# TASR-RAG: Taxonomy-guided Structured Reasoning for Retrieval-Augmented Generation

Anonymous ACL submission


## Abstract

Retrieval-Augmented Generation (RAG) helps large language models answer knowledge-intensive questions with external evidence, but existing systems often struggle with multi-hop queries over unstructured chunks. We argue that both information needs and factual evidence can be represented as relational triples $(h, r, t)$, turning retrieval into triple matching and multi-hop reasoning into chained matching with intermediate variables. The key challenge is matching evidence at the right semantic abstraction level: coarse abstractions improve recall but add ambiguity, while fine abstractions improve precision but suffer from sparsity and typing noise. We propose TASR-RAG, a taxonomy-guided framework for abstraction-aware retrieval and reasoning with typed triples. TASR-RAG projects query and document triples into a lightweight two-level type space, enabling hybrid matching over semantic similarity and structural compatibility. For complex questions, it decomposes the query into ordered triple sub-queries, performs sequential evidence selection with latent-variable binding, and refines intermediate entities without graph construction or exhaustive search. Experiments on multi-hop question answering benchmarks show that TASR-RAG outperforms strong RAG and structured-RAG baselines, while improving evidence matching, attribution, and reasoning traces.


## 1 Introduction

Large language models (LLMs) have demonstrated strong performance on knowledge-intensive tasks, yet their parametric knowledge alone remains insufficient for answering factual, long-tail, or time-sensitive questions (Petroni et al., 2021; Li et al., 2025b). Retrieval-Augmented Generation (RAG) addresses this limitation by retrieving external documents and conditioning generation on the retrieved evidence (Ram et al., 2023; Lewis et al., 2020). However, complex questions often require multi-hop reasoning: relevant evidence may be distributed across documents, intermediate entities must be resolved before later evidence can be retrieved, and the final answer depends on preserving these dependencies under limited context budgets. In such settings, the key challenge is not only whether relevant documents are retrieved, but whether evidence is selected and organized in the order required for reasoning.

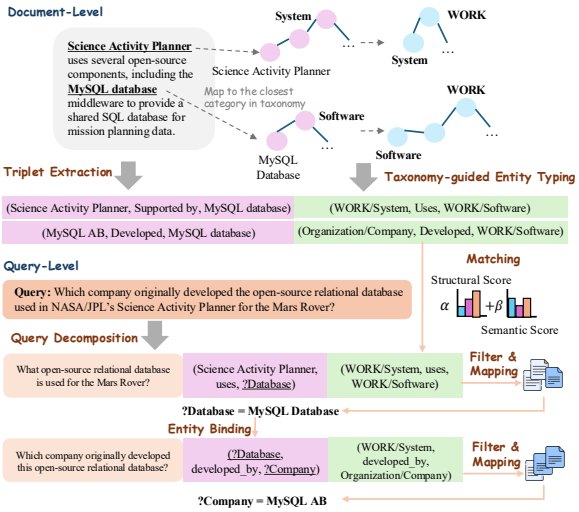



Figure 1: Overview of TASR-RAG. TASR-RAG reformulates multi-hop retrieval as abstraction-aware triple matching. Retrieved evidence and queries are represented as relational triples and further projected into taxonomy-guided typed abstraction spaces, enabling hybrid matching over both semantic similarity and structural compatibility. The framework then performs sequential latent-variable binding to progressively refine intermediate entities across reasoning hops.

Existing RAG systems only partially address this challenge. Standard dense-retrieval pipelines usually operate over unstructured text chunks and use one-shot prompting, which can introduce redundant context, low information density, and entity conflation (Yu et al., 2024; Shi et al., 2024). Recent structured RAG methods incorporate knowl-

edge graphs or extracted triples to improve factual grounding (Edge et al., 2024; Shi et al., 2025; Luo et al., 2025; Gutierrez et al., 2024; Li et al., 2025c; Wu et al., 2025), but graph construction can be costly and noisy, while extracted structures may be sparse or misaligned with the query's reasoning chain. More importantly, many methods either rely mainly on semantic relevance or commit to a single level of structural abstraction, making it difficult to balance recall and precision across reasoning hops.

Our motivation is that both information needs and factual evidence can be represented as relational triples $(h, r, t)$. Under this view, retrieval becomes triple matching, and multi-hop reasoning becomes chained matching with explicit variable binding. However, raw triples are often too brittle for multi-hop retrieval: semantically aligned evidence may fail to match because entities or relations differ lexically across documents. A natural solution is to introduce structural abstraction through entity typing, but matching at a single abstraction level creates a fundamental precision–recall trade-off. Overly coarse types improve recall but introduce ambiguity and entity conflation, while overly fine types improve precision at the cost of reduced recall due to sparsity and typing errors.

To address this problem, we introduce typed triples as a controllable semantic abstraction interface for multi-hop retrieval and reasoning. Instead of matching only raw triples, we additionally project them into taxonomy-guided typed triples $(\tau(h), r, \tau(t))$, enabling retrieval to operate jointly over semantic similarity and structural compatibility. We adopt a lightweight two-level taxonomy that balances broad semantic compatibility with fine-grained disambiguation, allowing more stable evidence matching and intermediate variable resolution across reasoning hops.

Based on this motivation, we propose **TaSR-RAG**, a **Ta**xonomy-guided **S**tructured **R**easoning framework for multi-hop RAG. The core idea of TASR-RAG is to perform retrieval and reasoning jointly in a taxonomy-guided abstraction space rather than relying solely on raw semantic matching. As shown in Figure 1, TASR-RAG represents both retrieved evidence and user queries as relational triples, and further projects them into typed triples using a lightweight two-level taxonomy. This abstraction-aware representation enables retrieval to balance semantic flexibility and structural precision across reasoning hops.

Built on top of this representation, TASR-RAG decomposes complex questions into ordered triple sub-queries with latent variables, performs hybrid matching over both raw and typed triples, and maintains explicit entity bindings across hops. For example, after resolving that *Science Activity Planner* uses *MySQL*, the next retrieval step becomes conditioned on the resolved intermediate entity *MySQL*. This design produces more stable multi-hop evidence matching and interpretable reasoning traces without requiring explicit graph construction or exhaustive graph search.

Our contributions are threefold. First, we reframe multi-hop RAG as abstraction-aware triple matching and introduce taxonomy-guided typed triples as a controllable semantic abstraction interface for both queries and documents. Second, we propose a hybrid matching mechanism that combines semantic similarity over raw triples with structural compatibility over typed triples, balancing recall and precision across reasoning hops. Third, we develop a sequential retrieval and reasoning framework with explicit latent-variable binding, turning multi-hop QA into an interpretable chain of abstraction-guided evidence matching. Experiments on multiple multi-hop QA benchmarks show that TASR-RAG consistently outperforms strong RAG and structured-RAG baselines while remaining training-free and compatible with existing dense retrievers and LLMs.

## 2 Method

### 2.1 Design Motivation

The key insight behind TASR-RAG is that complex multi-hop queries are naturally cast as chained matches between query triples and evidence triples, yet standard RAG relies on flat query–document similarity. Our design follows four principles.

**(1) Typed structural abstraction.** Typed triples provide a controllable abstraction interface: raw triples preserve semantic specificity, while taxonomy-guided types indicate whether two triples play compatible structural roles.

**(2) Decomposition for multi-hop reasoning.** Complex queries often involve intermediate answers that serve as inputs to later hops. We decompose each query into ordered triple sub-queries with explicit latent variables, so each hop can be matched and verified independently.

**(3) Multi-level matching.** A single granularity is insufficient for robust evidence selection. We use a

lightweight two-level taxonomy to combine broad type compatibility with finer structural cues, and integrate this typed signal with semantic similarity over raw triples.

**(4) Coupled evidence selection and reasoning.** Rather than selecting all context at once, we iteratively resolve latent variables and refine evidence selection. Each resolved binding constrains the next hop, producing an auditable trace without materializing a global graph.

### 2.2 Overview

We propose **TaSR-RAG** (**Ta**xonomy-guided **S**tructured **R**easoning for RAG), which (i) converts retrieved documents into relational and taxonomy-typed triples, (ii) decomposes the query into ordered typed triple sub-queries with latent variables, and (iii) traverses sub-queries left to right, reranking evidence and resolving variables at each hop.

Let $\mathcal{D}$ denote the document collection, and let $\mathcal{T} = \{\mathcal{T}^{(1)}, \mathcal{T}^{(2)}\}$ denote a two-level taxonomy. $\mathcal{T}^{(1)} = \{c_1^{(1)}, \ldots, c_L^{(1)}\}$ are first-level classes and $\mathcal{T}^{(2)}(c^{(1)}) = \{c_1^{(2)}, \ldots, c_M^{(2)}\}$ are second-level classes under $c^{(1)}$. We build $\mathcal{T}$ based on Schema.org [1] and adopt a two-level hierarchy to balance coverage and discriminability.

The pipeline consists of four stages:

1. **Initial Retrieval**: retrieve top-$K_0$ candidate documents via dense retrieval.
2. **Triple Extraction and Typing (Document-level)**: for each retrieved document, extract relational triples $\mathcal{R}_d$ and construct typed triples $\tilde{\mathcal{R}}_d$ by applying taxonomy-guided entity typing to the head and tail entities.
3. **Query Decomposition and Typing (Query-level)**: decompose $q$ into an ordered list of sub-queries $\{s_i\}_{i=1}^N$ (triples with latent variables) and construct their typed forms $\{\tilde{s}_i\}_{i=1}^N$.
4. **Structured Reasoning with Entity Binding**: resolve already-bound variables, rerank/filter documents with hybrid matching, answer the current sub-query, and bind its latent variable for subsequent hops.

**Running example.** For "*Which company originally developed the open-source relational database used in NASA/JPL's Science Activity Planner for the Mars Rover?*", TASR-RAG decomposes the query into (Science Activity Planner, `uses`, ?Database) and (?Database, `developed_by`, ?Company). Document triples such as (Science Activity Planner, `uses`, MySQL database) are also projected into typed forms, allowing hybrid matching to combine lexical evidence with type compatibility. The first hop binds ?Database to *MySQL database*; the second hop substitutes this value and resolves ?Company as *MySQL AB*. Full details are in Appendix G.

