KG-CF: Knowledge Graph Completion with Context Filtering under the Guidance of Large Language Models

Zaiyi Zheng[†], Yushun Dong[‡], Song Wang[†], Haochen Liu[†], Qi Wang[‡], and Jundong Li[†]

[†]*University of Virginia, Charlottesville, USA*, {sjc4fq, sw3wv, sat2pv, jundong}@virginia.edu
[‡]*Florida State University, Tallahassee, USA*, yd24f@fsu.edu
[‡]*Northeastern University, Boston, USA*, q.wang@northeastern.edu

Abstract—**Large Language Models (LLMs) have shown impressive performance in various tasks, including knowledge graph completion (KGC). However, current studies mostly apply LLMs to classification tasks, like identifying missing triplets, rather than ranking-based tasks, where the model ranks candidate entities based on plausibility. This focus limits the practical use of LLMs in KGC, as real-world applications prioritize highly plausible triplets. Additionally, while graph paths can help infer the existence of missing triplets and improve completion accuracy, they often contain redundant information. To address these issues, we propose KG-CF, a framework tailored for ranking-based KGC tasks. KG-CF leverages LLMs' reasoning abilities to filter out irrelevant contexts, achieving superior results on real-world datasets. The code and datasets are available at https://anonymous.4open.science/r/KG-CF.**

Index Terms—**knowledge graph, entity prediction, language models, inductive learning.**

I. Introduction

Knowledge Graphs (KGs) have become foundational in numerous real-world applications, including recommendation systems [1] and knowledge editing [2]. Structured as relational data, KGs encode vast factual information using triplets, where each triplet `(h, r, t)` specifies a relation `r` between entities `h` and `t` (e.g., `(Earth, orbits, Sun)`). Despite their utility, KGs are inherently sparse and incomplete, necessitating Knowledge Graph Completion (KGC) to predict missing triplets and thereby enrich the graph [3]. Traditional embedding-based methods, such as RotatE [4], have shown competitive results in KGC but often lack the ability to integrate external knowledge, including commonsense information not explicitly represented in the graph [5]. To address this, recent research leverages pretrained language models (PLMs) [6], with large language models (LLMs) in particular receiving significant attention for their robust reasoning and generalization capabilities [7].

Despite the growing interest in applying LLMs to KGC, current research still encounters significant limitations in practical applications. LLM-based KGC models primarily focus on triplet classification, performing binary evaluations (true or false) on potential missing triplets. However, most real-world KGs are sparse, with an imbalanced distribution of valid and invalid missing triplets. Conversely, traditional approaches (e.g., embedding-based models) emphasize ranking-based tasks, particularly entity prediction, which predicts missing entities for queries in the form `(?, r, t)` or `(h, r, ?)`. This task requires models to generate a ranked list of candidate head or tail entities based on relevance and plausibility, forming candidate triplets with the query. Such ranking is crucial in practice, as prioritizing likely missing triplets enhances efficiency and flexibility. However, two intrinsic challenges in LLM-based models hinder their performance on ranking-based tasks: (1) From the graphs's perspective, existing LLM-based frameworks [8], [9] primarily extract and input graph contextual information (e.g., graph topology, textual descriptions), so-called graph contexts, in textual form to enhance completion. However, in KGC tasks, some extracted graph contexts are irrelevant to the existence of given candidate triplets, introducing substantial redundancy and diverting the LLM's focus from the KGC task. (2) Sequential generation in LLMs makes them poorly suited for handling numerical values like floats [10], complicating the generation of precise plausibility scores for ranking candidate entities or triplets. Standard LLMs produce numbers digit by digit, leading to cumulative sequence errors [11]. This limitation also affects generating ranking lists. Moreover, triplet labels used in training are discrete (e.g., true/false), making it challenging to align these labels with continuous score outputs for autoregressive LLMs.

To address these challenges, we introduce KG-CF (**K**nowledge **G**raph Completion with **C**ontext **F**iltering). In this framework, LLMs are dedicated to filtering out irrelevant contextual information. Specifically, for any triplet `(h, r, t)` in a knowledge graph $\mathcal{G}$, we sample paths from the head entity `h` to the tail entity `t` in $\mathcal{G}$, forming a context set $\mathcal{C}$ to be filtered. The LLM then evaluates $\mathcal{C}$ for relevance to `(h, r, t)`. To reduce computational costs, we distill a smaller sequence classifier sc from the LLM to handle most of the context filtering. This allows us to effectively eliminate irrelevant paths and address the first challenge. Next, we train a smaller PLM, BERT [12], on the filtered context set $\mathcal{C}^*$ for path scoring. In the testing phase, we sample the corresponding

$\mathcal{C}$ for each triplet and use the highest score from $\mathcal{C}$ as the triplet's ranking score. By avoiding direct use of the LLM in ranking, we also overcome the second challenge.

