Path-enhanced Pre-trained Language Model for Knowledge Graph Completion

Anonymous ACL submission

Abstract

Pre-trained language models (PLMs) have achieved remarkable knowledge graph completion(KGC) success. However, previous methods derive KGC results mainly from triple-level and text-described learning, which lack the capability of capturing long-term relational and structural information. Moreover, the absence of a visible reasoning process leads to poor interpretability and credibility of the completions. In this paper, we propose a path-enhanced pre-trained language model-based knowledge graph completion method (PEKGC), which employs multi-view generation to infer missing facts in triple-level and path-level simultaneously to address lacking long-term relational information and interpretability issues. Furthermore, a neighbor selector module is proposed to filter neighbor triples to provide the adjacent structural information. Finally, we propose a fact-level re-evaluation and a heuristic fusion ranking strategy for candidate answers to fuse multi-view predictions. Extensive experiments on the benchmark datasets demonstrate that our model significantly improves the performance of the KGC task.

1 Introduction

Knowledge graphs (KGs) are designed to store knowledge in graph-structured format, as seen in Freebase (Bollacker et al., 2008), WordNet (Miller, 1995), and NELL (Carlson et al., 2010). The extensive application of various KGs has greatly benefited numerous downstream tasks, such as question answering, recommendation systems, and information retrieval. However, due to the limited scale of KGs, whether manually or automatically constructed, they invariably suffer from incomplete coverage and fail to encompass the vast expanse of real-world knowledge. This limitation has given rise to the task of knowledge graph completion (KGC), which involves predicting missing links by understanding the existing triples within KGs.

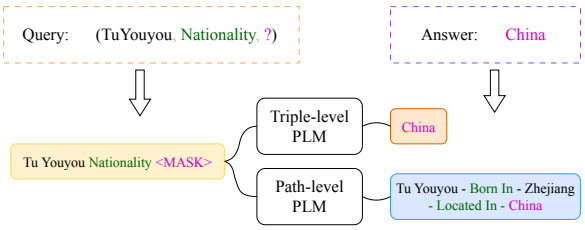


Figure 1: An example of PLM-based KGC in triple and path levels. In this triple, (Tu Youyou, Nationality, ?) is a query, and the answer is *China*.

Typically, most mainstream KGC methods fall into two categories: embedding-based and path-based. Embedding-based methods (Bordes et al., 2013; Yang et al., 2015; Dettmers et al., 2018; Vashishth et al., 2019) focus on mapping entities and relations into a low-dimensional, continuous vector space to capture intrinsic connections and predict missing links in the vector space. By comparison, path-based methods (Das et al., 2018; Qu et al., 2020; Zhu et al., 2021) aim to use paths between entities to predict and achieve more directly interpretable results. Whereas, both approaches aim to model all interactions among entities and relations directly, they hardly adapt to large-scale growth parameters and knowledge scenarios.

Recently, inspired by the success of pre-training language models (PLMs), such as BERT (Devlin et al., 2019), RoBERTa (Liu et al., 2019), and T5 (Raffel et al., 2020), in NLP tasks, encouraging increased interest in probing PLMs to complete KGs. According to the structure of the models, PLM-based models can be divided into two categories: encoder-only models, such as KG-BERT (Yao et al., 2019) and PKGC (Lv et al., 2022), and encoder-decoder models, such as KGT5 (Saxena et al., 2022) and KGS2S (Chen et al., 2022). The encoder-only models encode both the query and all candidate entities to calculate their matching confidence, while the encoder-decoder models encode the query and then decode possible

candidate entities. Overall, PLM-based KGC methods work by converting triples into serialized text descriptions sentences and feeding the sentences to PLMs to complete KGs. As illustrated in Figure 1, current PLM-based models (referred to as triple-level PLM) encode the concatenated query text, "*Tu Youyou Nationality [MASK]*", from a "flat" text-described view and directly generate the answer entity, *China*.

However, current PLM-based models have flattened triples into a simple triple-level text description presenting three significant limitations: (i) **low expressiveness**. The embeddings of entities and relations are learned at the triple level, which represents the embeddings as being focused only on a local perspective (i.e., one-hop structure). Previous research has indicated that relying solely on local relational information for KG learning is not enough (Guo et al., 2019). (ii) **inefficient information propagation**. These models depend exclusively on one-hop neighbors for aggregating and propagating information, which is inefficient for transferring semantics and knowledge between entities. (iii) **lack of interpretability**. KGC results are generated directly in response to queries without the necessary explanations or reasoning process, which harms the credibility of completions. Therefore, there is an urgent need to leverage rich structural information of KGs to capture more complex, long-term, or higher-level features to enhance expressiveness, propagation, and interpretability.

To address the limitations above, in this paper, we propose a Path-enhanced Pre-trained Language Model-based Knowledge Graph Completion model (PEKGC), which employs multi-view generation by an encoder-decoder architecture to tackle lacking long-term relational structures and poor interpretability issues. To enable PLM-based models to capture path-level knowledge and bridge long-term gaps between entities, our model generates possible entities with reasoning path chains, which also serve as evidence for the generated answers in response to queries. Besides, to extract local knowledge, our model simultaneously generates answers at a triple level. In this way, the model can focus on multi-level knowledge simultaneously and can be cross-validated occurring at both levels. To better use the adjacent structural information and enhance the inference ability of the model, we also design a neighbor selector module, which is pre-trained to filter the most relevant triples to the query as neighborhood information. In addition, we use KG soft prompts and position soft prompts to distinguish between KG knowledge and textual knowledge, as well as to mark the internal relationships of triples. To further re-evaluate and re-rank the joint results of both levels, we propose a fact-level re-evaluation and a heuristic fusion ranking strategy, which can re-calculate the confidence scores of candidate answers and efficiently re-rank them. In summary, the major contributions of this work are as follows:

- We propose a multi-level generation paradigm for knowledge graph completion that can capture local triple-level knowledge and long-term relational structures, and also model the text and structural information, achieving more directly interpretable results.
- We propose a heuristic fusion ranking strategy for multi-view generation during inference, further combining triple-level and path-level results to achieve better ranking performance.
- Extensive experiments are conducted on benchmark datasets, demonstrating that the proposed method outperforms existing state-of-the-art methods.

