What Breaks Knowledge Graph based RAG? Benchmarking and Empirical Insights into Reasoning under Incomplete Knowledge

Dongzhuoran Zhou[1,2], Yuqicheng Zhu[2,3], Xiaxia Wang[5], Hongkuan Zhou[2,3], Yuan He[4,5], Jiaoyan Chen[6], Steffen Staab[3,7], Evgeny Kharlamov[1,2]
[1]University of Oslo, [2]Bosch Center for AI, [3]University of Stuttgart, [4]Amazon,
[5]University of Oxford, [6]The University of Manchester, [7]University of Southampton
dongzhuoran.zhou@de.bosch.com

arXiv:2508.08344v4 [cs.AI] 12 Jan 2026

Abstract

Knowledge Graph-based Retrieval-Augmented Generation (KG-RAG) is an increasingly explored approach for combining the reasoning capabilities of large language models with the structured evidence of knowledge graphs. However, current evaluation practices fall short: existing benchmarks often include questions that can be directly answered using existing triples in KG, making it unclear whether models perform reasoning or simply retrieve answers directly. Moreover, inconsistent evaluation metrics and lenient answer matching criteria further obscure meaningful comparisons. In this work, we introduce a general method for constructing benchmarks and present **BRINK (Benchmark for Reasoning under Incomplete Knowledge)** to systematically assess KG-RAG methods under knowledge incompleteness. Our empirical results show that current KG-RAG methods have limited reasoning ability under missing knowledge, often rely on internal memorization, and exhibit varying degrees of generalization depending on their design.

1 Introduction

Retrieval-Augmented Generation (RAG) has become a widely adopted framework for enhancing large language models (LLMs) by incorporating external knowledge through a retrieve-then-generate paradigm (Khandelwal et al., 2020; Izacard and Grave, 2021; Borgeaud et al., 2022; Ram et al., 2023; Zhu et al., 2025b). By conditioning the generation on the retrieved documents, RAG enables LLMs to answer questions or perform tasks using more comprehensive and up-to-date knowledge than what is stored in their parameters. To improve retrieval accuracy, enable structured reasoning and support explanation, recent research has pivoted toward RAG methods based on Knowledge Graph (KG-RAG) (Han et al., 2024; Peng et al., 2024). Most of them directly use existing knowledge graphs (KGs) (Luo et al., 2024; Sun et al., 2024; Zhou et al., 2025b), while some of them construct and extend structured knowledge from unstructured documents (Fang et al., 2024). Such KG-RAG systems are expected to be well-suited for structured reasoning, where answers require synthesizing information from multiple connected facts.

Despite growing interest, current evaluation practices for KG-RAG fall short in two key ways. First, most existing benchmarks (Yih et al., 2016; Talmor and Berant, 2018a) are constructed on top of complete KGs, where direct evidence supporting the answer is readily available. For example, given the question "*Who is the brother of Justin Bieber?*", the KG contains the triple `hasBrother(JustinBieber, JaxonBieber)`, allowing the system to answer the question without reasoning. However, real-world KGs are often incomplete, and answering such questions in practice may require reasoning over alternative paths, e.g., combining `hasParent(JustinBieber, JeremyBieber)` and `hasChild(JeremyBieber, JaxonBieber)` to infer the sibling relationship. As a result, current benchmarks do not assess whether KG-RAG methods can reason over missing knowledge or simply retrieve answers directly from explicit evidence.

Second, evaluation protocols across KG-RAG studies lack standardization and rigor. We identify two pervasive issues: (1) ambiguous definitions, where metrics like "accuracy" fluctuate arbitrarily between Exact Match and permissive substring inclusion; and (2) implementation discrepancies, where official codebases frequently contradict their paper descriptions. For instance, several benchmarks describe using ranking-based metrics (e.g., Hits@1) but implement them as loose existence checks without any ranking. These methodological lapses result in inflated performance estimates and render cross-paper comparisons unreliable.

In this work, we introduce **BRINK (Benchmark**

for Reasoning under Incomplete Knowledge). Constructed using a novel generalizable method, BRINK includes an evaluation protocol designed to systematically assess KG-RAG methods under knowledge incompleteness. Each question in our benchmark is constructed such that it *cannot* be answered using a single explicit triple. Instead, the answer must be inferred by reasoning over alternative paths in the KG. To construct this benchmark, we follow a two-step process: (1) we mine high-confidence logical rules from the KG using a rule mining algorithm, and (2) we generate natural language questions based on rule groundings, ensuring that the directly supporting triple is removed while the remaining KG still contains sufficient evidence to infer the answer.

Our empirical study on BRINK reveals several key limitations of current KG-RAG systems. First, most models struggle to recover answers when direct supporting facts are removed, highlighting their limited reasoning capacity. Second, training-based methods (e.g., RoG, GNN-RAG) show stronger robustness under KG incompleteness compared to non-trained systems. Third, we find that textual entity labels substantially boost performance, suggesting that LLMs rely heavily on internal memorization.

2 Related Work

2.1 Knowledge Graph Question Answering (KGQA)

The KGQA task aims to answer natural language questions using the KG $\mathcal{G}$, where $\mathcal{G}$ is represented as a set of binary facts $r(s, o)$, with r denoting a predicate and s, o denoting entities. The answer to each question is one or more entities in $\mathcal{G}$.

Approaches to KGQA can be broadly categorized into the following two types: (1) *Semantic parsing-based methods* (Yih et al., 2016; Gu et al., 2021; Ye et al., 2021) translate questions into formal executable queries (e.g., SPARQL or logical forms), which are then executed over the KG to retrieve the answer entities. These methods offer high precision and interpretability, as the reasoning process is explicitly encoded. However, they face several challenges, including difficulty in understanding semantically and syntactically complex questions, handling diverse and compositional logical forms, and managing the large search space involved in parsing multi-relation queries(Lan et al., 2021). (2) *Embedding-based methods* (Yao et al., 2019; Baek et al., 2023), by contrast, encode questions and entities into a shared vector space and rank answer candidates based on embedding similarity. While these methods are end-to-end trainable and do not require annotated logical forms, they often struggle with multi-hop reasoning (Qiu et al., 2020), lack interpretability (Biswas et al., 2023), and exhibit high uncertainty in their predictions (Zhu et al., 2024, 2025c,a,d).

2.2 KGQA with RAG

Recently, a new line of work has emerged that integrates large language models more tightly into the KG reasoning process. These KG-RAG methods go beyond previous KGQA approaches by coupling structured retrieval with generative reasoning capabilities of LLMs. RoG (Luo et al., 2024) adopts a planning-retrieval-reasoning pipeline, where an LLM generates relation paths as plans, retrieves corresponding paths from the KG, and reasons over them for answer generation. G-Retriever (He et al., 2024) retrieves subgraphs via Prize-Collecting Steiner Tree optimization and provides them to LLMs as soft prompts or text prompts for question answering. GNN-RAG (Mavromatis and Karypis, 2024) uses a GNN to select candidate answers and retrieves shortest paths to them, which are verbalized for LLM-based answer generation. PoG (Chen et al., 2024) instead decomposes questions into subgoals and iteratively explores and self-corrects reasoning paths using a guidance-memory-reflection mechanism. ToG (Sun et al., 2024) treats the LLM as an agent that interactively explores and aggregates evidence on KGs through iterative beam search, StructGPT (Jiang et al., 2023) introduces an interface-based framework that enables LLMs to iteratively extract relevant evidence from structured data such as KGs and reason over the retrieved information to answer complex questions.

2.3 KGQA Benchmarks

Existing benchmarks such as WebQSP (Yih et al., 2016), CWQ (Talmor and Berant, 2018b), and GrailQA (Gu et al., 2021) are widely used to evaluate KGQA. However, during dataset construction, only question-answer pairs that yield a gold SPARQL answer on the reference KG are retained, while any unanswerable questions are discarded (Yih et al., 2016; Talmor and Berant, 2018b; Gu et al., 2021). This implicitly assumes that each question can be answered directly using an existing fact in the KG, overlooking the reality that

KGs are often incomplete. To address this, recent works (Xu et al., 2024; Zhou et al., 2025a) simulate KG incompleteness by either randomly deleting triples from the whole KG or removing those along the shortest path(s) between the question and answer entities. However, these approaches have a key limitation: they cannot ensure that sufficient knowledge remains in the KG to support answering each question. This can lead to misleading evaluations, as performance drops may stem from unanswerable questions rather than limitations in the model's reasoning ability.

2.4 LLM Reasoning Evaluation

Evaluation of LLM reasoning typically covers both logical question answering and logical consistency, with benchmarks spanning deduction, multiple-choice, proof generation, and consistency constraints such as entailment, negation, and transitivity. Datasets like LogicBench (Parmar et al., 2024), ProofWriter (Tafjord et al., 2020), and LogicNLI (Tian et al., 2021) are widely used for these tasks, but results indicate that LLMs still struggle with robust logical reasoning (Cheng et al., 2025; He et al., 2025).

