Reasoning with Ontology Graph: Toward Type-Constrained Knowledge Graph Question Answering

Yongxue Shan, Jie Peng, Zixuan Dong, Fei Hu and Xiaodong Wang*

National Key Laboratory of Parallel and Distributed Computing, College of Computer Science and Technology, National University of Defense Technology, Changsha, China
{shanyongxue001,pengjie,dongzixuan18,hufei,xdwang}@nudt.edu.cn

Abstract

Large language models (LLMs) have recently advanced knowledge graph question answering (KGQA), but current methods tend to rely on LLM-induced type systems with inconsistent granularity, or perform multi-hop reasoning without explicit target-type constraints. We introduce OntGQA, a type-constrained KGQA framework that reasons over a relation-centric ontology graph, where each relation is labeled with its head and tail entity types to provide a stable schema backbone. Built on this graph, OntGQA adopts a planner–judge architecture with generative backoff: a type planner proposes plausible head–tail type pairs, a judge verifies retrieved candidates and their paths, and a generator is invoked only when all candidates are rejected. By constraining both endpoints of reasoning in type space, OntGQA achieves state-of-the-art performance and produces ontology-grounded reasoning chains, with substantial Hit@1 gains (87.7%→91.5% on WebQSP and 67.6%→74.6% on CWQ).

1 Introduction

Knowledge graph question answering (KGQA) aims to answer natural language questions by reasoning over structured knowledge graphs (KGs), and has been applied in domains such as healthcare (Frisoni et al., 2024), agriculture (Ting et al., 2023), and multimedia (Chen et al., 2022; Liang et al., 2024). With the rapid progress of large language models (LLMs), there is a growing trend to combine LLMs with KGs to understand questions, design reasoning strategies, and guide path-based search over the graph (Pan et al., 2024).

Existing path-based multi-hop methods mainly follow two paradigms. Ontology-guided methods prompt an LLM to construct an ontology from observed relations and entity pairs, and then reason using a neighborhood ontology dictionary (Liu et al., 2025b). Subgraph-based methods reason over a local subgraph rooted at the question entity, then operate on its paths by retrieving related paths (Wen et al., 2024; Sun et al., 2024), generating path hypotheses with LLMs (Luo et al., 2024, 2025), or pruning noisy paths (Tan et al., 2025). By coupling structured KG search with the semantic understanding of LLMs, these approaches achieve strong empirical performance on multi-hop reasoning tasks.

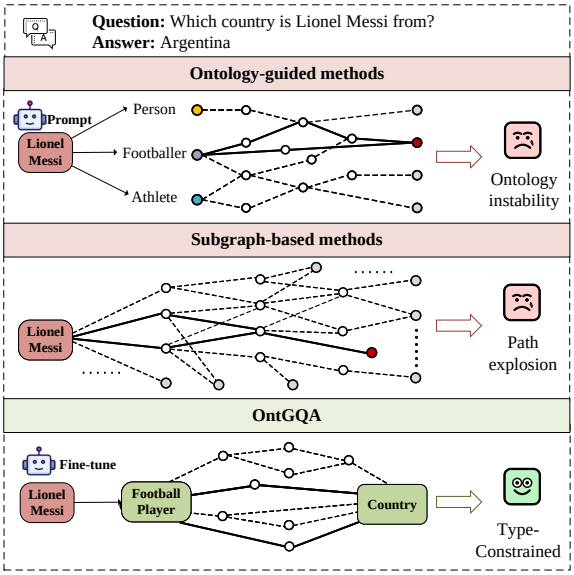


Figure 1: Example of previous methods and OntGQA. Solid lines indicate true reasoning paths, while dashed lines indicate spurious or incorrect paths.

Despite these advances, two major challenges remain, as shown in Figure 1. First, LLM-based ontology construction often suffers from granularity inconsistency. Without unified constraints, prior works may mix abstraction levels (e.g., Lionel Messi labeled as Person, Footballer, and Athlete), making the resulting ontologies hard to align. Second, subgraph-based methods expand large neighborhoods from the question entity and admit many paths with incorrect target types into the candidate space, which often leads to path explosion. These

*Xiaodong Wang is the corresponding author.

Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 10684–10697

issues motivate explicit constraints on type granularity and target types during multi-hop reasoning.

To tackle these challenges, we present OntGQA, which performs reasoning with an **Ont**ology **G**raph for type-constrained knowledge graph **Q**uestion **A**nswering. We build a reusable, relation-centric ontology graph grounded in the canonical Freebase type vocabulary, where each relation is labeled with its head and tail entity types to provide a stable schema backbone. On top of this resource, OntGQA follows a planner–judge architecture with generative backoff: a type planner predicts plausible head–tail type pairs for a question; retrieved candidates and their evidence paths are then verified by a judge, and a generator is invoked only when all candidates are rejected. By constraining both endpoints of reasoning in type space, OntGQA narrows the search space, mitigates path explosion from unconstrained neighborhood expansion, and improves the robustness of multi-hop answers.

In summary, our contributions include:

- We construct a reusable relation-centric ontology graph for Freebase, avoiding the instability of LLM-induced ontologies and providing a stable schema backbone for KGQA.
- We design a planner–judge architecture with generative backoff that constrains both endpoints of reasoning in type space, improving the precision and faithfulness of multi-hop reasoning paths.
- OntGQA achieves state-of-the-art performance on Freebase-based multi-hop QA benchmarks while delivering interpretable, ontology-grounded reasoning chains.

2 Preliminary

Knowledge Graph and Ontology Graph. A knowledge graph $\mathcal{G}$ represents factual information as triples of entities connected by relations. Its ontology graph $\mathcal{O}$ provides a type-level abstraction of $\mathcal{G}$, where the same set of relations is defined over entity types. While $\mathcal{G}$ captures instance-level facts, $\mathcal{O}$ offers a high-level semantic view by generalizing entities to their associated types.

Relation Paths and Reasoning Paths. Relation paths are sequences of relations, defined as $z = (r_1, r_2, \ldots, r_l)$, where each $r_i \in \mathcal{R}$ denotes the i-th relation in the path and l is the path length. Reasoning paths are concrete instantiations of relation paths in $\mathcal{G}$, represented as sequences of triples $w_z = (e_0 \xrightarrow{r_1} e_1 \xrightarrow{r_2} \cdots \xrightarrow{r_l} e_l)$.

Knowledge Graph Question Answering. KGQA is a core reasoning task that leverages structured knowledge to answer natural language questions. Given a question q and a knowledge graph $\mathcal{G} = (\mathcal{E}, \mathcal{R}, \mathcal{T})$, the objective is to predict the answers. Following previous studies (Jiang et al., 2023; Luo et al., 2024), we assume valid answers are reachable from the topic entities via one or more reasoning paths in $\mathcal{G}$.

3 Method

The overall architecture is shown in Figure 2. We first construct a relation-centric ontology graph as a schema backbone by linking each relation to its head and tail types via data extraction, filtering, and dataset-aware completion (Section 3.1). On top of this graph, we design a planner–judge architecture with generative backoff. The planner predicts a head–tail type plan to guide candidate and path retrieval, and the judge verifies candidate answers and their paths, falling back to a generator only when all are rejected (Section 3.2). Finally, we introduce an optimization framework to jointly train these components (Section 3.3).

3.1 Ontology Graph Construction

Entities connected by the same relation in a knowledge graph often share similar types and attributes. In large KGs, such regularities can be abstracted into an ontology, providing a compact, schema-level view. In multi-hop QA, questions exhibit recurring type patterns, making answer-type inference easier and more reliable than direct entity prediction. Motivated by this, we build an ontology graph grounded in the Freebase type vocabulary, mapping each relation to canonical head and tail types to mitigate granularity inconsistency. The left side of Figure 2 illustrates this abstraction from the original KG to the ontology graph.