2 Related Work

2.1 Knowledge Graph Completion

Traditional KGC methods aim to map entities and relations into a low-dimensional and continuous vector space to capture inner connections. These methods can be further subdivided into translation-based methods (Bordes et al., 2013; Sun et al., 2018), semantic matching methods (Yang et al., 2015; Balažević et al., 2019), convolutional neural network-based (CNN-based) methods (Dettmers et al., 2018; Ren et al., 2020), and graph neural network-based (GNN-based) methods (Schlichtkrull et al., 2018; Vashishth et al., 2019). However, as the scale of KGs has increased, the extraction and expression ability of these methods has gradually encountered bottlenecks.

2.2 Path-based KGC

To bridge the long-term relational gaps between entities and improve the interpretability of completion results, some methods introduce the multi-hop paths into the KGC task, i.e., path-based methods. These methods can be further subdivided into rule-based methods (Qu et al., 2020; Sadeghian et al., 2019), reinforcement learning-based (RL-based)

methods (Das et al., 2018; Lin et al., 2018; Jiang et al., 2023), and propagation path-based methods (Zhu et al., 2021; Zhang et al., 2023). Rule-based methods utilize logical inference and symbolic rules to complete KGs. They usually suffer from poor generalization and high complexity due to their direct operation on symbols when applied to large-scale KGs. RL-based methods frame multi-hop path reasoning as a finite horizon deterministic partially observed Markov decision process (MDP) and train an agent to navigate on KGs to locate target entities. However, these methods are also limited in the vast search space and sparse rewards encountered during training. Propagation path-based methods use relational paths to encode and transmit the intermediate knowledge between entities. Limited by the complexity of propagation, their dimensions are generally small, resulting in weak expression ability.

2.3 Pre-trained Language Model-based KGC

Recent research has focused on fine-tuning PLMs for KGC tasks to leverage the implicit knowledge of PLMs and the structured knowledge of KGs. KG-BERT (Yao et al., 2019) is the first to use BERT for KGC by simply concatenating triples' names as text-based input. Subsequent methods can be categorized into two main types based on their model structures: encoder-only models and encoder-decoder models. Encoder-only models such as StAR (Wang et al., 2021), which integrates graph embedding techniques to introduce structured knowledge, CoLE (Liu et al., 2022), which distills selective knowledge between graph embedding and PLMs, and PKGC (Lv et al., 2022), which employs soft prompts to convert triples into natural prompt sentences Encoder-decoder models such as KGT5 (Saxena et al., 2022), which uses a Seq2Seq generative framework to encode query and decode candidate entities, GenKGC (Xie et al., 2022), which introduces entity-aware hierarchical decoding for fast inference, and KG-S2S (Chen et al., 2022), which unifies triples into "flat" text and advance KG soft prompts.

3 Methodology

3.1 Notions

We formally represent a knowledge graph (KG) as $\mathcal{G} = (\mathcal{E}, \mathcal{R}, \mathcal{T})$, where $\mathcal{E}$ is the set of entities, $\mathcal{R}$ is the set of relations, and $\mathcal{T}$ is the set of triples in the KG. Each triple can be expressed as $l = (e_h, r, e_t) \in \mathcal{T}$. For a triple l, there are a set of relational paths $p = \{(e_h, r_1, e_1, ..., r_t, e_t) | (e_h, r_1, e_1) \in \mathcal{T}, ..., (e_{t-1}, r_t, e_t) \in \mathcal{T}\}$ connecting the head entity e_h and the tail entity e_t. Following (Guo et al., 2019), we use Biased Random Walks to generate these paths. Given a query *(Tu Youyou, nationality, ?)*, we can formalize it as $(e_h, r, ?)$, where e_h is the head entity of the query, and r is the relation between the head entity and the tail entity. The task of link prediction in KGC is to infer the tail entity e_t and, similarly, predict the head entity e_h for a query $(?, r, e_t)$. Following previous work, for each triple l, we add an inverse triple (e_t, r^{-1}, e_h) into KG, where r^{-1} is the inverse relation of r.

3.2 A Seq2Seq Architecture

As illustrated in Figure 2, the proposed PEKGC model follows a sequence-to-sequence (Seq2Seq) architecture comprising an encoder and a decoder. To train the model to capture knowledge from multi-views, we design two sub-tasks to train the model's capability at different levels (i.e., triple and path levels). Overall, it can be represented as:

$$P(y|x_q) = \prod_{k=1}^{N} P(y_k|x_q, y_{<k}), \quad (1)$$

where x_q is the input sequence, consisting of the concatenated query. For the triple-level generation, y denotes the concatenated target entities' names and descriptions. For the path-level generation, y denotes the concatenated multi-hop reasoning relational paths, target entities' names, and entities' descriptions.