3 Benchmark Construction

This section presents a general method for constructing benchmarks, evaluating KG-RAG methods that can support different settings of knowledge incompleteness. The key objective is to create natural language questions whose answers are not directly stated in the KG but can be logically inferred through reasoning over alternative paths.

To achieve this, we first mine high-confidence logical rules from the KG to identify triples that are inferable via reasoning. We then remove a subset of these triples while preserving the supporting facts required for inference. Natural language questions are generated based on the removed triples, meaning that models must rely on reasoning rather than direct retrieval to answer the questions.

3.1 Rule Mining

To ensure that questions in our benchmark require reasoning rather than direct lookup, we first identify triples that are logically inferable from other facts. We achieve this by mining high-confidence Horn rules from the original KG using the AMIE3 algorithm (Lajus et al., 2020).

AMIE3 is a widely used rule mining system designed to operate efficiently over large-scale KGs. A logical rule discovered by AMIE3 has the following form (*Horn rules* (Horn, 1951)):

$$B_1 \wedge B_2 \wedge \cdots \wedge B_n \Rightarrow H\,,$$

where each item is called an *atom*, a binary relation of the form $r(X, Y)$, in which r is a predicate and X, Y are variables. The left-hand side of the rule is a conjunction of *body atoms*, denoted as $\mathbf{B} = B_1 \wedge \cdots \wedge B_n$, and the right-hand side is the *head atom* H. Intuitively, a rule expresses that if the body $\mathbf{B}$ holds, then the head H is likely to hold as well.

A *substitution* σ maps every variable occurring in an atom to an entity that exists in $\mathcal{G}$. For example applying $\sigma = \{X \mapsto \texttt{Justin}, Y \mapsto \texttt{Jaxon}\}$ to the atom `hasSibling(X,Y)` yields the grounded fact `hasSibling(Justin,Jaxon)`. A *grounding* of a rule $\mathbf{B} \Rightarrow H$ is

$$\sigma(B_1) \wedge \ldots \sigma(B_n) \Rightarrow \sigma(H)\,.$$

Quality Measure. AMIE3 uses the following metrics to measure the quality of a rule:

- **Support.** The *support* of a rule is defined as the number its groundings for which all grounded facts are observed in the KG:

$$support(\mathbf{B} \Rightarrow H) = \\ |\{\sigma(H) \mid \forall i, \sigma(B_i) \in \mathcal{G} \wedge \sigma(H) \in \mathcal{G}\}|\,.$$

- **Head coverage.** *Head coverage (hc)* measures the proportion of observed head groundings in the KG that are successfully explained by the rule. It is defined as the ratio of the rule's support to the number of head groundings in the KG:

$$hc(\mathbf{B} \Rightarrow H) = \frac{support(\mathbf{B} \Rightarrow H)}{|\{\sigma \mid \sigma(H) \in \mathcal{G}\}|}\,.$$

- **Confidence.** *Confidence* measures the proportion of body groundings that also lead to the head being observed in the KG. It is defined as the ratio of the rule's support to the number body groundings in the KG:

$$confidence(\mathbf{B} \Rightarrow H) = \frac{support(\mathbf{B} \Rightarrow H)}{|\{\sigma \mid \sigma(\mathbf{B}) \in \mathcal{G}\}|}\,.$$

We retain only rules with high confidence and sufficient support, filtering out noisy or spurious

patterns. Specifically, we run AMIE3 with a confidence threshold of 0.3, a head coverage threshold of 0.1, and a maximum rule length of 4. AMIE3 incrementally generates candidate rules via breadth-first refinement (Lajus et al., 2020) and evaluates them using confidence and head coverage; only those meeting the specified thresholds are retained. Additional details on the rule generation and filtering process are provided in Appendix A.

3.2 Dataset Generation

We aim to generate questions that cannot be answered using direct evidence, but for which sufficient information is implicitly available in the KG. The core idea is to first remove triples that can be reliably inferred using high-confidence rules mined by AMIE3, and then generate questions based on these removed triples.

Triple Removal. For each mined rule $\mathbf{B} \Rightarrow H$, we extract up to 30 groundings $\sigma(\mathbf{B} \Rightarrow H)$ such that both the grounded body $\sigma(\mathbf{B})$ and grounded head $\sigma(H)$ exist in the KG. For each such grounding, we remove the head triple $\sigma(H)$ from the KG while preserving all body triples $\sigma(\mathbf{B})$. To ensure that the removed triples remain logically inferable from the remaining KG, we enforce the following two constraints:

- All grounded body triples required to infer a removed head must remain in the KG.
- Removing a head triple must not eliminate any body triple used by other selected groundings.

This guarantees that every removed triple can be inferred through at least one reasoning path provided by a mined rule.

Question Generation. For each removed triple $r(e_h, e_t)$, we use GPT-4 to generate a natural-language question that asks for the answer entity based on the predicate and a specified topic entity. To promote diversity, we randomly designate either the head e_h or the tail e_t as the topic entity, with the other serving as the answer entity. The exact prompt template is provided in Appendix D.

Dataset Balancing. KGs typically exhibit a "long-tail" distribution, where a small number of entities participate in a disproportionately large number of triples, while the majority appear only infrequently (Mohamed et al., 2020; Chen et al., 2023). This imbalance can cause many generated questions to share the same answer entity, leading to biased evaluation.

To reduce answer distribution bias, we apply frequency-based downsampling to the generated questions $\mathcal{Q}$, yielding a more balanced subset $\mathcal{Q}' \subseteq \mathcal{Q}$. As described in Algorithm 1, for each answer entity a, we retain at most $\tau \cdot |\mathcal{Q}|$ questions if a exceeds the frequency threshold τ; otherwise, all associated questions are kept.

Algorithm 1 Downsampling Procedure

Require: Question set $\mathcal{Q}$; threshold $\tau \in (0, 1]$
Ensure: Balanced subset $\mathcal{Q}' \subseteq \mathcal{Q}$

```
Let A ← set of unique answer entities in Q
Q' ← ∅
for all a ∈ A do
    Q_a ← {q ∈ Q | answer(q) = a}
    if |Q_a| > τ · |Q| then
        Randomly sample S_a ⊂ Q_a
        of size ⌊τ · |Q|⌋
    else
        S_a ← Q_a
    Q' ← Q' ∪ S_a
return Q'
```

Answer Set Completion. Although each question is initially generated based on a single deleted triple, there may exist multiple correct answers in the KG. For example, the question "*Who is the brother of Justin Bieber?*" may have several valid answers beyond the one used to generate the question (e.g., `Jaxon Bieber`). To ensure rigorous and unbiased evaluation, we construct for each question a complete set of correct answers using the full KG before any triple deletions. Specifically, for a given topic entity and predicate, we identify all tail entities such that the triple `(topic, predicate, tail)` exists in the KG. All such entities are collected as the answer set for that question.

3.3 BRINK Overview

Knowledge Graphs. To support a systematic evaluation of reasoning under knowledge incompleteness, we construct BRINK based on three well established KGs: **Family** (Sadeghian et al., 2019), **FB15k-237** (Toutanova and Chen, 2015) and **Wikidata5m** (Wang et al., 2021). These datasets differ in size, structure, and domain coverage, enabling evaluation across both synthetic and real-world settings.

- **Family** dataset is a synthetic KG encoding well-defined familial relationships, such as `father`, `mother`, `uncle`, and `aunt`. It is constructed from multiple family units with logically consistent relation patterns and interpretable schema.
- **FB15k-237** is a widely used benchmark derived from Freebase (Dong et al., 2014). The graph spans 14,541 entities and 237 predicates, covering real-world domains such as people, locations, and organizations.
- **Wikidata5m** is a large-scale real-world KG constructed from Wikidata (Vrandečić and Krötzsch, 2014). It contains about 4.6 million entities, 822 relations, and over 20 million triples spanning diverse domains such as people, places, organizations, and events (Wang et al., 2021).

Mined Rules. Table 1 summarizes the number of mined rules for each dataset, categorized by rule type. The listed types (e.g., symmetry, inversion, composition) correspond to common logical patterns, while the *other* category includes more complex or irregular patterns (See Appendix G for details).

Rule Type	Family	FB15k-237	Wikidata5m
Symmetry: $r(x,y) \Rightarrow r(y,x)$	0	27	19
Inversion: $r_1(x,y) \Rightarrow r_2(y,x)$	6	50	48
Hierarchy: $r_1(x,y) \Rightarrow r_2(x,y)$	0	76	0
Composition: $r_1(x,y) \wedge r_2(y,z) \Rightarrow r_3(x,z)$	56	343	0
Other	83	570	168
Total	**145**	**1,066**	**235**

Table 1: Statistics of mined rules.

Datasets in BRINK. Each dataset instance consists of (1) a natural-language question, (2) a topic entity referenced in the question, and (3) a complete set of correct answer entities derived from the original KG. Table 2 presents representative examples from each dataset. The final question set is randomly partitioned into training, validation, and test sets using an 8:1:1 ratio. This split is applied uniformly across all three datasets to ensure consistency.