### 2.3 Initial Retrieval

Given query $q$, we first retrieve top-$K_0$ candidate documents via dense retrieval. This step is shared by all methods and serves as a common preprocessing stage for fair comparison:

$$\mathcal{D}_{\text{cand}} = \underset{d \in \mathcal{D}}{\text{Top-}K_0} \cos\big(f_{\text{enc}}(q), f_{\text{enc}}(d)\big) \quad (1)$$

### 2.4 Triple Extraction and Typing (Document-level)

#### 2.4.1 Triple Extraction

For each candidate document $d \in \mathcal{D}_{\text{cand}}$, we employ an LLM to extract relational triples:

$$\mathcal{R}_d = \text{LLM}_{\text{extract}}(d) = \{(h, r, t)_1, \ldots, (h, r, t)_{|\mathcal{R}_d|}\} \quad (2)$$

**Typed Triple Construction.** Each raw triple $(h, r, t)$ is projected into a typed triple $(\tau(h), r, \tau(t))$, where $\tau(\cdot)$ maps entities into the taxonomy:

$$\tilde{\mathcal{R}}_d = \big\{(\tau(h), r, \tau(t)) : (h, r, t) \in \mathcal{R}_d\big\} \quad (3)$$

Typed triples provide abstraction-level structural alignment, while raw triples preserve semantic specificity. The two representations are therefore complementary during downstream matching: raw triples support flexible semantic comparison, and typed triples reduce entity conflation by enforcing type-level compatibility. Relations $r$ retain their surface forms because relation abstraction often reduces semantic sharpness and harms matching precision.

We perform query-time extraction so the LLM focuses on relations relevant to the current query rather than exhaustively extracting all possible triples. We compare this design with query-agnostic pre-extraction in Section 3.4.

When extracting relations, we instruct the LLM to avoid both overly generic predicates (e.g., *related to*) and overly specific paraphrases that rarely

[1] Schema.org

match across documents. This produces triples that are compact enough for comparison while still preserving the relation semantics needed for multi-hop evidence selection.

#### 2.4.2 Taxonomy-Guided Entity Typing

For each entity $e$ from document or query triples, an LLM assigns a hierarchical type $\tau(e) = (c^{(1)}, c^{(2)})$ in two stages:

*Stage 1: First-Level LLM Selection.* The LLM selects the best first-level class from $\mathcal{T}^{(1)}$:

$$c^{(1)} = \text{LLM}_{\text{select}}\big(\mathcal{T}^{(1)}, e\big) \tag{4}$$

*Stage 2: Second-Level LLM Selection.* Conditioned on $c^{(1)}$, the LLM selects the best second-level class from its children:

$$c^{(2)} = \text{LLM}_{\text{select}}\big(\mathcal{T}^{(2)}(c^{(1)}), e\big) \tag{5}$$

The final type $\tau(e) = (c^{(1)}, c^{(2)})$ is also inferred for query-level latent variables and used as a matching constraint. We use two levels because the two ends of the hierarchy serve different retrieval needs. The first level captures broad compatibility (e.g., whether an entity is a person, organization, location, or work), which improves recall when surface forms differ. The second level adds finer distinctions that help disambiguate competing candidates during multi-hop matching. Deeper taxonomies can become sparse and sensitive to typing errors, while a single level is often too coarse; this trade-off is evaluated empirically in Section 3.4.

### 2.5 Query Decomposition and Typing (Query-level)

After constructing document triples, we decompose query $q$ into ordered sub-queries that expose intermediate unknowns and guide step-wise reranking:

$$\{s_1, s_2, \ldots, s_N\} = \text{LLM}_{\text{decomp}}(q) \tag{6}$$

where each sub-query $s_i$ is represented as a relational triple:

$$s_i = (h_i, r_i, t_i) \tag{7}$$

Here $h_i$ and $t_i$ may be concrete entities or latent variables (prefix "?") to be resolved during reasoning. Because downstream matching uses both raw and taxonomy-typed triples, the decomposition prompt also infers type constraints for explicit entities and latent variables.

**Query-level Typed Triples.** We then obtain the typed form

$$\tilde{s}_i = (\tau(h_i), r_i, \tau(t_i)). \tag{8}$$

These typed sub-queries provide type-level constraints for downstream matching and complement the document-level typed triples above. We analyze decomposition errors in Appendix B.1.

This query-level structure is important because it determines the order in which evidence is retrieved. A later sub-query may contain a latent variable from an earlier one; after the variable is resolved, the next retrieval step becomes more specific and less prone to topic drift. Thus, decomposition is not only a parsing step, but also defines the control flow for sequential evidence selection.

### 2.6 Structured Reasoning with Entity Binding

#### 2.6.1 Hybrid Triple Matching for Document Reranking

For each resolved sub-query $s'_i$ and its typed form $\tilde{s}'_i$, we score each candidate document $d$ by matching the query triple against the document's extracted triples and matching the typed query triple against the document's typed triples. We then combine the two signals into a single score used to rerank documents for the current reasoning step.

**Structural Score.** The structural score measures type-level compatibility between the query triple and document triples. For a query-document triple pair, we compute:

$$\begin{aligned} S_{\text{type}}(\tau_q, \tau_d) = w^{(1)} \cdot \mathbb{1}[c_q^{(1)} = c_d^{(1)}] \\ + w^{(2)} \cdot \mathbb{1}[c_q^{(2)} = c_d^{(2)}] \end{aligned} \tag{9}$$

where $\tau_q = (c_q^{(1)}, c_q^{(2)})$ and $\tau_d = (c_d^{(1)}, c_d^{(2)})$ are hierarchical types, and $w^{(1)} + w^{(2)} = 1$ control the relative importance of each level.

The structural score for a triple pair is:

$$\begin{aligned} S_{\text{struct}}(\tilde{s}_i, \tilde{\rho}) = w_h \cdot S_{\text{type}}(\tau(h_i), \tau(h)) \\ + w_t \cdot S_{\text{type}}(\tau(t_i), \tau(t)) \end{aligned} \tag{10}$$

where $\tilde{\rho} = (\tau(h), r, \tau(t)) \in \tilde{\mathcal{R}}_d$ and $w_h + w_t = 1$.

**Semantic Score.** The semantic score captures embedding-level similarity between raw triples. We encode each component separately with role-specific prefixes:

$$\begin{aligned} \mathbf{v}_h &= f_{\text{enc}}(\texttt{"S: "} \| h), \\ \mathbf{v}_r &= f_{\text{enc}}(\texttt{"P: "} \| r), \\ \mathbf{v}_t &= f_{\text{enc}}(\texttt{"O: "} \| t) \end{aligned} \tag{11}$$

The semantic score between query triple $s_i = (h_i, r_i, t_i)$ and document triple $\rho = (h, r, t)$ is:

$$\begin{aligned} S_{\text{sem}}(s_i, \rho) = \lambda_h \cos(\mathbf{v}_{h_i}, \mathbf{v}_h) \\ + \lambda_r \cos(\mathbf{v}_{r_i}, \mathbf{v}_r) \\ + \lambda_t \cos(\mathbf{v}_{t_i}, \mathbf{v}_t) \end{aligned} \tag{12}$$

where $\lambda_h + \lambda_r + \lambda_t = 1$.

**Triple-Level Score.** The overall score for a query-document triple pair combines structural and semantic components:

$$\begin{aligned} S_{\text{triple}}(s_i, \rho) = \alpha \cdot S_{\text{struct}}(\tilde{s}_i, \tilde{\rho}) \\ + (1 - \alpha) \cdot S_{\text{sem}}(s_i, \rho) \end{aligned} \tag{13}$$

**Document-Level Aggregation.** To obtain a document-level score, we first compute the best matching score for each query triple within the document:

$$S_{\text{best}}(s_i, d) = \max_{\rho \in \mathcal{R}_d} S_{\text{triple}}(s_i, \rho) \tag{14}$$

The document score aggregates across all query triples using a mixture of max and mean:

$$\begin{aligned} S(d) = \gamma \cdot \max_i S_{\text{best}}(s_i, d) \\ + (1 - \gamma) \cdot \frac{1}{|\mathcal{S}_t|} \sum_{s \in \mathcal{S}_t} S_{\text{best}}(s, d) \end{aligned} \tag{15}$$

where $\mathcal{S}_t$ denotes the top-$t$ sub-queries ranked by their best matching scores.

**Threshold Filtering and Reranking.** Documents with scores below a threshold $\theta$ are filtered out, and the remaining documents are reranked by $S(d)$:

$$\begin{aligned} \mathcal{D}_i = \text{Rank}(\mathcal{C}_i;\ S(d)), \\ \mathcal{C}_i = \{d \in \mathcal{D}_{\text{cand}} : S(d) \geq \theta\} \end{aligned} \tag{16}$$

This reranked list $\mathcal{D}_i$ is then passed to the LLM to answer the current sub-query and resolve its latent variables.

#### 2.6.2 Entity Binding and Sub-query Execution

We maintain an entity binding table $\mathcal{B}$ that maps latent variables to their resolved values and reranks evidence after each update:

$$\mathcal{B}^{(0)} = \emptyset \tag{17}$$

For each sub-query $s_i$, we first substitute existing bindings,

$$s_i' = \text{Resolve}(s_i, \mathcal{B}^{(i-1)}) \tag{18}$$

where Resolve$(\cdot)$ replaces latent variables with their bound values when available. We then score and rerank documents using the resolved query:

$$\begin{aligned} \mathcal{D}_i = \text{Rank}(\mathcal{C}_i';\ S(d \mid s_i')), \\ \mathcal{C}_i' = \{d \in \mathcal{D}_{\text{cand}} : S(d \mid s_i') \geq \theta\} \end{aligned} \tag{19}$$

Given the reranked documents, the LLM answers the current sub-query,

$$\hat{o}_i = \text{LLM}_{\text{answer}}(s_i', \mathcal{D}_i) \tag{20}$$

and updates the binding table,

$$\mathcal{B}^{(i)} = \mathcal{B}^{(i-1)} \cup \{v_i \mapsto \hat{o}_i\} \tag{21}$$

where $v_i$ is the latent variable in $s_i$ (if any). The final answer is $\hat{a} = \hat{o}_N$, supported by the last-step evidence.

## 3 Experiments

To demonstrate the efficacy of our TaSR-RAG, we conduct extensive evaluations on a variety of open-domain and multi-hop Question Answering (QA) benchmarks.