3.3 Two-level Generation

Triple-level Generation To avoid ambiguity issues of entity names (Chen et al., 2022), we use entity descriptions to enrich the context information of entities. For a query $(e_h, r, ?)$, we concatenate the head entity's name x_h, the head entity's description d_h, and the relation's name x_r to form its representation on the encoding side, that is:

$$x_q = (x_h, d_h, [\text{SEP}], x_r, [\text{MASK}]), \quad (2)$$

where [SEP] denotes a special separator token, and [MASK] denotes the "?" at the corresponding position to distinguish the queries between $(e_h, r, ?)$ and $(?, r, e_t)$.

On the decoding side, the triple-level task aims to predict the target entity's name and description

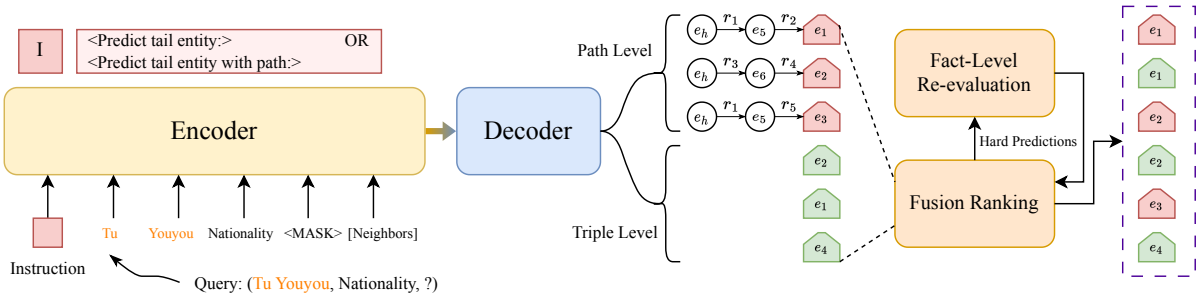


Figure 2: Overview of proposed PEKGC model, where e_* and r_* represent entities and relations, respectively.

jointly,

$$y_t = (x_t, d_t). \tag{3}$$

Similar to tail entity predictions, the representations of query and answer for predicting head entities, $(?, r, e_t)$, are as follows:

$$x_q = ([\text{MASK}], x_r, [\text{SEP}], x_t, d_t), \tag{4}$$

$$y_t = (x_h, d_h). \tag{5}$$

Path-level Generation The input of path-level generation, x_q, is the same as triple-level generation. The goal of its generation is to predict multi-hop reasoning relational paths from source to target entities, target entity's name, and target entity's description simultaneously,

$$y_p = (p_t, x_t, d_t), \tag{6}$$

$$y_p = (p_h, x_h, d_h), \tag{7}$$

where p_t (p_h) is a concatenated sequence of the relational reasoning paths $(e_h, r_1, e_1, ..., r_t, e_t)$ from the source entity e_h to the target entity e_t. This sequence serves as a reasoning chain and evidence for the answer,

$$p_t = (x_h, x_{r_1}, x_{e_1}, ..., x_{r_t}, x_t), \tag{8}$$

where x_{r_*} and x_{e_*} represent the name texts of relations and entities in the path, respectively.

KG Soft Prompt We insert KG soft prompts into the encoder's input to differentiate between KG and text knowledge and emphasize structural knowledge (Chen et al., 2022). Specifically, we define a set of additional trainable prompt embeddings, which are associated with relations of KG. It is worth noting that the number of relations in the KG is significantly smaller than that of entities, thus minimizing the risk of excessive overhead. These embeddings are denoted as $\mathbf{P_r} \in \mathbb{R}^{|\mathcal{R}| \times d}$, where d is the dimension of the encoder. Each relation (r) in the input sequence has a corresponding KG soft prompt ($\mathbf{p_r}$) that is inserted in front of it.

Position Soft Prompt We define position soft prompts, $\mathbf{P_o} \in \mathbb{R}^{3 \times d}$, to indicate and prompt the positional relationships among the head entity, relation, and tail entity within a triple. This allows the model to learn the internal positional relationship of the triple and effectively distinguish between the predicted head and tail entities.

Instructions Moreover, to further distinguish two-level generation tasks, we insert two specific instruction tags, I_t and I_p, at the beginning of the input sequence: "*Predict tail / head entity:*" for triple level and "*Predict tail / head entity with path:*" for path level. Consequently, the input embedding with soft prompts and instructions is updated to:

$$x_q = (I, p_0, x_h, d_h, x_s, p_1, p_r, x_r, p_2, x_m), \tag{9}$$

where I denotes the instruction tag, x_h denotes the head entity's name, d_h denotes the head entity's description, x_r denotes the relation's name, p_r represents the KG soft prompt related to r, p_0, p_1 and p_2 are position soft prompts, and x_s and x_m are special tokens, [SEP] and [MASK], respectively.

