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

Anonymous ACL submission

Abstract

Recent years have witnessed the unprecedented performance of Large Language Models (LLMs) in various downstream tasks, where knowledge graph completion stands as a representative example. Nevertheless, despite the emerging explorations of utilizing LLMs for knowledge graph completion, most LLMs pose challenges in quantitative triplet score generation. This disadvantage fundamentally conflicts with the inherently ranking-based nature of the knowledge graph completion task and its associated evaluation protocols. In this paper, we propose a novel framework KG-CF for knowledge graph completion. In particular, KG-CF not only harnesses the exceptional reasoning capabilities of LLMs through context filtering but also aligns with ranking-based knowledge graph completion tasks and the associated evaluation protocols. Empirical evaluations on real-world datasets validate the superiority of KG-CF in knowledge graph completion tasks.

1 Introduction

Knowledge Graphs (KGs) have become ubiquitous in a plethora of real-world applications (Zou, 2020), such as recommendation systems (Bobadilla et al., 2013) and question answering (Yani and Krisnadhi, 2021). Specifically, KGs are a type of relational data where abundant factual information can be represented with triplets (Ji et al., 2022). Each triplet is formulated as `(h, r, t)`, indicating the existence of relation `r` between those two entities `h` and `t`, e.g. `(Earth, orbits, Sun)`. In practice, KGs are inherently sparse and incomplete, and thus, Knowledge Graph Completion (KGC) has become a widely studied task. The goal of KGC is to predict missing triplets in a KG, which helps enrich the KG with more comprehensive knowledge (Chen et al., 2020a). Traditionally, embedding-based methods, such as TransE (Bordes et al., 2013a), DistMult (Yang et al., 2015), ConvE (Dettmers et al., 2018), and RotatE (Sun et al., 2019), have been empirically proven to achieve competitive performance in KGC. Nevertheless, these approaches fail to leverage the information that goes beyond the KGs, such as certain common sense that is not in the KG, to perform prediction (Yao et al., 2023). To address this issue, researchers have explored methods for achieving better performance in KGC tasks via taking advantage of the knowledge encoded in pretrained language models (PLMs) (Li et al., 2022; Youn and Tagkopoulos, 2023; Yao et al., 2019). Among existing PLMs, Large Language Models (LLMs) naturally bear significant potential owing to their exceptional reasoning and generalization capabilities (Hao et al., 2023).

Despite the rising interest in using LLMs for KGC, it remains a daunting task. Specifically, three inherent limitations of LLM-based models pose key challenges: 1) From the task's perspective, existing LLM-based frameworks (Wang et al., 2020; Chepurova et al., 2023) predominantly extract and input graph contextual information (e.g., topology, textual description) in the form of text. However, in KGC tasks, certain contextual information is irrelevant to the given triplet. Irrelevant context may introduce substantial redundancy, thereby diverting the LLM's focus from the KGC task. 2) From the model's perspective, the sequential output LLMs are inherently inadaptable to numerical values (Jin et al., 2024). Moreover, typical LLMs generate numerical values digit by digit rather than yielding these values as a whole, where errors usually accumulate in such a sequential process (Yang, 2024). Generating a ranking list directly using LLMs also faces a similar challenge. 3) From the data's perspective, the labels corresponding to all the triplets for training are inherently discrete (e.g., existence or not), which makes it challenging to formulate proper supervision (between discrete labels and digits with a varying length) to fine-tune the LLM to yield quantitative measures for ranking.

To handle the above challenges, we propose a principled framework named KG-CF (**K**nowledge **G**raph Completion with **C**ontext **F**iltering). In this framework, LLMs are solely employed for filtering irrelevant contextual information. Specifically, for an arbitrary triplet `(h, r, t)` in a knowledge graph $\mathcal{G}$, we employ a randomly sampled set of paths in $\mathcal{G}$ from the head entity `h` to the tail entity `t` as the context set $\mathcal{C}$ to be filtered. Then, we utilize an LLM to perform filtering on $\mathcal{C}$ according to each path's relevance with `(h, r, t)`. In fact, to reduce the computational cost, we distill a smaller sequence classifier model sc from the LLM for most of the contextual information filtering in this task. This approach allows us to eliminate irrelevant contexts and successfully address challenge 1). Subsequently, a smaller PLM model BERT (Devlin et al., 2019) is trained on the remaining context set $\mathcal{C}^*$ to perform path scoring. During the testing phase, we also sample the corresponding $\mathcal{C}$ for each triplet and select the highest score from $\mathcal{C}$ as the triplet's score for ranking. By refraining from directly utilizing the LLM for the ranking tasks, challenges 2) and 3) are effectively circumvented. 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.