### 3.1 Datasets and Evaluation Metrics

Our evaluation spans seven representative knowledge-intensive datasets, selected to cover a broad spectrum of reasoning challenges. For single-step retrieval, we utilize Natural Questions (NQ) (Kwiatkowski et al., 2019), TriviaQA (Joshi et al., 2017), and PopQA (Mallen et al., 2023). To investigate complex reasoning, we include several multi-hop datasets: HotpotQA (Yang et al., 2018), 2WikiMultiHopQA (Ho et al., 2020), MuSiQue (Trivedi et al., 2022), and Bamboogle (Press et al., 2023). Together, these benchmarks provide a rigorous testing ground for both basic retrieval-augmented generation and sophisticated multi-step reasoning.

We employ standard QA metrics, specifically Exact Match (EM) and F1 scores, as the primary performance indicators, following prior studies (Jin et al., 2025; Wu et al., 2025; Li et al., 2025a).

### 3.2 Baselines

We compare the performance of TaSR-RAG against two distinct groups of baseline models. The first category consists of Retrieval-free Methods: (1) Direct Inference, which relies solely on the generator's internal knowledge; (2) Chain-of-Thought (CoT) (Wei et al., 2022), utilizing step-by-step

Table 1: Performance comparison (EM / F1) of different methods on Qwen2.5-7B-Instruct and Qwen2.5-72B-Instruct. The best results are in **bold** and the second best are underlined.

| Methods | General QA | | | | | | Multi-Hop QA | | | | | | | | Avg. EM | Avg. F1 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | NQ | | TriviaQA | | PopQA | | HotpotQA | | 2WikimQA | | Musique | | Bamboogle | | | |
| | EM | F1 | EM | F1 | EM | F1 | EM | F1 | EM | F1 | EM | F1 | EM | F1 | | |
| **Qwen2.5-72B-Instruct** | | | | | | | | | | | | | | | | |
| Direct Inference | 19.9 | 28.3 | 45.4 | 53.1 | 19.7 | 24.4 | 20.6 | 29.0 | 15.3 | 19.6 | 5.5 | 10.3 | 17.6 | 27.8 | 20.6 | 27.5 |
| CoT | 23.8 | 33.8 | 51.6 | 60.4 | 24.4 | 30.3 | 26.4 | 37.2 | 20.5 | 26.2 | 9.9 | 18.6 | 19.6 | 31.0 | 25.2 | 33.9 |
| RAG | 34.2 | **48.6** | 53.4 | 62.5 | 33.2 | 41.2 | 32.8 | 46.2 | 22.7 | 29.1 | 10.6 | 19.9 | 21.6 | 34.1 | 29.7 | 40.2 |
| IRCoT | 23.3 | 33.1 | **71.9** | **84.2** | **52.0** | **64.5** | 21.9 | 30.8 | 25.1 | 32.1 | 13.2 | **24.8** | 23.2 | 36.6 | 32.9 | 43.7 |
| GraphRAG | 26.6 | 38.5 | 61.7 | 70.9 | 36.7 | 45.8 | 31.2 | 42.7 | 30.9 | 38.5 | 9.2 | 17.9 | 6.4 | 15.8 | 29.0 | 38.6 |
| HippoRAG | 36.5 | 48.2 | 61.9 | 71.9 | 41.1 | 49.7 | 37.2 | **48.5** | **55.5** | 63.6 | 11.7 | 19.8 | 29.6 | 35.9 | 39.1 | 48.2 |
| HyperGraphRAG | 25.7 | 37.3 | 57.4 | 68.1 | 36.0 | 45.9 | 29.6 | 42.8 | 42.1 | 51.5 | 9.0 | 17.4 | 19.2 | 27.0 | 31.3 | 41.4 |
| StructRAG | 27.9 | 40.6 | 60.8 | 71.9 | 36.5 | 45.0 | 27.5 | 41.8 | 24.5 | 37.2 | 8.8 | 17.2 | 44.8 | **54.5** | 33.0 | 44.0 |
| **Ours** | **40.6** | 48.3 | 64.5 | 69.9 | 37.7 | 43.2 | **38.7** | **48.5** | 51.8 | **66.2** | **18.3** | 24.4 | **45.6** | 49.1 | **42.5** | **49.9** |
| **Qwen2.5-7B-Instruct** | | | | | | | | | | | | | | | | |
| Direct Inference | 13.4 | 23.2 | 40.8 | 49.7 | 14.0 | 16.4 | 18.3 | 27.1 | 12.6 | 30.1 | 3.1 | 11.9 | 12.0 | 17.2 | 16.3 | 25.1 |
| CoT | 4.8 | 19.3 | 18.5 | 49.7 | 5.4 | 16.4 | 9.2 | 13.0 | 10.8 | 15.1 | 2.2 | 5.1 | 23.2 | 20.8 | 10.6 | 19.9 |
| RAG | 32.9 | 40.0 | 35.5 | **67.2** | 19.2 | 44.3 | 19.9 | 41.9 | 13.5 | 47.7 | 5.8 | 13.0 | 20.8 | 22.4 | 21.1 | 39.5 |
| IRCoT | 22.4 | 34.7 | 47.8 | 65.9 | 30.1 | **60.2** | 13.3 | 20.2 | 14.9 | 21.3 | 7.2 | 20.7 | 22.4 | **32.1** | 22.6 | 36.4 |
| GraphRAG | 22.0 | 21.4 | 35.9 | 48.0 | 12.8 | 27.4 | 22.1 | 33.6 | 30.5 | 37.8 | 4.1 | 11.8 | 8.0 | 14.9 | 17.0 | 27.9 |
| HippoRAG | 18.2 | 25.1 | 37.0 | 51.0 | 26.8 | 37.0 | 18.2 | 32.2 | 19.7 | 38.2 | 4.4 | 12.8 | 15.2 | 23.7 | 18.8 | 31.4 |
| HypergraphRAG | 1.3 | 2.7 | 3.8 | 6.2 | 0.8 | 1.6 | 4.4 | 6.2 | 1.6 | 1.9 | 0.4 | 1.2 | 3.2 | 3.9 | 2.2 | 3.4 |
| StructRAG | 20.8 | 32.2 | 50.0 | 60.4 | 31.4 | 40.1 | 18.5 | 29.8 | 16.7 | 23.9 | 4.1 | 10.7 | 13.6 | 24.5 | 22.2 | 31.7 |
| **Ours** | **38.2** | **45.2** | **55.1** | 60.5 | **36.5** | 43.9 | **35.1** | **42.0** | **45.9** | **65.6** | **14.6** | **22.0** | **25.6** | 28.6 | **35.9** | **44.0** |

prompting. The second category encompasses Retrieval-augmented Frameworks, with a particular focus on state-of-the-art graph-based approaches: (1) RAG (Shi et al., 2024), the conventional dense retrieval-generation pipeline; (2) IRCoT (Trivedi et al., 2023), which interleaves retrieval with intermediate reasoning steps to iteratively gather multi-hop evidence; (3) GraphRAG (Edge et al., 2024), which leverages community detection and global summarization over knowledge graphs to capture holistic dataset themes; (4) HippoRAG (Gutierrez et al., 2024), a neurobiologically inspired framework that employs Personalized PageRank to achieve long-term memory integration and multi-hop associative retrieval; (5) HyperGraphRAG (Luo et al., 2025), which models high-order relationships using hyperedges to better represent complex, multi-entity interactions; and (6) StructRAG (Li et al., 2025c), a structured retrieval-augmented approach that dynamically adapts the retrieval granularity and structural organization based on the complexity of the query. These baselines encompass a wide array of prompting and fine-tuning strategies, providing a comprehensive context for our results.

### 3.3 Main Results

Table 1 reports Exact Match (EM) results on seven QA benchmarks using `Qwen2.5-72B-Instruct` and `Qwen2.5-7B-Instruct` as the underlying generators. Across both model scales, TASR-RAG achieves the best overall average performance, indicating that taxonomy-typed triples and sequential latent-variable binding improve evidence selection robustness.

**Qwen2.5-72B-Instruct.** Our method achieves the highest average EM (42.5), substantially outperforming standard RAG (29.7) and other retrieval-augmented baselines. At the dataset level, TASR-RAG attains the best scores on NQ (40.6), HotpotQA (38.7), Musique (18.3), and Bamboogle (45.6), while remaining competitive on TriviaQA (64.5), PopQA (37.7), and 2WikimQA (51.8). Notably, the strongest competing methods are highly dataset-dependent (e.g., IRCoT on TriviaQA and HippoRAG on 2WikimQA), whereas our approach provides consistently strong results across both general and multi-hop QA.

**Qwen2.5-7B-Instruct.** The gains are even more pronounced in the smaller-model setting. TASR-RAG achieves the highest average EM (37.0), improving over standard RAG (21.1) by a large margin. Moreover, it ranks first on all seven datasets, suggesting that explicit structure and step-by-step binding can effectively compensate for limited model capacity by guiding the model to focus on high-density, type-consistent evidence.

Table 2: Ablation to disentangle the effects of Document-level and Query-level structuredization under the Qwen2.5-72B-Instruct setting. Doc-level Structure applies triple extraction and taxonomy typing to documents while keeping the original query unchanged. Query-level Structure enables query decomposition and typed sub-queries but disables document-level structural representations.

| Method | HotpotQA | 2WikiMQA | Bamboogle | Avg. |
|---|---|---|---|---|
| RAG | 32.8 | 22.7 | 21.6 | 25.7 |
| Doc-level Structure | 44.0 | 34.0 | 40.8 | 39.6 |
| Query-level Structure | 42.5 | 31.5 | 43.2 | 39.1 |
| Full | 46.2 | 35.6 | 45.1 | 42.3 |

Table 3: Effect of taxonomy granularity on multi-hop QA performance under the Qwen2.5-72B-Instruct setting.

| Method | HotpotQA | 2WikiMQA | Bamboogle | Avg. |
|---|---|---|---|---|
| No Typing | 32.8 | 22.7 | 21.6 | 25.7 |
| Top-level Taxonomy | 43.0 | 34.5 | 33.6 | 37.0 |
| Top-two level Taxonomy | 46.2 | 35.6 | 45.1 | 42.3 |
| Top-three level Taxonomy | 42.0 | 34.1 | 36.0 | 37.4 |

Table 4: Ablation on matching strategies under the Qwen2.5-72B setting. Semantic-only matching relies solely on triple embedding similarity, structural-only matching uses typed triple consistency, and hybrid matching combines both signals.

| Matching Strategy | HotpotQA | 2WikiMQA | Bamboogle | Avg. |
|---|---|---|---|---|
| Semantic-only Matching | 43.5 | 23.5 | 43.2 | 36.7 |
| Structural-only Matching | 44.0 | 33.5 | 44.8 | 40.8 |
| Hybrid Matching (Full) | 46.2 | 35.6 | 45.1 | 42.3 |

### 3.4 Ablation Studies

To better understand the contribution of each component in TaSR-RAG, we conduct a series of ablation studies that systematically remove or modify key design choices. All ablation variants are evaluated under the same experimental settings and retrieval budgets for fair comparison. Results are reported using Exact Match (EM).