We provide two retrieval sources per dataset:

- **Complete KG**: the original KG containing all triples.
- **Incomplete KG**: a modified version where selected triples, deemed logically inferable via AMIE3-mined rules, are removed (cf. Section 3.2).

Table 3 summarizes the number of KG triples and generated questions in each split for all three datasets, under complete and incomplete KG settings. In Appendix H, we quantitatively verify that removing triples from the incomplete KG does not render the resulting questions unanswerable.

4 Evaluation Protocol

4.1 Standardization Gaps in Prior Work

A major obstacle in KG-RAG research is the inconsistency of evaluation protocols, which often leads to unfair and misleading comparisons. We identify two primary sources of this inconsistency:

Ambiguity in Metric Definitions. Many studies rely on vague textual descriptions such as "whether the top-1 predicted answer is correct" for Hits@1 without specifying critical details (Luo et al., 2024; Mavromatis and Karypis, 2024; He et al., 2024). It is often unclear how LLM outputs are parsed, how "top-1" is defined in the absence of explicit ranking, and what constitutes correctness (e.g., exact match vs. substring inclusion). Some works omit definitions entirely, citing "previous work" without verification (Sun et al., 2024; Chen et al., 2024).

Discrepancies in Implementation. We observe significant misalignments between metric descriptions and their codebase implementations. For instance, although Luo et al. (2024), Mavromatis and Karypis (2024), and He et al. (2024) describe their primary metric as the ranking-based "Hits@1", their official implementations entirely bypass ranking mechanisms. Instead, Luo et al. (2024) and Mavromatis and Karypis (2024) adopt a permissive criterion where *any* occurrence of the gold answer within the generated text is counted as correct. Similarly, the definition of "accuracy" fluctuates between strict Exact Match and loose substring matching across different studies (Sun et al., 2024; Chen et al., 2024; Jiang et al., 2023). Because many works reuse reported numbers under the assumption of comparability, these inconsistencies propagate, rendering cross-paper conclusions unreliable.

To address these issues, we adhere to the fully standardized and formally defined protocol in BRINK.

Dataset	Example
Family	*Question:* Who is 139's brother? *Topic Entity:* 139 — *Answer:* [205, 138, 2973, 2974] *Direct Evidence:* `brotherOf(139,205)` *Alternative Paths:* `fatherOf(139,14) ∧ uncleOf(205,14) ⇒ brotherOf(139,205)`

FB15k-237	*Question:* What is the currency of the estimated budget for 5297 (Annie Hall)? *Topic Entity:* 5297 (Annie Hall)
	—
	Answer: [1109 (United States Dollar)]
	Direct Evidence: `filmEstimatedBudgetCurrency(5297, 1109)`
	Alternative Paths: `filmCountry(5297 (Annie Hall), 2896 (United States of America))`
	`∧ locationContains(2896 (United States of America), 9397 (New York))`
	`∧ statisticalRegionGdpNominalCurrency(9397 (New York), 1109 (United States Dollar))`
	`⇒ filmEstimatedBudgetCurrency(5297 (Annie Hall), 1109 (United States Dollar))`

Wikidata5m	*Question:* What series is 3729461 (Lumia 610) a part of? *Topic Entity:* 3729461 (Lumia 610) — *Answer:* [1059823 (Nokia Rise)] *Direct Evidence:* `partOfSeries(3729461 (Lumia 610), 1059823 (Nokia Rise))` *Alternative Paths:* `follows(2846157 (Nokia Lumia 620), 3729461 (Lumia 610)) ∧ partOfSeries(2846157 (Nokia Lumia 620), 1059823 (Nokia Rise))` `⇒ partOfSeries(3729461 (Lumia 610), 1059823 (Nokia Rise))`

Table 2: Examples from BRINK datasets. Each instance includes a *natural-language question*, a *topic entity* (provided to the KG-RAG model), and the full *set of correct answers*. The red-highlighted answer denotes the *hard answer*—i.e., the one whose supporting triple has been removed in the incomplete KG setting. We also show the corresponding *direct evidence* (the deleted triple) and an *alternative path* derived from a mined rule that enables inference of the answer.

Dataset	#Triples	Train	Val	Test	Total Qs
Family-Complete	17,615	1,749	218	198	2,165
Family-Incomplete	15,785	1,749	218	198	2,165
FB15k-237-Complete	204,087	4,374	535	540	5,449
FB15k-237-Incomplete	198,183	4,374	535	540	5,449
Wikidata5m-Complete	20,510,107	27,720	3,466	3,465	34,651
Wikidata5m-Incomplete	20,478,006	27,720	3,466	3,465	34,651

Table 3: Statistics of BRINK.

4.2 Evaluation Setup

Given a natural-language question $q \in \mathcal{Q}$, access to a KG, and a topic entity, the model is tasked with returning a set of predicted answer entities $\mathcal{P}_q$.

Since KG-RAG models typically produce raw text sequences as output, we extract the final prediction set $\mathcal{P}_q$ by applying string partitioning and normalizing, following Luo et al. (2024). Details of this postprocessing step are provided in Appendix E. Without specific justification, all entities are represented by randomly assigned indices without textual labels (e.g., "Barack Obama" becomes `39`) to ensure that models rely solely on knowledge from the KG rather than memorized surface forms.

4.3 Evaluation Metrics

Given a set of test questions $\mathcal{Q}$, we denote the predicted answer set and the gold answer set for each question $q \in \mathcal{Q}$ as $\mathcal{P}_q$ and $\mathcal{A}_q$, respectively. The evaluation metrics are defined as follows:

Hits@Any. Hits@Any measures the proportion of questions for which the predicted answer set overlaps with the gold answer set, i.e., at least one correct answer is predicted:

$$\text{Hits@Any} = \frac{1}{|\mathcal{Q}|} \sum_{q \in \mathcal{Q}} \mathbb{1}[\mathcal{P}_q \cap \mathcal{A}_q \neq \varnothing] .$$

Precision and **Recall.** Precision measures the fraction of predicted answers that are correct, while recall measures the fraction of gold answers that are predicted:

$$\text{Precision} = \frac{1}{|\mathcal{Q}|} \sum_{q \in \mathcal{Q}} \frac{|\mathcal{P}_q \cap \mathcal{A}_q|}{|\mathcal{P}_q|} ,$$

$$\text{Recall} = \frac{1}{|\mathcal{Q}|} \sum_{q \in \mathcal{Q}} \frac{|\mathcal{P}_q \cap \mathcal{A}_q|}{|\mathcal{A}_q|} .$$

F1-score. The F1-score is the harmonic mean of precision and recall, computed per question and averaged across all questions:

$$\text{F1} = \frac{1}{|\mathcal{Q}|} \sum_{q \in \mathcal{Q}} \frac{2 \cdot |\mathcal{P}_q \cap \mathcal{A}_q|}{|\mathcal{P}_q| + |\mathcal{A}_q|} .$$

Hits@Hard. We define the *hard answer* for each question, denoted as $a_q \in \mathcal{A}_q$, as the specific answer entity selected during the question generation process, i.e., the one whose supporting triple was intentionally removed from the KG. Hits@Hard measures the proportion of predictions including the hard answer. It is defined as:

$$\text{Hits@Hard} = \frac{1}{|\mathcal{Q}|} \sum_{q \in \mathcal{Q}} \mathbb{1}[a_q \in \mathcal{P}_q] .$$

Hard Hits Rate. We define the *Hard Hits Rate* (HHR) as the fraction of correctly answered questions (i.e., Hits@Any) that include the hard answer in predictions:

$$\text{HHR} = \frac{Hits@Hard}{Hits@Any}.$$

5 Experiments and Results

5.1 Overall Performance

We evaluate six representative KG-RAG methods—RoG (Luo et al., 2024), G-Retriever (He et al., 2024), GNN-RAG (Mavromatis and Karypis, 2024), PoG (Chen et al., 2024), StructGPT (Jiang et al., 2023), and ToG (Sun et al., 2023)—on the BRINK datasets introduced in Section 3.

Figure 1 reports Hits@Any and F1-scores across all methods and datasets. In most cases, **both metrics drop noticeably when moving from the complete to the incomplete KG setting**, highlighting the challenge posed by missing direct evidence. Precision and recall follow a similar trend (see Appendix C.1).

G-Retriever presents a unique pattern: it shows similarly low performance across both complete and incomplete KGs. This is because its retrieval is based on textual similarity, which often retrieves both topic and answer entities regardless of KG completeness. The GNN encoder can partially recover missing links via neighborhood aggregation, making it less sensitive to missing triples. However, the lack of explicit reasoning and noisy k-NN retrieval lead to irrelevant candidates and overall low F1.