2 Preliminary

Notations. We use script uppercase letters to represent sets, the dataset ($\mathcal{D}$) as well as the loss function ($\mathcal{L}$). As for neural network models, we use Greek letters (e.g., θ) to represent its parameters. Moreover, in the subscripts and superscripts used in the following text, '$*$' denotes fixed (e.g., model parameters that are no longer subject to change), while '$+$' and '$-$' respectively signify positive and negative. Bolded variable names denote the embeddings of the original variables.

2.1 Problem Formulation

We denote the knowledge graph by $\mathcal{G} = \{\mathcal{E}, \mathcal{R}, \mathcal{T}\}$, while $\mathcal{R}$ corresponds to the set of relation types, $\mathcal{E}$ corresponds to the set of entities, and $\mathcal{T}$ consists of all the triplets in $\mathcal{G}$. Specifically, a triplet t in $\mathcal{T}$ is denoted as $t = (e_h, r, e_t)$, where e_h is the head entity and e_t is the tail entity. In this work, we focus on entity prediction, which encompasses two subtasks: head prediction (Glorot et al., 2013) and tail prediction (Bordes et al., 2013b). Below, we provide the definition for tail prediction, noting that head prediction is 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. Suppose that we have a knowledge graph that contains information about countries and their capitals. An exemplar 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:

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

2.2 Pretrained KG Embedding

KG embeddings represent entities and relationships in a knowledge graph in a numerical format, typically as vectors in a high-dimensional space (Chen et al., 2020b). In scenarios involving non-textual inputs, employing pretrained KG embeddings can enhance the model's expressive capability. In our framework, we default to using KG embeddings generated by TransE (Bordes et al., 2013a).

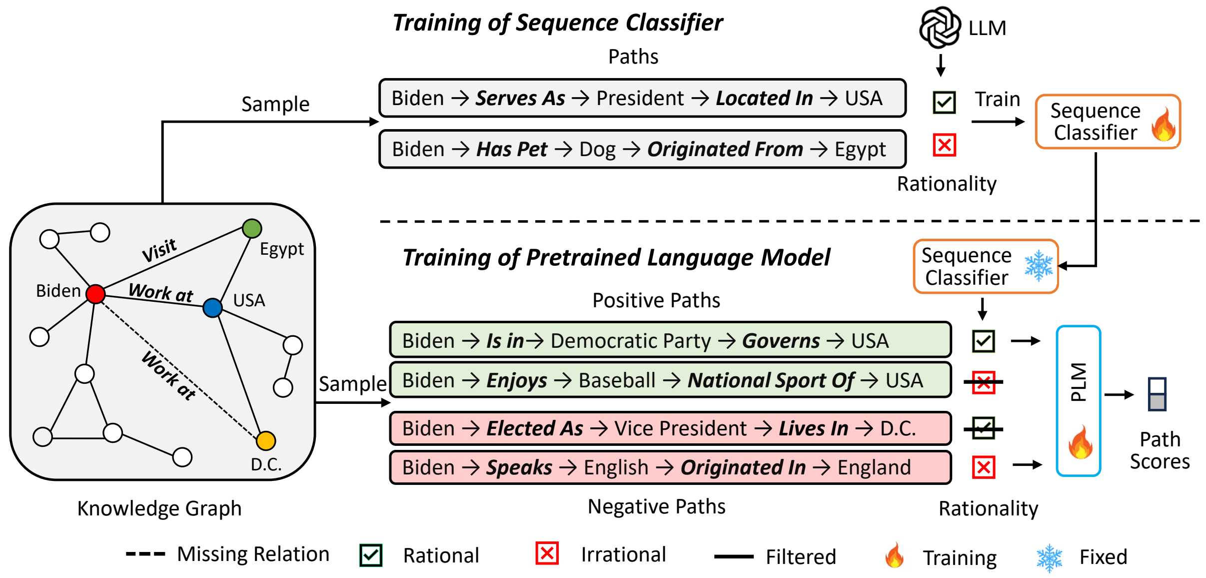

Figure 1: **The pipeline of KG-CF**. The model operates in three primary steps: 1) Sample a small set of paths and use LLMs to generate rationality labels for them. 2) Train our sequence classifier on the sampled path set. Then, filter all paths using the sequence classifier, retaining only "rational" 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.

