DrKGC: Dynamic Subgraph Retrieval-Augmented LLMs for Knowledge Graph Completion across General and Biomedical Domains

Anonymous ACL submission

Abstract

Knowledge graph completion (KGC) aims to predict missing triples in knowledge graphs (KGs) by leveraging existing triples and textual information. Recently, generative large language models (LLMs) have been increasingly employed for graph tasks. However, current approaches typically encode graph context in textual form, which fails to fully exploit the potential of LLMs for perceiving and reasoning about graph structures. To address this limitation, we propose **DrKGC** (**D**ynamic Subgraph **R**etrieval-Augmented LLMs for **K**nowledge **G**raph **C**ompletion). DrKGC employs a flexible lightweight model training strategy to learn structural embeddings and logical rules within the KG. It then leverages a novel bottom-up graph retrieval method to extract a subgraph for each query guided by the learned rules. Finally, a graph convolutional network (GCN) adapter uses the retrieved subgraph to enhance the structural embeddings, which are then integrated into the prompt for effective LLM fine-tuning. Experimental results on two general domain benchmark datasets and two biomedical datasets demonstrate the superior performance of DrKGC. Furthermore, a realistic case study in the biomedical domain highlights its interpretability and practical utility.

1 Introduction

Knowledge graphs (KGs) are structured representations of real-world facts, typically formulated as a set of triples that consist of entities and their relationships (Nickel et al., 2015; Ji et al., 2021). Biomedical Knowledge Graphs (BKGs) are specialized forms of KGs tailored to the biomedical domain. In a BKG, nodes represent biomedical entities—such as molecules, diseases, and genes—while edges capture various relationships among these entities, typically through functional predicates relevant to the biomedical domain (e.g., "treats," "inhibits," and "causes") (Walsh et al., 2020). BKGs have proved instrumental in numerous biological tasks, including drug repurposing, side-effect prediction, and drug–drug interaction detection (Himmelstein et al., 2017; Zitnik et al., 2018; Lin et al., 2020).

BKGs, like other KGs, often suffer from incompleteness, typically manifested as missing edges between nodes (Chen et al., 2020) . This incompleteness may arise because (1) the facts are absent from the data source, or (2) they remain undiscovered by humans. Such issues are particularly prevalent in BKGs, as their data primarily originates from experimental results, clinical trials, and scientific literature.

To address the challenge of predicting missing information in knowledge graphs, a wide range of Knowledge Graph Completion (KGC) models have been developed. These include structure-based models such as TransE (Bordes et al., 2013) and graph neural network (GNN) based models such as R-GCN (Schlichtkrull et al., 2018), rule-based approaches like Neural-LP (Yang et al., 2017), text-based methods such as KG-BERT (Yao et al., 2019). Recently, the advent of generative large language models (LLMs) has given rise to a new class of generation-based KGC approaches. Unlike traditional text-based methods that encode entity and relation descriptions into fixed embeddings, these approaches leverage LLMs to generate missing triples in a sequence-to-sequence manner, often relying on prompting or fine-tuning strategies (e.g., KICGPT (Wei et al., 2024), KoPA (Zhang et al., 2024)). Although generation-based methods have shown promise in KGC, they face several key limitations: ❶ **Structural Information Loss**: These methods often fail to preserve the rich structural information inherent in knowledge graphs. While graph paths or subgraphs can be encoded as text prompts, overly long inputs introduce noise and increase computational costs. ❷ **Static Embedding Limitations**: Incorporating structural embeddings into LLMs offers a partial solution but remains limited, as such embeddings are static and do not adapt to the query-specific context or dynamic subgraph structure. ❸ **Generic Responses from LLMs**: In the absence of additional constraints, LLMs tend to generate generic predictions influenced by pretraining data. This is especially problematic in biomedical KGs, where high-degree entities and many-to-many relations make multiple answers plausible—yet not all are contextually correct or desirable.

To tackle these challenges, we propose **D**ynamic Subgraph **R**etrieval-Augmented LLMs for **K**nowledge **G**raph **C**ompletion (**DrKGC**). Our approach begins by converting incomplete triples into natural language questions using an automatically generated template lexicon. It then employs a lightweight model to learn structure embeddings for entities and rank candidate entities based on their relevance to the query. This anchors the reasoning process in semantic and structural context without requiring long or noisy text prompts (❶). To overcome the limitations of static embeddings (❷), DrKGC dynamically constructs a query-specific subgraph using retrieved candidates and learned logical rules. This enables the model to focus on relevant local structures and incorporate adaptive, context-aware structural cues during inference. Finally, to mitigate the risk of generic or irrelevant responses (❸), DrKGC restricts the output space by explicitly defining a candidate entity set. The prompt is enriched with both global and local graph signals, guiding the LLM to generate contextually grounded and targeted predictions—especially in cases involving complex, many-to-many biomedical relations.

The key contributions of our work are as follows:

- We propose DrKGC, a novel and flexible framework for knowledge graph completion that effectively supports both general KGs and domain BKGs.
- We develop two critical components of DrKGC to effectively integrate graph-structural information into the generative model. Specifically, we extend the standard retrieval-augmented generation to the graph scenario where we leverage logical rules to obtain a local subgraph that represents entities of potential interest. Then, we develop a technique that applies graph convolutional networks to the retrieved subgraphs to further generate local embeddings of entities, effectively supplying structural information for LLM-based prediction.
- We perform comprehensive experiments on both benchmark datasets and biomedical use cases to evaluate the performance of DrKGC and show its significant improvement over state-of-the-art baseline approaches. We further conduct a biomedical case study on drug repurposing to demonstrate the practical applicability of DrKGC.

2 Related Work

2.1 Structure-based Methods

Knowledge graph completion (KGC) can be approached by leveraging the structural information of nodes and edges in large heterogeneous graphs. Early methods learn low-dimensional embeddings for entities and relations based on individual triples—for example, TransE (Bordes et al., 2013) views a relation as a translation from the subject to the object, while RotatE (Sun et al., 2019) extends TransE into a complex space to model symmetric relations. Semantic matching approaches (e.g., ComplEx (Trouillon et al., 2016), DistMult (Yang et al., 2014)) compute the similarity of entity and relation representations. However, these triple-based methods handle each triple independently and ignore higher-order neighborhood information. To address this, GNN-based methods, such as R-GCN (Schlichtkrull et al., 2018) and CompGCN (Vashishth et al., 2019), introduce message passing and neighborhood aggregation.

2.2 Rule-based Methods

Because two entities in a KG may be linked by a few one-hop paths but numerous multi-hop paths, rule-based methods have emerged to learn probabilistic logic rules from these relation paths for inferring missing triples. For example, Neural-LP (Yang et al., 2017) offers an end-to-end differentiable framework that jointly learns the parameters and structures of first-order logical rules by combining a neural controller with attention and memory, composing differentiable TensorLog operations. NCRL (Cheng et al., 2023) learns logical rules by splitting rule bodies into smaller parts, encoding them via a sliding window, and then merging them recursively with an attention mechanism, achieving efficient and scalable reasoning.

2.3 Text-based Methods

Knowledge graphs often include extensive textual information—such as names and descriptions of entities and relations—which text-based methods can exploit using pre-trained language models (PLMs) to predict missing triples. For example, KG-BERT (Yao et al., 2019) computes triple scores by feeding the text of head entities, relations, and tail entities into a BERT model. SimKGC (Wang et al., 2022) applies contrastive learning with three types of negative samples to build more discriminative KGC models. KGLM (Youn and Tagkopoulos, 2022) combines learning the structure of the knowledge graph with fine-tuning PLM.