5.2 Impact of Removing Direct Evidence

To examine the impact of removing direct evidence, we report HHR in Figure 2. Ideally, models capable of reasoning over alternative paths should maintain a similar HHR across both complete and incomplete KG settings. However, across all models and datasets, **we observe a significant drop in HHR when moving from the complete to the incomplete KG setting**. This highlights the limited reasoning capabilities of current KG-RAG methods: while they perform well with direct evidence, their effectiveness declines sharply when they need to retrieve alternative paths and reason over it to infer the answer. Notably, even on the Family dataset, where relation patterns are simple and should be easily recognized by LLMs, models exhibit a substantial decline in recovering the correct answer via alternative paths when direct evidence is absent.

Training-based methods (e.g., RoG and GNN-RAG) show a smaller drop in HHR compared to non-trained methods (e.g., PoG and ToG), suggesting that exposure to incomplete KGs during training helps models generalize over indirect reasoning paths. In contrast, non-trained methods perform well when direct evidence is available but struggle significantly when such evidence is missing, revealing a stronger sensitivity to incompleteness and more limited reasoning capabilities.

5.3 Fine-Grained Analysis by Rule Type

To enable a deeper understanding of model reasoning, we conduct a fine-grained analysis of HHR across different rule types on FB15k-237, focusing on two representative models: RoG and PoG (Figure 3). Overall, RoG exhibits greater robustness to KG incompleteness than PoG across most rule types, indicating that training-based methods can better generalize to multi-hop reasoning when direct evidence is removed. An exception arises in symmetric patterns, where PoG outperforms RoG under the incomplete setting. This is notable because symmetric relations (e.g., `sibling`) are inherently more robust to directionality. If a model retrieves `sibling(Bieber, Jaxon)`, it is straightforward for an LLM to infer `sibling(Jaxon, Bieber)`. RoG's lower robustness in this seemingly trivial case suggests that training-based methods may overfit to relational patterns seen during training, at the cost of generalizing over simpler relations.

5.4 Influence of Entity Labeling

To assess how different entity labeling schemes affect KG-RAG performance, we evaluate three settings for representing entities in the input: (1) **Private ID**—randomly assigned IDs with no semantic content, (2) **Entity ID**—official Freebase IDs (e.g., `/m/02mjmr` for `Barack Obama`), and (3) **Text Label**—natural language labels of entities. Figure 4 shows the F1-scores of all models under each representation, for both incomplete (left) and complete (right) KG settings. We observe two key trends: First, **text labels significantly boost performance.** Models consistently achieve higher scores when entity labels are expressed in natural language. This suggests that LLMs can effectively leverage their internal knowledge when text labels are provided. Second, **entity IDs provide**

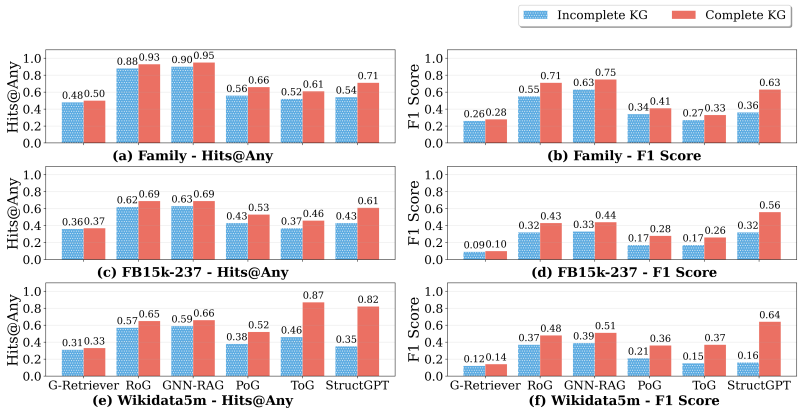

Figure 1: Performance comparison of KG-RAG models under incomplete (blue) and complete (red) KG settings, measured by **Hits@Any** (left) and **F1-Score** (right) on KGs: (a-b) Family, (c-d) FB15k-237, (e-f) Wikidata5m.

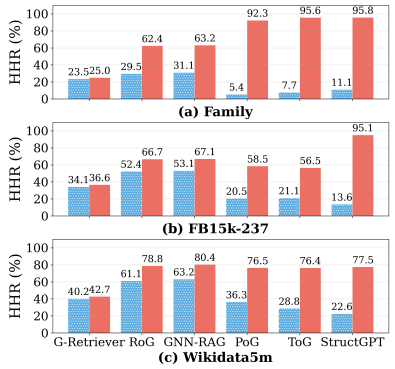

Figure 2: HHR under different KG settings. (a) Family. (b) FB15k-237. (c) Wikidata5m.

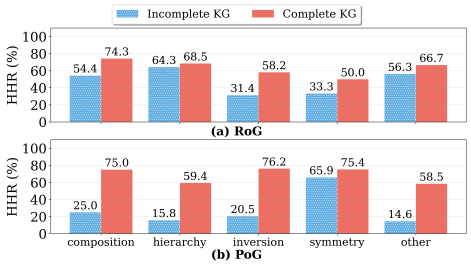

Figure 3: HHR across rule types on FB15k-237 for (a) RoG and (b) PoG, comparing performance under complete and incomplete KG settings.

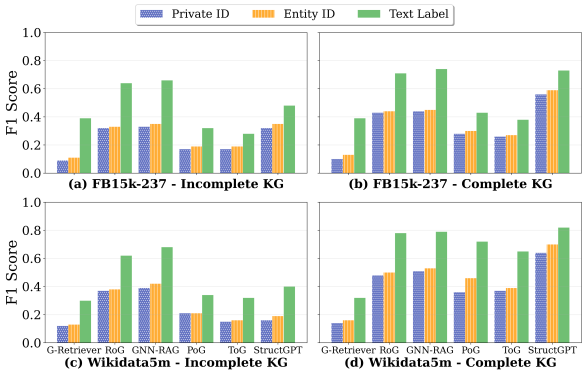

Figure 4: F1-Score comparison of models using Private ID, Entity ID, and Text Label representations on FB15k-237 (a-b) and Wikidata5m (c-d) datasets under incomplete and complete KG settings.

limited benefit over random IDs. Surprisingly, using official entity IDs like `/m/02mjmr` results in performance nearly identical to that of randomly assigned private IDs. This indicates that, despite LLMs potentially memorizing mappings between surface forms and IDs, they are unable to reason with these identifiers in generation tasks. Instead, they appear to treat IDs as opaque tokens unless the text label is explicit.

5.5 Case Study

To better understand the limitations of current KG-RAG methods, we analyze representative failure cases from BRINK. Table 4 shows two illustrative examples: (1) failure to retrieve relevant reasoning paths and (2) incorrect answer generation despite correct retrieval. A quantitative analysis is provided in Appendix J.

(1) Example 1: Retrieval Failure. The question requires reasoning over the relations `contains(New York City, Manhattan)` and `source(Manhattan, HUD)` to infer the answer `source(New York City, HUD)`. However, the retriever fails to locate any of these relevant facts. Instead, it returns spurious paths involving entities like `Pace University`, based on weak co-occurrence signals. As a result, the generator hallucinates an incorrect answer. This type of error is especially common among non-trained retrieval methods, which are more sensitive to missing links and lack robust multi-hop retrieval capabilities.

(2) Example 2: Reasoning Failure. The retriever correctly retrieves `spouse(Ian Holm, Penelope Wilton)`, which should enable inference of the reverse relation. However, the model outputs `Marriage`, likely influenced by unrelated `typeOfUnion` paths, indicating difficulty in filtering irrelevant context even with accurate retrieval.

6 Discussion and Conclusion

This work introduces a general methodology for constructing benchmarks aimed at evaluating KG-

Example 1: *Question:* What is the source of the estimated number of mortgages for **New York City**? *Answer:* [United States Department of Housing and Urban Development]
Alternative Path: `contains(New York City, Manhattan)` ∧ `source(Manhattan, HUD)` ⇒ `source(New York City, HUD)`
—
Prediction: [Pace University]
Retrieved Paths: `organizationExtra(New York City, Phone Number)` ∧ `serviceLocation(Phone Number, Pace University)`
(no path involving mortgage source)

Example 2: *Question:* Who is **Ian Holm**'s spouse? *Answer:* [Penelope Wilton]
Alternative Paths: `spouse(Ian Holm, Penelope Wilton)` ⇒ `spouse(Penelope Wilton, Ian Holm)`
—
Prediction: [Marriage]
Retrieved Paths: `spouse(Ian Holm, Penelope Wilton)`
`awardNominee(Ian Holm, Cate Blanchett)` ∧ `typeOfUnion(Cate Blanchett, Marriage)`
`awardNominee(Ian Holm, Kate Beckinsale)` ∧ `typeOfUnion(Kate Beckinsale, Domestic Partnership)`
... (many additional unrelated paths via award nominees)

Table 4: Case study examples from our benchmark illustrating typical failure modes of KG-RAG models. In both examples, the bold text highlights the topic entity, the green text denotes the expected alternative path, and the red text marks incorrect predictions.