### 3.5 Query-level vs. Document-level Structuredization

Our framework introduces structure on both the document side and the query side, raising a natural question: which component contributes more to the overall performance gains? To disentangle their individual effects, we conduct a targeted ablation study under the Qwen2.5-72B setting on three representative multi-hop benchmarks. Specifically, we compare (i) a Doc-level variant, which converts retrieved documents into relational triples and taxonomy-typed triples while keeping the query as a single, undecomposed request, and (ii) a Query-level variant, which enables query decomposition into sequential, typed sub-queries but leaves documents in their original unstructured form. As shown in Table 2, both variants substantially outperform the unstructured RAG baseline, indicating that structured representations on either side already provide strong inductive bias for multi-hop reasoning. Interestingly, the two variants achieve comparable average performance (39.6 vs. 39.1 EM), but exhibit different dataset-specific strengths: Document-level structure is slightly more effective on HotpotQA and 2WikiMQA, while query-level structure yields stronger gains on Bamboogle, which involves more implicit multi-step inference. The full model, which combines structured queries with structured documents, achieves the best results across all datasets, improving the average EM to 42.3. Crucially, their combination is complementary rather than redundant, and both are necessary to fully realize the benefits of structured, sequential reasoning in retrieval-augmented generation. Moreover, the ablation helps pinpoint where each form of structure matters most: query-level decomposition primarily stabilizes hop ordering and reduces reasoning drift, while document-level tripleization improves evidence precision by filtering distractor spans. This motivates our full design, which aligns structured queries with structured documents to maximize both recall of relevant hops and faithfulness of the final answer.

The full TaSR-RAG consistently outperforms both variants, with especially pronounced improvements on multi-hop datasets. While No-Seq suffers from limited reasoning depth, Seq-NoBind exhibits clear degradation, highlighting the importance of maintaining consistent entity bindings across reasoning steps. These results demonstrate that sequential reasoning alone is insufficient without explicit variable binding.

### 3.6 Impact of Taxonomy-Guided Entity Typing

We next investigate how the granularity of entity typing affects structured retrieval and reasoning. Using the same pipeline and retrieval budget, we vary the depth of the taxonomy used for typing entities on both the query and document sides. Here, **Top-level Taxonomy** uses only the top-level taxonomy (L1), **Top-two level Taxonomy** uses the top two levels (L1+L2), and **Top-three level Taxon-**

**omy** uses three levels (L1+L2+L3). As shown in Table 3, introducing even Top-level typing leads to a substantial improvement over the untyped variant, confirming that explicit type constraints are essential for reducing retrieval noise and entity conflation. Moving from Top-level to Top-two level yields further gains across all datasets, with especially pronounced improvements on Bamboogle, suggesting that mid-level semantic distinctions provide stronger guidance for multi-step inference.

However, increasing the granularity to Top-three level results in consistent performance degradation, despite using more detailed type information. This result reveals a key property of abstraction-aware retrieval: effective multi-hop reasoning requires operating at an intermediate semantic granularity. Coarse abstractions improve coverage but introduce ambiguity, while overly fine-grained abstractions reduce structural overlap and destabilize retrieval due to sparsity and typing noise.

The strong performance of the two-level taxonomy suggests that typed retrieval is fundamentally a granularity-control problem rather than simply a structure-injection problem. This also supports our central hypothesis that multi-hop retrieval benefits from controllable semantic abstraction rather than fixed structural representations.

### 3.7 Effect of Matching Strategy

We further analyze the contribution of the proposed hybrid matching mechanism by comparing it against single-mode alternatives. Specifically, we evaluate semantic-only matching, which scores documents solely based on embedding similarity between query and document triples, and structural-only matching, which relies exclusively on taxonomy-typed triple consistency. As shown in Table 4, semantic-only matching performs well on datasets with strong lexical or semantic overlap, but suffers significantly on 2WikiMQA, indicating vulnerability to topic drift and insufficient structural constraints during multi-hop retrieval. In contrast, structural-only matching yields higher and more stable performance, confirming that typed structural constraints effectively reduce noise and entity conflation; however, its performance remains limited by reduced recall when exact structural matches are sparse. The hybrid strategy consistently outperforms both alternatives across all datasets, achieving the highest average EM. These results demonstrate that semantic similarity and structural consistency provide complementary signals: semantic matching ensures coverage and recall, while structural matching enforces type-level precision.

**Additional analysis.** Due to space constraints, we place further analyses in the appendix, including the error breakdown, efficiency comparison, sensitivity to retrieval budget, performance by reasoning depth, implementation details, case study, and prompt templates (Appendix B and following sections).

## 4 Related Work

Prior work on retrieval-augmented generation improves factual grounding by conditioning LLMs on external evidence (Li et al., 2025b; Guu et al., 2020; Lewis et al., 2020; Petroni et al., 2021), with recent variants further optimizing retriever–generator interaction or adaptively triggering retrieval (Shi et al., 2024; Sun et al., 2025b,a; Asai et al., 2024; Sun et al., 2026). Structure-aware RAG methods additionally incorporate knowledge graphs, topic graphs, or extracted triples for multi-hop reasoning (Sun et al., 2024; Edge et al., 2024; Gutierrez et al., 2024; Luo et al., 2025; Li et al., 2025c; Wu et al., 2025). In contrast, TASR-RAG focuses on abstraction-aware context selection by representing queries and documents as taxonomy-guided typed triples, enabling hybrid semantic–structural matching with explicit latent-variable binding. A fuller discussion is provided in Appendix A.

## 5 Conclusion

We presented TASR-RAG, a taxonomy-guided framework that reformulates multi-hop retrieval as abstraction-aware triple matching. Our central insight is that effective multi-hop reasoning depends critically on retrieving evidence at the appropriate semantic abstraction level. By introducing typed triples as a controllable structural abstraction interface, TASR-RAG jointly leverages semantic similarity and structural compatibility while maintaining explicit entity bindings across reasoning hops. Experiments across multiple QA benchmarks demonstrate that controllable semantic abstraction substantially improves retrieval robustness, reduces entity conflation, and enables more faithful multi-hop reasoning without requiring explicit graph construction.

## Limitations

**Taxonomy coverage and domain transfer.** Our experiments rely on a general-purpose two-level taxonomy adapted from Schema.org (Table 5), which fits the encyclopedic nature of the benchmarks studied here but is not intended to be universal. Specialized domains such as biomedicine or chemistry would require both levels to be reorganized around domain-specific entity types. Because TASR-RAG cleanly decouples the taxonomy from the matching, decomposition, and binding modules, the hierarchy can be swapped for a domain schema (e.g., derived from UMLS or ChEBI) without modifying the rest of the pipeline.

**No error-recovery mechanism.** TASR-RAG resolves latent variables in a single forward pass: once a binding enters $B^{(i)}$, it is not revisited, even when downstream matches are weak or inconsistent with it. Our error analysis (§B.1) shows that propagation and decomposition errors together account for a meaningful share of failures. We leave this to future work, with the natural direction being an agentic extension in which the model monitors per-hop confidence and selectively backtracks when later evidence contradicts an earlier binding.

## Ethical Statement

This work focuses on improving evidence selection and reasoning faithfulness in retrieval-augmented generation. It does not introduce new datasets containing private or sensitive information, and our experiments are conducted on publicly available question answering benchmarks. Since TASR-RAG relies on LLM-based extraction, typing, and answering modules, it may still inherit biases, factual errors, or omissions from the underlying models and retrieved corpora. In practical deployments, users should therefore inspect the retrieved evidence and reasoning traces, especially in high-stakes domains such as medicine, law, or finance. The proposed taxonomy-guided structure is intended to improve transparency and controllability, but it should not be interpreted as a guarantee of correctness or fairness.

## References


Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. 2024. Self-rag: Learning to retrieve, generate, and critique through self-reflection. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

Darren Edge, Ha Trinh, Newman Cheng, Joshua Bradley, Alex Chao, Apurva Mody, Steven Truitt, and Jonathan Larson. 2024. From local to global: A graph RAG approach to query-focused summarization. *CoRR*, abs/2404.16130.

Bernal Jimenez Gutierrez, 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 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*.

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Ming-Wei Chang. 2020. Realm: retrieval-augmented language model pre-training. In *International Conference on Machine Learning*.

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, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Bowen Jin, Hansi Zeng, Zhenrui Yue, Dong Wang, Hamed Zamani, and Jiawei Han. 2025. Search-r1: Training llms to reason and leverage search engines with reinforcement learning. *CoRR*, abs/2503.09516.

Mandar Joshi, Eunsol Choi, Daniel Weld, and Luke Zettlemoyer. 2017. TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1601–1611, Vancouver, Canada. Association for Computational Linguistics.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. 2019. Natural questions: A benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:452–466.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Xiaoxi Li, Guanting Dong, Jiajie Jin, Yuyao Zhang, Yujia Zhou, Yutao Zhu, Peitian Zhang, and Zhicheng

Dou. 2025a. Search-o1: Agentic search-enhanced large reasoning models. *CoRR*, abs/2501.05366.

Yangning Li, Weizhi Zhang, Yuyao Yang, Wei-Chieh Huang, Yaozu Wu, Junyu Luo, Yuanchen Bei, Henry Peng Zou, Xiao Luo, Yusheng Zhao, Chunkit Chan, Yankai Chen, Zhongfen Deng, Yinghui Li, Hai-Tao Zheng, Dongyuan Li, Renhe Jiang, Ming Zhang, Yangqiu Song, and Philip S. Yu. 2025b. Towards agentic RAG with deep reasoning: A survey of rag-reasoning systems in llms. *CoRR*, abs/2507.09477.