2.4 Generation-based Methods

With the rise of generative large language models (LLMs), generation-based approaches have gained attention by transforming KGC into a sequence-to-sequence text generation task. These methods still rely on textual information from KGs, but they reframe a KGC query as a natural language question, prompt the LLM for an answer, and map that output back to KG entities. For example, KICGPT (Wei et al., 2024) introduces an in-context learning strategy that uses explicit instructions to guide LLM reasoning. KG-LLM (Yao et al., 2025) applies LLM to triple classification and relation prediction tasks in KGC. KoPA (Zhang et al., 2024) introduces the Knowledge Prefix Adapter to integrate pre-trained structural embeddings into LLMs, enhancing structure-aware reasoning. From a prompting perspective, LPNL (Bi et al., 2024) uses a two-stage sampling and divide-and-conquer method for scalable link prediction via natural language prompts. KC-GenRea (Wang et al., 2024) reformulates KGC as a re-ranking task for LLMs, and DIFT (Liu et al., 2024) implements KGC using discriminant instructions.

2.5 Biomedical Knowledge Graph Completion

BKGs have gained substantial attention for modeling structured knowledge in complex biomedical systems. Notable BKGs include Hetionet (Himmelstein et al., 2017), unifying 29 databases into a single network, PharmKG (Zheng et al., 2021), integrating 6 databases plus text-mined knowledge, and PrimeKG (Chandak et al., 2023), a precision medicine–focused graph consolidating 20 resources. For BKGs, KGC is vital in identifying missing triples to generate new hypotheses—for example, ICInet (Zhao et al., 2023) integrates GNNs, biological KGs, and gene expression profiles to predict cancer immunotherapy outcomes, while FuseLinker (Xiao et al., 2024) fuses pre-trained LLM text embeddings with Poincaré graph embeddings for improved GNN-based link prediction in drug repurposing.

3 Methodology

In this section, we introduce the proposed DrKGC. We begin with the preliminary and an overview, followed by a detailed description of each component.

3.1 Preliminary

Knowledge graph (KG). A KG (or BKG) can be represented as a directed multigraph, $\mathcal{G} = (\mathcal{E}, \mathcal{R}, \mathcal{T})$, where $\mathcal{E}$ is the set of entities, $\mathcal{R}$ is the set of relations and $\mathcal{T} = \{(h, r, t) | h, t \in \mathcal{E}, t \in \mathcal{R}\}$ is the set of triples. Each triple (h, r, t), with h and t the head and tail entities, and r representing the relation between them, describes a fact in KG.

Knowledge graph Completion (KGC). KGC aims to infer novel or missing triples from those already present in the graph. Let triples $\{(h', r', t') | h', t' \in \mathcal{E}, t' \in \mathcal{R}\}$, with $(h', r', t') \notin \mathcal{T}$, represent facts that are not unobserved in the KG. In this work, we cast KGC as the tasks of identifying missing entities in incomplete triples $(?, r_q, t_q)$ and $(h_q, r_q, ?)$, which are referred to as head prediction and tail prediction, respectively. Here, we call h_q or t_q the query entity and r_q the the query relation.

3.2 Overview

For simplicity, we only consider the head prediction scenario for illustration. Figure 1 illustrates the overall framework of DrKGC. DrKGC first employs a Question Generator to convert the incomplete triples $(?, r_q, t_q)$ into well-formed question Q. Then, a pre-trained lightweight model score each entity $\{e \in \mathcal{E} \mid (e, r_q, t_q) \notin \mathcal{T}\}$ for $(?, r_q, t_q)$, and selects the top k entities, where k is a hyperparameter, to form a candidates set $C = [e_1, e_2, e_3 ..., e_k]$. Subsequently, Subgraph Retriever retrieves a subgraph G based on the query entity t_q, all entities in C and the logic rules of r_q. A GCN-based adapter then leverages G to refine the embeddings of the t_q and the entities in C. Finally, the LLM selects the most plausible entity from the C, using both its own knowledge and the structured embeddings, to answer the question Q.

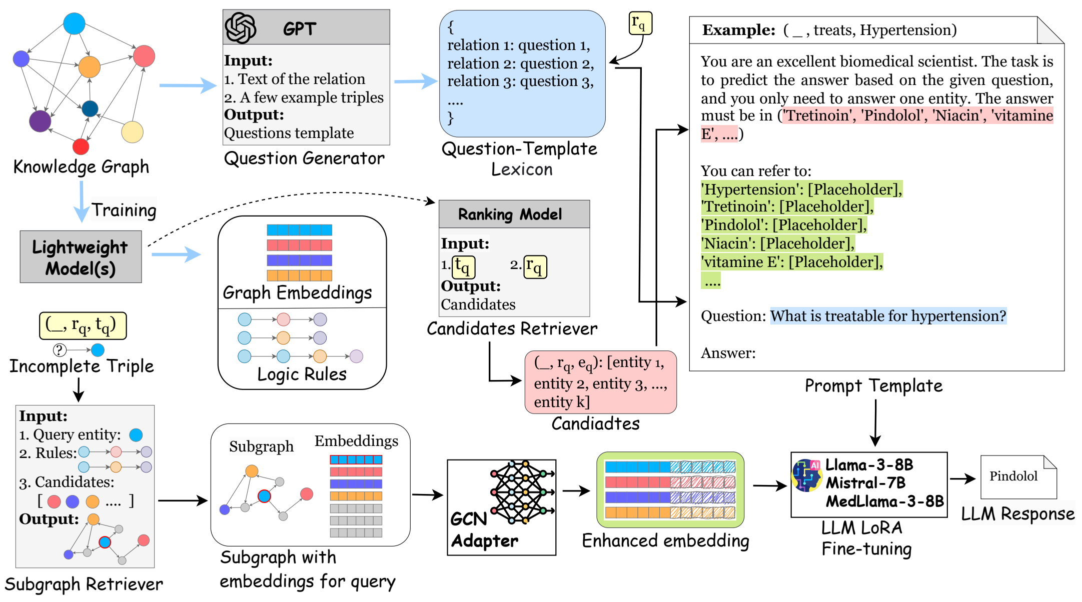

Figure 1: Overview of the DrKGC framework. Light-blue arrows denote the dataset-level workflow (run once per KG); black arrows denote the per-triple workflow (run for each incomplete triple).

3.3 Question Generator

To more accurately express the relations in the KG and convey the specific functional semantics of relations in BKGs, we reformulate the KGC task into a question-answering paradigm that aligns with LLMs. To achieve this, we introduce a simple yet effective approach comprising two main stages: Template Generation and Question Generation.

Template Generation. For each KG, we conduct a one-time process using GPT's few-shot context learning. Specifically, GPT-o1 is provided with a relation's name, its textual description, and a small set of sample triples, and is then instructed to generate a corresponding question template (with a placeholder for the query entity) via pattern induction. After processing each relation, we compile a question-template lexicon L (distinguishing between head and tail predictions) covering the entire relations set. The Appendix A.3 provides the complete lexicon for all four datasets.