2.3 Encoder-only Language Models

Unlike other models that may have both encoder and decoder components, an encoder-only model focuses solely on the embedding generation of the sentences (Naseem et al., 2021). Models of this kind, represented by Bert (Devlin et al., 2019), excel at classification tasks. In practice, encoder-only models accept a single text input and prepend a [CLS] token at the beginning. For processing classification tasks, we take advantage of the embedding at the [CLS] token as an aggregation of the textual information for the entire sentence content.

3 Methodology

3.1 Model Overview

In this section, we introduce the details of our proposed principled framework KG-CF, which utilizes the inference capabilities of LLMs when training sequence classifiers for triplets scoring on the task of KGC. Figure 1 shows our model pipeline. Our model can fundamentally be bifurcated into three distinct stages: path labeling, sequence classification for filtering, and PLM scoring. Given the exponential increase in path quantity with the rise in truncation length and our assertion that paths in knowledge graphs can be abstracted into more general meta-paths, we train a new sequence classifier for filtering paths to reduce the computational costs of LLM.

It is worth noting that we constrain the use of LLMs to filter a small portion of the context, thereby avoiding the substantial overhead associated with fine-tuning and inference. The paths filtered are then used as the training set for the PLM, with our ranking evaluation following thereafter being indistinguishable from conventional methods.

3.2 Path Labeling using 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 $c(q, e_t)$:

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

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 although inverse relationships are permitted in the paths, in prompt generation, all triplets in the path are represented in the standard forward order. For example, triplet $(Lakers, inv(works\ for), Lebron\ James)$ will be interpreted as *"Lebron James plays for Lakers"*, where $inv()$ represents the function of inversing.

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.

$\mathcal{D}_{sc} \leftarrow \emptyset$
for all $r \in \mathcal{R}$ **do**
    $r_{count} \leftarrow 0$
end for
for all triples $t \in T$ **do**
    $e_h, r, e_t \leftarrow t$
    if $r_{count} > n$ **then**
        continue
    end if
    $\mathcal{P} \leftarrow$ All simple paths from e_h to $e_t \in \mathcal{T} \setminus \{t\}$ with up to m
    $\mathcal{L} \leftarrow$ Label each path using LLM
    $\mathcal{D}_{sc} \leftarrow \mathcal{D}_{sc} \cup \{(\mathcal{P}[i], \mathcal{L}[i]) \mid 0 \leq i \leq |\mathcal{P}|\}$
    $r_{count} \leftarrow r_{count} + 1$
end for
return $\mathcal{D}_{sc}$

3.3 Sequence Classifier

In this section, we aim to obtain a sequence classifier $M_{sc} : \mathcal{P} \rightarrow \{0, 1\}$ that implements functionality similar to that described in Equation (5). We employ an LSTM (Hochreiter and Schmidhuber, 1997) model to implement the sequence classifier due to its expressiveness in modeling

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, Sequence Classifier sc.
Ensure: Dataset $\mathcal{D}_{PLM}$ for PLM training.

$\mathcal{D}_{PLM} \leftarrow \emptyset$
for all triples $t \in \mathcal{T}$ **do**
    $e_h, r, e_t \leftarrow t$
    $\mathcal{P}_{pos} \leftarrow$ All simple paths from e_h to $e_t \in \mathcal{T} \setminus \{t\}$ with up to m
    $\mathcal{P}_{pos} \leftarrow \{p | p \in \mathcal{P}_{pos} \wedge sc(p) > th\}$
    $\mathcal{D}_{pos} \leftarrow \{(p, true) \mid p \in \mathcal{P}_{pos}\}$
    $\mathcal{D}_{PLM} \leftarrow \mathcal{D}_{PLM} \cup \mathcal{D}_{pos}$
    for $i \leftarrow 1$ **to** neg_num **do**
        Pick an $e \in \mathcal{E} \setminus \{e_h\}\ s.t.\ (e_h, r, e) \notin \mathcal{T}$
        $\mathcal{P}_{neg} \leftarrow$ All simple paths from e_h to $e_t \in \mathcal{T}$ with up to max_hops
        $\mathcal{P}_{neg} \leftarrow \{p | p \in \mathcal{P}_{neg} \wedge sc(p) < th\}$
        $\mathcal{D}_{neg} \leftarrow \{(p, false) \mid p \in \mathcal{P}_{neg}\}$
        $\mathcal{D}_{PLM} \leftarrow \mathcal{D}_{PLM} \cup \mathcal{D}_{neg}$
    end for
end for
return $\mathcal{D}_{PLM}$

sequential information. 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:

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

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 our implementation, we did not assign a separate, unique embedding for each entity. Instead, we allowed embeddings to be shared among entities within the same category. Our intuition behind this approach is to enable the sequence classifier to learn more abstract and generalized context information.