Zhuoqun Li, Xuanang Chen, Haiyang Yu, Hongyu Lin, Yaojie Lu, Qiaoyu Tang, Fei Huang, Xianpei Han, Le Sun, and Yongbin Li. 2025c. Structrag: Boosting knowledge intensive reasoning of llms via inference-time hybrid information structurization. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net.

Haoran Luo, Haihong E, Guanting Chen, Yandan Zheng, Xiaobao Wu, Yikai Guo, Qika Lin, Yu Feng, Zemin Kuang, Meina Song, Yifan Zhu, and Luu Anh Tuan. 2025. Hypergraphrag: Retrieval-augmented generation via hypergraph-structured knowledge representation. *Preprint*, arXiv:2503.21322.

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), ACL 2023, Toronto, Canada, July 9-14, 2023*, pages 9802–9822. Association for Computational Linguistics.

Fabio Petroni, Aleksandra Piktus, Angela Fan, Patrick Lewis, Majid Yazdani, Nicola De Cao, James Thorne, Yacine Jernite, Vladimir Karpukhin, Jean Maillard, Vassilis Plachouras, Tim Rocktäschel, and Sebastian Riedel. 2021. KILT: a benchmark for knowledge intensive language tasks. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2021, Online, June 6-11, 2021*, pages 2523–2544. Association for Computational Linguistics.

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, Singapore, December 6-10, 2023*, pages 5687–5711. Association for Computational Linguistics.

Ori Ram, Yoav Levine, Itay Dalmedigos, Dor Muhlgay, Amnon Shashua, Kevin Leyton-Brown, and Yoav Shoham. 2023. In-context retrieval-augmented language models. *Transactions of the Association for Computational Linguistics*, pages 1316–1331.

Jimeng Shi, Sizhe Zhou, Bowen Jin, Wei Hu, Runchu Tian, Shaowen Wang, Giri Narasimhan, and Jiawei Han. 2025. Hypercube-based retrieval-augmented generation for scientific question-answering. *Preprint*, arXiv:2505.19288.

Weijia Shi, Sewon Min, Michihiro Yasunaga, Minjoon Seo, Richard James, Mike Lewis, Luke Zettlemoyer, and Wen-tau Yih. 2024. REPLUG: Retrieval-augmented black-box language models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 8371–8384, Mexico City, Mexico. Association for Computational Linguistics.

Jiashuo Sun, Pengcheng Jiang, Saizhuo Wang, Jiajun Fan, Heng Wang, Siru Ouyang, Ming Zhong, Yizhu Jiao, Chengsong Huang, Xueqiang Xu, Pengrui Han, Peiran Li, Jiaxin Huang, Ge Liu, Heng Ji, and Jiawei Han. 2026. Rethinking the reranker: Boundary-aware evidence selection for robust retrieval-augmented generation. *Preprint*, arXiv:2602.03689.

Jiashuo Sun, Shixuan Liu, Zhaochen Su, Xianrui Zhong, Pengcheng Jiang, Bowen Jin, Peiran Li, Weijia Shi, and Jiawei Han. 2025a. GRACE: generative representation learning via contrastive policy optimization. *CoRR*, abs/2510.04506.

Jiashuo Sun, Chengjin Xu, Lumingyuan Tang, Saizhuo Wang, Chen Lin, Yeyun Gong, Lionel M. Ni, Heung-Yeung Shum, and Jian Guo. 2024. Think-on-graph: Deep and responsible reasoning of large language model on knowledge graph. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

Jiashuo Sun, Xianrui Zhong, Sizhe Zhou, and Jiawei Han. 2025b. Dynamicrag: Leveraging outputs of large language model as feedback for dynamic reranking in retrieval-augmented generation. *CoRR*, abs/2505.07233.

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.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2023. Interleaving retrieval with chain-of-thought reasoning for knowledge-intensive multi-step questions. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2023, Toronto, Canada, July 9-14, 2023*, pages 10014–10037. Association for Computational Linguistics.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. 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 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*.

Junlin Wu, Xianrui Zhong, Jiashuo Sun, Bolian Li, Bowen Jin, Jiawei Han, and Qingkai Zeng. 2025. Structure-r1: Dynamically leveraging structural knowledge in LLM reasoning through reinforcement learning. *CoRR*, abs/2510.15191.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 2369–2380. Association for Computational Linguistics.

Yue Yu, Wei Ping, Zihan Liu, Boxin Wang, Jiaxuan You, Chao Zhang, Mohammad Shoeybi, and Bryan Catanzaro. 2024. Rankrag: Unifying context ranking with retrieval-augmented generation in llms. In *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*.

# A Detailed Related Work

## A.1 Retrieval-Augmented Generation

RAG enhances LLM performance by incorporating external evidence, improving factual grounding and reducing hallucination in knowledge-intensive tasks (Li et al., 2025b; Guu et al., 2020). Beyond chunk-based retrieval, work on retrieval-augmented next-token prediction (Lewis et al., 2020; Petroni et al., 2021) and end-to-end training frameworks (Shi et al., 2024; Sun et al., 2025b,a, 2026) aims to tighten the integration between retriever and generator. Self-reflective RAG systems such as Asai et al. (2024) employ special control tokens and a critic model to adaptively trigger retrieval and improve reasoning stability.

While these efforts address retrieval quality or retriever–generator integration, our work targets a different bottleneck: multi-hop context selection. Instead of retrieving unstructured passages and delegating all reasoning to the LLM, TASR-RAG represents both documents and queries as relational triples and augments them with taxonomy-guided entity types. This enables a hybrid matching function that combines semantic similarity over raw triples with structural consistency over typed triples, and supports sequential reasoning via explicit latent-variable binding.

## A.2 Structure-aware Retrieval and Reasoning

A complementary line of work introduces structural supervision into retrieval through knowledge graphs, LLM-generated topic graphs, or symbolic abstractions (Sun et al., 2024). GraphRAG (Edge et al., 2024), HippoRAG (Gutierrez et al., 2024), and HyperGraphRAG (Luo et al., 2025) construct large hierarchical graphs from LLM-generated summaries, supporting multi-hop reasoning but suffering from hallucinated nodes, misaligned structure, and prohibitive construction cost. StructRAG (Li et al., 2025c) and Structure-R1 (Wu et al., 2025) and related triple-based systems extract entity-level triples to reduce noise, yet these triples remain sparse, overly fine-grained, and lack structural alignment with the query, limiting multi-hop generalization. Ontology-based information extraction has also been explored in traditional IR and semantic parsing, but prior methods typically assume fixed schemas and do not integrate typed reasoning patterns from the query itself.

Our approach differs from prior structured RAG systems in both representation and retrieval philosophy. Existing graph-based or triple-based methods typically operate at a fixed structural granularity, making retrieval either overly sparse or overly coarse during multi-hop reasoning. In contrast, TASR-RAG treats typed triples as a controllable semantic abstraction interface, enabling retrieval to jointly leverage semantic similarity and structural compatibility across multiple abstraction levels. Instead of relying on explicit graph traversal, TASR-RAG performs abstraction-aware triple matching with sequential latent-variable binding. This allows the retrieval process to progressively refine intermediate entities while maintaining interpretable reasoning traces, without requiring hallucination-prone graph construction or exhaustive graph search.

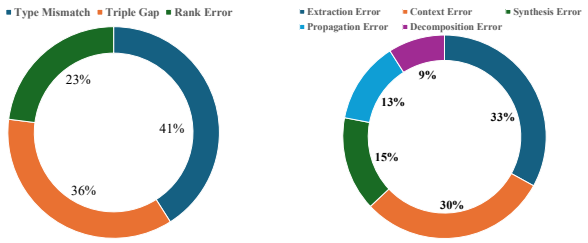



Figure 2: Error distribution in our manual analysis. The left figure illustrates Matching Errors and right figure illustrates Generation Errors.

# B Analysis

Beyond aggregate performance, we provide further analyses to shed light on the behavior and robustness of TASR-RAG.

## B.1 Error Breakdown Analysis.

We manually inspect a subset of error cases where the ground-truth evidence is present in the retrieved pool, and conduct an error analysis to isolate failures introduced by our pipeline. Following our two-tier taxonomy, we group errors into (i) **MATCHING ERRORS** arising from typed triple matching and reranking, and (ii) **GENERATION ERRORS** arising from sequential reasoning over the selected context.

As shown in Figure 2, most failures (64.5%) occur after retrieval and reranking, indicating that the dominant bottleneck is reasoning and answer construction rather than document ranking.

**Generation errors.** Within sequential reasoning, the most frequent issues are **Extraction Error** (33%) and **Context Error** (30%), where relevant documents are highly ranked but the LLM

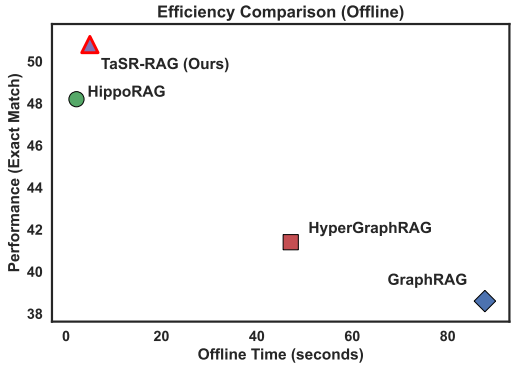
either fails to extract the exact answer string (format/normalization) or fails to derive the correct step result from the provided evidence. This suggests that, once relevant evidence is retrieved, the dominant bottleneck shifts to faithful utilization of the selected context: extraction failures are often triggered by aliasing/normalization (e.g., abbreviations, date formats) and by answers expressed implicitly rather than as a single span, while context errors indicate that hop-level evidence can still be noisy or incomplete, causing the model to follow a locally plausible but globally incorrect chain. We also observe **Synthesis Error** (15%) where intermediate steps appear consistent but the final answer becomes inconsistent, and **Propagation Error** (13%) caused by incorrect variable substitution across sub-queries. Finally, **Decomposition Error** (10%) suggests that imperfect sub-query formation remains a non-negligible source of downstream failures.

**Matching errors.** Among typed triple matching failures, **Type Mismatch** (41%) and **Triple Gap** (36%) dominate. This highlights a key trade-off in our structured pipeline: typing constraints improve precision, but they also amplify upstream typing mistakes and brittleness in triple extraction/coverage. The remaining **Rank Error** (23%) suggests that even when types and triples are adequate, the hybrid (semantic+structural) scoring can still misrank evidence.

