?"</i>	
Gold Decomposition (Reasoning Chain)	
Step 1	Q: Who is the manufacturer of Kentucky Tavern? $\hookrightarrow$ A: Sazerac Company
Step 2	Q: What is the headquarters location of Sazerac Company? $\hookrightarrow$ A: New Orleans
Step 3	Q: What is the saints symbol of New Orleans called? $\hookrightarrow$ A: fleur-de-lis
Model Prediction (Failure Case)	
Step 1	Q: What is the headquarters location of Kentucky Tavern ? $\hookrightarrow$ A: Insufficient Information
Error Analysis: <i>The model missed the "manufacturer" hop and erroneously attempted to link the product directly to a location.</i>	

Figure 14: A failure case study in QD. The model fails to recognize the dependency structure.

Query: <i>"Where is Sazerac Company's headquarters location?"</i>
Failure Chain: TRA Construction $\rightarrow$ Retrieval $\rightarrow$ Missed Evidence The target entity information was structured as follows: <pre>{ "title": "Sazerac", "attributes": { "description": "A local New Orleans variation of a cognac or whiskey cocktail", } }</pre>
Construction Error During TRA construction, the key clue ("New Orleans" as headquarters location) was stored inside a generic 'description' field instead of being mapped to an explicit 'headquarters'-related key.
Similarity Effect In retrieval, the query asks for a 'headquarters' attribute. Because the relevant clue is embedded in unstructured descriptive text, the similarity score becomes low for this query intent.
Ranking Consequence With low similarity, this entity falls below the Top- K cutoff and is not selected as supporting evidence.
Final Outcome "Insufficient Information"

Figure 15: Failure Case Analysis. Structural mismatch caused the retriever to miss the relevant passage.

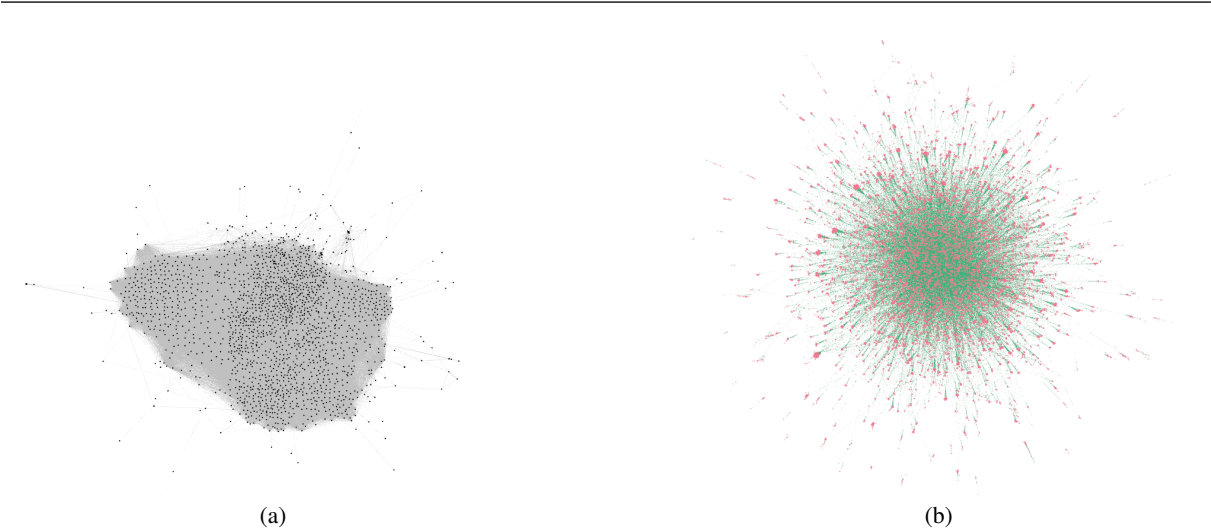


Figure 16: Graph visualization illustrating the extensibility of the TRA structure. (a) Nodes are represented as individual documents, with edges created when nodes share a common value in corresponding TRA. (b) A representation where nodes contain values, including entities. The graph layout is generated using ForceAtlas2.