Training Both triple-level and path-level generation goals are to predict the answer sequence y_t or y_p. Therefore, the optimization objective is given by:

$$L_g = -\log P(y|x_q). \tag{10}$$

3.4 Neighbor Selector

To better use the adjacent structural information around the query, we design an additional neighbor selector module, which is pre-trained to filter the most relevant triples to the query as neighborhood information. For a query, $(e_h, r, ?)$, there are a k-th order neighborhood subgraph,

$$\begin{aligned} \mathcal{N}_h = &\{(e_h, r_1^1, e_1^1), ...(e_h, r_m^1, e_n^1), ... \\ &, (e_1^1, r_2^2, e_2^2), ..., (e_{n-1}^{k-1}, r_m^k, e_n^k)\} \\ &, 0 \le m \le |\mathcal{R}|, 0 \le n \le |\mathcal{E}|. \end{aligned} \tag{11}$$

Subsequently, we can filter query-relevant neighbors from $\mathcal{N}_h$ by a pre-trained encoder-decoder selector. The input of encoder is:

$$x = (x_h, x_r, [\text{MASK}], [\text{SEP}], x_a, x_b, x_c), \quad (12)$$

where x_h and x_r are entity and relation names of the query repectively, and (x_a, x_b, x_c) is the name of $(e_a, r_b, e_c) \in \mathcal{N}_h$, which is a neighbor triple related to the source entity e_h. The prediction goal of the selector is $y \in \{yes, no\}$, where yes indicates the target entity present in the neighbor triple, considered a positive sample, while conversely, it constitutes a negative sample. Therefore, considering reducing the sensitivity of the model in the learning process to prevent over-fitting, the pre-training loss of this module is:

$$\begin{aligned} L_n = &- \log P(y|x) \quad (13) \\ &+ \log(1 + \sum_{i \in \Omega_{neg}, j \in \Omega_{pos}} e^{\mu(s_i - s_j)}), \end{aligned}$$

where μ is a margin value, Ω_{neg} denotes the decode scores set of negative samples, and Ω_{pos} denotes the decode scores set of positive samples.

This selector allows us to filter the most relevant triples, $\mathcal{N}'_h \subset \mathcal{N}_h$. Consequently, concatenating $\mathcal{N}'_h$ to x_q of Equation 9 can be updated as:

$$x'_q = (x_q, x_n), \quad (14)$$

where x'_q is the final input to equation 1, and x_n represents the textualized representation of the filtered neighbor triples $\mathcal{N}'_h$.

3.5 Re-evaluation and Fusion Ranking

Fact-level Re-evaluation To re-evaluate and re-order the two group results generated by triple and path levels, we also pre-train the encoder of model at fact level. For instance, given a candidate triple $\hat{l} = (e_h, r, \hat{e}_t)$ generated by triple or path level, its confidence score is calculated as:

$$\phi(\mathbf{e}_{hr}, \hat{\mathbf{e}}_t) = \cos(\mathbf{e}_{hr}, \hat{\mathbf{e}}_t) = \mathbf{e}_{hr} \cdot \hat{\mathbf{e}}_t, \quad (15)$$

where $\mathbf{e}_{hr}$ is the max pooling of the combination encoding of e_h and r, $\hat{\mathbf{e}}_t$ is the encoding of the predicted candidate entity $\hat{e}_t$, and $\cdot$ denotes a dot product operation.

As a result, the training objective of the fact-level encoder is to maximize the confidence scores of the correct triples $\{(e_h, r, e_t)|(e_h, r, e_t) \in \mathcal{T}\}$:

$$\text{argmax}_t \phi(\mathbf{e}_{hr}, \mathbf{e}_t). \quad (16)$$

Consequently, we take correct triples as positive targets and the other entities as negative targets in the same batch during training. Following (Chen et al., 2020; Wang et al., 2022), we use the InfoNCE loss to achieve this goal:

$$L_f = -\log \frac{e^{(\phi(\mathbf{e}_{hr}, \mathbf{e}_t) - \gamma)/\tau}}{e^{(\phi(\mathbf{e}_{hr}, \mathbf{e}_t) - \gamma)/\tau} + \sum_{i=1}^{N} e^{\phi(\mathbf{e}_{hr}, \mathbf{e}'_t)/\tau}}, \quad (17)$$

where $\gamma > 0$ is a margin factor that encourages the model to increase the confidence scores of the correct triples, and τ is a temperature factor to adjust the relative importance of negatives. The loss of head entity prediction is similar.

Fusion Ranking Given the extra computing resources consumed by re-evaluation operations, it is necessary to prune this process. Consequently, we propose a heuristic fusion ranking strategy to eliminate unnecessary re-evaluations and improve performance. First, we introduce two definitions:

Definition 1. *(Consistent Prediction) Given a query with triple-level predicted rank list $\hat{A}_t$ and path-level predicted rank list $\hat{A}_p$, it is a consistent prediction if there exists $\hat{A}_{t_0} = \hat{A}_{p_0}$, where $\hat{A}_{t_0}$ and $\hat{A}_{p_0}$ are highest ranking of their list. A set of consistent predictions is given by:*

$$A_C = \{\hat{A}_t | \hat{A}_{t_0} = \hat{A}_{p_0}\}. \quad (18)$$

Definition 2. *(Hard Prediction) Given a query with triple-level predicted rank list $\hat{A}_t$ and path-level predicted rank list $\hat{A}_p$, it is a hard prediction if there exists $|\hat{A}_t \cap \hat{A}_p| \leq \alpha$ or $|\hat{A}_t| \geq \beta$, $\alpha, \beta \geq 0$. A set of hard predictions is given by:*

$$A_H = \{\hat{A}_t \cup \hat{A}_p || \hat{A}_t \cap \hat{A}_p| \leq \alpha, |\hat{A}_t| \geq \beta\}. \quad (19)$$

Consistency prediction indicates that the predicted entities from the two views are relatively aligned, which is a simple prediction. Therefore, we can directly select the path-level ranking as the final result. Hard prediction indicates significant discrepancies between the two views, suggesting a more complex prediction. In these cases, it is necessary to re-evaluate and re-order all the candidate entities using the fact-level evaluator. For other cases that belong to general prediction, we can use any rank or re-evaluation rank as the final result. we also choose the path-level rank as the final result to simplify the actual process.