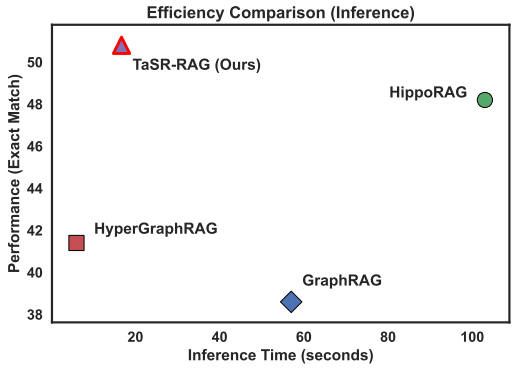
### B.2 Efficiency

We analyze the computational efficiency of different RAG frameworks by separating the cost into two stages: offline preprocessing and inference-time execution. Figure 3 reports the trade-off between performance and time cost for GRAPHRAG, HIPPORAG, HYPERGRAPHRAG, and TASR-RAG.

**Offline Efficiency.** Offline preprocessing exhibits substantial variance across methods. Graph-based approaches such as GRAPHRAG and HYPERGRAPHRAG incur high offline costs due to expensive graph construction and indexing, with GRAPHRAG being particularly costly. In contrast, HIPPORAG achieves very low preprocessing time, while TASR-RAG requires only a modest offline cost despite incorporating structured retrieval and matching. Importantly, TASR-RAG achieves the highest performance among all methods while maintaining an offline time that is an order of magnitude lower than heavy graph-based baselines.




(a) Offline efficiency.




(b) Inference efficiency.

Figure 3: Performance–efficiency trade-off under offline preprocessing and inference-time execution. TASR-RAG (ours) achieves the best overall balance, avoiding the high offline cost of graph-based methods while maintaining low inference-time overhead.

**Inference Efficiency.** Inference-time efficiency reveals a different trade-off. While HYPERGRAPHRAG achieves fast inference by relying on precomputed structures, HIPPORAG suffers from significant inference overhead. GRAPHRAG also incurs non-trivial inference cost. In contrast, TASR-RAG maintains consistently low inference time while delivering the best performance, demonstrating that its structured reasoning and retrieval design does not introduce prohibitive online overhead.

Across both offline and inference settings, TASR-RAG occupies a favorable position on the performance–efficiency frontier. It avoids the extreme offline cost of graph-heavy methods and the high inference latency of retrieval-intensive approaches, achieving a balanced and practical trade-

off without sacrificing accuracy.

# C Additional Experiments

## C.1 Sensitivity to Retrieval Budget.

We study the effect of retrieval budget by varying the number of retrieved documents as follows:

$$K \in \{1, 3, 5, 10, 15, 20, 30\}. \tag{22}$$

As shown in Figure 5, TASR-RAG consistently outperforms standard RAG across all retrieval budgets and datasets, demonstrating robust performance under different resource constraints.

The performance gap between TASR-RAG and RAG is particularly pronounced in certain regimes. On HotpotQA, TASR-RAG achieves 38.7% at Top-10, compared to RAG's 32.8%, representing an 18% relative improvement. On 2WikimQA, the advantage is even more striking: TaSR-RAG reaches 51.8% at Top-10 versus RAG's 22.7%, more than doubling the baseline performance. For Bamboogle, TaSR-RAG attains 45.6% at Top-10 compared to RAG's 21.6%, demonstrating substantial gains across multi-hop reasoning scenarios.

Interestingly, TaSR-RAG exhibits different scaling behaviors across datasets. On HotpotQA and Bamboogle, performance generally improves with larger $K$, peaking around Top-15 to Top-20 before slight degradation at Top-30, suggesting that retrieval noise accumulates beyond optimal budget. In contrast, on 2WikimQA, TaSR-RAG shows remarkable efficiency at lower budgets, with performance jumping dramatically from Top-1 (25.2%) to Top-10 (51.8%), then maintaining stable high performance through Top-30. This suggests that TaSR-RAG's structured reasoning and dynamic matching mechanism enable it to identify and utilize relevant evidence more efficiently, particularly when the retrieval budget is constrained.

These results highlight TaSR-RAG's ability to extract maximal value from retrieved documents through its question decomposition and adaptive reasoning strategy, rather than simply relying on retrieving more documents.

## C.2 Performance by Reasoning Depth.

To explicitly evaluate multi-hop reasoning ability, we analyze model performance as a function of reasoning depth. Specifically, we group test instances by hop count and report performance for hop values with non-empty support. Figure 4 compares TASR-RAG with standard RAG across three representative multi-hop QA benchmarks.

Across all datasets, both methods exhibit comparable performance on shallow reasoning cases. However, as reasoning depth increases, standard RAG degrades rapidly, reflecting its reliance on flat retrieval and one-shot generation. In contrast, TASR-RAG maintains strong performance and shows increasingly larger gains in deeper reasoning regimes. The advantage becomes particularly pronounced at 3-hop and beyond, where explicit sequential structure and entity binding are critical.

The trend is most evident on Bamboogle, which emphasizes implicit and indirect multi-hop inference, where TASR-RAG substantially outperforms RAG at higher hop counts. Similar patterns are observed on 2WikiMultiHopQA and HotpotQA, indicating that the benefits of taxonomy-guided sequential context selection generalize across different multi-hop reasoning styles.

# D Two-level Entity Taxonomy

Table 5 lists the two-level entity taxonomy used throughout TASR-RAG for taxonomy-guided typing. We use the top-level label (L1) to provide coarse semantic constraints (e.g., `PERSON`, `ORGANIZATION`), and the second-level label (L2) to capture finer distinctions (e.g., `Scientist` vs. `Politician`) that improve precision during multi-hop evidence selection.

# E Dataset Statistics

Our method does not require any additional training; therefore, we only report the evaluation datasets used in our experiments. Table 6 summarizes dataset sizes and task types.

# F Implementation Details

We evaluate our method using Qwen2.5-7B-Instruct and Qwen2.5-72B-Instruct as the backbone generators. All components involving prompting—including query decomposition, relational triple extraction, entity typing decisions, sub-query answering, and final answer synthesis—are implemented via LLM inference. For initial dense retrieval, we retrieve top-$K_0$ candidate documents with $K_0 = 10$. During structured reasoning (evidence selection), the number of selected documents $K$ is dynamic and determined by score-based filtering with a threshold $\theta = 0.3$. For taxonomy-guided entity typing, we use a retrieval-first, coarse-to-fine

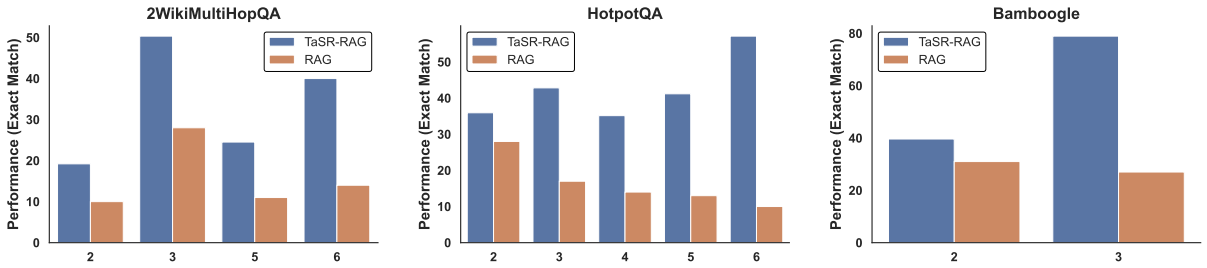
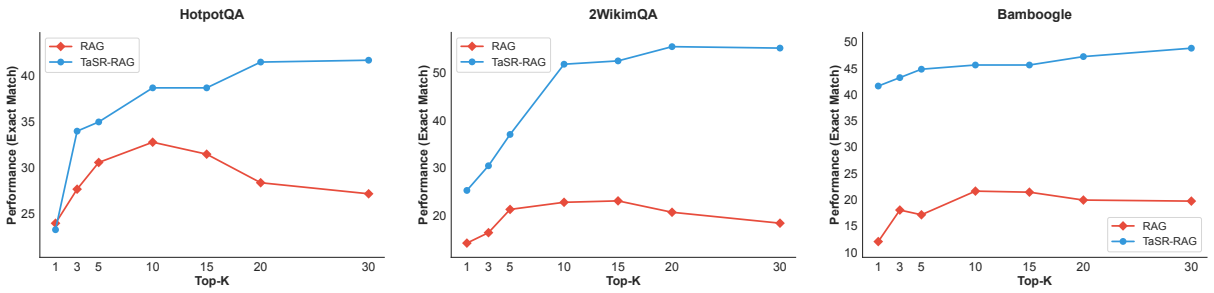



Figure 4: Performance by reasoning depth (hop count). We report results for hop counts with non-empty support. While RAG performance degrades as reasoning depth increases, TASR-RAG maintains strong performance and exhibits increasingly larger gains in deeper reasoning regimes, with particularly pronounced improvements at 3-hop and beyond.




(a) HotpotQA (b) 2WikimQA (c) Bamboogle

Figure 5: Performance comparison between TASR-RAG and standard RAG across different retrieval budgets (Top-K) on three multi-hop QA datasets. TASR-RAG consistently outperforms RAG across all settings, with particularly strong gains on 2WikimQA (over 100% relative improvement at Top-10) and substantial improvements on HotpotQA and Bamboogle.

strategy: for each entity, top-$N$ first-level type candidates are retrieved using embedding similarity with $N = 10$, from which the LLM selects the top-3 labels; under each selected first-level label, top-$M$ second-level candidates are retrieved with $M = 20$, and the LLM selects the final $(\mathrm{L1}, \mathrm{L2})$ pair from their union. Document-level scoring aggregates over the top-$t = 3$ sub-queries. Hybrid matching combines structural and semantic signals with $\alpha = 0.5$; type-level matching uses $w^{(1)} = 0.5$ for L1 and $w^{(2)} = 0.5$ for L2, head and tail contributions use $w_h = 0.5$ and $w_t = 0.5$, and semantic matching weights are set to $\lambda_h = 0.3$, $\lambda_r = 0.3$, and $\lambda_t = 0.4$. All embedding-based retrieval, including taxonomy label retrieval, uses an E5 encoder with L2-normalized embeddings and inner-product FAISS search, which is equivalent to cosine similarity. Hierarchical typing is implemented with a two-stage FAISS setup consisting of a global L1 index and separate L2 indexes for each L1 branch, enabling efficient coarse-to-fine candidate retrieval. To reduce LLM overhead, we apply simple rule-based typing for structured entities such as years, dates, and percentages before invoking embedding- or LLM-based typing. For robustness, all classification-style prompting steps use deterministic decoding (temperature $= 0$), enforce JSON-only outputs, and optionally include lightweight disambiguation context constructed from document titles and evidence sentences associated with extracted triples. Unless otherwise specified, all experiments use the same settings across datasets and model scales.