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

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

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. In particular, we use the sequence classifier to filter and construct the dataset $\mathcal{D}_{plm}$ for PLM model training in Sec. 3.4. The detailed process is described in Algorithm 2.

3.4 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 generate the text representation and compute its score as follows:

$$P_{text} = text(c(q, e_t)) \otimes text(T), \quad (8)$$
$$score(P) = \hat{y}_P = \sigma(PLM(P_{text})), \quad (9)$$

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.

Optimization. To train the PLM model, we utilize the same loss function as Eq. (7). It is important to note that although both the sequence classifier and the PLM model process sequential input to output binary results, these two models do not serve the same task. The sequence classifier solely focuses on assessing the rationality of the reasoning process (without considering the accuracy of the reasoning outcome). Hence, it uses the judgments of LLMs as labels. On the other hand, the PLM model is utilized to determine the presence of a target triplet candidate in the KG, thereby using the ground truth as labels, which indicate whether the triplet exists.

Scoring and Ranking. To provide a basis for entity ranking, inspired by BERTRL (Zha et al., 2021), we represent the confidence score of each completion $c(q, e_t)$ using the most rational path corresponding to e_t. Specifically, we first calculate the score for each path in $\mathcal{P}_{c(q,e_t)}$ and assign the highest value to $c(q, e_t)$:

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

This score will be utilized for triplets ranking and metrics computation. A special case occurs when $\mathcal{P}_{c(q,e_t)} = \emptyset$. In this scenario, we manually assign the lowest score to the completion. Detailed settings of ranking are provided in Sec. 4.3.

4 Empirical Evaluation

In this section, we introduce the details of experiments for evaluating our KG-CF model. Particularly, we conduct experiments on two knowledge graphs in the real world. We will answer the following four questions through experiments: (1) How well can KG-CF perform in knowledge graph completion tasks? (2) How do the results of path filtering align with human intuition? (3) How do different filtering choices contribute to the overall performance of KG-CF?

4.1 Datasets

In this subsection, we provide details of the datasets used in our experiments. In particular, three widely used real-world knowledge graphs are utilized for the evaluation: Nell-995 (Xiong et al., 2017), FB15K-237 (Bordes et al., 2013c), and WN18RR (Shang et al., 2018). NELL-995 and FB15K-237 are datasets focused on relation extraction, composed of rigorously labeled instances derived from web-sourced text, emphasizing entity and relationship identification. WN18RR is a benchmark dataset for knowledge graph completion, derived from WordNet with refined relations, emphasizing the evaluation of triplet prediction methodologies. To expedite training, we separately sample a subset from each corresponding source dataset as our evaluation dataset.

4.2 Experimental Settings

Dataset Configurations. When training KG-CF, we extract positive and negative samples at a ratio of 1:5. For path searches on all three datasets, we set the truncation length of trajectories (i.e., the maximum number of triplets that a trajectory can contain) to 3. In addition to the traditional transductive scenario, we also conduct experiments on inductive scenarios. Following (Teru et al., 2020), we construct the inductive dataset where the entity sets in the training graph $\mathcal{E}_{train}$ and the testing graph $\mathcal{E}_{test}$ do not completely overlap. We provide the source code as well as the detailed dataset configurations in `https://anonymous.4open.science/r/KG-CF`.

Baselines. In the experimental part, we intend to adopt methodologies from several pre-existing works as our baselines. Among these, both the rule-based method RuleN (Meilicke et al., 2018) and the GNN-based method GRAIL (Teru et al., 2020) are applicable to both inductive and transductive scenarios. In contrast, the reinforcement learning-based MINERVA (Das et al., 2018) and the embedding-based TuckER (Balazevic et al., 2019) are unable to handle entities and relations that were unseen during training. In addition to these traditional models, we also included two methods based on pretrained encoder-only lan-

Table 1: Performances on **transductive** entity prediction of traditional methods (top) and PLM-based approaches (bottom). Metrics contain Hits@1 and MRR. Results are in percentage, and the best ones are 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 (Ours)	**67.5**	**70.3**	**62.3**	**70.9**	**73.1**	**82.0**

Table 2: Performances on **inductive** entity prediction of traditional methods (top) and PLM-based approaches (bottom). Metrics contain Hits@1 and MRR. Results are in percentage, and the best ones are shown in **Bold**.