Question Generation. After obtaining the lexicon L, we first map the query relation r_q to its corresponding question template. Next, we place the query entity r_q into the placeholder position to generate the complete question Q, which can be expressed as $Q = P(L(r_q), t_q)$.

3.4 Candidates Retriever

To mitigate the issues of an excessively large search space, limited LLM input capacity, and the tendency of LLMs to produce generic responses, we constrain the LLM's input and output using candidate sets. Similar to some previous works (e.g., (Zhang et al., 2024; Wei et al., 2024; Liu et al., 2024)), we also employ lightweight KGC methods to obtain entity rankings, which are then used to collect candidate entities.

Lightweight Model Training. Unlike previous work, we require more than just entity rankings. Therefore, we train not only lightweight structure-based models to obtain the structural embeddings of entities, but also lightweight rule-based models to learn the logical rules of relations in the KG, which guide the subsequent subgraph retrieval. By "lightweight," we refer to simpler, more resource-efficient approaches that do not rely on large-scale pretraining. This process is inherently flexible and any state-of-the-art method that can generate structured embeddings and perform rule mining may serve as a substitute. In our implementation, we focus on leveraging open-source methods that have demonstrated strong performance on the KGC task. The best structure-based model, M_S, generates embeddings for all entities, which we denote as the global embeddings $\mathbf{E}_{global} = \{\mathbf{e}_{global} \mid e \in \mathcal{E}\}$. The best rule-based model, M_R extracts logic rules for each relation. For every relation $r \in \mathcal{R}$, we denote the corresponding set of rules as $\mathcal{L}_r$.

Candidates Collection. The best performing lightweight model is used to rank the candidates. Specifically, for head prediction, we specify the query relation and query entity, replace the head with each entity $\{e \in \mathcal{E} \mid (e, r_q, t_q) \notin \mathcal{T}\}$ for $(?, r_q, t_q)$, and compute a likelihood score for each resulting triple, which reflects its plausibility. The replaced entities in top k triples are then selected to form the candidate set $C = [e_1, e_2, e_3..., e_k]$.

3.5 Dynamic Subgraph RAG

Retrieval-augmented generation (RAG) integrates retrieval-based methods with generative models to enhance the quality and accuracy of generated text (Lewis et al., 2020). Inspired by this idea, we propose a dynamic subgraph RAG strategy tailored for KGC tasks, which comprises two key components: Dynamic Subgraph Retrieval and Structure-Aware Embedding Enhancement.

Dynamic Subgraph Retrieval. To enable the LLM to select the correct answer from C based on the query entity t_q and query relation r_q, it is crucial to augment the graph context by retrieving an informative subgraph. To this end, we propose a bottom-up dynamic subgraph retrieval scheme, which is dynamic in that it does not mechanically retrieve the subgraph solely based on the t_q and r_q, but rather adapts to variable candidates sets. Specifically, we first ensure that both the t_q and all candidate entities $e \in C$ are included in the subgraph G, and then retrieve the shortest paths connecting each $e \in C$ to the t_q to guarantee connectivity. Next, we sort the logical rules in $\mathcal{L}_{r_q}$ by their assigned confidence scores and sequentially use them to search the paths from the $e \in C$ to t_q, thereby enriching the subgraph. This process continues until the number of triples reaches a preset threshold τ, which serves to constrain the subgraph's size. Finally, if the number of triples remains below τ after these steps, we augment the subgraph with additional triples connected to other entities from $e \in C$ and t_q via the r_q and its logical rules. Further details are provided in Appendix A.4.

Structure-Aware Embedding Enhancement.

Unlike traditional RAG, integrating structured subgraphs directly into the prompts is challenging. Even if described in text, much of the structural information is lost, and the text may be excessively long due to the richness of the subgraphs. To overcome this limitation, we exploit the subgraph's structural information to vectorize the graph context. We refer to the resulting embeddings as local embeddings $\mathbf{E}_{local} = \{\mathbf{e}_{local} \mid e \in \mathcal{E}\}$.

To obtain local embeddings and enhance the overall structural representation, we design a graph convolutional network (GCN)-based adapter. It comprises a low-dimensional relational GCN and a subsequent adapter that projects the resulting vectors to the LLM input layer's dimensionality. Specifically, for each query subgraph, the GCN is initialized with the global embeddings of all entities and then updates these representations via the neighborhood aggregation mechanism to produce the local embeddings. We concatenate the global and local embeddings to form the final enhanced structural embedding, i.e., $\mathbf{e}_{enhance} = [\mathbf{e}_{global}; \mathbf{e}_{local}]$. To reduce computational overhead for graph, GCN computations are performed in a low-dimensional space. Consequently, we employ an adapter to map the resulting structural embeddings to the LLM input dimension for seamless integration. During LoRA fine-tuning, we allow gradients to flow through the entire model, including the GCN adapter.

3.6 Prompt Template

Appendix A.2 presents the detailed prompt template. In summary, for each queried incomplete triple, our prompt comprises the following components: the instruction I for KGC; the candidates set C; special `[Placeholder]` tokens for the structured embeddings—they will be replaced by the actual enhanced structural embeddings of t_q and each $e \in C$ after token vectorization; and the question Q generated by the Question Generator.

4 Experiments

4.1 Experiment Setup

Dataset. We evaluate our proposed method on two benchmark KG datasets, WN18RR (Dettmers et al., 2018) and FB15k-237 (Toutanova et al., 2015), and two widely used BKG datasets, PharmKG (Zheng et al., 2021) and PrimeKG (Chandak et al., 2023). Dataset statistics, detailed descriptions and processing procedures are provided in Appendix A.1.

Baseline Methods. For the KG and BKG datasets, we selected two sets of baselines.

(1) For the WN18RR and FB15k-237, we consider baselines spanning multiple categories: structure-based methods: TransE (Bordes et al., 2013), DistMult (Yang et al., 2014), RotatE (Sun et al., 2019) and CompGCN (Vashishth et al., 2019); rule-based methods: Neural-LP (Yang et al., 2017), RLogic (Cheng et al., 2022), and NCRL (Cheng et al., 2023); text-based methods: KG-BERT (Yao et al., 2019), SimKGC (Wang et al.,

Methods		WN18RR				FB15k-237			
		MRR	Hits@1	Hits@3	Hits@10	MRR	Hits@1	Hits@3	Hits@10
Structure-based	TransE	0.243	0.043	0.441	0.532	0.279	0.198	0.376	0.441
	DistMult	0.444	0.412	0.470	0.504	0.281	0.199	0.301	0.446
	RotatE	0.476	0.428	0.492	0.571	0.338	0.241	0.375	0.533
	CompGCN	0.479	0.443	0.494	0.546	0.355	0.264	0.390	0.535
Rule-based	Neural-LP	0.381	0.368	0.386	0.408	0.237	0.173	0.259	0.361
	RLogic	0.470	0.443	–	0.537	0.310	0.203	–	0.501
	NCRL	0.670	0.563	–	**0.850**	0.300	0.209	–	0.473
Text-based	KG-BERT	0.216	0.041	0.302	0.524	–	–	–	0.420
	SimKGC	0.671	0.595	0.719	0.802	0.336	0.249	0.362	0.511
	KGLM	0.467	0.330	0.538	0.741	0.298	0.200	0.314	0.468
	GHN	0.678	0.596	0.719	0.821	0.339	0.251	0.364	0.518
Generation-based	KICGPT	0.564	0.478	0.612	0.677	0.412	0.327	0.448	0.554
	COSIGN	0.641	0.610	0.654	0.715	0.368	0.315	0.434	0.520
	DIFT	0.686	0.616	0.730	0.806	0.439	0.364	0.468	0.586
Hybrid	StAR	0.551	0.459	0.594	0.732	0.365	0.266	0.404	0.562
	CoLE	0.587	0.532	0.608	0.694	0.389	0.294	0.430	0.574
DrKGC (Ours)		**0.716**	**0.654**	**0.757**	0.813	**0.472**	**0.406**	**0.498**	**0.599**