Methods	WN18RR				FB15k-237			
	MRR	Hits@1	Hits@3	Hits@10	MRR	Hits@1	Hits@3	Hits@10
KG-BERT (Yao et al., 2019)	21.6	4.1	30.2	52.4	-	-	-	42.0
StAR (Wang et al., 2021)	40.1	24.3	49.1	70.9	29.6	20.5	32.2	48.2
GenKGC (Xie et al., 2022)	-	28.7	40.3	53.5	-	19.2	35.5	43.9
KGT5 (Saxena et al., 2022)	50.8	48.7	-	54.4	27.6	21.0	-	41.4
CoLE (Liu et al., 2022)	58.5	53.2	60.7	68.9	38.7	29.3	42.6	**57.0**
SimKGC (Wang et al., 2022)	66.6	58.7	71.7	80.0	33.6	24.9	36.2	51.1
KG-S2S (Chen et al., 2022)	57.4	53.1	59.5	66.1	33.6	25.7	37.3	49.8
CSProm-KG (Chen et al., 2023)	57.5	52.2	59.6	67.8	35.8	26.9	39.3	53.8
PDKGC (Geng et al., 2023)	57.7	50.5	60.9	71.3	37.9	28.5	41.5	56.6
BMKGC (Kong et al., 2024)	66.9	59.0	72.0	**80.7**	33.2	24.7	36.5	51.4
COSIGN (Li et al., 2024)	64.1	61.0	65.4	71.4	36.8	31.5	43.4	52.0
PEMLM (Qiu et al., 2024)	55.6	50.9	57.3	64.8	35.5	26.4	38.9	53.8
PEKGC (ours)	**69.7**	**64.9**	**73.5**	78.7	**39.2**	**33.6**	**46.6**	54.4

Table 1: The performance of PLM-based KGC models on the link prediction task. The Hits@1, Hits@3, Hits@10, and MRR metrics are multiplied by 100. We **highlight** the best results and underline the second-best results.

4 Experiments

To investigate our model's effectiveness and efficiency, we evaluate the performance of proposed model on KGC (link prediction) task, which aims to produce a ranking list of all entities for a query $((e_h, r, ?)$ or $(?, r, e_t))$, on benchmark KGs. We also conduct an ablation study to demonstrate the impact of each proposed module. Additionally, we present examples of reasoning paths in a case study to illustrate that PEKGC effectively generates high-quality reasoning paths.

4.1 Experiment Setup

Datasets We adopt two benchmark datasets for the link prediction task, i.e., WN18RR (Toutanova et al., 2015) and FB15k-237 (Dettmers et al., 2018). Table 2 lists the details of these two datasets.

Metrics Similar to recent works, we use the mean reciprocal rank (MRR) and Hits@k to evaluate the performance of all models, where Hits@k represents the fraction of positive triples ranked in the top k positions.

Baselines Based on link prediction task, we compare our approach with twelve PLM-based KGC methods: KG-BERT (Yao et al., 2019), StAR (Wang et al., 2021), GenKGC (Xie et al., 2022), KGT5 (Saxena et al., 2022), CoLE (Liu et al., 2022), SimKGC (Wang et al., 2022), KG-S2S (Chen et al., 2022), CSProm-KG (Chen et al., 2023), PDKGC (Geng et al., 2023), BMKGC (Kong et al., 2024),COSIGN (Li et al., 2024), and PEMLM (Qiu et al., 2024).

Datasets	#Ent	#Rel	#Tri	#Degree
WN18RR	40945	11	86835	2.19
FB15k-237	14505	237	272115	19.74

Table 2: Datasets are used in the experiments.

4.2 Main Results

Table 1 presents the link prediction results for PLM-based models on the WN18RR and FB15k-237 datasets. The experimental results confirm that our PEKGC model achieves satisfactory performance compared to baseline models across most metrics. Notably, our PEKGC improves the Hits@1 metric by 6.4% and 6.7% on the WN18RR and FB15k-237 datasets compared to the previous best PLM-based models. We attribute this to Definition 1 of the fusion ranking paradigm, which employs cross-verification of two-level predictions on the top-one results, further improving Hits@1, as evidenced in Table 4. Additionally, the results in the table also suggest that both definitions should be used together to achieve more significant improvement.

We observe that the performance of PEKGC is weaker than some models in Hits@10, which can be attributed to common limitations of encoder-decoder models. Since these models rely on generating candidate answers through a decoder, the diversity of answers is constrained. As a result, encoder-decoder models generally face a significant disadvantage on larger rank scales compared to encoder-only models, which can match all entities of the KGs by calculating their matching confidence scores. However, our proposed model still outperforms previous encoder-decoder meth-

ID	Levels	Answers
Query: (?, MemberOfDomainRegion, Facer[A dated Briticism]) ⇒ Answer: United Kingdom of Great Britain and Northern Ireland		
1	Triple Level	United Kingdom of Great Britain and Northern Ireland
	Path Level	Facer $\overset{MemberOfDomainRegion^{-1}}{\longrightarrow}$ England $\overset{HasPart^{-1}}{\longrightarrow}$ United Kingdom of Great Britain and Northern Ireland
Query: (Telephone, VerbGroup, ?]) ⇒ Answer: Call		
2	Triple Level	Call
	Path Level	Telephone $\overset{SynsetDomainTopicOf^{-1}}{\longrightarrow}$ Call
Query: (?, MusicArtistsGenre, Dio]) ⇒ Answer: Rock Music		
3	Triple Level	Hard Rock & *Heavy Rock* & *Thrash Metal* & *Doom Metal*
	Path Level	Dio $\overset{MusicArtistsGenre^{-1}}{\longrightarrow}$ Hard Rock $\overset{MusicParentGenre}{\longrightarrow}$ Rock Music
Query: (Asheville, LocationTimeZones, ?) ⇒ Answer: Eastern Time Zone		
4	Triple Level	Eastern Time Zone
	Path Level	Asheville $\overset{Contains^{-1}}{\longrightarrow}$ Buncombe County $\overset{LocationTimeZones}{\longrightarrow}$ Eastern Time Zone