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 (Ours)	**78.5**	**80.9**	51.2	58.3	**79.5**	**86.6**

guage models: KG-BERT (Yao et al., 2019) and BERT-RL (Zha et al., 2021).

Implementation Details. Our code is implemented through Python with Pytorch and HuggingFace libraries. The experiments were conducted on a server equipped with six A6000 GPUs. We utilized GPT-3.5 as the LLM and employed an LSTM (Hochreiter and Schmidhuber, 1997) model to implement a sequence classifier. For the sequence classifier, we train it over ten epochs with a learning rate of 1e-3. The PLM scorer is trained for two epochs with a learning rate of 1e-5. The threshold th in Algorithm 2 is set to be 0.1.

4.3 Evaluation Method

In both transductive and inductive scenarios, we separately evaluate our approach on two subtasks: tail prediction and head prediction. We then compute the average performance of two scenarios.

General Protocol. Following GRAIL (Teru et al., 2020) and BERTRL (Zha et al., 2021), we 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 will sort entities in T according to their scores and compute metrics by ranking t_{test}.

KG-CF. Noting that if there are no paths associated with tail entity t during the evaluation of KG-CF, we will set $score(t)$ to be 0 (the lower bound). Furthermore, if t happens to be the t_{test}, we will set the $rank(t_{test})$ to be the median rank of all tail candidates with a score of 0. Clearly, in this scenario, $rank(t_{test}) \geq 25$. Therefore, it will not affect the metrics of Hits@1 or Hits@10 and is also fair to other tail candidates.

4.4 Main Results (Question 1)

In this subsection, our KG-CF framework is evaluated on three knowledge graphs in both transductive (Table 1) and inductive scenarios (Table 2). We obtain the following observations through experiments: 1) Our KG-CF framework outperforms other baselines across most datasets and scenarios, revealing the effectiveness and versatility of using LLM and sequence classifiers for filtering graph contextual information. 2) Compared to the inductive scenario, KG-CF exhibits more consistent performance in the transductive scenario. 3) Compared to other baselines and datasets, our method demonstrates more substantial improvements on the NELL-995 dataset. While both NELL-995 and FB15K-237 are comprehensive knowledge graphs concerning real-world scenarios, NELL-995 offers richer textual information about entities (e.g., `person mexico Ryan Whitney` instead of merely `Ryan Whitney`). Intuitively, this feature enhances the LLM in handling rare nouns, leading to more accurate judgments and generations.

4.5 Case Study (Question 2)

Data Diversity. A meta-path (Jiao et al., 2022) is composed of a series of node types and edge types, culminating in a structured path pattern. we group the entities in the WN18RR dataset following (Lin et al., 2018). In the process of data filtering, while we eliminated a large number of anomalous positive samples, we simultaneously encountered the issue of insufficient coverage of meta-paths. To check this problem, we have also tallied the number of unique positive meta-paths traversed by the agent during the training process regarding the threshold value. This will measure the breadth of the KG context exploration. the results are shown in Table 3. We observe a dramatic decrease in the number of meta-paths at a threshold of 0.1, with minimal decline thereafter. This may be related to the sigmoid function's characteristics. Overall, we filtered 80% paths and meta-paths, respectively.

Table 3: The numbers of meta-paths and paths regarding different values of the threshold.

Threshold	#Mata-Paths	#Paths
0.0	**66,024**	**257,565**
0.1	13,059	52,119
0.2	12,488	50,034
0.3	12,124	48,177
0.4	11,839	47,283

Reasoning Path Explanation. In this section, we will intuitively assess the quality of reasoning paths at various stages within the dataset: immediately after sampling from the KG, following LLM filtering, and after sequence classifier selection. We select a triplet `(person Roger Mudd, person leads organization, television network CBS)` in the NELL-995 dataset as an example. In the initial sampling of paths, there exists a trajectory composed of a single triple, whose textual representation is: "person roger mudd person belongs to organization television network CBS. " This was labeled as true in the original dataset, but it is clearly not a valid reasoning path, as belonging to an organization does not necessarily mean leading an organization. This issue was corrected by the LLM, which reassigned it with a false label. Subsequently, the sequence classifier also accurately filtered out this path when preprocessing the PLM trainset.