Our contributions are summarized in three-fold:

- **Problem Formulation.** We summarize the challenges related to model design and training data for LLMs in KGC tasks. Moreover, we delineate a specific application (context filtering) of LLMs in this scenario.
- **Framework Design.** We propose a principled framework, KG-CF, which successfully leverages the knowledge encoded in the LLMs while still being able to align with the ranking-based tasks and evaluations in KGC.
- **Empirical Evaluation.** We conduct empirical evaluations on real-world KG datasets. The experiment results validate the superiority of the proposed model KG-CF compared with other alternatives in KGC tasks.

II. Preliminary

A. Problem Formulation

We denote the knowledge graph as $\mathcal{G} = \{\mathcal{E}, \mathcal{R}, \mathcal{T}\}$, where $\mathcal{R}$ represents relation types, $\mathcal{E}$ represents entities, and $\mathcal{T}$ includes all triplets in $\mathcal{G}$. A triplet $t \in \mathcal{T}$ is defined as $t = (e_h, r, e_t)$, with e_h as the head entity and e_t as the tail entity. Our focus is on entity prediction, covering two subtasks: head prediction and tail prediction [13]. We provide the definition for tail prediction, with head prediction defined analogously.

Definition 1 (***Tail Entity Prediction***). *Given a query $q = (e_h, r_q, ?)$ where r_q is the query relation, we define the completion of q by e_t as:*

$$c(q, e_t) = q|_{?=e_t} = (e_h, r_q, e_t), \tag{1}$$

where c denotes the completion function. Firstly, we need to identify the candidate set $\mathcal{C}$ for the tail:

$$\begin{aligned} &\mathcal{C} = \{e_i\}_{i=1\to n} \subseteq \mathcal{E} \setminus \{e_h\}, \\ &s.t.\ \forall e_t \in \mathcal{C}, c(q, e_t) \notin \mathcal{T}, \end{aligned} \tag{2}$$

where n is a predefined integer. Our objective is to identify a ranking list A of all candidates:

$$\forall i \in [1, n), score(A_i) \geq score(A_{i+1}) \tag{3}$$

where $score$ is the scoring function.

Example. Consider a KG of countries and their capitals. An example query in this graph is presented as follows:

$$q = (Japan, Capital, ?).$$

We have sampled a series of tail candidates: $\mathcal{C} = \{Paris, Tokyo, Peking, Berlin, Kyoto, London\}$. If there already exists a comprehensive KGC model, the ranking list could possibly be:

$$A = \{Tokyo, Kyoto, Peking, Paris, London\}.$$

III. Methodology

A. Model Overview

In this section, we present our principled framework, KG-CF, which leverages LLM inference capabilities to train sequence classifiers for context filtering in KGC tasks. Figure 1 outlines our model pipeline, divided into three key stages: path labeling, sequence classification for filtering, and PLM scoring. To address the exponential growth in path numbers with increasing truncation length, we introduce a sequence classifier to filter paths, leveraging the generalizability of graph topology in knowledge graphs to reduce computational costs.

B. Path Labeling through LLM

Path Formulation. For a query $q = (e_h, r_q, ?)$ and a potential completion $c(q, e_t)$, we can execute a breadth-first search algorithm on the graph to acquire a straightforward inferential path from e_h to e_t. Each trajectory T is formulated as a list of triplets $\{t_i\}_{i=0\to n}$ that starts from e_h and ends at a potential tail entity e_t:

$$T = ((e_h, r_0, e_1), (e_1, r_1, e_2), ..., (e_n, r_n, e_t)).$$

We define an inference path P as the concatation of a trajectory T_q along with the completion $c(q, e_t) = (e_h, r_q, e_t)$:

$$P = ((e_h, r_q, e_t), T). \tag{4}$$

LLM Inference. So far, we have formalized the objects that need to be filtered. Subsequently, we transform the paths into character sequences to adapt the inference paths to the input of LLMs. Therefore, we obtain labels for all the paths associated with $c(q, e_t)$:

$$\mathcal{Y}_{c(q,e_t)} = LLM(instruction \oplus f(\mathcal{P}_{c(q,e_t)})), \tag{5}$$