3.1.1 Data source and extraction.

Freebase aggregates facts from high-quality open resources (e.g., Wikipedia (Wales and Sanger, 2001), WordNet (Miller, 1995)) and releases an RDF dump in N-Triples format (Microsoft, 2015). After downloading and decompressing the dump, we process triples under the canonical namespace `http://rdf.freebase.`

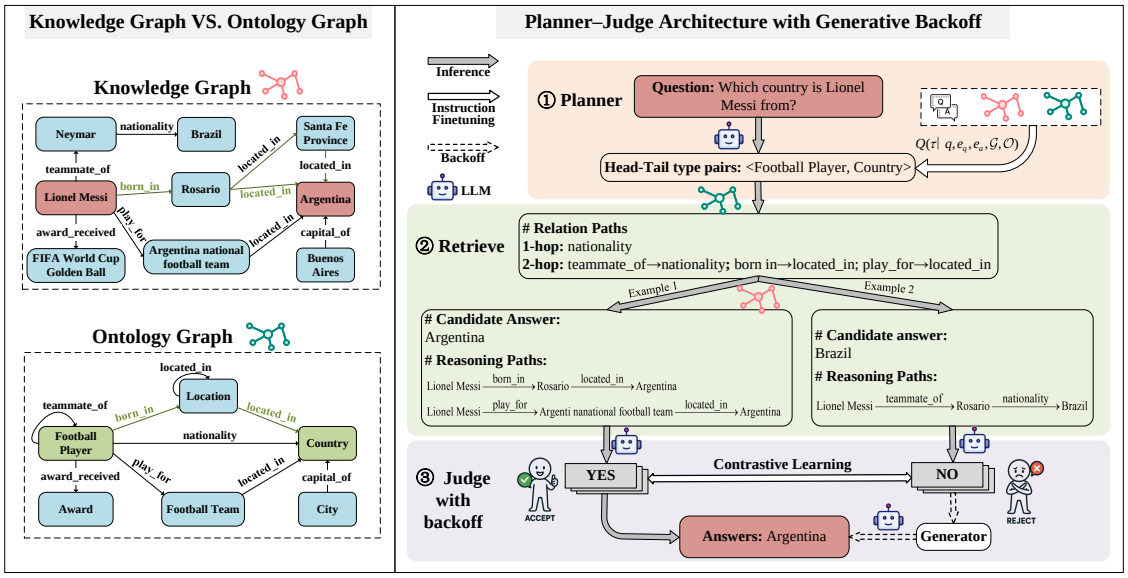


Figure 2: An overview of the OntGQA model.

com/ns/. We extract property-level schema using type.property.schema and type.property.expected_type, treating the former as the head (subject) type and the latter as the tail (object) type. We stream-parse the dump line by line, pair each property's domain and range, keep canonical identifiers, and record all entries.

3.1.2 Filtering and dataset-aware completion

Direct projection of head–relation–tail type signatures from the RDF dump can be incomplete or noisy. We remove non-semantic or administrative types (e.g., common.topic) and retain only relations with complete head–tail signatures. To improve coverage, we then perform dataset-aware completion for a small number of missing or underspecified relations (listed in Appendix D). For each KGQA dataset, we aggregate head and tail entity sets from the training split only and infer the corresponding types for these relations. A completion is accepted only when both sides can be inferred.

The resulting ontology graph is derived from the Freebase schema rather than model predictions, providing consistent type constraints across Freebase-based QA datasets and shrinking the search space for multi-hop reasoning. Unlike entity-centric subgraph expansion, our retrieval is anchored by fixed head and tail types, tightly bounding admissible relation paths. Detailed processing statistics (e.g., runtime and counts of relation–type triples) are provided in Appendix D.

3.2 Planner–Judge Architecture with Generative Backoff

Based on the constructed ontology graph, OntGQA performs reasoning in three steps: (i) given a question, a planner predicts plausible head–tail entity type pairs; (ii) conditioned on these type plans, we retrieve candidate answers and their reasoning paths from the ontology graph $\mathcal{O}$ and the KG $\mathcal{G}$; and (iii) a judge determines whether each candidate is correct, invoking a fallback generator only when all candidates are rejected.

3.2.1 Step I. Ontology-Guided Planner

The planner outputs ontology-aligned head–tail entity type pairs $\tau = (\tau_h, \tau_t)$ as plans for a question q. Treating type pairs as plans imposes explicit endpoint constraints for multi-hop retrieval and provides supervision that aligns the planner with the canonical Freebase type vocabulary.

Constraint-induced posterior from graphs. Given a question q with question entity e_q and gold answer e_a, we extract the shortest factual path $w_z(e_q, e_a) = (e_q \xrightarrow{r_1} \cdots \xrightarrow{r_l} e_a)$ with relation sequence $z = (r_1, \ldots, r_l)$ in $\mathcal{G}$. We use this shortest path only as a minimal supervision signal. Aligning z to the ontology graph $\mathcal{O}$ yields a head–tail type transition $(\tau_h \overset{z}{\Longrightarrow} \tau_t)$, which we treat as a valid plan summarizing an ontology-consistent way to reach e_a from e_q. We define a constraint-induced

target distribution $Q(\tau)$ over valid type pairs:

$$Q(\tau) \simeq Q(\tau \mid q, e_q, e_a, \mathcal{G}, \mathcal{O}) = \begin{cases} \frac{1}{|\mathcal{T}|}, & \tau \in \mathcal{T}, \\ 0, & \text{otherwise.} \end{cases} \tag{1}$$

$$\mathcal{T} = \{(\tau_h, \tau_t) : \exists w_z(e_q, e_a) \in \mathcal{G} \text{ s.t. } (\tau_h \overset{z}{\Longrightarrow} \tau_t) \in \mathcal{O}\} \tag{2}$$

where we assume a uniform distribution over all valid type pairs $\mathcal{T}$, following Luo et al. (2024), as a simple and stable target.

Prior from an LLM. To leverage instruction-following capabilities, we prompt an LLM to generate a serialized list of type pairs (e.g., <PAIR> τ_h <SEP> τ_t </PAIR>). This defines a conditional prior $P_\phi(\tau \mid q)$ parameterized by the planner LLM, which factorizes autoregressively as

$$\log P_\phi(\tau \mid q) = \sum_{i=1}^{|\tau|} \log P_\phi(s_i \mid s_{<i}, q) \tag{3}$$

where $P_\phi(s_i \mid s_{<i}, q)$ is the next-token probability under parameters ϕ.

3.2.2 Step II. Plan-Conditioned Path Retrieval

Given the predicted type plans $\widehat{\mathcal{T}} = \{(\tau_h, \tau_t)\}$ and question entities $\mathcal{E}_q \subseteq \mathcal{E}$, enumerating all KG paths consistent with each (τ_h, τ_t) is intractable. We therefore perform hop-wise retrieval under ontology constraints. We first enumerate admissible relation paths in the ontology graph $\mathcal{O}$ using the endpoint types (τ_h, τ_t) and then instantiate them in the KG $\mathcal{G}$ to obtain reasoning paths and candidate answers.

We implement this retrieval strategy from 1-hop to multi-hop paths:

- **1-hop templates.** For each question entity $e_q \in \mathcal{E}_q$, we enumerate ontology-licensed outgoing relations $\mathcal{R}^+(\tau_h) = \{r : \exists \tau' \text{ s.t. } (\tau_h \xrightarrow{r} \tau') \in \mathcal{O}\}$. For each $r \in \mathcal{R}^+(\tau_h)$, we instantiate KG edges and collect the corresponding 1-hop reasoning paths $(e_q \xrightarrow{r} a')$, where the terminal entity a' is a candidate answer.