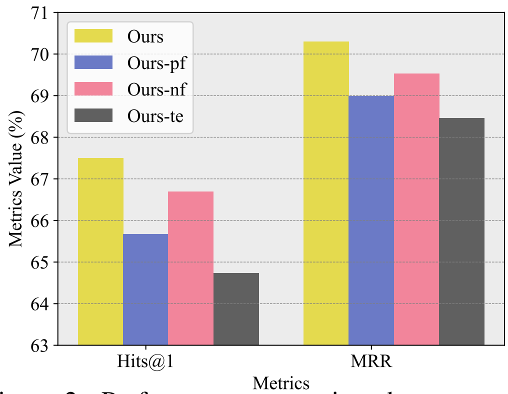

Figure 2: Performance comparison between our approach, Ours-pf, Ours-nf, and Ours-te. Here, -pf, -nf, -te represent positive path filtering, negative path filtering, and trajectory entities being removed, respectively.

4.6 Ablation Study (Question 3)

We conducted an ablation study on the WN18RR dataset 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 experimental results in Figure 2.

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 of this ablation study showed a slight decline compared to the original model, indicating that our sequence classifier can enhance the rationality of positive paths.

Negative Path Filtering. Within this setting, we posit that for a negative triplet $(e_h, r_q, e_t) \notin \mathcal{T}$, all paths from e_h to e_t fail to validate the existence of r_q (owing to the incompleteness of KGs, we claim this assumption to be false). This type of ablation also led to a slight decrease in results, suggesting that the sequence classifier can effectively eliminate false negatives caused by KG incompleteness. Conversely, the performance degradation is considerably smaller compared to -pf. This observation indicates that the irrelevant contextual information from existing triplets in the KG is more extensive and exerts a more pronounced negative effect on performance than that from the missing triplets.

Trajectory Entities. In this ablation, we replaced all entity names in the path trajectories with anonymous names (e.g., "entity1"). However, we retained the necessary relation information within each path to evaluate our model's filtering capability based solely on topological information. The objective is to investigate and avoid the presence of data leakage within both the LLM and pretrained language model, namely whether the knowledge stored internally in the language models confers an unfair advantage to performance. The significant performance drop indicates that the issue indeed exists. This also corroborates our assessment in Section 4.4 that textual description substantially impacts the performance of LLM filtering. Meanwhile, since we still filter out a certain number of paths in this setting, the model endures performance degradation due to the reduced dataset size.

5 Related Work

5.1 Knowledge Graph Completion (KGC)

In general, existing KGC methods can be approximately divided into two groups: 1) *embedding-based Methods.* These methods model entities and relations within a knowledge graph by mapping them into an embedding space. Approaches such as TransE (Bordes et al., 2013c), DistMult (Yang et al., 2015), ComplEx (Trouillon et al., 2016), ConvE (Dettmers et al., 2018), and RotatE (Sun et al., 2019) vary in handling relational semantics, with RotatE currently recognized for its adept handling of various relation types through geometric intuition, positioning it as a leading method in the field. 2) *PLM-based method:* KG-BERT (Yao et al., 2019) represents an early model in this category, leveraging the inherent knowledge within BERT. Subsequent advancements, such as BERTRL (Zha et al., 2021), have sought to improve upon this by incorporating reasoning paths between entities. Recent developments also include prompt engineering and the use of LoRA adapters (Hu et al., 2021) for fine-tuning, alongside innovations like soft prompts and the Open World Assumption (OWA) to enhance training and evaluation. Beyond the aforementioned methods, soft prompts (Chen et al., 2023) and OWA (Open World Assumption) (Lv et al., 2022) are also introduced for model training and evaluation. Different from them, we have not only leveraged the outstanding reasoning capabilities of LLMs to examine the context of graphs from a new perspective, but we have also trained a sequence classifier to learn this particular ability of LLMs.

5.2 Reasoning with Large Language Models

Currently, there are primarily two strategies (Qiao et al., 2023) employed to leverage LLMs for accomplishing reasoning tasks: 1) *Strategy Enhanced Reasoning.* These approaches place a greater emphasis on enhancing the reasoning standards and strategies of LLMs. Specifically, given that LLMs excel at comprehending and adhering to manually provided instructions (Liu et al., 2023), existing works (Wei et al., 2023) attempt to boost model performance directly through prompt engineering (Sahoo et al., 2024). Another line of work (Zelikman et al., 2022; Huang et al., 2022) focuses on optimizing the reasoning process through iterative methods. External reasoning engines (e.g., physical simulators, code interpreters) (Madaan et al., 2022; Lyu et al., 2023) have also been introduced to assist LLMs in reasoning. 2) *Knowledge Enhanced Reasoning.* In general, knowledge plays a vital role in AI reasoning systems (Pan et al., 2024). Some efforts (Liu et al., 2022; Fu et al., 2023) aim to mine information stored internally within LLMs, while other researchers endeavor to incorporate external data sources (Yang et al., 2022), including Knowledge Graphs, wiki documents, and more. In this work, we mainly focus on the first approach since we do not furnish LLMs with additional knowledge information during each session.