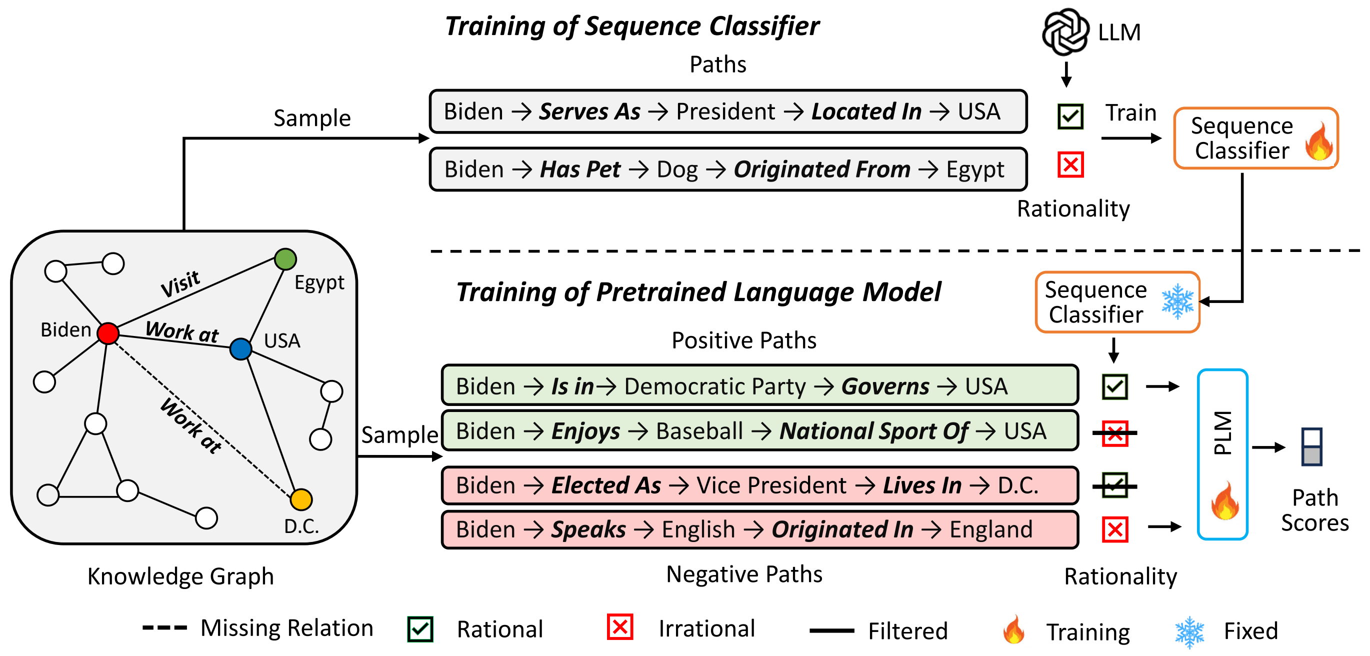
where $\oplus$ denotes the concatenation operation, $\mathcal{P}_{c(q,e_t)}$ contains all the possible paths related to $c(q, e_t)$ and f transform the paths into texts. The result $\mathcal{Y}_{c(q,e_t)}$ contains labels for paths in $\mathcal{P}_{c(q,e_t)}$ while each label is in $\{0, 1\}$. Based on this operation, we construct a dataset $\mathcal{D}_{sc}$ for the sequence classifier training, and we introduce the details in the next section. The detailed process is presented in Algorithm 1.

Note that inverse relationships are allowed (suitable for head prediction) and all triplets in the path are represented in the standard forward order. For example, triplet $(Lakers, inv(plays\ for), Lebron\ James)$ will be interpreted as *"Lebron James plays for Lakers"*, where $inv()$ represents the function of inversing.

C. Sequence Classifier

We employ an LSTM [14] model as the sequence classifier $M_{sc} : \mathcal{P} \to \{0, 1\}$ to implements functionality similar to LLM in Equation (5). Considering a path $P = ((e_h, r_q, e_t), ((e_h, r_0, e_1), ..., (e_{n-1}, r_{n-1}, e_t)))$, we have:

$$\begin{aligned} &\boldsymbol{h_0} = R(0, \boldsymbol{e_h} \oplus \boldsymbol{r_0} \oplus \boldsymbol{e_1} \oplus \boldsymbol{r_q}), \\ &\boldsymbol{h_i} = R(\boldsymbol{h_{i-1}}, \boldsymbol{e_i} \oplus \boldsymbol{r_i} \oplus \boldsymbol{e_{i+1}} \oplus \boldsymbol{r_q}), i \leq n-1, \\ &\hat{y} = \sigma(fc(h_{n-1})), \end{aligned} \tag{6}$$



Training of Sequence Classifier
Paths
LLM
Sample
Biden → Serves As → President → Located In → USA
Biden → Has Pet → Dog → Originated From → Egypt
Train
Sequence Classifier
Rationality
Training of Pretrained Language Model
Sequence Classifier
Positive Paths
Biden → Is in → Democratic Party → Governs → USA
Biden → Enjoys → Baseball → National Sport Of → USA
Biden → Elected As → Vice President → Lives In → D.C.
Biden → Speaks → English → Originated In → England
Sample
Negative Paths
Rationality
PLM
Path Scores
Visit
Egypt
Biden
Work at
USA
Work at
D.C.
Knowledge Graph
Missing Relation
Rational
Irrational
Filtered
Training
Fixed

Fig. 1. **The pipeline of KG-CF**. The model operates in three primary steps: 1) Sample a small set of paths and use LLMs to generate rationality labels for them. 2) Train our sequence classifier on the sampled path set. Then, filter all paths using the sequence classifier, retaining only “rational” positive and “irrational” negative sample paths. 3) Feed all data, including queries, tail nodes, and inference paths, into a PLM for binary classification training. The PLM scorer will output a number between 0 and 1 as the score for the current triplet candidate.