- **2-hop with boundary gates.** The first hop is restricted to $\mathcal{R}^+(\tau_h)$ and the second hop to ontology-licensed incoming relations $\mathcal{R}^-(\tau_t) = \{r : \exists \tau' \text{ s.t. } (\tau' \xrightarrow{r} \tau_t) \in \mathcal{O}\}$. Concretely, we expand from e_q using relations in $\mathcal{R}^+(\tau_h)$, and only keep 2-hop paths whose second relation lies in $\mathcal{R}^-(\tau_t)$.

- **Backoff to longer paths.** If the union of 1–2 hop retrieval is empty, we back off to $K{=}3$ (and then $K{=}4$) under the same boundary gates to recover recall on sparse graphs.

After retrieval, we deduplicate the obtained paths and textualize each path as an evidence string $\mathbf{w}_{a'}$, taking the terminal entity as the candidate a'. Aggregating all $(a', \mathbf{w}_{a'})$ for a plan τ yields the candidate–evidence set $\mathcal{C}(\tau, \mathcal{O}, \mathcal{G})$ for downstream judging or generation.

3.2.3 Step III. Judge with Generative Backoff

Judge. Given q, a retrieved candidate set $\widehat{\mathcal{A}}(q)$, and a small set of shortest textualized evidence paths $\mathbf{W}_{a'}$ for each candidate $a' \in \widehat{\mathcal{A}}(q)$, the judge prompts an instruction-tuned LLM to decide whether a' is a correct answer to q, outputting the literal label YES or NO. We use balanced positives/negatives and a contrastive margin.

Let $\mathcal{V}_{\text{yes}}, \mathcal{V}_{\text{no}} \subseteq \mathcal{V}$ be tokenizer-derived sets of single tokens for the two labels. For $c \in \{\text{yes}, \text{no}\}$, define the log probability mass assigned to label c:

$$\ell_c(q, a', \mathbf{w}_{a'}) = \log \sum_{v \in \mathcal{V}_c} P_\theta\big(v \,\big|\, q, a', \mathbf{w}_{a'}\big) \tag{4}$$

where $P_\theta(\cdot \mid \cdot)$ is the LLM next-token distribution with parameters θ. The signed margin is

$$m(q, a', \mathbf{w}_{a'}) = \ell_{\text{yes}}(q, a', \mathbf{w}_{a'}) - \ell_{\text{no}}(q, a', \mathbf{w}_{a'}) \tag{5}$$

with positive values favoring YES and negative values favoring NO.

With a certainty threshold $\lambda_{\text{margin}} > 0$, we accept high-confidence YES decisions and form the judged answer list:

$$\text{Judge}(q) = \{\, a' \in \widehat{\mathcal{A}}(q) : m > 0 \wedge |m| \geq \lambda_{\text{margin}} \,\} \tag{6}$$

Here, $m > 0$ denotes a YES decision, and $|m| \geq \lambda_{\text{margin}}$ indicates that the model is highly confident in the answer.

Generative backoff. If all candidates are rejected or none are available, we invoke a generator $\text{Gen}(q)$ with a generation-style prompt to produce a fallback set from a single decode. The final answer set is

$$\mathcal{A}(q) = \begin{cases} \text{Judge}(q), & \text{if Judge}(q) \neq \varnothing, \\ \text{Gen}(q), & \text{otherwise.} \end{cases} \tag{7}$$

This design yields precise, evidence-grounded acceptance when the judge is confident, and resorts to generation only as a last fallback.

3.3 Optimization Framework

We formulate OntGQA as an optimization problem over three instruction-tuned modules: (i) a type planner that predicts head-tail entity type pairs $\tau = (\tau_h, \tau_t)$ for a question q; (ii) a judge that verifies candidate answers $a' \in \mathcal{E}$ with their evidence paths $\mathbf{w}_{a'}$; and (iii) a generator that directly produces a fallback answer when no candidate is accepted. The optimization objective is detailed in Appendix A.

The planner is trained by minimizing the KL divergence between the posterior in (1) and the prior in (3):

$$\mathcal{L}_{\text{plan}} = D_{\text{KL}}\big(Q(\tau) \,\|\, P_\phi(\tau \mid q)\big) \;\simeq\; -\tfrac{1}{|\mathcal{T}|} \sum_{\tau \in \mathcal{T}} \log P_\phi(\tau \mid q) \tag{8}$$

Here, $P_\phi(\tau \mid q)$ is the planner prior with parameters ϕ; $Q(\tau)$ is a variational posterior; D_{KL} denotes Kullback–Leibler divergence.

The judge is trained with a listwise InfoNCE objective over contrastively constructed positives and negatives:

$$\mathcal{L}_{\text{judge}} = \tfrac{1}{|\mathcal{Y}_q|} \sum_{a' \in \mathcal{Y}_q} \left[-\log \frac{\exp\big(\ell_{\text{yes}}\big)}{\exp\big(\ell_{\text{yes}}\big) + \sum_{a' \in \mathcal{N}_q} \exp\big(\ell_{\text{no}}\big)} \right] \tag{9}$$

where $\mathcal{Y}_q = \{a' \in \widehat{\mathcal{A}}(q) : \text{YES}\}$ and $\mathcal{N}_q = \{a' \in \widehat{\mathcal{A}}(q) : \text{NO}\}$ are the positive/negative sets for question q; ℓ_{yes} and ℓ_{no} are judge logits for acceptance and rejection in (4), respectively.

An optional generative backoff is trained by maximum likelihood:

$$\mathcal{L}_{\text{gen}} = -\log P_\omega(a \mid q) \tag{10}$$

where $a \in \mathcal{E}$ is the gold answer for q; $P_\omega(a \mid q)$ is the generator distribution with parameters ω. The generator may be trained separately, replaced with a pretrained model, or invoked via an external API. If fine-tuned jointly, ω may share parameters with the judge and the planner; otherwise, it denotes frozen weights or API settings.

The overall training loss is the sum of the three terms:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{plan}} + \mathcal{L}_{\text{judge}} + \mathcal{L}_{\text{gen}} \tag{11}$$

4 Experiments

4.1 Experiment Settings

Datasets. We evaluate the reasoning performance of OntGQA on two benchmark KGQA datasets: WebQSP (Yih et al., 2016) and CWQ (Talmor and Berant, 2018). Both datasets are built upon the Freebase knowledge graph (Bollacker et al., 2008), which contains roughly 88M entities, 20K relations, and 126M factual triples. WebQSP includes 4,737 natural language questions focusing on relatively simple reasoning, while CWQ comprises 34,689 more complex questions requiring up to four-hop reasoning with compositional or constraint-based queries. We follow the standard training and test splits from prior work (Luo et al., 2024), and summarize dataset statistics in Table 1.

Datasets	#Train	#Test	Max #Hop	#Relation
WebQSP	2,826	1,628	2	5726
CWQ	27,639	3,531	4	6576

Table 1: Statistics of the datasets.

Evaluation Metrics. Following prior work (Luo et al., 2024), we mainly evaluate OntGQA with Hit@1 and F1. Hit@1 is the fraction of questions whose top-ranked prediction matches any gold answer under the dataset's normalization rules. F1 is macro-averaged over questions by computing per-question precision and recall between the predicted and gold answer sets. In ablation studies, we additionally report precision and recall to further analyze answer coverage and accuracy.