RAG systems under conditions of knowledge incompleteness—a realistic yet often overlooked challenge in real-world KGs.

Our empirical study on datasets derived from Family, FB15k-237 and Wikidata5m reveals several key limitations of current KG-RAG methods. Most notably, existing KG-RAG models struggle to recover answers when direct evidence is missing, indicating limited robustness to incomplete knowledge. While training-based methods exhibit greater resilience, our fine-grained analysis reveals potential overfitting to specific relation patterns, sometimes at the expense of generalizing over trivial structures. Furthermore, we find that textual entity labels substantially improve performance, suggesting that models rely more on retrieving memorized knowledge than performing symbolic reasoning over structured data.

These findings suggest several research directions: (1) developing retrieval strategies that identify alternative paths when evidence is missing, (2) designing reasoning modules with stronger generalization to avoid overfitting to specific relation patterns, and (3) fine-tuning methods that enhance retrieval without impairing reasoning.

7 Limitation

Our benchmark construction relies on Horn-style logical rules mined by AMIE3 to construct reasoning questions. While this format may appear restrictive, it in fact covers a broad range of reasoning patterns frequently observed in real-world KGs. As shown in Figure 3, the benchmark encompasses four major rule types that dominate real-world KGs, with additional diverse patterns summarized in Appendix G.

This focus on Horn rules also offers a practical advantage: such rules are interpretable and easy to verify for plausibility, ensuring the reliability and consistency of the constructed questions. In contrast, more expressive rule types are challenging to mine reliably, and even SOTA rule mining methods often produce noisy or implausible rules. To maintain benchmark quality, we therefore focus on Horn rules in this paper. Extending to richer rule types is a promising direction for future work, specifically, incorporating advanced rule-learning methods like DRUM (Sadeghian et al., 2019) and AnyBURL (**?**), or aggregating high-confidence rules from multiple algorithms, offers a principled way to expand coverage beyond the capabilities of any single system.

8 Acknowledgements

The authors thank the International Max Planck Research School for Intelligent Systems (IMPRS-IS) for supporting Yuqicheng Zhu and Hongkuan Zhou. The work was partially supported by EU Projects Graph Massivizer (GA 101093202), enRichMyData (GA 101070284), SMARTY (GA 101140087), and the EPSRC project OntoEm (EP/Y017706/1).

References

Jinheon Baek, Alham Fikri Aji, Jens Lehmann, and Sung Ju Hwang. 2023. Direct fact retrieval from knowledge graphs without entity linking. *arXiv preprint arXiv:2305.12416*.

Russa Biswas, Lucie-Aimée Kaffee, Michael Cochez, Stefania Dumbrava, Theis E Jendal, Matteo Lissandrini, Vanessa Lopez, Eneldo Loza Mencía, Heiko Paulheim, Harald Sack, and 1 others. 2023. Knowledge graph embeddings: open challenges and opportunities. *Transactions on Graph Data and Knowledge*, 1(1):4–1.

Sebastian Borgeaud, Arthur Mensch, Jordan Hoffmann, Trevor Cai, Eliza Rutherford, Katie Millican, George

van den Driessche, Jean-Baptiste Lespiau, Bogdan Damoc, Aidan Clark, Diego de Las Casas, Aurelia Guy, Jacob Menick, Roman Ring, Tom Hennigan, Saffron Huang, Loren Maggiore, Chris Jones, Albin Cassirer, and 9 others. 2022. Improving language models by retrieving from trillions of tokens. In *ICML*, volume 162 of *Proceedings of Machine Learning Research*, pages 2206–2240. PMLR.

Lihu Chen, Simon Razniewski, and Gerhard Weikum. 2023. Knowledge base completion for long-tail entities. In *Proceedings of the First Workshop on Matching From Unstructured and Structured Data (MATCHING 2023)*, pages 99–108.

Liyi Chen, Panrong Tong, Zhongming Jin, Ying Sun, Jieping Ye, and Hui Xiong. 2024. Plan-on-graph: Self-correcting adaptive planning of large language model on knowledge graphs. In *Proceedings of the 38th Conference on Neural Information Processing Systems*.

Fengxiang Cheng, Haoxuan Li, Fenrong Liu, Robert van Rooij, Kun Zhang, and Zhouchen Lin. 2025. Empowering llms with logical reasoning: A comprehensive survey. *arXiv preprint arXiv:2502.15652*.

Xin Luna Dong, Evgeniy Gabrilovich, Geremy Heitz, Wiko Horn, Kevin Murphy, Shaohua Sun, and Wei Zhang. 2014. From data fusion to knowledge fusion. *Proceedings of the VLDB Endowment*, 7(10).

Jinyuan Fang, Zaiqiao Meng, and Craig Macdonald. 2024. Reano: Optimising retrieval-augmented reader models through knowledge graph generation. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2094–2112.

Luis Galárraga, Christina Teflioudi, Katja Hose, and Fabian M Suchanek. 2015. Fast rule mining in ontological knowledge bases with amie+. *The VLDB Journal*, 24(6):707–730.

Yu Gu, Sue Kase, Michelle Vanni, Brian Sadler, Percy Liang, Xifeng Yan, and Yu Su. 2021. Beyond iid: three levels of generalization for question answering on knowledge bases. In *Proceedings of the Web Conference 2021*, pages 3477–3488. ACM.

Haoyu Han, Yu Wang, Harry Shomer, Kai Guo, Jiayuan Ding, Yongjia Lei, Mahantesh Halappanavar, Ryan A Rossi, Subhabrata Mukherjee, Xianfeng Tang, and 1 others. 2024. Retrieval-augmented generation with graphs (graphrag). *arXiv preprint arXiv:2501.00309*.

Xiaoxin He, Yijun Tian, Yifei Sun, Nitesh Chawla, Thomas Laurent, Yann LeCun, Xavier Bresson, and Bryan Hooi. 2024. G-retriever: Retrieval-augmented generation for textual graph understanding and question answering. *Advances in Neural Information Processing Systems*, 37:132876–132907.

Yuan He, Bailan He, Zifeng Ding, Alisia Lupidi, Yuqicheng Zhu, Shuo Chen, Caiqi Zhang, Jiaoyan Chen, Yunpu Ma, Volker Tresp, and 1 others. 2025. Supposedly equivalent facts that aren't? entity frequency in pre-training induces asymmetry in llms. *arXiv preprint arXiv:2503.22362*.

Alfred Horn. 1951. On sentences which are true of direct unions of algebras1. *The Journal of Symbolic Logic*, 16(1):14–21.

Gautier Izacard and Edouard Grave. 2021. Leveraging passage retrieval with generative models for open domain question answering. In *EACL*, pages 874–880. Association for Computational Linguistics.

Jinhao Jiang, Kun Zhou, Zican Dong, Keming Ye, Wayne Xin Zhao, and Ji-Rong Wen. 2023. Structgpt: A general framework for large language model to reason over structured data. *arXiv preprint arXiv:2305.09645*.

Urvashi Khandelwal, Omer Levy, Dan Jurafsky, Luke Zettlemoyer, and Mike Lewis. 2020. Generalization through memorization: Nearest neighbor language models. In *ICLR*. OpenReview.net.

Jonathan Lajus, Luis Galárraga, and Fabian Suchanek. 2020. Fast and exact rule mining with amie 3. In *The Semantic Web: 17th International Conference, ESWC 2020, Heraklion, Crete, Greece, May 31–June 4, 2020, Proceedings 17*, pages 36–52. Springer.

Yunshi Lan, Gaole He, Jinhao Jiang, Jing Jiang, Wayne Xin Zhao, and Ji-Rong Wen. 2021. A survey on complex knowledge base question answering: Methods, challenges and solutions. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence*, pages 4483–4491. International Joint Conferences on Artificial Intelligence Organization.

Linhao Luo, Yuan-Fang Li, Gholamreza Haffari, and Shirui Pan. 2024. Reasoning on graphs: Faithful and interpretable large language model reasoning. In *ICLR*. OpenReview.net.

Costas Mavromatis and George Karypis. 2024. Gnn-rag: Graph neural retrieval for large language model reasoning. *arXiv preprint arXiv:2405.20139*.

Aisha Mohamed, Shameem Parambath, Zoi Kaoudi, and Ashraf Aboulnaga. 2020. Popularity agnostic evaluation of knowledge graph embeddings. In *Conference on Uncertainty in Artificial Intelligence*, pages 1059–1068. PMLR.

OpenAI. 2024. Chatgpt(3.5)[large language model]. https://chat.openai.com.

Mihir Parmar, Nisarg Patel, Neeraj Varshney, Mutsumi Nakamura, Man Luo, Santosh Mashetty, Arindam Mitra, and Chitta Baral. 2024. Logicbench: Towards systematic evaluation of logical reasoning ability of large language models. *arXiv preprint arXiv:2404.15522*.