Algorithm 1 Dataset for Sequence Classifer

```
Require: KG $\mathcal{G} = (\mathcal{E}, \mathcal{R}, \mathcal{T})$, Maximum path length $m$ and
    path numbers per relation $n$.
Ensure: Dataset $\mathcal{D}_{sc}$ for Sequence Classifer.
 1: $\mathcal{D}_{sc} \leftarrow \emptyset$
 2: for all $r \in \mathcal{R}$ do
 3:     $r_{count} \leftarrow 0$
 4: end for
 5: for all triples $t \in T$ do
 6:     $e_h, r, e_t \leftarrow t$
 7:     if $r_{count} > n$ then
 8:         continue
 9:     end if
10:     $\mathcal{P} \leftarrow$ All simple paths from $e_h$ to $e_t \in \mathcal{T} \setminus \{t\}$ with
    path length up to $m$
11:     $\mathcal{L} \leftarrow$ Label each path using LLM
12:     $\mathcal{D}_{sc} \leftarrow \mathcal{D}_{sc} \cup \{(\mathcal{P}[i], \mathcal{L}[i]) \mid 0 \le i \le |\mathcal{P}|\}$
13:     $r_{count} \leftarrow r_{count} + 1$
14: end for
15: return $\mathcal{D}_{sc}$
```

where R denotes the LSTM model, $\hat{y}$ is the prediction by applying classifier layer fc and Sigmoid function σ to the last hidden state h_{n-1}. In particular, we use the sequence classifier to filter and construct the dataset $\mathcal{D}_{plm}$ for PLM model training in Sec. III-D. The detailed process is described in Algorithm 2.

Optimization. We use the cross-entropy loss to train the sequence classifier model:

$$\mathcal{L} = \sum_{i=1}^{N} \left[y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i) \right]. \tag{7}$$

Algorithm 2 Dataset for PLM

```
Require: KG $\mathcal{G} = (\mathcal{E}, \mathcal{R}, \mathcal{T})$, Number of negative instances
    $neg\_num$, Threshold $th$, Maximum path length $m$, Se-
    quence Classifier $sc$.
Ensure: Dataset $\mathcal{D}_{PLM}$ for PLM training.
 1: $\mathcal{D}_{PLM} \leftarrow \emptyset$
 2: for all triples $t \in \mathcal{T}$ do
 3:     $e_h, r, e_t \leftarrow t$
 4:     $\mathcal{P}_{pos} \leftarrow$ All simple paths from $e_h$ to $e_t \in \mathcal{T} \setminus \{t\}$ with
    up to $m$
 5:     $\mathcal{P}_{pos} \leftarrow \{p | p \in \mathcal{P}_{pos} \wedge sc(p) > th\}$
 6:     $\mathcal{D}_{pos} \leftarrow \{(p, true) \mid p \in \mathcal{P}_{pos}\}$
 7:     $\mathcal{D}_{PLM} \leftarrow \mathcal{D}_{PLM} \cup \mathcal{D}_{pos}$
 8:     for $i \leftarrow 1$ to $neg\_num$ do
 9:         Pick an $e \in \mathcal{E} \setminus \{e_h\}$ $s.t.$ $(e_h, r, e) \notin \mathcal{T}$
10:         $\mathcal{P}_{neg} \leftarrow$ All simple paths from $e_h$ to $e_t \in \mathcal{T}$ with
    path length up to $max\_hops$
11:         $\mathcal{P}_{neg} \leftarrow \{p | p \in \mathcal{P}_{neg} \wedge sc(p) < th\}$
12:         $\mathcal{D}_{neg} \leftarrow \{(p, false) \mid p \in \mathcal{P}_{neg}\}$
13:         $\mathcal{D}_{PLM} \leftarrow \mathcal{D}_{PLM} \cup \mathcal{D}_{neg}$
14:     end for
15: end for
16: return $\mathcal{D}_{PLM}$
```

TABLE I
PERFORMANCES ON **TRANSDUCTIVE** ENTITY PREDICTION OF TRADITIONAL (TOP) AND PLM-BASED APPROACHES (BOTTOM). RESULTS ARE IN PERCENTAGE WITH THE BEST ONES SHOWN IN **BOLD**.