Implementation details. OntGQA uses LLaMA2-Chat-7B (Touvron et al., 2023) as the backbone LLM and is instruction-tuned for three epochs with a learning rate of 2e-5 on four A100-40G GPUs. We use a fixed random seed and report 95% confidence intervals (CIs) via single-seed bootstrap over test predictions with 10,000 resamples. At inference, the planner uses beam search to output the top-3 entity type pairs per question. For evidence retrieval, we cap reasoning depth based on the hop statistics in Figure 7 (two hops on WebQSP and four on CWQ). In the judge module, we set $\lambda_{\text{margin}} = 1.0$ and provide a detailed hyperparameter analysis in Section G.

4.2 Main Results

We evaluate OntGQA on WebQSP and CWQ against graph-only, LLM-only, and KG+LLM baselines to assess its multi-hop reasoning performance. We consider two main variants: OntGQA (Llama-2-7B), which jointly fine-tunes the planner and judge, and OntGQA (Llama-2-7B + GPT-4o), which keeps Llama-2-7B frozen and invokes GPT-4o in

Type	Methods	WebQSP		CWQ	
		Hit@1	F1	Hit@1	F1
Graph Reasoning	GraftNet (Sun et al., 2018)	66.7	62.4	36.8	32.7
	NSM (He et al., 2021)	68.7	62.8	47.6	42.4
	SR+NSM (Zhang et al., 2022)	68.9	64.1	50.2	47.1
	UniKGQA (Jiang et al., 2023)	77.2	72.2	51.2	49.1
LLM Reasoning	Llama-2-7B (Touvron et al., 2023)	56.4	36.5	28.4	21.4
	Llama-3.1-8B (Meta, 2024)	55.5	34.8	28.1	22.4
	ChatGPT (OpenAI, 2022)	59.3	43.5	34.7	30.2
	GPT-4o (OpenAI, 2024)	61.8	43.6	38.2	32.9
	DeepSeek-v3 (DeepSeek-AI et al., 2024)	64.0	43.9	41.1	33.8
KG+LLM	ToG (GPT-4) (Sun et al., 2024)	82.6	—	67.6	—
	RoG (Llama-2-7B) (Luo et al., 2024)	85.7	70.8	62.6	56.2
	SymAgent (Llama-2-7B) (Liu et al., 2025a)	55.5	41.3	35.1	31.2
	GNN-RAG (Llama-2-7B) (Mavromatis and Karypis, 2025)	85.7	71.3	66.8	**59.4**
	ORT (GPT-4o) (Liu et al., 2025b)	87.7	71.8	65.4	58.7
	GCR (Llama-2-7B + GPT-4o) (Luo et al., 2025)	87.5	71.5	66.0	58.5
	OntGQA (Llama-2-7B + GPT-4o)	**91.5**	76.6	73.8	55.8
	OntGQA (Llama-2-7B)	**91.5**	**77.1**	**74.6**	56.8

Table 2: Comparison of methods on WebQSP and CWQ. The backbone model used by each method is shown in parentheses.

the judge with generative backoff module. As shown in Table 2, OntGQA achieves state-of-the-art performance on both benchmarks. To further assess the stability of these results, we report bootstrap confidence intervals in Appendix E.

Compared with graph-reasoning models such as UniKGQA, OntGQA improves Hit@1 by about 14 points on WebQSP and over 23 points on CWQ, indicating that purely structural graph learning underutilizes the information available in the KG. Pure LLM baselines also lag behind, with the best model reaching at most 64.0% Hit@1 on WebQSP and 41.1% on CWQ, suggesting that free-form LLM reasoning without explicit schema guidance struggles to maintain accurate multi-hop grounding.

Within the KG+LLM family, OntGQA achieves the strongest results. Relative to the best prior KG+LLM systems, it improves Hit@1 by 3.8 points on WebQSP over ORT and by 7.0 points on CWQ over ToG. Although OntGQA's F1 on CWQ (56.8%) is slightly below GNN-RAG (59.4%), this largely reflects a precision–recall trade-off under strict set matching: CWQ often has multi-answer gold sets, and OntGQA may retrieve extra answers beyond the gold set even when top-1 is correct. A top-k diagnostic curve and a 50-case manual audit support this explanation (Appendix F).

Overall, these results show that type-constrained multi-hop reasoning in our ontology graph substantially improves answer accuracy and yields a robust, ontology-aligned reasoning framework.

4.3 Ablation Studies

We conduct ablation studies to assess the contribution of three key components in OntGQA: the generative backoff module, the judge module, and the ontology graph. Since removing the judge requires using the OpenAI API, we take OntGQA (Llama-2-7B + GPT-4o) as the reference benchmark. Table 3 reports results on WebQSP and CWQ across Hit@1, F1, Precision, and Recall.

w/o Backoff Analysis. Removing the generative backoff module prevents the architecture from recovering answers that the judge fails to validate. As shown in Table 3, all four metrics consistently drop on both datasets, indicating that backoff effectively recovers correct answers that either lack reachable reasoning paths or are filtered out in earlier stages.

w/o Judge Analysis. Replacing the judge with a purely generative decision makes the model directly output answers from GPT-4o given the retrieved candidates and reasoning paths. Although precision increases, this variant reduces Hit@1 by 2.82% on WebQSP and 7.78% on CWQ, indicating that the model becomes overly conservative and discards many low-confidence candidate answers. Additionally, compared with GCR—which similarly feeds candidates and reasoning paths into

Method	WebQSP				CWQ			
	Hit@1	F1	Precision	Recall	Hit@1	F1	Precision	Recall
OntGQA	**91.46**	**76.59**	75.38	**86.52**	**73.80**	55.84	52.95	**69.94**
w/o Backoff	89.68	75.27	73.97	85.00	67.77	50.76	47.50	64.69
w/o Judge	88.64	75.94	**81.26**	78.08	66.02	**57.37**	**58.28**	61.59
w/o Ontology Graph	86.55	74.98	77.30	79.09	57.07	48.41	49.07	52.69

Table 3: Ablation experiment results on WebQSP and CWQ.

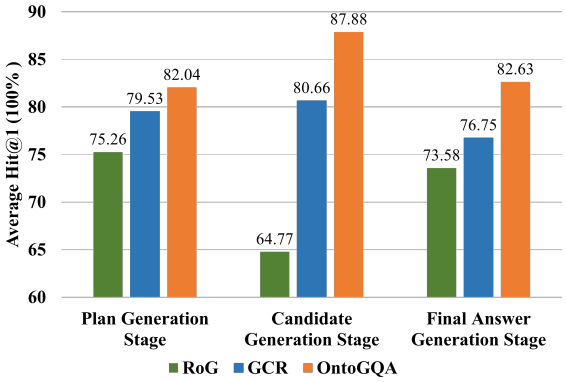
an LLM—our approach remains more competitive, highlighting that OntGQA already produces more reliable candidates and paths before the judge.

w/o Ontology Graph Analysis. Without the ontology graph, the architecture cannot obtain type-constrained relation paths from entity-type pairs. For a fair comparison, we use RoG-generated relation paths as the planning component for the subsequent inference. Performance drops substantially across all metrics on both datasets, confirming that the ontology graph is crucial for retrieving type-consistent reasoning paths, improving recall, and ultimately enhancing overall QA performance.