Boci Peng, Yun Zhu, Yongchao Liu, Xiaohe Bo, Haizhou Shi, Chuntao Hong, Yan Zhang, and Siliang Tang. 2024. Graph retrieval-augmented generation: A survey. *arXiv preprint arXiv:2408.08921*.

Nico Potyka, Yuqicheng Zhu, Yunjie He, Evgeny Kharlamov, and Steffen Staab. 2024. Robust knowledge extraction from large language models using social choice theory. In *Proceedings of the 23rd International Conference on Autonomous Agents and Multiagent Systems*, pages 1593–1601.

Yunqi Qiu, Yuanzhuo Wang, Xiaolong Jin, and Kun Zhang. 2020. Stepwise reasoning for multi-relation question answering over knowledge graph with weak supervision. In *Proceedings of the 13th international conference on web search and data mining*, pages 474–482.

Ori Ram, Yoav Levine, Itay Dalmedigos, Dor Muhlgay, Amnon Shashua, Kevin Leyton-Brown, and Yoav Shoham. 2023. In-context retrieval-augmented language models. *Trans. Assoc. Comput. Linguistics*, 11:1316–1331.

Ali Sadeghian, Mohammadreza Armandpour, Patrick Ding, and Daisy Zhe Wang. 2019. Drum: End-to-end differentiable rule mining on knowledge graphs. *Advances in neural information processing systems*, 32.

Jiashuo Sun, Chengjin Xu, Lumingyuan Tang, Saizhuo Wang, Chen Lin, Yeyun Gong, Lionel M Ni, Heung-Yeung Shum, and Jian Guo. 2023. Think-on-graph: Deep and responsible reasoning of large language model on knowledge graph. *arXiv preprint arXiv:2307.07697*.

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 *ICLR*. OpenReview.net.

Oyvind Tafjord, Bhavana Dalvi Mishra, and Peter Clark. 2020. Proofwriter: Generating implications, proofs, and abductive statements over natural language. *arXiv preprint arXiv:2012.13048*.

Alon Talmor and Jonathan Berant. 2018a. The web as a knowledge-base for answering complex questions. In *NAACL-HLT*, pages 641–651. Association for Computational Linguistics.

Alon Talmor and Jonathan Berant. 2018b. The web as a knowledge-base for answering complex questions. *arXiv preprint arXiv:1803.06643*.

Jidong Tian, Yitian Li, Wenqing Chen, Liqiang Xiao, Hao He, and Yaohui Jin. 2021. Diagnosing the first-order logical reasoning ability through logicnli. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3738–3747.

Kristina Toutanova and Danqi Chen. 2015. Observed versus latent features for knowledge base and text inference. In *Proceedings of the 3rd workshop on continuous vector space models and their compositionality*, pages 57–66.

Denny Vrandečić and Markus Krötzsch. 2014. Wikidata: a free collaborative knowledgebase. *Communications of the ACM*, 57(10):78–85.

Xiaozhi Wang, Tianyu Gao, Zhaocheng Zhu, Zhengyan Zhang, Zhiyuan Liu, Juanzi Li, and Jian Tang. 2021. Kepler: A unified model for knowledge embedding and pre-trained language representation. *Transactions of the Association for Computational Linguistics*, 9:176–194.

Yao Xu, Shizhu He, Jiabei Chen, Zihao Wang, Yangqiu Song, Hanghang Tong, Guang Liu, Kang Liu, and Jun Zhao. 2024. Generate-on-graph: Treat llm as both agent and kg in incomplete knowledge graph question answering. *arXiv preprint arXiv:2404.14741*.

Liang Yao, Chengsheng Mao, and Yuan Luo. 2019. Kg-bert: Bert for knowledge graph completion. *arXiv preprint arXiv:1909.03193*.

Xi Ye, Semih Yavuz, Kazuma Hashimoto, Yingbo Zhou, and Caiming Xiong. 2021. Rng-kbqa: Generation augmented iterative ranking for knowledge base question answering. *arXiv preprint arXiv:2109.08678*.

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 *ACL (2)*. The Association for Computer Linguistics.

Dongzhuoran Zhou, Yuqicheng Zhu, Yuan He, Jiaoyan Chen, Evgeny Kharlamov, and Steffen Staab. 2025a. Evaluating knowledge graph based retrieval augmented generation methods under knowledge incompleteness. *arXiv preprint arXiv:2504.05163*.

Dongzhuoran Zhou, Yuqicheng Zhu, Xiaxia Wang, Hongkuan Zhou, Jiaoyan Chen, Steffen Staab, Yuan He, and Evgeny Kharlamov. 2025b. Gr-agent: Adaptive graph reasoning agent under incomplete knowledge. *arXiv preprint arXiv:2512.14766*.

Yuqicheng Zhu, Daniel Hernández, Yuan He, Zifeng Ding, Bo Xiong, Evgeny Kharlamov, and Steffen Staab. 2025a. Predicate-conditional conformalized answer sets for knowledge graph embeddings. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 4145–4167, Vienna, Austria. Association for Computational Linguistics.

Yuqicheng Zhu, Nico Potyka, Daniel Hernández, Yuan He, Zifeng Ding, Bo Xiong, Dongzhuoran Zhou, Evgeny Kharlamov, and Steffen Staab. 2025b. Argrag: Explainable retrieval augmented generation using quantitative bipolar argumentation. In *Conference on Neurosymbolic Learning and Reasoning*, pages 697–718. PMLR.

Yuqicheng Zhu, Nico Potyka, Mojtaba Nayyeri, Bo Xiong, Yunjie He, Evgeny Kharlamov, and Steffen Staab. 2024. Predictive multiplicity of knowledge graph embeddings in link prediction. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 334–354, Miami, Florida, USA. Association for Computational Linguistics.

Yuqicheng Zhu, Nico Potyka, Jiarong Pan, Bo Xiong, Yunjie He, Evgeny Kharlamov, and Steffen Staab. 2025c. Conformalized answer set prediction for knowledge graph embedding. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 731–750.

Yuqicheng Zhu, Jingcheng Wu, Yizhen Wang, Hongkuan Zhou, Jiaoyan Chen, Evgeny Kharlamov, and Steffen Staab. 2025d. Certainty in uncertainty: Reasoning over uncertain knowledge graphs with statistical guarantees. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 8741–8763.

A Details of Rule Mining

A.1 AMIE3 Candidate Rule Refinement

Refinement is carried out using a set of operators that generate new candidate rules:

- Dangling atoms, which introduce a new variable connected to an existing one;
- Closing atoms, which connect two existing variables;
- Instantiated atoms, which introduce a constant and connect it to an existing variable.

AMIE3 generates candidate rules by a refinement process using a classical breadth-first search (Lajus et al., 2020). It begins with rules that contain only a head atom (e.g. $\Rightarrow \texttt{hasSibling}(X, Y)$) and refines them by adding atoms to the body. For example, it may generate the refined rule:

$$\texttt{hasParent}(X, Z) \wedge \texttt{hasChild}(Z, Y) \Rightarrow \texttt{hasSibling}(X, Y)\,.$$

This refinement step connects existing variables and introduces new ones, gradually building meaningful patterns.

A.2 AMIE3 Hyperparameter Settings

We use AMIE3 with a confidence threshold of 0.3 and a PCA confidence threshold θ_{PCA} of 0.4 for all three datasets. The maximum rule length is set to 3 for **Family** to avoid overly complex patterns, and 4 for **FB15k-237** and **Wikidata5m** to allow richer rules. For the AMIE3 confidence and head coverage thresholds (Section 3.1), we tested values from 0.1–0.6 and chose 0.3 as a balance: lower values produced unreliable rules, while higher ones biased the distribution toward certain relations. For τ, we tested 0.01–0.2 and set it to limit domination by frequent entities, ensuring a balanced answer distribution. See Appendix B.1 for the definition of PCA confidence.

B Properties of Horn Rules Mined by AMIE3

AMIE3 mines logical rules from knowledge graphs in the form of (Horn) rules:

$$B_1 \wedge B_2 \wedge \cdots \wedge B_n \implies H$$

where B_i and H are atoms of the form $r(X, Y)$. To ensure interpretability and practical utility, AMIE3

imposes the following structural properties on all mined rules:

- **Connectedness:** All atoms in the rule are transitively connected via shared variables or entities. This prevents rules with independent, unrelated facts (e.g., $\texttt{diedIn}(x, y) \implies \texttt{wasBornIn}(w, z)$). Two atoms are connected if they share a variable or entity; a rule is connected if every atom is connected transitively to every other atom.
- **Closedness:** Every variable in the rule appears at least twice (i.e., in at least two atoms). This avoids rules that merely predict the existence of some fact without specifying how it relates to the body, such as $\texttt{diedIn}(x, y) \implies \exists z : \texttt{wasBornIn}(x, z)$.
- **Safety:** All variables in the head atom also appear in at least one body atom. This ensures that the rule's predictions are grounded by the body atoms and avoids uninstantiated variables in the conclusion.