Datasets	**WN18RR**		**FB15K-237**		**NELL-995**	
	Hits@1	**MRR**	**Hits@1**	**MRR**	**Hits@1**	**MRR**
RuleN	64.6	67.1	60.2	67.5	63.6	73.7
GRAIL	64.4	67.6	49.4	59.7	61.5	72.7
MINERVA	63.2	65.6	53.4	57.2	55.3	59.2
TuckER	60.0	64.6	61.5	68.2	72.9	80.0
BERTRL	66.3	68.7	61.9	69.6	68.6	78.2
KG-CF	**67.5**	**70.3**	**62.3**	**70.9**	**73.1**	**82.0**

TABLE II
PERFORMANCES ON **INDUCTIVE** ENTITY PREDICTION.

Datasets	**WN18RR**		**FB15K-237**		**NELL-995**	
	Hits@1	**MRR**	**Hits@1**	**MRR**	**Hits@1**	**MRR**
RuleN	74.6	78.2	41.5	46.3	63.8	71.1
GRAIL	76.9	79.9	39.0	46.9	55.4	67.5
KG-BERT	43.6	57.4	34.1	50.0	24.4	41.9
BERTRL	75.3	79.5	**54.1**	**60.6**	71.7	81.0
KG-CF	**78.5**	**80.9**	51.2	58.3	**79.5**	**86.6**

Here, N is the number of samples, y_i represents the true label of the i-th sample (with a value of 0 or 1), and $\hat{y}_i$ denotes the predicted probability of the i-th sample being in class 1. The label y is generated by LLMs.

D. PLM Scoring

In this section, we demonstrate the scoring and training process of our PLM scorer. Considering a path $P = (c(q, e_t), T)$, we compute the text representation and its score as follows:

$$P_{text} = text(c(q, e_t)) \otimes text(T), \tag{8}$$

$$score(P) = \hat{y}_P = \sigma(PLM(P_{text})), \tag{9}$$

where $text(\cdot)$ stands for the textualize function, $\otimes$ denotes concatenating and independently annotating two segments of text, and $\hat{y}_P$ is the score of the path P by applying the sigmoid function $\sigma(\cdot)$ on the outputs of the PLM model. We utilize the same loss function as Eq. (7) for PLM training, while the label represent the existence of triplets in KG.

Scoring and Ranking. To provide a basis for entity ranking, inspired by BERTRL [15], we represent the confidence score of each completion $c(q, e_t)$ using the highest path score corresponding to it:

$$score(e_t) = max\{\hat{y}_P | P \in \mathcal{P}_{c(q,e_t)}\} \tag{10}$$

A special case occurs when $\mathcal{P}_{c(q,e_t)} = \emptyset$. In this scenario, we manually assign the lowest score to the completion.

IV. EMPIRICAL EVALUATION

In this section, we will answer the following four questions through experiments: (1) How well can KG-CF perform in knowledge graph completion tasks? (2) How do different filtering choices contribute to the overall performance of KG-CF? (3) How does the maximum path length affect the accuracy of the completion?

A. Experimental Settings

Datasets. Three widely utilized real-world knowledge graphs: NELL-995 [16], FB15K-237 [13], and WN18RR [17]. NELL-995 and FB15K-237 are relation extraction datasets sourced from web text. WN18RR is derived from WordNet with refined relations. To streamline training, we sample a subset from each source dataset for evaluation.

Baselines. We incorporate methods from prior research as baselines. Among them, RuleN [18] (rule-based) and GRAIL [19] (GNN-based) support both inductive and transductive scenarios, while MINERVA [20] (reinforcement learning) and TuckER [21] (embedding-based) are limited to transductive settings. Additionally, we include KG-BERT [22] and BERT-RL [15], both leveraging pretrained language models.

B. Evaluation Method

In both transductive and inductive scenarios, we separately evaluate our approach on two subtasks: tail prediction and head prediction. The average metrics of two scenarios are shown as the final results. Following GRAIL [19] and BERTRL [15], we randomly select another 49 tail entities $\{t_i\}_{i=1\to 49}$ for each test triplet $(h_{test}, r_{test}, t_{test})$ and form a candidate set $T_{test} = \{t_{test}\} \cup \{t_i\}_{i=1\to 49}$. Despite t_{test}, we make sure that for any other $t \in T$, $(h_{test}, r_{test}, t) \notin \mathcal{G}$. By the end, we rank t_{test} based on scores and compute metrics.

C. Main Results (Question 1)

In this subsection, we assess our KG-CF framework on three knowledge graphs across transductive (Table I) and inductive scenarios (Table II), leading to these insights: (1) KG-CF surpasses most baselines across datasets and scenarios, highlighting the effectiveness of using LLMs and sequence classifiers to refine graph context. (2) KG-CF demonstrates greater stability in transductive scenarios compared to inductive ones. (3) Our method achieves the most significant improvements on NELL-995, which, unlike FB15K-237, includes richer textual descriptions of entities (e.g., "person Mexico Ryan Whitney" rather than "Ryan Whitney"). This detail allows the LLM to better handle rare nouns, enhancing its precision.