Methods		PharmKG				PrimeKG			
		MRR	Hits@1	Hits@3	Hits@10	MRR	Hits@1	Hits@3	Hits@10
Structure-based	TransE	0.091	0.034	0.092	0.198	0.281	0.194	0.315	0.451
	RotatE	-	-	-	-	0.382	0.285	0.419	0.588
	DistMult	0.063	0.024	0.058	0.133	0.212	0.148	0.238	0.341
	ComplEx	0.075	0.030	0.071	0.155	0.204	0.141	0.266	0.340
	R-GCN	0.067	0.027	0.062	0.139	0.640	0.569	0.680	0.761
	HRGAT	0.154	0.075	0.172	0.315	0.443	0.347	0.489	0.637
DrKGC (Ours)		**0.266**	**0.183**	**0.293**	**0.436**	**0.658**	**0.592**	**0.705**	**0.770**

Table 1: Comparison of DrKGC (using Llama-3-8B) and baselines on WN18RR, FB15k-237, PharmKG and PrimeKG. For each metric, the best performance is highlighted in **bold**, and the second-best is underlined.

2022), KGLM (Youn and Tagkopoulos, 2022) and GHN (Qiao et al., 2023); generation-based methods: KICGPT (Wei et al., 2024), COSIGN (Li et al., 2024) and DIFT (Liu et al., 2024); and hybrid methods: StAR (Wang et al., 2021) and CoLE (Liu et al., 2022). The baseline comparisons in this paper are based on the reported performance values of these methods.

(2) For PharmKG and PrimeKG, we focus on structure-based methods well-suited for BKG, including TransE, RotatE, DistMult, ComplEx, R-GCN (Schlichtkrull et al., 2018), and HRGAT (Liang et al., 2023). The baseline performance for PharmKG is taken from the reported values reported in the original PharmKG (Zheng et al., 2021) paper; while for PrimeKG, the baseline comparisons were conducted by ourselves.

Implementation Details. In the lightweight model training stage, we trained NCRL to mine logical rules for the four datasets. For global structural embeddings, we employed RotatE for WN18RR and FB15k-237, and HRGAT for PharmKG, with hyperparameters consistent with the original publications. For PrimeKG, we used R-GCN with our optimal hyperparameter settings to obtain global embeddings. For WN18RR and FB15k-237, we additionally utilize the ranking results from SimKGC and CoLE, whereas, for PharmKG and PrimeKG, we directly employ HRGAT and R-GCN for ranking. The candidates set size is fixed at 20. For the fine-tuning stage, we compared Llama-3-8B (Dubey et al., 2024), Llama-3.2-3B(Dubey et al., 2024), MedLlama-3-8B (johnsnowlabs, 2024) and Mistral-7B (Jiang et al., 2023) as our LLMs. We employed LoRA for efficient parameter tuning, with the primary hyperparameters set to $r = 64$, $\alpha = 16$, a dropout rate of 0.1 and a learning rate of 2×10^{-4}. Model performance was evaluated using ranking-based metrics, including Mean Reciprocal Rank (MRR) and Hits@1, Hits@3, and Hits@10 under the "filtered" setting (Bordes et al., 2013). Additional training details are in Appendix A.5.

All experiments were conducted on an AMD EPYC 7763 64-Core CPU, an NVIDIA A100-

SXM4-40GB GPU (CUDA 12.4), and Rocky Linux 8.10.

4.2 Main Results

Table 1 demonstrates that the proposed DrKGC achieves state-of-the-art performance on WN18RR, FB15k-237, PharmKG, and PrimeKG across most metrics. On WN18RR, although DrKGC trails NCRL and GHN in Hits@10, it outperforms all generation-based approaches on all evaluated metrics. Notably, both text-based and generation-based methods yield lower Hits@10 scores than NCRL, and the gap between DrKGC and text-based GHN is minimal (only -0.97%). For FB15k-237, DrKGC outperformed all baselines across every metrics, achieving improvements of 7.5% in MRR and 11.4% in Hits@1. Given FB15k-237's diverse set of relations and semantic patterns, these results underscore demonstrate the advantage of DrKGC in capturing diverse relations and semantic nuances. For PharmKG and PrimeKG, DrKGC also outperforms all baselines across all metrics. This demonstrates that, even though BKGs lack extensive text information and LLMs have not been pre-trained on specialized biomedical corpora, DrKGC can still achieve strong performance by leveraging LLMs' understanding of semantics and structural embeddings.

4.3 Ablation Studies

We conducted ablation studies on all four datasets to assess the contribution of each component in DrKGC, with the results presented in Table 2. In the first ablation study, we removed the rule restrictions during subgraph retrieval. The results show that DrKGC's performance declined across all four datasets, with a more pronounced drop in KGs than in BKGs. In the second study, we eliminated local embeddings and relied solely on global embeddings as the structural reference for entities. This change also led to performance degradation on all datasets. In the third study, we removed the structural embeddings entirely, forcing the LLM to select the correct answer directly from the candidates set without any structural reference. The significant performance decline observed for both KGs and BKGs confirms the importance of incorporating structural information into LLM predictions. Finally, we omitted the question template and instead directly instructed the LLM to complete the incomplete triple. While it resulted in only a slight performance drop on KGs, it had a substantial impact on BKGs. This can be attribute that the relations in BKGs are inherently functional and mechanism; for instance, asking the LLM "What gene causes Parkinson's disease?" provides clearer instruction than simply prompting it to complete an incomplete triple such as (?, causes, Parkinson's disease).

w/o	**WN18RR**		**FB15k-237**	
	MRR ($\Delta\%$)	Hits@1 ($\Delta\%$)	MRR ($\Delta\%$)	Hits@1 ($\Delta\%$)
rules	0.684 (-4.47)	0.612 (-6.42)	0.448 (-5.08)	0.375 (-7.64)
local embedding	0.676 (-5.59)	0.596 (-8.87)	0.439 (-6.99)	0.361 (-11.1)
embedding	0.669 (-6.56)	0.582 (-11.0)	0.433 (-8.26)	0.351 (-13.5)
question template	0.711 (-0.70)	0.647 (-1.07)	0.469 (-0.64)	0.401 (-1.23)
DrKGC (raw)	0.716	0.654	0.472	0.406
w/o	**PharmKG**		**PrimeKG**	
	MRR ($\Delta\%$)	Hits@1 ($\Delta\%$)	MRR ($\Delta\%$)	Hits@1 ($\Delta\%$)
rules	0.264 (-0.75)	0.181 (-1.09)	0.648 (-1.52)	0.578 (-2.36)
local embedding	0.261 (-0.88)	0.176 (-3.83)	0.631 (-4.10)	0.539 (-8.95)
embedding	0.260 (-2.26)	0.174 (-4.92)	0.619 (-5.93)	0.510 (-13.9)
question template	0.258 (-3.01)	0.172 (-6.01)	0.613 (-6.83)	0.510 (-13.9)
DrKGC (raw)	0.266	0.183	0.658	0.592

Table 2: Ablation study results on four datasets.