6 Conclusion & Future Works

In this paper, we propose KG-CF, a pretrained language model (PLM)-based knowledge graph completion method enhanced by the LLM-guided context filtering. Specifically, we distilled a sequence classifier from an LLM to assess the rationality of reasoning paths, thereby curating high-quality KG contexts for training of the Bert scorer. Experiments results indicate that KG-CF demonstrates exceptional performance across the majority of datasets and scenarios. Furthermore, our approach judiciously leverages generative LLMs for a reasonable scope applicable to the entity ranking protocol. However, the current evaluation metrics (Hits@n) remain flawed: missing triplets in KGs might be included as negative samples. This issue is more pronounced for PLM-based methods. Employing LLMs to refine the evaluation protocol represents a valuable research direction.

7 Limitaions

This work mainly focuses on the problem of utilizing LLM's reasoning ability on knowledge graph completion tasks. Specifically, we exclude irrational reasoning paths by querying the LLM. 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.

8 Ethics Statement

This paper proposes a novel LLM-based framework to perform KGC tasks, which aligns with the inherent ranking-based nature of this task and the corresponding evaluation protocols. We do not foresee any ethical issue that needs to be specifically highlighted here.

References

Ivana Balazevic, Carl Allen, and Timothy Hospedales. 2019. Tucker: Tensor factorization for knowledge graph completion. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Association for Computational Linguistics.

Jesús Bobadilla, Fernando Ortega, Antonio Hernando, and Abraham Gutiérrez. 2013. Recommender systems survey. *Knowledge-based systems*, 46:109–132.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Durán, Jason Weston, and Oksana Yakhnenko. 2013a. Translating embeddings for modeling multi-relational data. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'13, page 2787–2795, Red Hook, NY, USA. Curran Associates Inc.

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

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013c. Translating embeddings for modeling multi-relational data. In *Advances in Neural Information Processing Systems*, volume 26. Curran Associates, Inc.

Chen Chen, Yufei Wang, Aixin Sun, Bing Li, and Kwok-Yan Lam. 2023. Dipping PLMs sauce: Bridging structure and text for effective knowledge graph completion via conditional soft prompting. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 11489–11503, Toronto, Canada. Association for Computational Linguistics.

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

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

Alla Chepurova, Aydar Bulatov, Yuri Kuratov, and Mikhail Burtsev. 2023. Better together: Enhancing generative knowledge graph completion with language models and neighborhood information.

Rajarshi Das, Shehzaad Dhuliawala, Manzil Zaheer, Luke Vilnis, Ishan Durugkar, Akshay Krishnamurthy, Alex Smola, and Andrew McCallum. 2018. Go for a walk and arrive at the answer: Reasoning over paths in knowledge bases using reinforcement learning.

Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. 2018. Convolutional 2d knowledge graph embeddings.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding.

Yao Fu, Hao Peng, Litu Ou, Ashish Sabharwal, and Tushar Khot. 2023. Specializing smaller language models towards multi-step reasoning.

Xavier Glorot, Antoine Bordes, Jason Weston, and Yoshua Bengio. 2013. A semantic matching energy function for learning with multi-relational data.

Shibo Hao, Yi Gu, Haodi Ma, Joshua Jiahua Hong, Zhen Wang, Daisy Zhe Wang, and Zhiting Hu. 2023. Reasoning with language model is planning with world model.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models.

Jiaxin Huang, Shixiang Shane Gu, Le Hou, Yuexin Wu, Xuezhi Wang, Hongkun Yu, and Jiawei Han. 2022. Large language models can self-improve.

Shaoxiong Ji, Shirui Pan, Erik Cambria, Pekka Marttinen, and Philip S. Yu. 2022. A survey on knowledge graphs: Representation, acquisition, and applications. *IEEE Transactions on Neural Networks and Learning Systems*, 33(2):494–514.

Fangkai Jiao, Yangyang Guo, Xuemeng Song, and Liqiang Nie. 2022. Merit: Meta-path guided contrastive learning for logical reasoning.

Ming Jin, Shiyu Wang, Lintao Ma, Zhixuan Chu, James Y. Zhang, Xiaoming Shi, Pin-Yu Chen, Yuxuan Liang, Yuan-Fang Li, Shirui Pan, and Qingsong Wen. 2024. Time-llm: Time series forecasting by reprogramming large language models.