These restrictions are widely adopted in KG rule mining (Galárraga et al., 2015; Lajus et al., 2020) to guarantee that discovered rules are logically well-formed and meaningful for downstream reasoning tasks.

B.1 PCA Confidence

To understand the concept of rule mining better for the reader we simplified notation of confidence in main body. Note AMIE3 also supports a more optimistic confidence metric known as *PCA confidence*, which adjusts standard confidence to account for incompleteness in the KG.

Motivation. Standard confidence for a rule is defined as the proportion of its correct predictions among all possible predictions suggested by the rule. However, this metric is known to be pessimistic for knowledge graphs, which are typically incomplete: many missing triples may be true but unobserved, unfairly penalizing a rule's apparent reliability.

Definition. To address this, AMIE3 introduces *PCA confidence* (Partial Completeness Assumption confidence) (Galárraga et al., 2015), an optimistic variant that partially compensates for KG incompleteness. Given a rule of the form

$$B_1 \wedge \cdots \wedge B_n \implies r(x, y)$$

the **standard confidence** is

$$\text{conf}(R) = \frac{|\{(x, y) : B_1 \wedge \cdots \wedge B_n \wedge r(x, y)\}|}{|\{(x, y) : B_1 \wedge \cdots \wedge B_n\}|}$$

where the denominator counts all predictions the rule could possibly make, and the numerator counts those that are actually present in the KG.

PCA confidence modifies the denominator to include only those (x, y) pairs for which at least one $r(x, y')$ triple is known for the subject x. That is, the rule is only penalized for predictions about entities for which we have observed at least some information about the target relation. Formally,

$$\text{conf}_{\text{PCA}}(R) = \frac{|\{(x, y) : B_1 \wedge \cdots \wedge B_n \wedge r(x, y)\}|}{|\{(x, y) : B_1 \wedge \cdots \wedge B_n \wedge \exists y' : r(x, y')\}|}$$

Here, the denominator sums only over those x for which some y' exists such that $r(x, y')$ is observed in the KG.

Intuition. This approach assumes that, for any entity x for which at least one fact $r(x, y')$ is known, the KG is "locally complete" with respect to r for x—so if the rule predicts other $r(x, y)$ facts for x, and they are missing, we treat them as truly missing (i.e., as counterexamples to the rule). But for entities where no $r(x, y)$ fact is observed at all, the rule is not penalized for predicting additional facts.

Comparison. PCA confidence thus provides a more optimistic and fairer assessment of a rule's precision in the presence of incomplete data. It is widely adopted in KG rule mining, and is the default metric for filtering and ranking rules in AMIE3.

For further details, see (Galárraga et al., 2015).

B.2 Rule Mining Procedure

AMIE3 generates candidate rules by a refinement process using a classical breadth-first search (Lajus et al., 2020). Algorithm 2 summarizes the rule mining process of AMIE3. The algorithm starts with an initial set of rules that contain only a head atom (i.e. $\top \Rightarrow r(X, Y)$, where $\top$ denotes an empty body) and maintains a queue of rule candidates (Line 1). At each step, AMIE3 dequeues a rule R from the queue and evaluates whether it satisfies three criteria (Line 5):

- the rule is *closed* (i.e., all variables in at least two atoms),

Algorithm 2 AMIE3

Require: Knowledge graph $\mathcal{G}$, maximum rule length l, PCA confidence threshold θ_{PCA}, and head coverage threshold θ_{hc}.
Ensure: Set of mined rules $\mathcal{R}$.

1: $q \leftarrow$ all rules of the form $\top \Rightarrow r(X, Y)$
2: $\mathcal{R} \leftarrow \emptyset$
3: **while** q is not empty **do**
4: $\quad R \leftarrow q$.dequeue()
5: $\quad$ **if** `SatisfiesRuleCriteria`(R) **then**
6: $\quad\quad \mathcal{R} \leftarrow \mathcal{R} \cup \{R\}$
7: $\quad$ **if** `len`$(R) < l$ and $\theta_{PCA}(R) < 1.0$ **then**
8: $\quad\quad$ **for all** $R_c \in$ `refine`(R) **do**
9: $\quad\quad\quad$ **if** $hc(R_c) \geq \theta_{hc}$ and $R_c \notin q$ **then**
10: $\quad\quad\quad\quad q$.enqueue(R_c)
11: **return** $\mathcal{R}$

- its PCA confidence is higher than θ_{PCA},
- its PCA confidence is higher than the confidence of all previously mined rules with the same head atom as R and a subset of its body atoms.

If these conditions are met, the rule is added to the final output set $\mathcal{R}$.

If R has fewer than l atoms and its confidence can still be improved (Line 8), AMIE3 applies a `refine` operator (Line 9) that generates new candidate rules by adding a body atom (details in Appendix A). Refined rules are added to the queue only if they have sufficient head coverage (Line 11) and have not already been explored. This process continues until the queue is empty, at which point all high-quality rules satisfying the specified constraints have been discovered.

B.3 Benchmark Construction Code and Data

We release the source code for benchmark construction, along with the Family, FB15k-237 and Wikidata5m benchmark datasets, at `https://anonymous.4open.science/r/INCK-EA16`.

C Additional Results of the experiment

C.1 Recall and Precision

In complement to Figure 1, we provide a detailed breakdown of recall and precision for all settings to offer a more comprehensive evaluation of the F1 scores. Figure 5 presents the recall and precision values for all evaluated KG-RAG models across the constructed benchmarks.

C.2 Evaluation of Generate-on-Graph (GoG) (Xu et al., 2024)

This section provides a detailed analysis of GoG (Xu et al., 2024), following the experimental results presented in the main text. While GoG explicitly addresses knowledge incompleteness as part of its motivation, we did not include this aspect in the main body for several reasons.

GoG's implementation is heavily optimized for specific datasets, such as CWQ (Talmor and Berant, 2018a) and WebQSP (Yih et al., 2016), and relies on several domain-specific heuristics:

- Hard-coded Templates: Prompting mechanisms and relation-name heuristics tailored to specific KG schemas.
- Entity-Type Linking: Reliance on entity-type information that is often unavailable in general-purpose or incomplete KGs.
- Textual Dependency: Assumptions regarding entity text labels, which prevents the model from functioning on datasets using anonymized numeric IDs—a necessary measure we take to prevent knowledge leakage and evaluate pure structural reasoning.

Despite these limitations, we adapted GoG for our benchmark. The results are shown in Table 5. Our evaluation indicates a significant performance decline, particularly in HHR, our metric designed to assess reasoning capacity when direct supporting facts are missing.

The primary limitation lies in GoG's underlying reasoning mechanism. Rather than performing multi-hop reasoning over alternative paths within the KG, GoG tends to compensate for missing triples by querying the LLM's internal parametric priors. While this strategy may succeed in benchmarks with significant knowledge overlap between the KG and the LLM's training data, it fails in our setting. By anonymizing entity names, we isolate the model's ability to reason over the provided structure.

Table 5: Performance of GoG on Family and FB15k-237.

Dataset	Hits@Any	HHR	F1
Family Complete	0.60	0.96	0.48
Family Incomplete	0.58	0.28	0.36
FB15k-237 Complete	0.46	0.57	0.26
FB15k-237 Incomplete	0.20	0.23	0.11

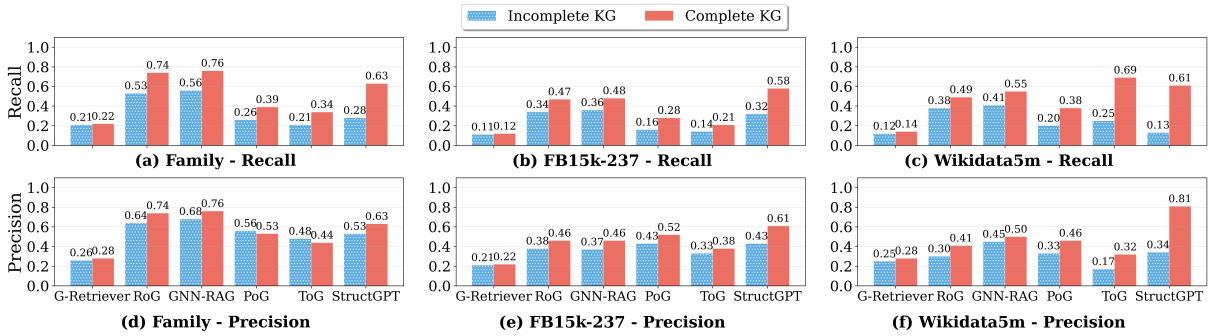

Figure 5: Performance comparison of KG-RAG models under incomplete (blue) and complete (red) KG settings, measured by **Recall** (top) and **Precision** (bottom) on Family, FB15k-237, and Wikidata5m.