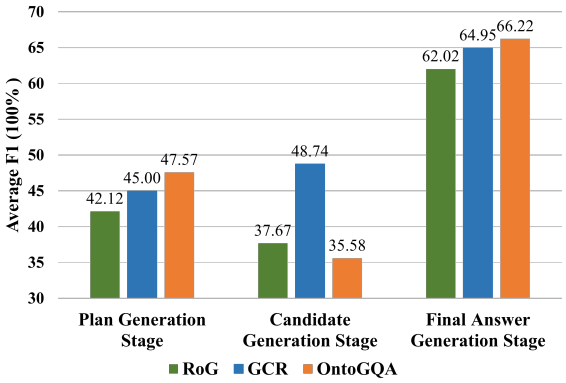
4.4 Hop-wise Analysis of Boundary Gating

To further examine whether the effect of boundary gating extends beyond fully constrained short-hop cases, we conduct a hop-wise analysis on CWQ. As shown in Figure 7, WebQSP contains only 1–2 hop questions, whereas CWQ includes a smaller but non-negligible portion of $\geq$3-hop questions (20.75%). We therefore perform the hop-wise analysis only on CWQ. Table 4 reports two complementary perspectives: hop-wise performance for both final answers and reasoning paths before answer generation, and the average number of candidate paths with and without boundary gating.

The results show that OntGQA remains effective on the $\geq$3-hop subsets rather than benefiting only from short-hop questions. For final answers, OntGQA achieves Hit@1/F1 of 67.02/52.57 on 3-hop questions and 59.52/47.38 on 4-hop questions. At the reasoning-path level, Hit@1 also remains high across hop buckets, suggesting that ontology-guided retrieval can still identify useful paths even when intermediate hops are only partially constrained. At the same time, boundary gating is critical for tractability. Removing the boundary gates leads to severe candidate-path explosion, increasing the average number of candidate paths from 170.77 to 114,849.10 overall, with



(a) Average Hit@1 at each stage.



(b) Average F1 at each stage.

Figure 3: Stage-wise performance comparison of RoG, GCR, and OntGQA.

particularly sharp growth on the longer-hop subsets. These results indicate that boundary gating is important not only for preserving answer quality, but also for keeping multi-hop retrieval computationally manageable.

4.5 Stage-wise Performance

We conduct a stage-wise comparison with RoG and GCR to quantify the contribution of each OntGQA component beyond strong KG–LLM baselines, where the average bars report the mean performance across WebQSP and CWQ.

At the plan generation stage, RoG and GCR di-

(a) Hop-wise performance on CWQ						
Result Type	Metric	Overall	1-hop	2-hop	3-hop	4-hop
Final answers	Hit@1	73.80	83.59	82.19	67.02	59.52
	F1	55.84	56.74	63.75	52.57	47.38
Reasoning paths	Hit@1	94.37	96.69	94.01	81.91	95.24
	F1	23.26	8.18	29.58	19.96	28.67
(b) Candidate-path explosion on CWQ						
Model		Overall	1-hop	2-hop	3-hop	4-hop
OntGQA		170.77	341.12	107.14	80.12	42.88
w/o Boundary gates		114849.10	204776.85	75317.65	138404.72	154345.74

Table 4: Hop-wise performance and candidate-path explosion on CWQ.

Types	Variants	WebQSP		CWQ	
		Hit@1	F1	Hit@1	F1
Only fine-tuned	Qwen2-1.5B	89.93	76.27	70.35	51.61
	Qwen2-7B	90.66	**79.06**	**74.91**	**58.46**
	Llama-2-7B	**91.52**	77.11	74.57	56.76
Fine-tuned +prompt	GPT-4o	91.46	76.59	73.80	55.84
	DeepSeek-v3	91.46	76.64	73.21	55.11

Table 5: Performance of different model variants on WebQSP and CWQ.

Metric	GCR	OntGQA
Fine-tuning samples on WebQSP	28,307	19,692
Fine-tuning samples on CWQ	181,602	100,693
Fine-tuning time (GPU hours)	13.93	7.90
Average inference runtime (s)	3.60	2.57

Table 6: Efficiency comparison of different methods.

rectly decode relation paths, whereas OntGQA predicts head–tail entity-type pairs and derives type-constrained paths. As shown in Figure 3, OntGQA achieves the highest average Hit@1 and F1, indicating that type-based planning is more stable and informative than generating full relation paths.

The candidate generation stage evaluates the quality of retrieved answer sets before making final predictions. OntGQA achieves the best Hit@1 but a lower F1, deliberately producing a high-recall, low-precision candidate pool where correct answers are usually covered but mixed with many distractors. This motivates a dedicated judge rather than aggressive pruning during generation.

At the answer generation stage, OntGQA surpasses both RoG and GCR in Hit@1 and F1, demonstrating the effectiveness of our ontology-guided design. By constraining both endpoints of reasoning in type space and combining a planner–judge architecture with generative backoff, OntGQA delivers consistent gains over existing pipelines at every stage.

4.6 Variant Performance

Table 5 reports the effect of different backbone LLMs in our framework on WebQSP and CWQ. Among the purely fine-tuned open-source models, Qwen2-7B delivers the best overall F1 and a strong balance across both datasets. Using API-based models with our prompts does not bring further gains: GPT-4o and DeepSeek-v3 achieve similar Hit@1 but lower F1 scores, especially on CWQ. This suggests that our ontology-guided framework can already make effective use of supervision with a moderate-size open-source backbone, and stronger proprietary LLMs offer only limited additional benefit.

4.7 Efficiency Analysis

Table 6 summarizes the fine-tuning and inference costs of GCR and OntGQA. GCR enumerates all shortest KG reasoning paths for each question and creates one training instance per path. OntGQA instead collapses these paths into entity-type pairs, yielding 4,288 and 34,293 planner supervisions on WebQSP and CWQ. It then constructs a balanced contrastive judge set by pairing each positive with one negative (7,702×2 on WebQSP and 33,200×2 on CWQ). Thus, the total number of fine-tuning samples is 4,288+7,702×2=19,692 on WebQSP

and 34,293+33,200×2=100,693 on CWQ, consistent with Table 6. Overall, OntGQA uses roughly 30–45% fewer training samples than GCR and reduces training time to 7.9 GPU hours. At inference time, OntGQA achieves its best performance with an average runtime of 2.57 seconds, which is both faster and more economical than GCR. These gains stem from our ontology graph, which enables reasoning in a compact type space and supports an efficient judge without enumerating KG paths.

5 Related Work

Knowledge-graph question answering (KGQA). Early studies explored compact neural representations that map entities and relations into vector spaces (e.g., KV-Mem (Miller et al., 2016); NSM (He et al., 2021)) and retrieval-style architectures that explicitly assemble subgraphs or paths for inference (e.g., GraftNet (Sun et al., 2018); SR (Zhang et al., 2022). While effective for single-hop or shallow compositions, these models often underutilize long-range structure and struggle to maintain faithfulness in multi-hop reasoning. With the advent of large language models (LLMs), fine-tuned approaches such as RoG (Luo et al., 2024), UniKGQA (Jiang et al., 2023), and DeCAF (Yu et al., 2022) achieve strong compositional reasoning by learning to produce paths and answers end-to-end. In parallel, training-free prompting methods (e.g., MindMap (Wen et al., 2024); relation-inventory prompting (Chen et al., 2024)) reduce engineering overhead by steering a frozen LLM to explore the KG, yet they often lack deep structural grounding, yielding noisy or incomplete paths and unstable answers under KG sparsity.