D. Ablation Study (Question 2)

We conducted an ablation study on the WN18RR dataset (both transductive and inductive) where three components are removed separately: positive path filtering ($-pf$), negative path filtering ($-nf$), and trajectory entities in the paths ($-te$, i.e., relation only). We present the results in Figure 3.

Positive Path Filtering. Under this setting, we assume that all paths from e_h to e_t in the positive triplet (e_h, r_q, e_t) conform to standard reasoning logic, thus preserved during the data filtering phase. The results showed a slight decline compared to the original model, indicating that our sequence classifier can enhance the rationality of positive paths.

Negative Path Filtering. In this setting, we assume that for a negative triplet $(e_h, r_q, e_t) \notin \mathcal{T}$, all paths from e_h to e_t fail to confirm r_q's existence (though, given KG incompleteness, we consider this assumption incorrect). This ablation led to

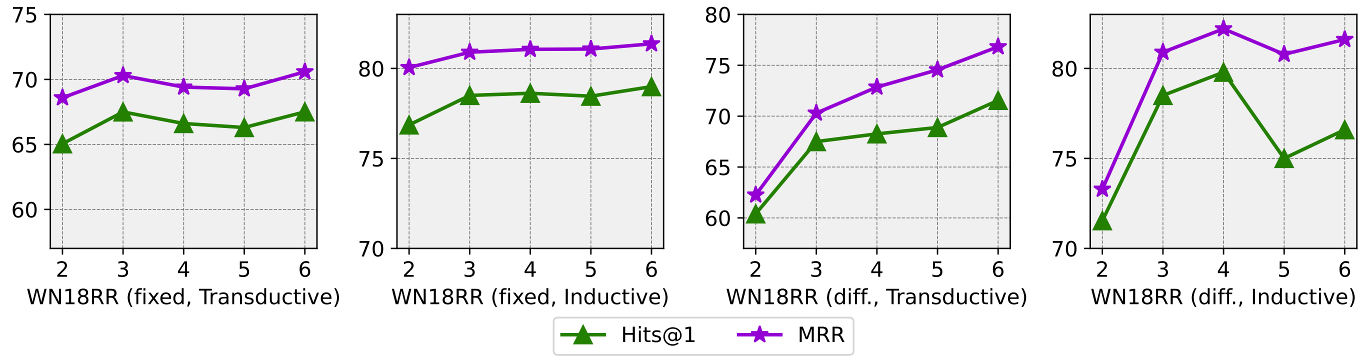


Fig. 2. **Path length scalability study.** The horizontal axis represents the maximum path length, and the vertical axis represents the metric values.

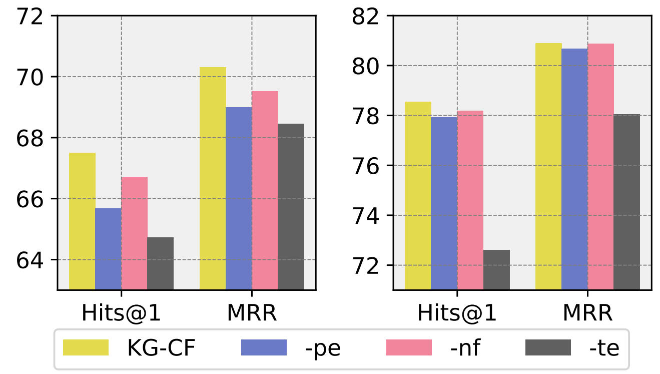


Fig. 3. Transductive (left) and inductive (right) performance comparison between KG-CF, KG-CF-pf, KG-CF-nf, and KG-CF-te. Here, -pf, -nf, -te represent positive path filtering, negative path filtering, and trajectory entities being removed, respectively.

a slight performance drop, suggesting the sequence classifier effectively filters out false negatives caused by KG incompleteness. In contrast, the performance drop is notably smaller than in -pf, indicating that irrelevant contextual information from existing KG triplets has a more substantial negative impact on performance than missing triplets.

Trajectory Entities. In this ablation, we replaced all entity names in path trajectories with anonymous labels (e.g., "entity1") but retained relation information to test the filtering of our model based solely on topology. This aims to detect data leakage, assessing whether internal knowledge in language models provides an unfair advantage. The significant performance drop confirms this issue, supporting our finding in Section 4.4 that textual descriptions strongly influence LLM filtering. Additionally, with fewer paths in this setup, performance declines due to reduced dataset size. In the inductive scenario, these findings hold in all ablation types, confirming the robustness and effectiveness of KG-CF.

E. Path Length Scalability Study (Question 3)

Intuitively, providing a knowledge graph completion model with richer contextual information can boost confidence and accuracy in training and prediction. However, the exponential cost of graph sampling at larger scales limits the practical amount of context that can be used. Here, we evaluate the impact of scalability on model performance in two settings.