D Prompt Template

Prompt for generating questions from triples:

```
You are an expert in knowledge graph question
    generation.

Given:
Removed Triple: ({entity_h}, {predicate_T}, {
    entity_t})
Question Entity: {topic_entity}
Answer Entity: {answer_entity}

Write a clear, natural-language question that
    asks for the Answer Entity, using the given
    predicate and Topic Entity.

Requirements:
- Express the predicate {predicate_T} naturally
    (paraphrasing allowed, but preserve core
    meaning; e.g., "wife_of" -> "wife").
- Mention the Topic Entity {topic_entity}.
- The answer should be the Answer Entity {
    answer_entity}.
- Do not mention the Answer Entity {
    answer_entity} in the question.
- Do not ask a yes/no question.
- Output only the question as plain text.

Example:
Removed Triple: ("Alice", "wife_of", "Carol")
Question Entity: Carol
Answer Entity: Alice

Output:
Who is Carol's wife?

Now, generate the question for:
Removed Triple: ({entity_h}, {predicate_T}, {
    entity_t})
Question Entity: {topic_entity}
Answer Entity: {answer_entity}
```

To ensure reproducibility and mitigate randomness in LLM outputs (Potyka et al., 2024), we set the generation temperature to 0 in all experiments.

E Detailed Evaluation Settings

All evaluated models are required to produce their predictions as a *structured list of answers*, but in practice, the model output is often a raw string P_{str} (e.g., `"Paris, London"` or `"Paris London"`). To obtain a set-valued prediction suitable for evaluation, we first apply a splitting function $\text{split}(P_{\text{str}})$, which splits the raw string into a list of answer strings $P = [p_1, p_2, \ldots, p_n]$ using delimiters such as commas, spaces, or newlines as appropriate.

We then define a normalization function $\text{norm}(\cdot)$, which converts each answer string to lowercase, removes articles (`a`, `an`, `the`), punctuation, and extra whitespace, and eliminates the special token `<pad>` if present. The final prediction set is then defined as $\mathcal{P} = \{\text{norm}(p) \mid p \in P\}$, i.e., the set of unique normalized predictions. The same normalization is applied to each gold answer in the list A to obtain the set $\mathcal{A}$.

All evaluation metrics are computed based on the resulting sets of normalized predictions $\mathcal{P}$ and gold answers $\mathcal{A}$.

Algorithm 3 Output Processing

Require: Model output string P_{str}, gold answer list A

Ensure: Normalized prediction set $\mathcal{P}$, normalized gold set $\mathcal{A}$

$P \leftarrow \text{split}(P_{\text{str}})$
$\mathcal{P} \leftarrow \{\, \text{norm}(p) \mid p \in P \,\}$
$\mathcal{A} \leftarrow \{\, \text{norm}(a) \mid a \in A \,\}$
return $\mathcal{P}, \mathcal{A}$

F Baseline Details

Unless otherwise specified, for all methods we use the LLM backbone and hyperparameters as described in the original papers.

RoG, G-Retriever, and GNN-RAG are each trained and evaluated separately on the 8:1:1 training split of each dataset (Family, FB15k-237

and Wikidata5m) using a single NVIDIA H200 GPU, as described in Section 3.3. For RoG, we use LLaMA2-Chat-7B as the LLM backbone, instruction-finetuned on each dataset for 3 epochs separately. The batch size is set to 4, the learning rate to 2×10^{-5}, and a cosine learning rate scheduler with a warmup ratio of 0.03 is adopted (Luo et al., 2024). For G-Retriever, the GNN backbone is a Graph Transformer (4 layers, 4 attention heads per layer, hidden size 1024) with LLaMA2-7B as the LLM. Retrieval hyperparameters and optimization follow He et al. (2024). For GNN-RAG (Mavromatis and Karypis, 2024), we use the recommended ReaRev backbone and sBERT encoder; the GNN component is trained for 200 epochs with 80 epochs of warmup and a patience of 5 for early stopping. All random seeds are fixed for reproducibility. For PoG (Chen et al., 2024), StructGPT (Jiang et al., 2023), and ToG (Sun et al., 2024), we use GPT-3.5-turbo as the underlying LLM, following the original papers, and adopt the original prompt and generation settings from each method. The temperature of LLMs is set to 0.

G Detailed Analysis of Other Rule Types

The *Other* category in Table 1 encompasses a broad range of logical rules that do not fall into standard symmetry, inversion, hierarchy, or composition classes. Below we summarize the main patterns observed, provide representative examples, and discuss their impact on model performance.

Longer Compositional Chains. Rules involving three,

$$r_1(x, y) \wedge r_2(y, z) \wedge r_3(z, w) \Rightarrow r_4(x, w)$$

Triangle Patterns. Rules connecting three entities in a triangle motif,

$$r_1(x, y) \wedge r_2(x, z) \Rightarrow r_3(y, z)$$

Intersection Rules. Rules where multiple body atoms share the same argument,

$$r_1(x, y) \wedge r_2(x, y) \Rightarrow r_3(x, y)$$

Other Patterns. Some rules do not exhibit simple interpretable motifs, involving unusual variable binding or rare predicate combinations. Like recursive rules (check AMIE3 (Lajus et al., 2020) for more details)

H Quality Check of the Generated Questions

To ensure that the constructed benchmark does not produce unanswerable questions, we conduct a quality check on the generated question set. Although the construction process guarantees that all grounded body triples required to infer a removed head remain in the KG, it is still necessary to verify that the removed triples can indeed be inferred from the remaining facts. Ensuring the answerability of generated questions is therefore essential, and a quantitative verification is required to confirm that the removed triples remain inferable from the retained facts.

To examine this, we evaluate whether each removed triple can be recovered when the model is given sufficient contextual information. Specifically, we provide both the alternative reasoning paths and the corresponding logical rules for each question, thereby simulating a perfect retriever with guided reasoning. This setup ensures that all necessary evidence for deriving the target entity is explicitly available. Performance is reported as the proportion of cases in which the target entity was correctly predicted.

The results in Table 6 show that, under this setting, LLMs answer most questions correctly, indicating that the benchmark questions are indeed answerable by LLMs when sufficient information is available.

Table 6: Performance of backbone LLMs when provided with sufficient information

Backbone	Family	FB15k-237	Wikidata5m
GPT-3.5-turbo	0.88	0.86	0.84
GPT-4o	0.91	0.94	0.90

I Results under Permissive Evaluation Metrics

We also report the performance of representative KG-RAG models under the commonly used permissive metrics adopted in prior works, such as Hits@1 and relaxed F1/Recall/Precision (Luo et al., 2024; Chen et al., 2024). These metrics treat partial string matches as correct answers and may therefore overestimate model performance, for example by counting a response that contains the gold entity in a negated form as correct (e.g., treating "not Barack Obama" as correct when the gold answer

is "Barack Obama"). Nevertheless, as shown in Table 7, the results remain consistent with those obtained using our stricter evaluation protocol, revealing substantial performance differences between complete and incomplete KGs.

Table 7: Performance of representative KG-RAG models under standard permissive metrics.

Method	Settings	Hits@1	Permissive F1	Permissive Recall	Permissive Precision
RoG	complete KG	69.54	43.97	47.25	46.08
RoG	incomplete KG	62.97	33.24	33.63	37.81
PoG	complete KG	57.59	30.21	27.49	56.99
PoG	incomplete KG	45.26	18.56	18.47	45.66

The consistent performance gap across both permissive and strict evaluation settings further validates the robustness of our findings and highlights the distinct behaviors of KG-RAG models when reasoning over incomplete knowledge graphs. In particular, both representative methods (RoG and PoG) exhibit significant performance drops across permissive and strict evaluation settings, and other models show similarly clear declines under incomplete KGs.

J Quantitative Error Analysis

To complement the qualitative case studies presented in Section 5.5, we further quantify the distribution of failure types across datasets and methods. Based on the taxonomy introduced in Section 5.5, we aggregate the results into two broad categories: *Retrieval Failure* and *Reasoning Failure*. On average, the ratio between these two categories is approximately 7:3, with the majority of failures attributable to Retrieval Failure rather than Reasoning Failure. The ratio is averaged across datasets and models, following the classification criteria in Section 5.5: a case is marked as *Retrieval Failure* if no alternative path supporting the gold answer is retrieved, and as *Reasoning Failure* if such a path is retrieved but the final answer is incorrect.

This quantitative summary reinforces the qualitative findings discussed in Section 5.5, suggesting that Retrieval Failure remains the primary bottleneck for KG-RAG models operating under incomplete KGs.

K Personal Identification Issue in FB15k-237 and Wikidata5m

While FB15k-237 and Wikidata5m contain information about individuals, they typically focus on well-known public figures such as celebrities, politicians, and historical figures. Since this information is already widely available online and in various public sources, their inclusion in Freebase and Wikidata doesn't significantly compromise individual privacy compared to datasets containing sensitive personal information.

L AI Assistants In Writing

We use ChatGPT (OpenAI, 2024) to enhance our writing skills, abstaining from its use in research and coding endeavors.