## G Case Study

We present a representative case aligned with Figure 1 to illustrate how TASR-RAG performs taxonomy-guided structured reasoning for evidence selection. The question asks for two linked facts: (i) which database *Science Activity Planner* uses, and (ii) which company developed that database. After dense retrieval (Step 0), the candidate pool $\mathcal{D}_{\text{cand}}$ contains heterogeneous snippets: Doc[1] directly states that *Science Activity Planner* uses *MySQL*, Doc[3] mentions *PostgreSQL* and *MySQL* only as deployment options, and Doc[6] provides a developer fact about *MySQL*. TASR-RAG first converts each candidate document into relational triples and

Table 5: Two-level entity taxonomy used for type constraints (L1 and L2).

| L1 | L2 categories |
|---|---|
| PERSON | Scientist, Engineer, Academic, Politician, Businessperson, Athlete, Actor, Musician, Writer, Journalist, Inventor, MilitaryPerson |
| ORGANIZATION | Company, University, ResearchInstitute, GovernmentAgency, Nonprofit, InternationalOrganization, MilitaryUnit, SportsTeam, PoliticalParty, MediaOutlet, Hospital, School |
| LOCATION | Country, StateOrProvince, City, Region, Continent, River, Lake, Mountain, Island, SeaOrOcean, Desert, Park |
| FACILITY | Building, Bridge, Airport, Station, Port, Museum, Stadium, Campus, Laboratory, PowerPlant |
| EVENT | War, Election, Tournament, Conference, Festival, Disaster, Protest, LaunchEvent, MergerEvent, Trial |
| WORK | Book, Film, TVSeries, Song, Album, VideoGame, SoftwareProject, ResearchPaper, LawOrPolicy, Dataset |
| PRODUCT | CloudService, Database, ProgrammingLanguage, HardwareDevice, VehicleModel, Drug, Chemical, ConsumerProduct, ModelOrAlgorithm |
| BIOENTITY | Animal, Plant, Bacteria, Virus, Disease, ProteinOrGene |
| TIME | Year, Date, TimePeriod |
| QUANTITY | Count, Money, Percentage, Measurement |
| CONCEPT | Technology, Method, Theory, FieldOfStudy, RoleOrTitle |
| OTHER | Other |

Table 6: Dataset statistics for evaluation.

| Split | Dataset | #Examples | Task Type |
|---|---|---|---|
| Eval | NQ | 3,610 | Single-hop |
| | TriviaQA | 11,313 | Single-hop |
| | PopQA | 14,267 | Single-hop |
| | HotpotQA | 7,405 | Multi-hop |
| | 2WikiMultiHopQA | 12,576 | Multi-hop |
| | Musique | 2,417 | Multi-hop |
| | Bamboogle | 125 | Multi-hop |

taxonomy-typed triples (Step 1), then decomposes the question into an ordered chain of triple subqueries with latent variables (Step 2).

$$s_1 = (\text{Science Activity Planner}, \texttt{uses}, ?x),$$
$$s_2 = (?x, \texttt{developed_by}, ?y),$$

with corresponding typed forms that enforce the intended reasoning pattern,

$$\tilde{s}_1 = \begin{pmatrix} \textsc{Work/System}, \\ \texttt{uses}, \\ \textsc{Work/Database} \end{pmatrix},$$
$$\tilde{s}_2 = \begin{pmatrix} \textsc{Work/Database}, \\ \texttt{developed_by}, \\ \textsc{Organization/Company} \end{pmatrix}.$$

This typed decomposition makes the required reasoning pattern explicit and prevents the second hop from drifting to irrelevant entities or types.

With the binding table initialized (Step 3), TaSR-RAG performs hop-wise evidence selection via hybrid (semantic + structural) triple matching. For hop 1 (Step 4), we compute HybridMatch scores $S(d)$ between $(s_1, \tilde{s}_1)$ and each $(\mathcal{R}_d, \tilde{\mathcal{R}}_d)$, then filter by $S(d) \geq \theta$ and rank. In this pool, Doc[1] is retained because it contains a relation-consistent and type-consistent triple supporting

$$(\text{Science Activity Planner}, \texttt{uses}, \text{MySQL}),$$

whereas Doc[3] is filtered since it does not entail a uses relation (it only lists options). The hop-1 answerer updates the binding table (Step 5) with $?x \leftarrow$ MySQL, which turns hop 2 into the grounded query

$$s_2' = (\text{MySQL}, \texttt{developed_by}, ?y).$$

For hop 2 (Step 6), HybridMatch is recomputed conditioned on the resolved entity, promoting Doc[6] and down-weighting any non-developer mentions of *MySQL*; the final hop binds $?y \leftarrow$ MySQL AB and returns **MySQL; MySQL AB** (Step 7). This example highlights how sequential binding interacts with hybrid matching: the first hop resolves an intermediate entity with high precision, and the second hop uses that binding to retrieve developer evidence that would be hard to isolate under a single, unconstrained retrieval step.

## H Algorithm

We present algorithm of TaSR-RAG in Algorithm 1.

| **Question:** Which database does *Science Activity Planner* use, and which company developed it? | | | |
|---|---|---|---|
| **Step** | **Sub-query / operation** | **Retrieved/selected content** | **Binding update** |
| 0 | Candidate retrieval | Dense retrieval returns an evidence pool $\mathcal{D}_{\text{cand}}$ containing Doc[1], Doc[3], Doc[6]. | — |
| 1 | Document triples + typing | Extract triples $\mathcal{R}_d$ and typed triples $\tilde{\mathcal{R}}_d$ for each $d \in \mathcal{D}_{\text{cand}}$. | — |
| 2 | Query decomposition + typing | $s_1$ = (Science Activity Planner, `uses`, $?x$); $s_2$ = ($?x$, `developed_by`, $?y$).<br>$\tilde{s}_1$ = (WORK/SYSTEM, `uses`, WORK/DATABASE);<br>$\tilde{s}_2$ = (WORK/DATABASE, `developed_by`, ORGANIZATION/COMPANY). | — |
| 3 | Initialize bindings | Initialize entity binding table $\mathcal{B} \leftarrow \emptyset$. | $\mathcal{B}_0 = \emptyset$ |
| 4 | Hop 1: context selection | Compute HybridMatch scores $S(d)$ for $d \in \mathcal{D}_{\text{cand}}$ w.r.t. $(s_1, \tilde{s}_1)$, then filter by $S(d) \geq \theta$ and rank. In this pool, Doc[1] is kept (relation- and type-consistent), while Doc[3] is filtered (option list; weak match). Select $\mathcal{D}_1 = \{\text{Doc[1]}\}$. | — |
| 5 | Hop 1: bind variable | $\hat{o}_1 \leftarrow \text{LLM}_{\text{answer}}(s_1, \mathcal{D}_1)$; bind the latent variable in $s_1$. | $\mathcal{B}_1 : ?x \leftarrow$ MySQL |
| 6 | Hop 2: context selection | Resolve $s_2' =$ (MySQL, `developed_by`, $?y$), then recompute $S(d)$ for all $d \in \mathcal{D}_{\text{cand}}$ conditioned on the binding. Filter by $S(d) \geq \theta$ and rank; Doc[6] is kept as the top match, while non-developer mentions of MySQL are down-weighted. Select $\mathcal{D}_2 = \{\text{Doc[6]}\}$. | — |
| 7 | Hop 2: bind + answer | $\hat{o}_2 \leftarrow \text{LLM}_{\text{answer}}(s_2', \mathcal{D}_2)$; update bindings and return $\hat{a} = \hat{o}_2$. | $\mathcal{B}_2 : ?y \leftarrow$ MySQL AB |
| **Answer: MySQL; MySQL AB** | | | |

Table 7: Case study aligned with Figure 1 and our method. TASR-RAG performs step-wise evidence selection via hybrid (semantic + structural) triple matching and maintains explicit bindings $(\mathcal{B}_1, \mathcal{B}_2)$ to carry resolved entities across hops.

# I Prompt Templates

This appendix lists the six prompts used by TASR-RAG that correspond one-to-one with the four LLM roles in Algorithm 1: $\text{LLM}_{\text{extract}}$ (Figure 6), $\text{LLM}_{\text{select}}$ (Figures 7 and 8), $\text{LLM}_{\text{decomp}}$ (Figures 9 and 10), and $\text{LLM}_{\text{answer}}$ (Figure 11). Auxiliary prompts used for output-format repair, decomposability gating, single-triple fallback, and variable-substitution rewriting do not alter the pipeline described in Section 2 and are released with our code.

## I.1 Document-Level Triple Extraction

## I.2 Taxonomy-Guided Entity Typing – Stage 1

## I.3 Taxonomy-Guided Entity Typing – Stage 2

## I.4 Query Decomposition

## I.5 Sub-queries to Typed Triples

## I.6 Sub-query Answering

**Triple Extraction** USER PROMPT

```
Your task is to extract factual triples, strictly as
(subject, relation, object), from the Document that might
      help
to answer the User Query.

USER QUERY: "{query}"

You must output ONLY a JSON array of strings:
[
  ["subject", "relation", "object"],
  ...
]
Each triple MUST contain exactly 3 non-empty strings.

## DEFINITION OF A "RELATION"
A relation is ANY link between two entities. Extract BOTH:
1. Explicit Actions (verbs): "A attacked B"
     -> ["A", "attacked", "B"]
2. Implicit Static Relations expressed by possessives,
   appositives, prepositions, or noun modifiers:
     "A's B"              -> (A, has, B)
     "A, the B,"          -> (A, is a, B)
     "A in B" / "A of B" -> (A, located in, B) / (A, part
      of, B)

## STRICT RULES
- Return ONLY a JSON array; no markdown, no extra text.
- Use simple, active, human-readable relation phrases
  (e.g., "is located in", "is a", "served as",
   "started on", "ended on").
- Capture every relation, including static / noun-only
      ones.
- Resolve coreference: replace pronouns with their
      referents.
- No empty objects: for intransitive verbs, infer minimal
  context from the document.
- Dates are entities. For dated events, emit explicit
  ["X", "started on", DATE] / ["X", "ended on", DATE] /
  ["X", "occurred on", DATE].
- For truncated sentences at chunk boundaries, infer the
  missing subject from nearby context.
- No hallucination: do not invent facts beyond the
      document.

## INPUT DOCUMENT
{doc_text}

## OUTPUT FORMAT
Return ONLY:
[
  ["A", "B", "C"],
  ["D", "E", "F"],
  ...
]
All fields must be non-empty strings.
```

Figure 6: Triple extraction prompt ($\text{LLM}_{\text{extract}}$, Section 2). Query-conditioned extraction over each candidate document, covering both verbal and implicit static relations with strict JSON output.