Fixed Setting. We create training datasets and train models with varying maximum path lengths, but fix the maximum path length at 3 during testing for all models.

Diff. Setting. Training follows the same approach as in the fixed setting, but each model is tested with the maximum path length used during its training.

We run experiments on the WN18RR dataset, showing results for both settings in Figure 2. Key observations are as follows: (1) Extending path length generally improves model performance. (2) Increasing path length from two to three significantly boosts performance in both transductive and inductive scenarios. However, performance differences for path lengths of three or more are minimal, likely because shorter paths capture more relevant information. (3) In some cases, training on longer paths may reduce the model's ability to reason over shorter paths.

V. Related Work

A. Knowledge Graph Completion (KGC)

Existing KGC methods generally fall into three categories: (1) *Embedding-based Methods:* These approaches map entities and relations into an embedding space, with notable methods like TransE [13] and DistMult [23]. Among them, RotatE is widely recognized for its geometric approach to relational semantics. (2) *Causality-based Methods:* This category [24] focuses on identifying and utilizing causal relationships within knowledge graphs. (3) *PLM-based Methods:* Initiated by KG-BERT [22], which leverages BERT's inherent knowledge, this approach has evolved with models like BERTRL [15] that incorporate reasoning paths. Advanced techniques include prompt engineering, LoRA adapters [25], and innovations in soft prompts [26] for enhanced training and evaluation.

B. Reasoning with Large Language Models

Currently, there are primarily two strategies [27] for leveraging LLMs in reasoning tasks: (1) *Strategy Enhanced Reasoning.* These methods focus on refining the reasoning capabilities and strategies of LLMs. Since LLMs excel in interpreting and following explicit instructions [28], prompt engineering [29] has been widely used to directly improve model performance [30]. Additionally, iterative methods [31] enhance reasoning processes, and external reasoning tools (e.g., code interpreters) [32] are employed to support LLMs. (2) *Knowledge Enhanced Reasoning.* Knowledge is crucial for AI reasoning systems [33]. Some studies [34] focus on extracting information embedded within LLMs, while others incorporate external data sources [35].

VI. Limitaions

This work mainly focuses on the problem of utilizing LLM's reasoning ability on KGC. We note that we only deploy simple reasoning paths as the graph context, which is not essential for evaluation. Therefore, new context type selection (e.g. ego-graph) can be a future direction that is worthwhile to explore.

VII. Conclusion & Future Works

This paper presents KG-CF, a knowledge graph completion method that enhances pretrained language models (PLMs) through LLM-guided context filtering. We distill a sequence classifier from an LLM to assess reasoning path validity, enabling high-quality KG context selection for training the BERT scorer. Experiments show KG-CF achieves strong performance across datasets and scenarios. Our approach efficiently applies autoregressive LLMs to entity ranking. We leave the incorporation of varying graph context types to future works.

Acknowledgement

This work is supported in part by the National Science Foundation under grants (IIS-2006844, IIS-2144209, IIS-2223769, CNS-2154962, BCS-2228534, and CMMI-2411248), the Commonwealth Cyber Initiative Awards under grants (VV-1Q24-011, VV-1Q25-004), and the research gift funding from Netflix and Snap.

References

[1] J. Bobadilla, F. Ortega, A. Hernando, and A. Gutiérrez, "Recommender systems survey," *Knowledge-based systems*, vol. 46, pp. 109–132, 2013.

[2] S. Wang, Y. Zhu, H. Liu, Z. Zheng, C. Chen, and J. Li, "Knowledge editing for large language models: A survey," *ACM Comput. Surv.*, Oct. 2024, just Accepted. [Online]. Available: https://doi.org/10.1145/3698590

[3] Z. Chen, Y. Wang, B. Zhao, J. Cheng, X. Zhao, and Z. Duan, "Knowledge graph completion: A review," *Ieee Access*, vol. 8, pp. 192 435–192 456, 2020.

[4] Z. Sun, Z.-H. Deng, J.-Y. Nie, and J. Tang, "Rotate: Knowledge graph embedding by relational rotation in complex space," 2019.

[5] L. Yao, J. Peng, C. Mao, and Y. Luo, "Exploring large language models for knowledge graph completion," 2023.

[6] D. Li, S. Yang, K. Xu, M. Yi, Y. He, and H. Wang, "Multi-task pre-training language model for semantic network completion," 2022.

[7] S. Hao, Y. Gu, H. Ma, J. J. Hong, Z. Wang, D. Z. Wang, and Z. Hu, "Reasoning with language model is planning with world model," 2023.

[8] X. Wang, T. Gao, Z. Zhu, Z. Zhang, Z. Liu, J. Li, and J. Tang, "Kepler: A unified model for knowledge embedding and pre-trained language representation," 2020.

[9] A. Chepurova, A. Bulatov, Y. Kuratov, and M. Burtsev, "Better together: Enhancing generative knowledge graph completion with language models and neighborhood information," 2023.