LLM–Ontology Integration. Research on integrating LLMs with ontologies has accelerated, spanning ontology construction (Cohen et al., 2023; Funk et al., 2023), enhancement (Zaitoun et al., 2023; Toro et al., 2024) and learning (Babaei Giglou et al., 2023). Foundationally, Chen and Zhao (2018) extract domain knowledge from Freebase RDF dumps to build semantic bases with high precision and coverage, while Kommineni et al. (2024) propose a capability-question–driven, semi-automated ontology construction workflow—together establishing feasibility and an engineering substrate for downstream tasks. For KGQA, Jiang et al. (2025) and Liu et al. (2025b) introduce an ontology-guided prompting strategy and a reverse-thinking framework to strengthen multi-hop reasoning and generalization, and OntoTune (Liu et al., 2025c) aligns LLMs to domain ontologies via ontology-driven self-training to enhance organization and interpretability. Despite these advances, most approaches depend on prompt engineering and model-internal knowledge, leaving them vulnerable to hallucination and consistency issues and unable to substitute expert-curated ontologies and community consensus.

6 Conclusion

In this paper, we construct a reusable, relation-centric ontology graph for Freebase and propose a Planner–Judge architecture with generative backoff for multi-hop knowledge graph question answering. The planner predicts head–tail type pairs, the ontology graph then guides the retrieval of candidate answers and reasoning paths, and the judge–generator module selects or generates the final answer. Experimental results show that our method improves answer accuracy and coverage, and the resulting ontology graph can be directly reused across all datasets built on Freebase.

Limitations

In this work, we highlight two main directions for improvement. First, our ontology graph is instantiated only on Freebase and evaluated on WebQSP and CWQ, so assessing the portability of the ontology construction pipeline and planner–judge architecture to other KGs (e.g., DBpedia or domain-specific graphs) remains an important direction. Second, OntGQA assumes an offline, precomputed ontology and moderate-size LLM backbones; extending it to dynamically evolving KGs and schemas is a promising avenue for future work.

Ethical Considerations

This work does not raise significant ethical concerns. All data and models used in this paper are publicly available and are used in accordance with their respective licenses, including the Creative Commons BY 4.0 License, the MIT License, the Apache License 2.0, and the LLaMA 2 Community License Agreement. Potential risks are limited but include reliance on external LLM APIs and the possibility of generating plausible yet incorrect reasoning chains; we mitigate these risks through type constraints and evidence-based judging.

References

Hamed Babaei Giglou, Jennifer D'Souza, and Sören Auer. 2023. LLMs4OL: Large language models for ontology learning. In *International Semantic Web Conference*, pages 408–427. Springer.

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*, pages 1247–1250.

Deyan Chen and Hong Zhao. 2018. Research on the method of extracting domain knowledge from the freebase RDF dumps. *IEEE Access*, 6:50306–50322.

Yixuan Chen, Dongsheng Li, Peng Zhang, Jie Sui, Qin Lv, Lu Tun, and Li Shang. 2022. Cross-modal ambiguity learning for multimodal fake news detection. In *Proceedings of the ACM on Web Conference 2022*, pages 2897–2905.

Zhongwu Chen, Long Bai, Zixuan Li, Zhen Huang, Xiaolong Jin, and Yong Dou. 2024. A new pipeline for knowledge graph reasoning enhanced by large language models without fine-tuning. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 1366–1381.

Roi Cohen, Mor Geva, Jonathan Berant, and Amir Globerson. 2023. Crawling the internal knowledge-base of language models. In *Findings of the Association for Computational Linguistics: EACL 2023*, pages 1856–1869. Association for Computational Linguistics.

DeepSeek-AI, Aixin Liu, Bei Feng, Bing Xue, Bing-Li Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Daya Guo, Dejian Yang, Deli Chen, Dong-Li Ji, Erhang Li, Fangyun Lin, Fucong Dai, and 179 others. 2024. DeepSeek-V3 Technical Report. *ArXiv*, abs/2412.19437.

Giacomo Frisoni, Alessio Cocchieri, Alex Presepi, Gianluca Moro, and Zaiqiao Meng. 2024. To generate or to retrieve? on the effectiveness of artificial contexts for medical open-domain question answering. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9878–9919.

Maurice Funk, Simon Hosemann, Jean Christoph Jung, and Carsten Lutz. 2023. Towards ontology construction with language models. In *Joint proceedings of the 1st workshop on Knowledge Base Construction from Pre-Trained Language Models (KBC-LM) and the 2nd challenge on Language Models for Knowledge Base Construction (LM-KBC) co-located with the 22nd International Semantic Web Conference (ISWC 2023)*, volume 3577.

Gaole He, Yunshi Lan, Jing Jiang, Wayne Xin Zhao, and Ji-Rong Wen. 2021. Improving multi-hop knowledge base question answering by learning intermediate supervision signals. In *Proceedings of the 14th ACM international conference on web search and data mining*, pages 553–561.

Jinhao Jiang, Kun Zhou, Xin Zhao, and Ji-Rong Wen. 2023. Unikgqa: Unified retrieval and reasoning for solving multi-hop question answering over knowledge graph. In *The Eleventh International Conference on Learning Representations*.

Longquan Jiang, Junbo Huang, Cedric Moller, and Ricardo Usbeck. 2025. Ontology-guided, hybrid prompt learning for generalization in knowledge graph question answering. In *2025 19th International Conference on Semantic Computing (ICSC)*, pages 28–35. IEEE Computer Society.

Vamsi Krishna Kommineni, Birgitta König-Ries, and Sheeba Samuel. 2024. From human experts to machines: An LLM supported approach to ontology and knowledge graph construction. *arXiv preprint arXiv:2403.08345*.

Ke Liang, Lingyuan Meng, Yue Liu, Meng Liu, Wei Wei, Suyuan Liu, Wenxuan Tu, Siwei Wang, Sihang Zhou, and Xinwang Liu. 2024. Simple yet effective: Structure guided pre-trained transformer for multimodal knowledge graph reasoning. In *Proceedings of the 32nd ACM International Conference on Multimedia*, pages 1554–1563.

Ben Liu, Jihai Zhang, Fangquan Lin, Cheng Yang, Min Peng, and Wotao Yin. 2025a. Symagent: A neural-symbolic self-learning agent framework for complex reasoning over knowledge graphs. In *Proceedings of the ACM on Web Conference 2025*, pages 98–108.

Runxuan Liu, Luobei Luobei, Jiaqi Li, Baoxin Wang, Ming Liu, Dayong Wu, Shijin Wang, and Bing Qin. 2025b. Ontology-guided reverse thinking makes large language models stronger on knowledge graph question answering. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15269–15284, Vienna, Austria. Association for Computational Linguistics.

Zhiqiang Liu, Chengtao Gan, Junjie Wang, Yichi Zhang, Zhongpu Bo, Mengshu Sun, Huajun Chen, and Wen Zhang. 2025c. Ontotune: Ontology-driven self-training for aligning large language models. In *Proceedings of the ACM on Web Conference 2025*, pages 119–133.

Linhao Luo, Yuanfang Li, Gholamreza Haffari, and Shirui Pan. 2024. Reasoning on graphs: Faithful and interpretable large language model reasoning. In *The Twelfth International Conference on Learning Representations*.

Linhao Luo, Zicheng Zhao, Gholamreza Haffari, Yuan-Fang Li, Chen Gong, and Shirui Pan. 2025. Graph-constrained reasoning: Faithful reasoning on knowledge graphs with large language models. In *Forty-second International Conference on Machine Learning*.

Costas Mavromatis and George Karypis. 2025. GNN-RAG: Graph neural retrieval for efficient large language model reasoning on knowledge graphs. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 16682–16699.

Meta. 2024. Build the Future of AI with Meta Llama 3.

Microsoft. 2015. Fastrdfstore-data.

Alexander Miller, Adam Fisch, Jesse Dodge, Amir-Hossein Karimi, Antoine Bordes, and Jason Weston. 2016. Key-value memory networks for directly reading documents. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1400–1409.