**Entity Typing – Stage 1 (L1 top-3)** USER PROMPT

```
Choose the best 3 L1 types for the entity, ordered best to
       worst.

Rules:
- Output MUST be valid JSON only:
    {"L1_top3": ["...", "...", "..."]}
- MUST choose ONLY from candidates.
- MUST be unique and length = 3.

Infer the semantic type from context. If the context
      implies
a specific type (e.g. "religion" -> CONCEPT, "political
party" -> ORGANIZATION, "city" -> LOCATION), prioritize
      that
over generic types.

Entity: "{e}"
Context:
{ctx}

Candidates (L1): {l1_candidates}

Output JSON:
```

Figure 7: Stage 1 of taxonomy-guided entity typing ($\text{LLM}_{\text{select}}$). Selects three coarse L1 candidates from the taxonomy $\mathcal{T}^{(1)}$, conditioned on local context.

**Entity Typing – Stage 2 ((L1, L2) pair)** USER PROMPT

```
Choose the best (L1, L2) type pair for the entity.

Rules:
- Output MUST be valid JSON only:
    {"L1": "...", "L2": "..."}
- You MUST choose a pair from candidates.
- The output (L1, L2) must correspond to one of the
  candidate pairs.

Use context to determine the SEMANTIC type. Prioritize
specific semantic types over generic ones; only use PERSON
types when the answer is actually a human being.

Entity: "{e}"
Context:
{ctx}

Candidates (L1/L2 pairs): {pair_candidates}

Output JSON:
```

Figure 8: Stage 2 of taxonomy-guided entity typing ($\text{LLM}_{\text{select}}$). Selects the final hierarchical type $\tau(e) = (c^{(1)}, c^{(2)})$ from the union of L2 candidates retrieved under the top-3 L1 labels.

**Query Decomposition** USER PROMPT

```
You are a query decomposition engine.

Task: Decompose the user query into a JSON array of short,
atomic sub-queries for retrieval.

Rules:
- Output MUST be valid JSON only (a list of strings).
  No extra text.
- Keep the same language as the query.
- Each sub-query must be TRULY ATOMIC -- one single fact
  lookup or one single operation.
- Break down nested constraints
  ("X of Y" -> "Find Y", then "Find X of Y").
- If there is a calculation step (double, half, sum, ...),
  make it a SEPARATE sub-query.
- Keep original entity surface forms when possible.
- Max {max_subqueries} sub-queries.

Examples:
Query: "Who was president of the United States in the year
        that Citibank was founded?"
Output: ["When was Citibank founded?",
         "Who was president of the United States in that
      year?"]

Query: "What is the population of the capital of France?"
Output: ["What is the capital of France?",
         "What is the population of that city?"]

Query: "Who wrote Harry Potter?"
Output: ["Who wrote Harry Potter?"]

Now decompose:

Query: "{query}"
Output:
```

Figure 9: Query decomposition prompt ($\text{LLM}_{\text{decomp}}$, first stage). Splits a complex multi-hop query into an ordered list of atomic natural-language sub-queries; nested constraints and calculation steps are forced to separate hops.

**Sub-queries to Triples** USER PROMPT

```
You are a structured semantic parser.

Task: Convert EACH sub-query into EXACTLY ONE triple
["subject", "predicate", "object"] for later matching.

Hard constraints:
- Output MUST be valid JSON only:
    {"triples": [...]}
- EXACTLY ONE triple per sub-query (1-to-1 mapping).
  Output EXACTLY {expected_count} triples.
- Keep the same order as sub-queries.
- Use concise predicates that reflect the actual
  relationship; never use vague predicates like
  "relates to", "about", "concerns".
- Use variables for unknowns:
  ?ans, ?person, ?date, ?place, ?num, ?entity, ?thing, ...
- Do NOT use outside knowledge; just convert the intent.

Critical rules:
1. NEVER use the entire question sentence as subject or
   object. Extract entities / concepts only.
2. Convert questions to declarative triple form by
   identifying WHO / WHAT does WHAT to WHOM / WHAT.
3. PRESERVE concrete entities mentioned in the sub-query.
   Use variables ONLY for the unknown being asked about.

Examples:
Sub-query: "What is the atomic number of hydrogen?"
  GOOD: ["hydrogen", "has atomic number", "?num"]
  BAD : ["?element", "has atomic number", "?num"]

Sub-query: "Who was the Roman emperor that declared war
            on the sea?"
  GOOD: ["?person", "declared war on", "the sea"]

Now convert.

Sub-queries (ordered):
{subqueries_json}

Output JSON:
```

Figure 10: Sub-query-to-triple parser ($\text{LLM}_{\text{decomp}}$, second stage). Casts each natural-language sub-query as a relational triple with explicit latent variables (?`ans`, ?`person`, ?`date`, ...), preserving concrete entities so that only the asked-for slot becomes a variable.

**Sub-query Answering** USER PROMPT

```
You are a fact-grounded QA system.

Your task is to answer the question using ONLY the
      evidence
documents provided below.

Critical rules:
- The answer MUST be taken DIRECTLY from the text in the
  evidence documents.
- Do NOT use any external knowledge or make inferences
  beyond what is explicitly stated.
- If the exact answer is not explicitly written in the
  documents, respond that facts are insufficient.

{context_section}

Question:
{question}

Evidence (ranked by relevance):
{docs}

Instructions:
1. First, analyse the evidence step by step inside
   <reasoning>...</reasoning> tags.
   - Quote the relevant sentence(s) from the documents
     that contain the answer.
   - The answer must appear verbatim in the quoted text.
2. Then, provide your final answer inside
   <answer>...</answer> tags.
3. The answer must be in the SHORTEST form possible
   (a name, number, or short phrase).
4. Do NOT add extra words, background, dates, or
   explanations in the answer.

Output format:
<reasoning>
[Quote the relevant text from documents, then explain.]
</reasoning>
<answer>[minimal answer string, copied from evidence]</
      answer>
```

Figure 11: Sub-query answering prompt ($\text{LLM}_{\text{answer}}$). Executed at each hop over the reranked, type-filtered document list $D_i$; enforces verbatim evidence grounding and a minimal answer span that is then written into the entity binding table $B$.

**Algorithm 1** Taxonomy-Guided Structured Reasoning for Evidence Selection (TaSR-RAG)

**Require:** Query $q$, document collection $\mathcal{D}$, two-level taxonomy $\mathcal{T}$, dense-retrieval top-$K_0$, threshold $\theta$, LLMs $\text{LLM}_{\text{extract}}$, $\text{LLM}_{\text{decomp}}$, $\text{LLM}_{\text{select}}$, $\text{LLM}_{\text{answer}}$

**Ensure:** Final answer $\hat{a}$ (returned as the output of the last sub-query)

$\mathcal{D}_{\text{cand}} \leftarrow$ Top-$K_0$DenseRetrieve$(q, \mathcal{D})$
**for all** $d \in \mathcal{D}_{\text{cand}}$ **do**
  $\mathcal{R}_d \leftarrow \text{LLM}_{\text{extract}}(d)$ ▷ Document triples
  **for all** $e \in \text{Entities}(\mathcal{R}_d)$ **do**
    $c^{(1)} \leftarrow \text{LLM}_{\text{select}}(\mathcal{T}^{(1)}, e)$
    $c^{(2)} \leftarrow \text{LLM}_{\text{select}}(\mathcal{T}^{(2)}(c^{(1)}), e)$
    $\tau(e) \leftarrow (c^{(1)}, c^{(2)})$
  **end for**
  $\tilde{\mathcal{R}}_d \leftarrow \{(\tau(h), r, \tau(t)) : (h, r, t) \in \mathcal{R}_d\}$
**end for**

$\{s_1, \ldots, s_N\} \leftarrow \text{LLM}_{\text{decomp}}(q)$ ▷ Sub-queries (triples with latent variables)
**for all** $e \in \text{Entities}(\{s_i\}_{i=1}^{N})$ **do**
  $c^{(1)} \leftarrow \text{LLM}_{\text{select}}(\mathcal{T}^{(1)}, e)$
  $c^{(2)} \leftarrow \text{LLM}_{\text{select}}(\mathcal{T}^{(2)}(c^{(1)}), e)$ ▷ Query-level typing (incl. variables)
  $\tau(e) \leftarrow (c^{(1)}, c^{(2)})$
**end for**
$\tilde{s}_i \leftarrow (\tau(h_i), r_i, \tau(t_i)) \quad \forall i$

$\mathcal{B} \leftarrow \emptyset$ ▷ Entity binding table
**for** $i = 1$ to $N$ **do**
  $s'_i \leftarrow \text{Resolve}(s_i, \mathcal{B})$; $\tilde{s}'_i \leftarrow \text{Resolve}(\tilde{s}_i, \mathcal{B})$
  **for all** $d \in \mathcal{D}_{\text{cand}}$ **do**
    $S(d) \leftarrow \text{HybridMatch}\big((s'_i, \tilde{s}'_i), (\mathcal{R}_d, \tilde{\mathcal{R}}_d)\big)$
  **end for**
  $\mathcal{D}_i \leftarrow \text{Rank}(\{d \in \mathcal{D}_{\text{cand}} : S(d) \geq \theta\}; S(d))$
  $\hat{o}_i \leftarrow \text{LLM}_{\text{answer}}(s'_i, \mathcal{D}_i)$
  $\mathcal{B} \leftarrow \mathcal{B} \cup \text{Bind}(s_i, \hat{o}_i)$
**end for**

$\hat{a} \leftarrow \hat{o}_N$
**return** $\hat{a}$