Da Li, Sen Yang, Kele Xu, Ming Yi, Yukai He, and Huaimin Wang. 2022. Multi-task pre-training language model for semantic network completion.

Xi Victoria Lin, Richard Socher, and Caiming Xiong. 2018. Multi-hop knowledge graph reasoning with reward shaping.

Jiacheng Liu, Alisa Liu, Ximing Lu, Sean Welleck, Peter West, Ronan Le Bras, Yejin Choi, and Hannaneh Hajishirzi. 2022. Generated knowledge prompting for commonsense reasoning.

Xiaoxia Liu, Jingyi Wang, Jun Sun, Xiaohan Yuan, Guoliang Dong, Peng Di, Wenhai Wang, and Dongxia Wang. 2023. Prompting frameworks for large language models: A survey.

Xin Lv, Yankai Lin, Yixin Cao, Lei Hou, Juanzi Li, Zhiyuan Liu, Peng Li, and Jie Zhou. 2022. Do pre-trained models benefit knowledge graph completion? a reliable evaluation and a reasonable approach. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 3570–3581, Dublin, Ireland. Association for Computational Linguistics.

Qing Lyu, Shreya Havaldar, Adam Stein, Li Zhang, Delip Rao, Eric Wong, Marianna Apidianaki, and Chris Callison-Burch. 2023. Faithful chain-of-thought reasoning.

Aman Madaan, Shuyan Zhou, Uri Alon, Yiming Yang, and Graham Neubig. 2022. Language models of code are few-shot commonsense learners.

Christian Meilicke, Manuel Fink, Yanjie Wang, Daniel Ruffinelli, Rainer Gemulla, and Heiner Stuckenschmidt. 2018. 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*, pages 3–20. Springer.

Usman Naseem, Imran Razzak, Shah Khalid Khan, and Mukesh Prasad. 2021. A comprehensive survey on word representation models: From classical to state-of-the-art word representation language models. *ACM Trans. Asian Low-Resour. Lang. Inf. Process.*, 20(5).

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

Shuofei Qiao, Yixin Ou, Ningyu Zhang, Xiang Chen, Yunzhi Yao, Shumin Deng, Chuanqi Tan, Fei Huang, and Huajun Chen. 2023. Reasoning with language model prompting: A survey.

Pranab Sahoo, Ayush Kumar Singh, Sriparna Saha, Vinija Jain, Samrat Mondal, and Aman Chadha. 2024. A systematic survey of prompt engineering in large language models: Techniques and applications.

Chao Shang, Yun Tang, Jing Huang, Jinbo Bi, Xiaodong He, and Bowen Zhou. 2018. End-to-end structure-aware convolutional networks for knowledge base completion.

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

Komal K. Teru, Etienne Denis, and William L. Hamilton. 2020. Inductive relation prediction by subgraph reasoning.

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.

Xiaozhi Wang, Tianyu Gao, Zhaocheng Zhu, Zhengyan Zhang, Zhiyuan Liu, Juanzi Li, and Jian Tang. 2020. Kepler: A unified model for knowledge embedding and pre-trained language representation.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. 2023. Chain-of-thought prompting elicits reasoning in large language models.

Wenhan Xiong, Thien Hoang, and William Yang Wang. 2017. Deeppath: A reinforcement learning method for knowledge graph reasoning. *arXiv preprint arXiv:1707.06690*.

Bishan Yang, Wen tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. 2015. Embedding entities and relations for learning and inference in knowledge bases.

Jinbiao Yang. 2024. Rethinking tokenization: Crafting better tokenizers for large language models.

Zhicheng Yang, Jinghui Qin, Jiaqi Chen, Liang Lin, and Xiaodan Liang. 2022. Logicsolver: Towards interpretable math word problem solving with logical prompt-enhanced learning.

Mohammad Yani and Adila Alfa Krisnadhi. 2021. Challenges, techniques, and trends of simple knowledge graph question answering: a survey. *Information*, 12(7):271.

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

Liang Yao, Jiazhen Peng, Chengsheng Mao, and Yuan Luo. 2023. Exploring large language models for knowledge graph completion.

Jason Youn and Ilias Tagkopoulos. 2023. Kglm: Integrating knowledge graph structure in language models for link prediction.

Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah D. Goodman. 2022. Star: Bootstrapping reasoning with reasoning.

Hanwen Zha, Zhiyu Chen, and Xifeng Yan. 2021. Inductive relation prediction by bert.

Xiaohan Zou. 2020. A survey on application of knowledge graph. In *Journal of Physics: Conference Series*, volume 1487, page 012016. IOP Publishing.