Table 3: Four examples are predicted at triple and path levels, respectively. The first two queries come from the WN18RR dataset, and the other two come from the FB15k-237 dataset. The inverse relations of existing relations are denoted by $^{-1}$.

Datasets	Models	MRR	Hits@1
WN18RR	PEKGC	**69.7**	**64.9**
	-Definition 1	68.4	63.6
	-Definition 2	67.8	62.3
FB15k-237	PEKGC	**39.2**	**33.6**
	-Definition 1	36.7	29.7
	-Definition 2	37.2	30.6

Table 4: Comparison of MRR and Hits@1 between PEKGC and models without partial fusion ranking.

Datasets	Models	MRR	Hits@1
WN18RR	PEKGC	**69.7**	**64.9**
	-Path Level	67.1	62.8
	-Soft Prompt	68.5	63.9
	-Fact Level	67.8	62.3
	-Fusion Ranking	65.8	60.9
	-Neighbors	60.0	55.3
FB15k-237	PEKGC	**39.2**	**33.6**
	-Path Level	36.7	30.6
	-Soft Prompt	38.2	32.7
	-Fact Level	37.2	30.6
	-Fusion Ranking	36.7	29.7
	-Neighbors	38.1	31.0

Table 5: Comparison of MRR and Hits@1 between PEKGC and models without path-level generation, soft prompt, fact-level re-evaluation, fusion ranking, or neighbors.

ods across all indicators. These phenomena demonstrate the effectiveness of the proposed PEKGC, as it indeed improves the generation abilities of encoder-decoder models.

4.3 Ablation Study

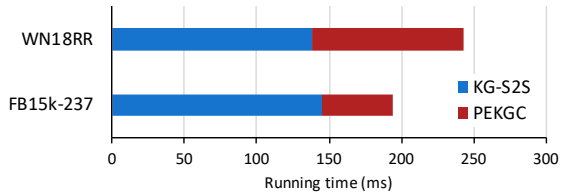
We conducted an ablation study on two benchmark datasets to evaluate the effectiveness of the proposed modules. As shown in Table 5, "-Path Level" refers to training and predicting solely at the triple level, "-Fact Level" indicates the exclusion of fact-level re-evaluations, "-Soft Prompt" involves removing KG and position prompts, and "-Fusion Ranking" means eliminating the heuristic fusion ranking paradigm and using fact-level re-evaluation ranking as the final results in the inference stage, "-Neighbors" denotes the removal of neighbor triples. We observe that removing any modules leads to a significant performance drop across both datasets. Additionally, the final performance is not good when using only the fusion ranking without re-evaluating hard predictions, further emphasizing the necessity of re-evaluating hard predictions as outlined in Definition 2. In brief, these ablations validate the effectiveness of the proposed modules.

4.4 Case Study

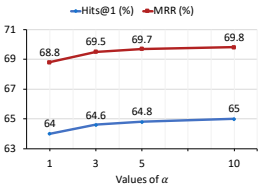
We present four typical queries answered at both the triple and path levels to gain insight into the difference between the two-level generation, as shown

in Table 3. Triple-level predictions provide entity answers directly, while path-level prediction offers both entity answers and reasoning paths, which serve as reasoning chains.

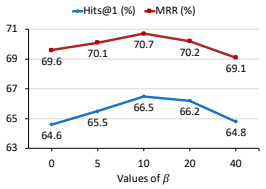
In particular, the third query shows the difference in accuracy between the two-level generation. Although "Hard Rock" is a correct answer, "Heavy Rock", "Thrash Metal" and "Doom Metal" are completely fabricated answers produced by the triple-level generator. The reasons for this phenomenon are twofold: i) Encoder-decoder models adopt a teacher-forcing strategy during training, leading to reduced scalability when predicting unseen triples, especially in 1-N or N-N triples. ii) At times, predicting the correct results directly can be challenging due to the lack of necessary prior knowledge or a step-by-step reasoning process. The paths between entities can provide the required prior knowledge and serve as a reasoning chain to address this issue. The above demonstrates the necessity of introducing paths to enhance the robustness and expansibility of completions. Additionally, the paths incorporated into the generation process play a crucial role in providing human-understandable interpretability.



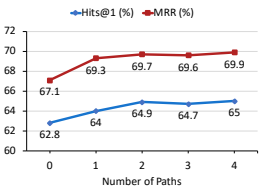
(a) Running times.



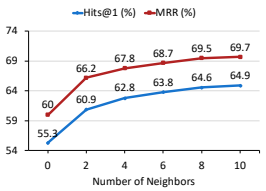
(b) The value of α.



(c) The value of β.



(d) The number of paths.



(e) The number of neighbors.

Figure 3: The further experimental results of PEKGC. The last four figures are the results of the WN18RR dataset.