4.4 DrKGC under Complex Conditions

To further verify DrKGC's robustness, we evaluated both its inductive prediction capability and its resilience under noisy conditions on WN18RR. Specifically, for the inductive setting, we extracted all test triples whose entities or relation never appear in the training set and measured DrKGC's performance on those unseen-entity cases. For the noise experiment, we replaced a fixed proportion of triples in the training set with random negative triples and then assessed the resulting impact on DrKGC's metrics. The results are summarized in Figure 2. Under the inductive setting, our model experiences only modest performance drops (MRR: -5.4%; Hits@1: -6.7%), and even when injecting noise into 20% of the KG, the reductions in MRR and Hits@1 remain limited to -7.9% and -7.6%, respectively, demonstrating DrKGC's robustness.

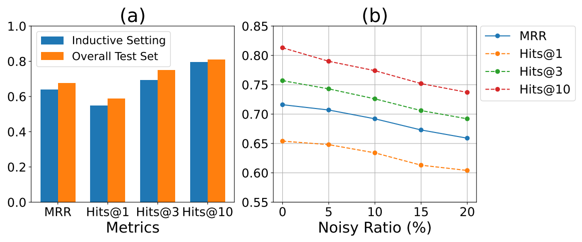

Figure 2: Robustness evaluation on WN18RR. (a) Comparison of evaluation metrics under the inductive setting versus the overall test set. (b) Impact of proportional noise addition on model performance.

4.5 Subgraph Size (τ) Sensitivity Analysis

We examine the model performance and efficiency under different τ values on WN18RR. The results are presented in Figure 3. As τ increases, model performance initially improves and then declines, with optimal results observed at $\tau = 100$ (our chosen hyperparameter) or 125; conversely, runtime grows linearly with τ. The performance trend is reasonable: a smaller τ restricts the information available in the subgraph, whereas an excessively large τ admits paths from low-confidence rules that degrade the quality of the local embeddings.

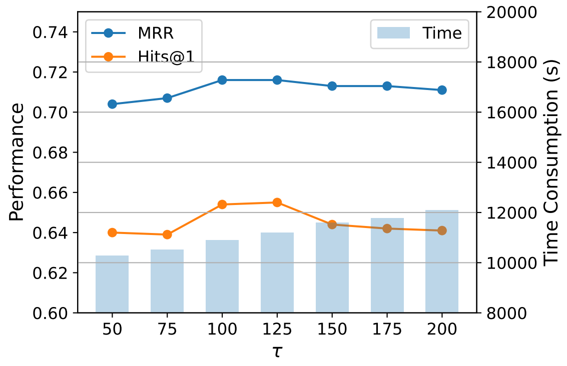

Figure 3: Impact of τ on DrKGC Performance and time consumption on WN18RR.

4.6 Case Study

To illustrate the practical utility of our approach, we conducted a drug repurposing case study for "breast cancer" using the PrimeKG dataset. In this study, we defined "Breast Cancer" as the query entity and "indication" as the query relation for head prediction. Recognizing that multiple drugs may be effective in treating breast cancer, we employed DrKGC to generate the top 10 predictions. This process was executed iteratively.

To validate our results, we conducted a manual evaluation by clinical trials and published literature (Zheng et al., 2021; Xiao et al., 2024). Specifically, if a predicted drug is documented on ClinicalTrials.gov, we record the corresponding NCT ID as evidence. If not, we search PubMed for supporting literature and record the corresponding PMID. In the absence of evidence from either source, "No evidence found" is recorded.

Table 3 shows the results. Figure 4 illustrates a portion of the subgraph from the first round of predictions, where three drugs—Troglitazone, Rosiglitazone, and Cardarine—share mechanism paths that are multi-hop connected to the breast

	Predicted Drugs	Evidence Source	PMID or NCT ID
1	Enzalutamide	Clinical Trial	NCT02750358
2	Troglitazone	Literature	31894283
3	Rosiglitazone	Clinical Trial	NCT00933309
4	Dichloroacetic Acid	Clinical Trial	NCT01029925
5	GTI 2040	Clinical Trial	NCT00068588
6	Uridine Monophosphate	Literature	32382150
7	Nimesulide	Clinical Trial	NCT01500577
8	Cardarine	Literature	15126355
9	Drospirenone	Clinical Trial	NCT00676065
10	Vitamin A	Literature	34579037

Table 3: Top 10 predicted drugs for Breast Cancer.

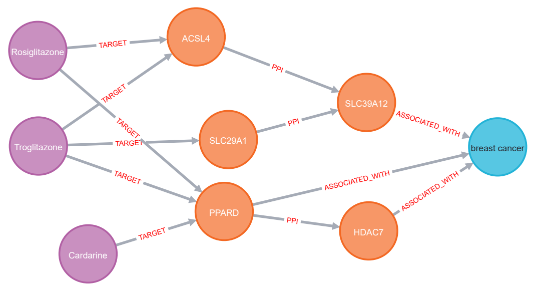

Figure 4: Example of multi-hop mechanism paths from drugs to Breast Cancer: purple, blue, and orange nodes represent drugs, diseases, and genes/proteins.

cancer entity. Consider the paths "Troglitazone–PPARD–breast cancer" and "Troglitazone–PPARD–HDAC7–breast cancer": Troglitazone targets PPARD, a druggable protein and a key molecular target in metastatic cancer (Zuo et al., 2017), and PPARD also interacts with HDAC7, which regulates genes critical for tumor growth and the maintenance of cancer stem cells (Caslini et al., 2019). This mechanism insight provided by the DrKGC's subgraph both supports and explains our predictions in biomedical domain.

5 Conclusion

In this paper, we propose a novel KGC framework, DrKGC. DrKGC fully exploits graph context information and flexibly integrates mechanisms such as dynamic subgraph information aggregation, embedding injection, and RAG, overcoming the limitations of previous generation-based methods in structural information loss, static entity representations, and generic LLM responses. Experimental results demonstrate that DrKGC achieves state-of-the-art performance on general KGs and performs exceptionally well on domain-specific KGs such as BKGs. By capturing graph context to generate informative subgraphs, DrKGC also enhances model interpretability, which is particularly valuable for biomedical applications.

6 Limitations

DrKGC relies on fine-tuning large language models, a process that is computationally intensive, and its performance is inherently constrained by the current capabilities of LLMs and lightweight models. Future work will focus on optimizing fine-tuning efficiency, enhancing LLM performance, and exploring extensions to other graph tasks such as reasoning and question answering. Moreover, retrieving more informative subgraphs may present additional challenges. In this work, we adopt a lightweight heuristic graph retrieval method; however, more rigorous rule-based detection and filtering technique and, alternative subgraph strategies, such as learning-driven subgraph retrieval, merit further investigation. We plan to explore these more sophisticated approaches in future research.

References

Baolong Bi, Shenghua Liu, Yiwei Wang, Lingrui Mei, and Xueqi Cheng. 2024. Lpnl: Scalable link prediction with large language models. In *Findings of the Association for Computational Linguistics ACL 2024*, pages 3615–3625.

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.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. *Advances in neural information processing systems*, 26.