George A Miller. 1995. WordNet: a lexical database for English. *Communications of the ACM*, 38(11):39–41.

OpenAI. 2022. Introducing ChatGPT.

OpenAI. 2024. Hello gpt-4o.

Shirui Pan, Linhao Luo, Yufei Wang, Chen Chen, Jiapu Wang, and Xindong Wu. 2024. Unifying large language models and knowledge graphs: A roadmap. *IEEE Transactions on Knowledge and Data Engineering*, 36(7):3580–3599.

Haitian Sun, Bhuwan Dhingra, Manzil Zaheer, Kathryn Mazaitis, Ruslan Salakhutdinov, and William Cohen. 2018. Open domain question answering using early fusion of knowledge bases and text. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4231–4242.

Jiashuo Sun, Chengjin Xu, Lumingyuan Tang, Saizhuo Wang, Chen Lin, Yeyun Gong, Lionel 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*.

Alon Talmor and Jonathan Berant. 2018. The web as a knowledge-base for answering complex questions. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 641–651.

Xingyu Tan, Xiaoyang Wang, Qing Liu, Xiwei Xu, Xin Yuan, and Wenjie Zhang. 2025. Paths-over-graph: Knowledge graph empowered large language model reasoning. In *Proceedings of the ACM on Web Conference 2025*, pages 3505–3522.

WANG Ting, WANG Na, CUI Yunpeng, and LIU Juan. 2023. Agricultural technology knowledge intelligent question-answering system based on large language model. *Smart agriculture*, 5(4):105.

Sabrina Toro, Anna V. Anagnostopoulos, Sue Bello, Kai Blumberg, Rhiannon Cameron, Leigh Carmody, Alexander D. Diehl, Damion M. Dooley, William Duncan, Petra Fey, Pascale Gaudet, Nomi L. Harris, marcin p. joachimiak, Leila Kiani, Tiago Lubiana, Monica C. Munoz-Torres, Shawn T. O'Neil, David Osumi-Sutherland, Aleix Puig, and 11 others. 2024. Dynamic retrieval augmented generation of ontologies using artificial intelligence (dragon-ai). *Journal of Biomedical Semantics*, 15(1):19.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, and 49 others. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Jimmy Donal Wales and Larry Sanger. 2001. Wikipedia.

Yilin Wen, Zifeng Wang, and Jimeng Sun. 2024. Mindmap: Knowledge graph prompting sparks graph of thoughts in large language models. In *62nd Annual Meeting of the Association for Computational Linguistics, ACL 2024*, pages 10370–10388. Association for Computational Linguistics (ACL).

Wen-tau Yih, Matthew Richardson, Christopher Meek, Ming-Wei Chang, and Jina Suh. 2016. The value of semantic parse labeling for knowledge base question answering. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 201–206.

Donghan Yu, Sheng Zhang, Patrick Ng, Henghui Zhu, Alexander Hanbo Li, Jun Wang, Yiqun Hu, William Yang Wang, Zhiguo Wang, and Bing Xiang. 2022. Decaf: Joint decoding of answers and logical forms for question answering over knowledge bases. In *The Eleventh International Conference on Learning Representations*.

Antonio Zaitoun, Tomer Sagi, Szymon Wilk, and Mor Peleg. 2023. Can large language models augment a biomedical ontology with missing concepts and relations? *arXiv preprint arXiv:2311.06858*.

Jing Zhang, Xiaokang Zhang, Jifan Yu, Jian Tang, Jie Tang, Cuiping Li, and Hong Chen. 2022. Subgraph retrieval enhanced model for multi-hop knowledge base question answering. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5773–5784.

A Optimization Objective

For each plan τ, retrieval returns a set of candidate–evidence pairs $\mathcal{C}(\tau, \mathcal{O}, \mathcal{G}) = \{(a', \mathbf{w}_{a'})\}$. We form the planner–judge answer distribution by marginalizing planning uncertainty and aggregating judged candidates:

$$P_{\text{pj}}(a) = \sum_{\tau \in \mathcal{T}} \underbrace{P_\phi(\tau \mid q)}_{\text{Type planner}} \sum_{(a', \mathbf{w}_{a'}) \in \mathcal{C}(\tau, \mathcal{O}, \mathcal{G})} \delta[a{=}a'] \underbrace{P_\theta(1 \mid q, a', \mathbf{w}_{a'})}_{\text{Judge(YES)}} \tag{12}$$

where $\tau = (\tau_h, \tau_t) \in \mathcal{T}$ is the head–tail type pair predicted by the planner; $P_\phi(\tau \mid q)$ is the planner prior over type pairs; $\mathcal{C}(\tau, \mathcal{O}, \mathcal{G})$ is the set of candidate–evidence pairs instantiated on $\mathcal{G}$ by ontology-compatible relation paths from the question entity of q; each element contains a candidate entity $a' \in \mathcal{E}$ and an evidence multiset of reasoning paths $\mathbf{w}_{a'} = \{w^1_{a'}, \ldots, w^m_{a'}\}$ $(m \geq 1)$; $\delta[\cdot]$ is the Kronecker delta selecting the mass for a; and $P_\theta(1 \mid q, a', \mathbf{w}_{a'})$ is the judge's acceptance probability for candidate a' given evidence $\mathbf{w}_{a'}$. The inner sum aggregates path-level acceptance for each candidate, and the outer sum marginalizes over plans.

B Prompt Template

OntGQA relies on three instruction-style prompt templates, one for each module. The ontology-guided type planner uses the template in Figure 4, the judge module adopts the verification template in Figure 5, and the generator module leverages the fallback generation template in Figure 6.

Ontology-Guided Planner

Input:
<TASK: TYPE_PAIRS>
Please generate a valid type pair that can be helpful for answering the following question:
Question: `[Question]`
Output:
`<PAIR> [Head type] <SEP> [Tail type] </PAIR>`

Figure 4: Prompt template for the ontology-guided type planner.

C Datasets Statistics

In Figure 7, we further report the distribution of question hops on WebQSP and CWQ. On WebQSP, the vast majority of questions are relatively shallow, with 65.49% of questions solvable with a single reasoning hop and the remaining 34.51% requiring two hops; no questions in our analysis fall into the category of three or more hops. In contrast, CWQ exhibits a much heavier multi-hop bias: only 40.91% of questions are 1-hop, 38.34% require 2 hops, and as many as 20.75% involve at least 3 hops. These statistics highlight that CWQ contains substantially more compositional and long-range reasoning queries than WebQSP, motivating the different maximum reasoning depths used for the two datasets.

Judge

Input:
You are a strict judge for knowledge-graph QA. Given a QUESTION, a CANDIDATE answer, and its EVIDENCE PATHS (if any), decide whether the candidate is a correct final answer to the question. Return strictly one token: "YES" or "NO". No explanations.
Question: `[Question]`
Candidate: `[Candidate answer]`
Evidence paths: `[Evidence paths]`
Output:
`[YES/NO]`

Figure 5: Prompt template for the judge module.

Generator

Input:
<TASK: ANSWERS>
Please generate ALL correct answer entities for the following question:
Question: `[Question]`
Output:
`[Answers]`

Figure 6: Prompt template for the generator module.