4.5 Further Analysis

Running Efficiency Analysis To visualize the efficiency of PEKGC, we compare its average running times to KG-S2S, as shown in Figure 3(a). While PEKGC needs to consider path and structural information compared to KG-S2S, its average runtime increases by 105ms and 49ms on two datasets, respectively. However, under the condition of sacrificing acceptable runtime efficiency, PEKGC's Hits@1 performance is on average improved by 22.2% and 30.7%, respectively, compared to KG-S2S.

Hyperparameters of Fusion Ranking For the two parameters in fusion ranking, a larger value of α or a smaller value of β means more candidate triples must be re-evaluated. As shown in Figure 3(b)-3(c), a smaller or too-large value of β cannot get better results. Consequently, choosing the proper range for re-evaluation can reduce the computational overhead and improve the results.

Path Numbers for Each Triple Figure 3(d) shows the performance impact of using different numbers of paths for each triple. We observe that the introduction of paths improves the model's performance. However, when the number of paths is ≥ 2, there is no significant improvement. This indicates that an appropriate number of paths is sufficient for satisfactory improvement.

Neighbor Numbers for Each Triple As shown in Figure 3(e), the addition of neighbor information greatly improves the performance of the generative model, but more neighbor triples will increase the length of the input sequence and final running time. We observed that when the number of neighbors is set to 8, the recall rates and satisfactory results can be achieved.

5 Conclusion

We propose a multi-view generation framework, PEKGC, for KGC tasks that captures triple-level knowledge, long-term path-level knowledge, and structural neighborhoods, achieving more directly interpretable results. We propose a heuristic fusion ranking strategy for multi-view generation during inference to further combine triple-level and path-level results to achieve better ranking performance. Experimental results demonstrate the effectiveness of our approach. In future work, we aim to address the challenges of diversity and generalization of PLM-based models.

6 Limitations

Our proposed PEKGC model significantly improves the performance of encoder-decoder KGC models. However, three challenges remain for future work: i) The diversity of generated results is limited, resulting in a high number of identical and homogeneous outputs, which affects the models' universality and robustness; ii) Decoders tend to favor generating seen entities, leading to challenges in flexibility and generalization, especially in 1-N and N-N triples.

7 Ethics Considerations

Our work adheres to the guidelines outlined in the ACL Code of Ethics. As knowledge graph completion is a widely accepted and long-standing research task, we do not see any significant ethical concerns. As for the scientific artifacts used in our experiments, we confirm to comply with the corresponding intended use and licenses.

References

Ivana Balažević, 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)*, pages 5185–5194.

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.

Andrew Carlson, Justin Betteridge, Bryan Kisiel, Burr Settles, Estevam R Hruschka, and Tom M Mitchell. 2010. Toward an architecture for never-ending language learning. In *Twenty-Fourth AAAI conference on artificial intelligence*.

Chen Chen, Yufei Wang, Bing Li, and Kwok-Yan Lam. 2022. Knowledge is flat: A seq2seq generative framework for various knowledge graph completion. In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 4005–4017.

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.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. 2020. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR.

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. In *International Conference on Learning Representations*.

Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. 2018. Convolutional 2d knowledge graph embeddings. In *Thirty-second AAAI conference on artificial intelligence*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186.

Yuxia Geng, Jiaoyan Chen, Yuhang Zeng, Zhuo Chen, Wen Zhang, Jeff Z Pan, Yuxiang Wang, and Xiaoliang Xu. 2023. Prompting disentangled embeddings for knowledge graph completion with pre-trained language model. *Available at SSRN 4790015*.

Lingbing Guo, Zequn Sun, and Wei Hu. 2019. Learning to exploit long-term relational dependencies in knowledge graphs. In *International conference on machine learning*, pages 2505–2514.

Chunyang Jiang, Tianchen Zhu, Haoyi Zhou, Chang Liu, Ting Deng, Chunming Hu, and Jianxin Li. 2023. Path spuriousness-aware reinforcement learning for multi-hop knowledge graph reasoning. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pages 3173–3184.

Yonghui Kong, Cunhang Fan, Yujie Chen, Shuai Zhang, Zhao Lv, and Jianhua Tao. 2024. Bilateral masking with prompt for knowledge graph completion. In *Findings of the Association for Computational Linguistics: NAACL 2024*, pages 240–249.

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.

Xi Victoria Lin, Richard Socher, and Caiming Xiong. 2018. Multi-hop knowledge graph reasoning with reward shaping. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3243–3253.

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*, pages 1329–1338.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Xin Lv, Yankai Lin, Yixin Cao, Lei Hou, Juanzi Li, Zhiyuan Liu, Peng Li, and Jie Zhou. 2022. Do pretrained 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.

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

Chenyu Qiu, Pengjiang Qian, Chuang Wang, Jian Yao, Li Liu, Fang Wei, and Eddie Eddie. 2024. Joint pre-encoding representation and structure embedding for efficient and low-resource knowledge graph completion. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 15257–15269.

Meng Qu, Junkun Chen, Louis-Pascal Xhonneux, Yoshua Bengio, and Jian Tang. 2020. Rnnlogic: Learning logic rules for reasoning on knowledge graphs. In *International Conference on Learning Representations*.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140):1–67.

Feiliang Ren, Juchen Li, Huihui Zhang, Shilei Liu, Bochao Li, Ruicheng Ming, and Yujia Bai. 2020. Knowledge graph embedding with atrous convolution and residual learning. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 1532–1543.