Corrado Caslini, Sunhwa Hong, Yuguang J Ban, Xi S Chen, and Tan A Ince. 2019. Hdac7 regulates histone 3 lysine 27 acetylation and transcriptional activity at super-enhancer-associated genes in breast cancer stem cells. *Oncogene*, 38(39):6599–6614.

Payal Chandak, Kexin Huang, and Marinka Zitnik. 2023. Building a knowledge graph to enable precision medicine. *Scientific Data*, 10(1):67.

Zhe Chen, Yuehan Wang, Bin Zhao, Jing Cheng, Xin Zhao, and Zongtao Duan. 2020. Knowledge graph completion: A review. *Ieee Access*, 8:192435–192456.

Kewei Cheng, Nesreen K. Ahmed, and Yizhou Sun. 2023. Neural compositional rule learning for knowledge graph reasoning. *Preprint*, arXiv:2303.03581.

Kewei Cheng, Jiahao Liu, Wei Wang, and Yizhou Sun. 2022. Rlogic: Recursive logical rule learning from knowledge graphs. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 179–189.

Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. 2018. Convolutional 2d knowledge graph embeddings. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Daniel Scott Himmelstein, Antoine Lizee, Christine Hessler, Leo Brueggeman, Sabrina L Chen, Dexter Hadley, Ari Green, Pouya Khankhanian, and Sergio E Baranzini. 2017. Systematic integration of biomedical knowledge prioritizes drugs for repurposing. *Elife*, 6:e26726.

Shaoxiong Ji, Shirui Pan, Erik Cambria, Pekka Marttinen, and S Yu Philip. 2021. A survey on knowledge graphs: Representation, acquisition, and applications. *IEEE transactions on neural networks and learning systems*, 33(2):494–514.

Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. 2023. Mistral 7b. *arXiv preprint arXiv:2310.06825*.

johnsnowlabs. 2024. Jsl-medllama-3-8b-v2.0. `https://huggingface.co/johnsnowlabs/JSL-MedLlama-3-8B-v2.0`. Accessed: 2024-12-21.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems*, 33:9459–9474.

Jinpeng Li, Hang Yu, Xiangfeng Luo, and Qian Liu. 2024. Cosign: Contextual facts guided generation for knowledge graph completion. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 1669–1682.

Shuang Liang, Anjie Zhu, Jiasheng Zhang, and Jie Shao. 2023. Hyper-node relational graph attention network for multi-modal knowledge graph completion. *ACM Trans. Multimedia Comput. Commun. Appl.*, 19(2).

Xuan Lin, Zhe Quan, Zhi-Jie Wang, Tengfei Ma, and Xiangxiang Zeng. 2020. Kgnn: Knowledge graph neural network for drug-drug interaction prediction. In *IJCAI*, volume 380, pages 2739–2745.

Yang Liu, Zequn Sun, Guangyao Li, and Wei Hu. 2022. I know what you do not know: Knowledge graph embedding via co-distillation learning. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, CIKM '22, page 1329–1338, New York, NY, USA. Association for Computing Machinery.

Yang Liu, Xiaobin Tian, Zequn Sun, and Wei Hu. 2024. Finetuning generative large language models with discrimination instructions for knowledge graph completion. *Preprint*, arXiv:2407.16127.

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

Maximilian Nickel, Kevin Murphy, Volker Tresp, and Evgeniy Gabrilovich. 2015. A review of relational machine learning for knowledge graphs. *Proceedings of the IEEE*, 104(1):11–33.

Zile Qiao, Wei Ye, Dingyao Yu, Tong Mo, Weiping Li, and Shikun Zhang. 2023. Improving knowledge graph completion with generative hard negative mining. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 5866–5878, Toronto, Canada. Association for Computational Linguistics.

Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne Van Den Berg, Ivan Titov, and Max Welling. 2018. Modeling relational data with graph convolutional networks. In *The semantic web: 15th international conference, ESWC 2018, Heraklion, Crete, Greece, June 3–7, 2018, proceedings 15*, pages 593–607. Springer.

Zhiqing Sun, Zhi-Hong Deng, Jian-Yun Nie, and Jian Tang. 2019. Rotate: Knowledge graph embedding by relational rotation in complex space. *arXiv preprint arXiv:1902.10197*.

Kristina Toutanova, Danqi Chen, Patrick Pantel, Hoifung Poon, Pallavi Choudhury, and Michael Gamon. 2015. Representing text for joint embedding of text and knowledge bases. In *Proceedings of the 2015 conference on empirical methods in natural language processing*, pages 1499–1509.

Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. 2016. Complex embeddings for simple link prediction. In *International conference on machine learning*, pages 2071–2080. PMLR.

Shikhar Vashishth, Soumya Sanyal, Vikram Nitin, and Partha Talukdar. 2019. Composition-based multi-relational graph convolutional networks. *arXiv preprint arXiv:1911.03082*.

Brian Walsh, Sameh K Mohamed, and Vít Nováček. 2020. Biokg: A knowledge graph for relational learning on biological data. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, pages 3173–3180.

Bo Wang, Tao Shen, Guodong Long, Tianyi Zhou, Ying Wang, and Yi Chang. 2021. Structure-augmented text representation learning for efficient knowledge graph completion. In *Proceedings of the Web Conference 2021*, pages 1737–1748.

Liang Wang, Wei Zhao, Zhuoyu Wei, and Jingming Liu. 2022. Simkgc: Simple contrastive knowledge graph completion with pre-trained language models. *arXiv preprint arXiv:2203.02167*.

Yilin Wang, Minghao Hu, Zhen Huang, Dongsheng Li, Dong Yang, and Xicheng Lu. 2024. Kc-genre: A knowledge-constrained generative re-ranking method based on large language models for knowledge graph completion. *arXiv preprint arXiv:2403.17532*.

Yanbin Wei, Qiushi Huang, James T Kwok, and Yu Zhang. 2024. Kicgpt: Large language model with knowledge in context for knowledge graph completion. *arXiv preprint arXiv:2402.02389*.

Yongkang Xiao, Sinian Zhang, Huixue Zhou, Mingchen Li, Han Yang, and Rui Zhang. 2024. Fuselinker: Leveraging llm's pre-trained text embeddings and domain knowledge to enhance gnn-based link prediction on biomedical knowledge graphs. *Journal of Biomedical Informatics*, 158:104730.

Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. 2014. Embedding entities and relations for learning and inference in knowledge bases. *arXiv preprint arXiv:1412.6575*.

Fan Yang, Zhilin Yang, and William W Cohen. 2017. Differentiable learning of logical rules for knowledge base reasoning. *Advances in neural information processing systems*, 30.

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

Liang Yao, Jiazhen Peng, Chengsheng Mao, and Yuan Luo. 2025. Exploring large language models for knowledge graph completion. In *ICASSP 2025-2025 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5. IEEE.

Jason Youn and Ilias Tagkopoulos. 2022. Kglm: Integrating knowledge graph structure in language models for link prediction. *arXiv preprint arXiv:2211.02744*.

Yichi Zhang, Zhuo Chen, Lingbing Guo, Yajing Xu, Wen Zhang, and Huajun Chen. 2024. Making large language models perform better in knowledge graph completion. In *Proceedings of the 32nd ACM International Conference on Multimedia*, pages 233–242.