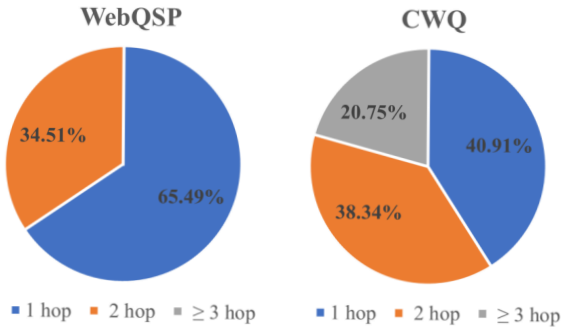


Figure 7: Statistics of the question hops in WebQSP and CWQ.

D Ontology Graph Statistics

Table 8 summarizes the statistics of the ontology graph construction pipeline. On a single

Category	Relation
Common to both datasets (19)	kp_lw.philosopher.main_interests meteorology.tropical_cyclone.strongest_storm_of kp_lw.philosophy_influencer.influencee base.sportbase.sport_sport_club.player_role owl#inverseOf location.australian_state.premiers sports.sports_team.rivalry kp_lw.philosopher.influenced_by base.saturdaynightlive.snl_musical_performance.song location.australian_state.governors time.participant.event kp_lw.philosophy_influencee.influencer kp_lw.philosopher.region time.event.participant kp_lw.philosopher.era time.event.previous_in_series kp_lw.philosopher.school time.event.next_in_series sports.team_rivalry.team
CWQ only (2)	biology.organism_classification.lower_classification_closure kp_lw.philosopher.influenced
WebQSP only (0)	—

Table 7: Missing or under-specified relations in the RDF dump, consolidated across datasets. Duplicates are merged into the "Common to both datasets" group.

Item	Value
Property–schema entries (single machine)	71,210
Total pipeline time (single machine)	$\approx$ 1.93 h
Missing relations detected in CWQ	21
Missing relations detected in WebQSP	19
Relations missing in both datasets	19
Relations in ontology graph	32,195
Entity types in ontology graph	12,369

Table 8: Statistics of the ontology construction pipeline.

machine, the pipeline extracts 71,210 property-schema entries and takes approximately 6,958 seconds ($\approx$1.93 hours). After filtering non-semantic/administrative types, we obtain an initial set of relations with complete head–tail type signatures, which constitutes the main body of our ontology graph.

We additionally audit KGQA datasets to identify relations whose head–tail signatures are missing or under-specified in the RDF-derived schema. This set is small: CWQ contains 21 such relations and WebQSP contains 19, of which 19 are common to both datasets (Table 7). For these relations only, we perform dataset-aware completion using training splits exclusively to avoid test leakage. Concretely, for each identified relation, we aggregate the head-entity set and tail-entity set observed in training triples/questions, infer their corresponding Freebase types, and accept a completion only when both head and tail types can be inferred. Since completion touches only a limited number of relations relative to the full ontology (32,195 relations in total), it primarily improves schema completeness and coverage for rare relations; removing this step affects only this small subset.

The final ontology graph contains 32,195 relations and 12,369 entity types. A consolidated list of missing/under-specified relations, grouped by whether they are common to both datasets or unique to one, is reported in Table 7. To facilitate reproducibility and further research, we will publicly release the constructed ontology graph (relation–type triples) together with the full list of completed relations upon publication.

E Bootstrap Confidence Intervals for Main Results

To assess the stability of the reported results, we further compute 95% confidence intervals (CIs) for Hit@1 and F1 using a single-seed bootstrap over test predictions. As shown in Table 9, the intervals are reasonably tight on both WebQSP and CWQ, indicating that the reported performance is stable under test-set resampling. Notably, both OntGQA variants achieve consistently narrow intervals across datasets and metrics, which provides additional evidence that the observed improvements are not caused by incidental variation on the test set.

Model	WebQSP Hit@1	WebQSP F1	CWQ Hit@1	CWQ F1
OntGQA (Llama-2-7B + GPT-4o)	91.46 [90.11, 92.75]	76.59 [75.02, 78.19]	73.80 [72.39, 75.22]	55.84 [54.52, 57.16]
OntGQA (Llama-2-7B)	91.52 [89.77, 93.18]	77.11 [75.11, 79.10]	74.57 [73.15, 75.98]	56.76 [55.44, 58.11]

Table 9: 95% confidence intervals (CIs) for OntGQA, estimated via single-seed bootstrap over test predictions with 10,000 resamples.

k	Hit	F1	Precision	Recall
1	54.9	47.0	54.9	45.0
2	66.1	52.9	54.4	57.8
3	70.5	54.9	54.0	63.6
5	73.0	56.1	53.7	67.9
10	74.3	56.7	53.6	70.4

Table 10: CWQ top-k diagnostic curve for OntGQA (macro-averaged). Increasing k improves recall/hit while precision decreases mildly, illustrating the precision–recall trade-off under set-based evaluation.

F CWQ Hit@1–F1 Diagnostics

Why Hit@1 can increase while F1 decreases on CWQ. CWQ uses set-based evaluation, where returning additional candidates can reduce precision and thus F1 even when the top prediction is correct. This effect is amplified because CWQ frequently contains multi-answer questions: 47.3% of questions have exactly one gold answer, while 16.1% have at least five. Therefore, methods that favor higher recall (e.g., outputting multiple candidates to avoid missing gold answers) may achieve higher Hit@1 but slightly lower F1.

Diagnostics. Table 10 reports a top-k diagnostic curve of OntGQA on CWQ, showing the expected recall gain with increasing k under a modest precision drop. To further understand cases where top-1 is correct but F1 is penalized, we conduct a manual audit of 200 sampled instances. As summarized in Table 11, most penalties come from incorrect predictions (81.5%), while the remainder mainly arise from evaluation-confounding factors such as ambiguous intent, surface-form aliasing, and missing gold labels. These cases can reduce set-based F1 even when Hit@1 improves.

G Sensitivity to the Judge Margin

Figure 8 reports the effect of the judge margin τ on Hit@1 and F1 for WebQSP and CWQ. On WebQSP, increasing τ from 0 to 2 slightly decreases Hit@1 (from $\sim$ 92 to $\sim$ 90) while F1 steadily improves and peaks around $\tau = 2$ ($\sim$ 78). Larger values make the judge overly conservative, causing both metrics to drop. CWQ exhibits a similar pattern: Hit@1 monotonically decreases as τ grows, whereas F1 gently rises and reaches its maximum around $\tau \in [2, 2.5]$ ($\sim$ 57) before declining again.

Overall, τ controls the trade-off between retaining more candidate paths (low τ, higher recall) and aggressively filtering noisy paths (high τ, higher precision but lower recall). Since both datasets show relatively stable performance for τ in a moderate range and $\tau = 1$ offers a good balance between high Hit@1 and competitive F1, we fix $\tau = 1$ in all main experiments.

Penalty Source	Count (#)	Proportion (%)
Incorrect Predictions	163	81.5
Ambiguous Intent	17	8.5
Surface Form Alias	16	8.0
Missing Gold Labels	4	2.0
Total	200	100.0

Table 11: Manual audit of 200 CWQ cases with set-based F1 penalties despite correct top-1 predictions.

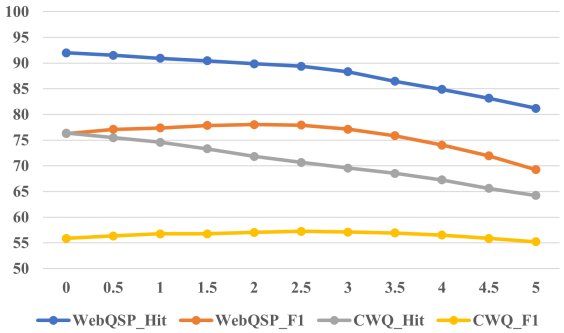


Figure 8: Effect of the margin threshold in WebQSP and CWQ.