Ali Sadeghian, Mohammadreza Armandpour, Patrick Ding, and Daisy Zhe Wang. 2019. Drum: End-to-end differentiable rule mining on knowledge graphs. *Advances in Neural Information Processing Systems*.

Apoorv Saxena, Adrian Kochsiek, and Rainer Gemulla. 2022. Sequence-to-sequence knowledge graph completion and question answering. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2814–2828.

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 *15th International Conference on Extended Semantic Web Conference, ESWC 2018*, pages 593–607.

Zhiqing Sun, Zhi-Hong Deng, Jian-Yun Nie, and Jian Tang. 2018. Rotate: Knowledge graph embedding by relational rotation in complex space. In *International Conference on Learning Representations*.

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.

Shikhar Vashishth, Soumya Sanyal, Vikram Nitin, and Partha Talukdar. 2019. Composition-based multi-relational graph convolutional networks. In *International Conference on Learning Representations*.

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. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4281–4294.

Xin Xie, Ningyu Zhang, Zhoubo Li, Shumin Deng, Hui Chen, Feiyu Xiong, Mosha Chen, and Huajun Chen. 2022. From discrimination to generation: Knowledge graph completion with generative transformer. In *ICLR 2022 Workshop on Deep Learning on Graphs for Natural Language Processing*.

Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. 2015. Embedding entities and relations for learning and inference in knowledge bases. In *International Conference on Learning Representations*.

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

Yongqi Zhang, Zhanke Zhou, Quanming Yao, Xiaowen Chu, and Bo Han. 2023. Adaprop: Learning adaptive propagation for graph neural network based knowledge graph reasoning. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 3446–3457.

Zhaocheng Zhu, Zuobai Zhang, Louis-Pascal Xhonneux, and Jian Tang. 2021. Neural bellman-ford networks: A general graph neural network framework for link prediction. *Advances in Neural Information Processing Systems*, 34.

A Implementation Details

A.1 Hyperparameter setting

We realize PEKGC on A100 GPU using the T5-base model (Raffel et al., 2020) with the Adam optimizer, and follow the standard T5 unsupervised training paradigm. Following KG-S2S (Chen et al., 2022), we enclose the entities' descriptions in square brackets, wrap paths in parentheses, and use the "|" token as a special separator. Answer texts are also enclosed by the T5 special tokens. During the data preparation phase, we use Biased Random Walks algorithm (Guo et al., 2019) to generate related paths for triples in the training set, tuning the number of paths per triple within the range of {1, 2, 3, 4}, and tuning the number of neighbors per triple within the range of {2, 4, 6, 8, 10}. In the training stage, the number of training epochs is set to 100 and 20, with the entity description lengths set to 40 and 80 for the WN18RR and FB15k-237 datasets, respectively. The mini-batch size for the main tasks is chosen from {16, 32, 64}. To maximize the performance of fact-level re-evaluation, we search for the optimal mini-batch size within {256, 512, 1024}. The maximum input token length for the encoder is set to 512. We use a learning rate of 0.001, an encode sequence dropout rate of 0.1, a margin factor of 0.02, and a temperature factor of 0.05. In the inference stage, the model generates the raw text without special tokens. We set the maximum output length to 150 and the number of samples for beam search to 40. For hard predictions, we search for α within {1, 2, 3, 4, 5, 10}, and β within {0, 5, 10, 20, 40}. The optimal parameter values are shown in Table 6.

Hyperparameters	Values	
	WN18RR	FB15k-237
batch size	64	32
learning rate	0.001	0.001
dropout rate	0.1	0.1
beam size	40	40
margin value μ	25	15
margin factor γ	0.02	0.02
temperature factor τ	0.05	0.05
input max length	512	512
output max length	150	150
description max length	40	80

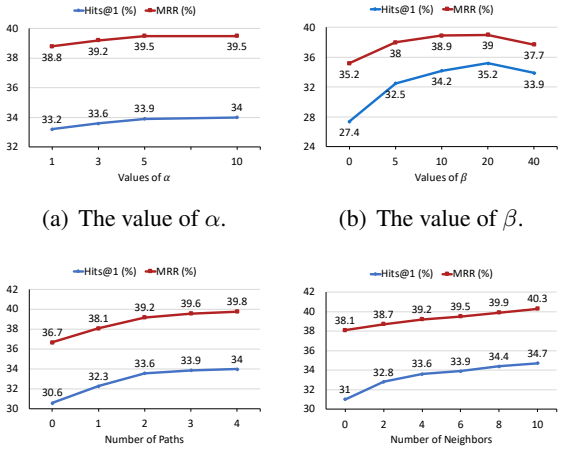
Table 6: The hyperparameters for two datasets.

A.2 Implementation Process

For the sake of clarity, we list the operation process of our framework as follows: i) Use the Biased Random Walks algorithm to extract the corresponding paths for all triples. ii) Pre-train the neighbor selector referring to Section 3.4. However, as the number of neighbor triples may be huge, we first sample 500-800 neighbor triples and then use the neighbor selector to filter them. iii) Pre-train the fact-level re-evaluator referring to Section 3.5. iv) Train PEKGC with the prepared data, referring to Section 3.3. v) Use trained PEKGC to generate results at the triple and path levels. vi) re-evaluate and re-order referring to heuristic fusion ranking strategy.

B More Analysis Results

As shown in Figure 4, the experimental results on the FB15k-237 dataset is consistent with the WN18RR dataset.



(a) The value of α. (b) The value of β.

(c) The number of paths. (d) The number of neighbors.

Figure 4: The further experimental results of PEKGC on the FB15k-237 dataset.