Lianhe Zhao, Xiaoning Qi, Yang Chen, Yixuan Qiao, Dechao Bu, Yang Wu, Yufan Luo, Sheng Wang, Rui Zhang, and Yi Zhao. 2023. Biological knowledge graph-guided investigation of immune therapy response in cancer with graph neural network. *Briefings in Bioinformatics*, 24(2):bbad023.

Shuangjia Zheng, Jiahua Rao, Ying Song, Jixian Zhang, Xianglu Xiao, Evandro Fei Fang, Yuedong Yang, and Zhangming Niu. 2021. Pharmkg: a dedicated knowledge graph benchmark for bomedical data mining. *Briefings in bioinformatics*, 22(4):bbaa344.

Marinka Zitnik, Monica Agrawal, and Jure Leskovec. 2018. Modeling polypharmacy side effects with graph convolutional networks. *Bioinformatics*, 34(13):i457–i466.

Xiangsheng Zuo, Weiguo Xu, Min Xu, Rui Tian, Micheline J Moussalli, Fei Mao, Xiaofeng Zheng, Jing Wang, Jeffrey S Morris, Mihai Gagea, et al. 2017. Metastasis regulation by ppard expression in cancer cells. *JCI insight*, 2(1).

A Appendix

A.1 Details of the Dataset

Table 4 presents the statistical details of the four datasets used in our study.

Datasets	Entities	Relations	Training	Validation	Testing
WN18RR	40,943	11	86,835	3,034	3,134
FB15K-237	14,541	237	272,115	17,535	20,466
PharmKG	7,601	28	400,788	49,536	50,036
PrimeKG	26,509	4	130,535	500	500

Table 4: Statistics of the four datasets.

WN18RR (*MIT License*), derived from WordNet (Miller, 1995), contains word sense entities and lexical-semantic relations like hypernymy. FB15k-237 (*CC BY 4.0*), from Freebase (Bollacker et al., 2008), consists of entities such as people and organizations with factual relations like affiliation and location. PharmKG (*Apache-2.0*) focuse on pharmaceutical data, capturing information about genes, diseases, chemicals. PrimeKG (*CC0 1.0*) is a multimodal BKG that unifies other biological entities like phenotypes and pathways for precision medicine analysis.

For WN18RR and FB15k-237, we adopted the node and relation texts provided by KG-BERT (Yao et al., 2019). For PharmKG, we utilized the PharmKG-8k version from the original work (Zheng et al., 2021),which filtered high-quality entities based on criteria such as FDA approval and MeSH tree classification and provided a partitioned dataset.

The PrimeKG dataset used in our study is a subset extracted from the original PrimeKG (Chandak et al., 2023) tailored for drug repurposing task. Specifically, we first selected triples from PrimeKG that have a head node of type "drug", a tail node of type "disease", and a relation of "indication". There are 9,388 such triples in total. Next, we randomly split them into 8,388 triples for training, 500 for validation, and 500 for testing, ensuring that the entities in the validation and test sets are also present in the training set. Finally, we enriched the training set by adding additional triples with the following (head, relation, tail) patterns: (drug, target, gene/protein), (gene/protein, associated with, disease), and (gene/protein, ppi, gene/protein). First, we added triples linking the existing drug and disease entities to gene/protein entities; then, we added triples connecting gene/protein entities to one another. In addition, to simplify the problem, we imposed an upper limit on the degree of gene/protein entities to mitigate the influence of hub nodes.

A.2 Prompt Template

As shown in Table 5, for both the general KG (WN18RR and FB15k-237) and the biomedical KG (PharmKG and PrimeKG), the prompt template remains consistent generally, comprising a simple instruction, a candidates set, corresponding structural embeddings (initially represented by [Placeholder]) for reference, and a question. The only difference is the role name assigned to the LLM (either linguist or biomedical scientist).

```
You are an excellent {linguist, biomedical
scientist}. The task is to predict the answer
based on the given question, and you only need to
answer one entity. The answer must be in
('candidate1', 'candidate2', 'candidate3',
'candidate4', 'candidate5', 'candidate6',
'candidate7', 'candidate8', 'candidate9',
'candidate10', 'candidate11', 'candidate12',
'candidate13', 'candidate14', 'candidate15',
'candidate16', 'candidate17', 'candidate18',
'candidate19', 'candidate20').
You can refer to the entity embeddings: 'query
entity': [Placeholder], 'candidate1':
[Placeholder], 'candidate2': [Placeholder],
'candidate3': [Placeholder], 'candidate4':
[Placeholder], 'candidate5': [Placeholder],
'candidate6': [Placeholder], 'candidate7':
[Placeholder], 'candidate8': [Placeholder],
'candidate9': [Placeholder], 'candidate10':
[Placeholder], 'candidate11': [Placeholder],
'candidate12': [Placeholder], 'candidate13':
[Placeholder], 'candidate14': [Placeholder],
'candidate15': [Placeholder], 'candidate16':
[Placeholder], 'candidate17': [Placeholder],
'candidate18': [Placeholder], 'candidate19':
[Placeholder], 'candidate20': [Placeholder].
Question: (The generated question)
Answer:
```

Table 5: Prompt template for DrKGC

A.3 Question-Template Lexicon

For each of the four datasets, two question-template lexicons are provided. One lexicon is designed to use the head node and relation to predict the tail node (corresponding to the tail prediction task), while the other is designed to use the tail node and relation to query the head node (corresponding to the head prediction task). In practice, the appropriate lexicon is selected based on the dataset and the prediction task (head or tail). For each incomplete triple, the corresponding question template is retrieved using the query relation, and then the query entity is inserted into the "{}" placeholder, generating the final question. Tables 6 and 7 illustrate the two question-template lexicons for WN18RR as examples.

```
# tail_prediction:
"also see":
"What is additionally relevant or similar to
{}?,"
"derivationally related form":
"What is a word or concept that is derivationally
related to {}?,"
"has part":
"What part does {} have?,"
"hypernym":
"What is a more general category or class that
includes {}?,"
"instance hypernym":
"Of what category or class is {} a specific
instance?,"
"member meronym":
"What is included as a member of {}?,"
"member of domain region":
"What is associated with {} in terms of regional
terms or concepts?,"
"member of domain usage":
"What is associated with {} in terms of specific
usage or context?,"
"similar to":
"What is similar to {}?,"
"synset domain topic of":
"What topic or field is {} associated with?,"
"verb group":
"What verb is in the same semantic or functional
group as {}?"
```

Table 6: Tail prediction question-template lexicon for WN18RR.

A.4 Rule Mining and Subgraph Retrieval Strategy

We first employ the lightweight NCRL model to mine logical rules from the knowledge graph. To further justify the use of NCRL, we evaluate DrKGC by replacing NCRL with RNNLogic and with randomly generated rules. The comparison results are presented in Table 8.

The results show that using rules mined by RNNLogic causes a slight decrease in DrKGC's performance, demonstrating that the choice of rule mining model can influence overall effectiveness. Employing randomly generated rules leads to a more pronounced degradation and falls behind both the NCRL and RNNLogic, which further validates the appropriateness of NCRL as our rule miner.