[10] M. Jin, S. Wang, L. Ma, Z. Chu, J. Y. Zhang, X. Shi, P.-Y. Chen, Y. Liang, Y.-F. Li, S. Pan, and Q. Wen, "Time-llm: Time series forecasting by reprogramming large language models," 2024.

[11] J. Yang, "Rethinking tokenization: Crafting better tokenizers for large language models," 2024.

[12] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," 2019.

[13] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, and O. Yakhnenko, "Translating embeddings for modeling multi-relational data," *Advances in neural information processing systems*, vol. 26, 2013.

[14] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[15] H. Zha, Z. Chen, and X. Yan, "Inductive relation prediction by bert," 2021.

[16] W. Xiong, T. Hoang, and W. Y. Wang, "DeepPath: A reinforcement learning method for knowledge graph reasoning," in *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, M. Palmer, R. Hwa, and S. Riedel, Eds. Copenhagen, Denmark: Association for Computational Linguistics, Sep. 2017, pp. 564–573. [Online]. Available: https://aclanthology.org/D17-1060

[17] C. Shang, Y. Tang, J. Huang, J. Bi, X. He, and B. Zhou, "End-to-end structure-aware convolutional networks for knowledge base completion," 2018.

[18] C. Meilicke, M. Fink, Y. Wang, D. Ruffinelli, R. Gemulla, and H. Stuckenschmidt, "Fine-grained evaluation of rule-and embedding-based systems for knowledge graph completion," in *The Semantic Web–ISWC 2018: 17th International Semantic Web Conference, Monterey, CA, USA, October 8–12, 2018, Proceedings, Part I 17*. Springer, 2018, pp. 3–20.

[19] K. K. Teru, E. Denis, and W. L. Hamilton, "Inductive relation prediction by subgraph reasoning," 2020.

[20] R. Das, S. Dhuliawala, M. Zaheer, L. Vilnis, I. Durugkar, A. Krishnamurthy, A. Smola, and A. McCallum, "Go for a walk and arrive at the answer: Reasoning over paths in knowledge bases using reinforcement learning," 2018.

[21] I. Balazevic, C. Allen, and T. Hospedales, "Tucker: Tensor factorization for knowledge graph completion." Association for Computational Linguistics, 2019.

[22] L. Yao, C. Mao, and Y. Luo, "Kg-bert: Bert for knowledge graph completion," 2019.

[23] B. Yang, W. tau Yih, X. He, J. Gao, and L. Deng, "Embedding entities and relations for learning and inference in knowledge bases," 2015.

[24] R. Das, A. Godbole, N. Monath, M. Zaheer, and A. McCallum, "Probabilistic case-based reasoning for open-world knowledge graph completion," 2020. [Online]. Available: https://arxiv.org/abs/2010.03548

[25] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen, "Lora: Low-rank adaptation of large language models," 2021.

[26] G. Qin and J. Eisner, "Learning how to ask: Querying lms with mixtures of soft prompts," 2021. [Online]. Available: https://arxiv.org/abs/2104.06599

[27] S. Qiao, Y. Ou, N. Zhang, X. Chen, Y. Yao, S. Deng, C. Tan, F. Huang, and H. Chen, "Reasoning with language model prompting: A survey," 2023.

[28] X. Liu, J. Wang, J. Sun, X. Yuan, G. Dong, P. Di, W. Wang, and D. Wang, "Prompting frameworks for large language models: A survey," 2023.

[29] P. Sahoo, A. K. Singh, S. Saha, V. Jain, S. Mondal, and A. Chadha, "A systematic survey of prompt engineering in large language models: Techniques and applications," 2024.

[30] J. Wei, X. Wang, D. Schuurmans, M. Bosma, B. Ichter, F. Xia, E. Chi, Q. Le, and D. Zhou, "Chain-of-thought prompting elicits reasoning in large language models," 2023.

[31] E. Zelikman, Y. Wu, J. Mu, and N. D. Goodman, "Star: Bootstrapping reasoning with reasoning," 2022.

[32] Q. Lyu, S. Havaldar, A. Stein, L. Zhang, D. Rao, E. Wong, M. Apidianaki, and C. Callison-Burch, "Faithful chain-of-thought reasoning," 2023.

[33] S. Pan, L. Luo, Y. Wang, C. Chen, J. Wang, and X. Wu, "Unifying large language models and knowledge graphs: A roadmap," *IEEE Transactions on Knowledge and Data Engineering*, 2024.

[34] J. Liu, A. Liu, X. Lu, S. Welleck, P. West, R. L. Bras, Y. Choi, and H. Hajishirzi, "Generated knowledge prompting for commonsense reasoning," 2022.

[35] Z. Yang, J. Qin, J. Chen, L. Lin, and X. Liang, "Logicsolver: Towards interpretable math word problem solving with logical prompt-enhanced learning," 2022.