To further ensure the quality and reliability of the automatically learned rules, we apply a two-stage post-processing pipeline comprising conflict resolution and redundancy elimination. First, for

```
# head_prediction:
"also see":
"What is related or similar to {}?,"
"derivationally related form":
"What word or concept leads to {}?,"
"has part":
"What includes {} as a part?,"
"hypernym":
"What is a example or specific instance of {}?,"
"instance hypernym":
"What entity is classified under {}?,"
"member meronym":
"What larger group does {} belong to?,"
"member of domain region":
"What is associated with the region of {}?,"
"member of domain usage":
"What is used in the same context as {}?,"
"similar to":
"What is considered similar to {}?,"
"synset domain topic of":
"What is associated with the field or topic of
{}?,"
"verb group":
"What other verb is in the same functional or
semantic group as {}?"
```

Table 7: Head prediction question-template lexicon for WN18RR.

Dataset	RNNLogic		Random Rules	
	MRR (Δ%)	Hits@1 (Δ%)	MRR (Δ%)	Hits@1 (Δ%)
WN18RR	0.706 (–1.40)	0.640 (–2.14)	0.682 (–4.75)	0.609 (–6.88)
FB15K-237	0.455 (–3.60)	0.384 (–5.42)	0.446 (–5.51)	0.376 (–7.39)
PharmKG	0.265 (–0.36)	0.182 (–0.55)	0.262 (–1.50)	0.180 (–1.63)
PrimeKG	0.652 (–0.91)	0.588 (–0.68)	0.637 (–3.19)	0.569 (–3.89)

Table 8: Performance of DrKGC with RNNLogic and random rules ($\Delta\%$ values indicate differences from using NCRL).

conflict resolution, we group rules by identical bodies and, when a group yields more than one distinct head, which indicates potential conflict, we retain only the rule with the highest confidence score. Next, to eliminate redundancy, we examine pairs of rules that share the same head: if the body of rule A is a strict subset of the body of rule B and A's confidence exceeds B's, we remove B as redundant.

During subgraph retrieval, we constrain the subgraph size by the hyperparameter τ, which limits the number of triples and is set to 100 after comparing the DrKGC performance of taking $\{50, 100, 200\}$, and control its depth by the length of the rule. The maximum rule length is defined during the training of the logical rule learning model; we set this to 3 to match the configuration of the original NCRL work.

A.5 Model Training

Inspired by previous work (Wei et al., 2024; Liu et al., 2024), our model training does not strictly follow the traditional paradigm of using fixed training, validation, and test sets. Specifically, we first use the KG dataset's standard splits for training, validation, and testing to train a lightweight model. We then employ this trained lightweight model to perform head and tail predictions on every triple in the validation set, generating candidate rankings that are used to construct prompts. In the LLM fine-tuning phase, we re-partition the validation set (Liu et al., 2024) and utilize it to fine-tune the LLM. Finally, model performance is evaluated on the test set in the usual manner. For each triple in the test set, both head and tail predictions are conducted to ensure fairness. This approach not only reduces the volume of training data required for fine-tuning but also avoids the issue where the trained lightweight model consistently ranks the correct answer for incomplete triples in the training set first, which could mislead the LLM selection.

In lightweight models training phase, for WN18RR, FB15k-237, and PharmKG, we use the hyperparameters consistent with the original publications of their corresponding methods. For PrimeKG, the optimal parameters identified via grid search are provided in Table 9. In the LLM fine-tuning phase, we adjust the learning rate $\{2 \times 10^{-3}, 2 \times 10^{-4}\}$, the number of GCN layers $\{1, 2\}$ and the size of GCN hidden dimension $\{128, 256\}$, and set the epoch size to 15 with early stopping. The time required for LLM fine-tuning is detailed in Table 10.

	TransE	RotatE	DistMult	ComplEx	R-GCN	HRGAT
Batch Size	512	512	512	512	256	128
Learning Rate	2e-3	1e-4	1e-4	2e-3	1e-3	1e-3
Negative Sampling	512	512	512	512	512	40
Hidden Dimension	1000	2000	2000	1000	200	200

Table 9: Optimal hyperparameters for lightweight model on PrimeKG.

	WN18RR	FB15k-237	PharmKG	PrimeKG
Runtime	3:01:31	19:12:11	2:22:27	37:09

Table 10: LLM fine-tuning time statistics.

A.6 Comparison of Alternative LLMs

In this section, we compare the performance of DrKGC by replacing different LLMs.

First, we evaluate the impact of the LLM's size by comparing Llama-3-8B with a smaller variant, Llama-3.2-3B. The comparison results are presented in Table 11. The performance achieved with the Llama-3.2-3B LLM is inferior compared to that of the Llama-3-8B, which is consistent with our expectations. This difference arises from the reduced number of parameters in the smaller model, inherently limiting its expressive power and reasoning capabilities.

Dataset	MRR (Δ)	Hits@1 (Δ)	Hits@3 (Δ)	Hits@10 (Δ)
WN18RR	0.709 (-0.07)	0.644 (-0.10)	0.754 (-0.03)	0.811 (-0.02)
FB15k237	0.466 (-0.06)	0.397 (-0.09)	0.494 (-0.04)	0.596 (-0.03)
PharmKG	0.260 (-0.06)	0.172 (-0.11)	0.292 (-0.01)	0.436 (-0.00)
PrimeKG	0.656 (-0.02)	0.595 (-0.03)	0.691 (-0.14)	0.762 (-0.08)

Table 11: DrKGC Performance with Llama-3.2-3B (Δ values indicate differences from Llama-3-8B).

Then, we further investigate the impact of employing other different LLMs within DrKGC on prediction performance. In addition to Llama-3-8B, we compare Mistral-7B and a biomedical-focused instruction-tuning variant, MedLlama-3-8B. The results of replacing the LLM component in DrKGC are presented in Figure 5. Overall, Llama-3-8B delivers the best performance, while Mistral-7B underperforms, despite achieving the highest Hits@10 on FB15k-237. Notably, MedLlama-3-8B's overall performance is slightly inferior to that of Llama-3-8B, even on the two BKGs; it only attained the best result in Hits@1 on PharmKG. It can be found that MedLlama-3 performs worse than Llama-3-8B on biomedical datasets overall. The main reason is that biomedical LLMs are not pre-trained or fine-tuned for knowledge graph generation (KGC) or link prediction tasks. MedLlama-3 is optimized for biomedical question answering and medical text generation. While it does incorporate some biomedical and clinical knowledge, this does not necessarily mean an improvement in structured relational reasoning capabilities, which are critical for knowledge graph completion.

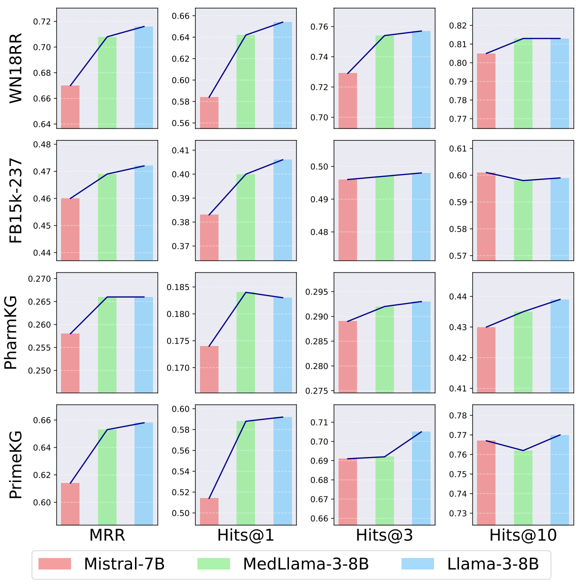

Figure 5: Comparison of DrKGC performance using different LLMs across four datasets.