Do Pre-trained Models Benefit Knowledge Graph Completion? A Reliable Evaluation and a Reasonable Approach

Xin Lv[1,2], **Yankai Lin**[3], **Yixin Cao**[4], **Lei Hou**[1,2*], **Juanzi Li**[1,2]
Zhiyuan Liu[1,2], **Peng Li**[5†], **Jie Zhou**[3]
[1]Department of Computer Science and Technology, BNRist
[2]KIRC, Institute for Artificial Intelligence, Tsinghua University, Beijing 100084, China
[3]Pattern Recognition Center, WeChat AI, Tencent Inc., China
[4]Singapore Management University, Singapore
[5]Institute for AI Industry Research (AIR), Tsinghua University, China
lv-xl8@mails.tsinghua.edu.cn, houlei@tsinghua.edu.cn

Abstract

In recent years, pre-trained language models (PLMs) have been shown to capture factual knowledge from massive texts, which encourages the proposal of PLM-based knowledge graph completion (KGC) models. However, these models are still quite behind the SOTA KGC models in terms of performance. In this work, we find two main reasons for the weak performance: (1) Inaccurate evaluation setting. The evaluation setting under the closed-world assumption (CWA) may underestimate the PLM-based KGC models since they introduce more external knowledge; (2) Inappropriate utilization of PLMs. Most PLM-based KGC models simply splice the labels of entities and relations as inputs, leading to incoherent sentences that do not take full advantage of the implicit knowledge in PLMs. To alleviate these problems, we highlight a more accurate evaluation setting under the open-world assumption (OWA), which manually checks the correctness of knowledge that is not in KGs. Moreover, motivated by prompt tuning, we propose a novel PLM-based KGC model named PKGC. The basic idea is to convert each triple and its support information into natural prompt sentences, which are further fed into PLMs for classification. Experiment results on two KGC datasets demonstrate OWA is more reliable for evaluating KGC, especially on the link prediction, and the effectiveness of our PKCG model on both CWA and OWA settings.

1 Introduction

Knowledge graph (KG) has gradually become the cornerstone of many Natural Language Processing (NLP) tasks (Cui et al., 2017; Zhou et al., 2018), as one of the most effective ways to represent world knowledge. To improve the coverage, researchers have automated knowledge extraction techniques or relied on collaborative editing, while these KGs still hardly cover the massive emerging knowledge in the real world. This problem motivates knowledge graph completion (KGC), the task of predicting missing links through understanding existing structures in KGs.

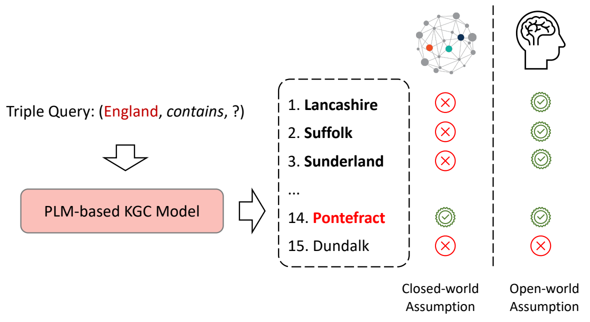

Figure 1: Evaluation results for link prediction under different settings. The bolded entities in the dashed box are all correct answers, but only the red entities are considered correct under the closed-world assumption.

Soon sweeping across the entire NLP field, the potential of pre-trained language models (PLMs) for KGC has attracted much attention. Petroni et al. (2019); Shin et al. (2020) reveal that PLMs have captured factual knowledge implicitly from massive unlabeled texts. This could be helpful to complete missing knowledge. KG-BERT (Yao et al., 2019) first introduces PLMs into KGC. It splices the labels of entities and relation in the triple as the input of PLMs to verify its correctness. Kim et al. (2020) further introduces multi-task learning on the basis of KG-BERT. However, the above PLM-based KGC models do not present promising results, or are even quite behind conventional knowledge graph embedding (KGE) models (about 20.8% lower than the SOTA model in Hits@10).

[*] Corresponding Author
[†] Part of the work was done while Peng Li was working at Tencent.

This raises a question: why the learned factual knowledge in PLMs cannot be beneficial for KGC?

In this work, we find two main reasons for the weak performance of PLM-based KGC models: (1) **Inaccurate evaluation setting**. Most existing KGC models are evaluated under the closed-world assumption (CWA), which assumes that any knowledge unseen in given KGs is incorrect. Such a setting can benefit the automatical dataset construction without manual annotation. However, the introduction of PLMs brings in much unseen knowledge, which is considered to be incorrect under CWA, wrongly lowering the performance. As shown in Figure 1, for a triple query (England, *contains*, ?), the PLM-based KGC model gives many correct tail entities (highlight with boldface), but only *Pontefract* is considered correct under CWA since it exists in KGs. (2) **Inappropriate utilization of PLMs**. Existing PLM-based KGC models simply splice the labels of the entities and relations in the triples as the input of PLMs. This results in incoherent sentences, which gaps with the pre-trained task and thus cannot take full advantage of the knowledge in PLMs.

To alleviate the above two problems, we propose a new benchmark setting for rectification of this line of research and a novel PLM-based model. To make the KGC evaluation more credible, we highlight a new evaluation setting based on the open-world assumption (OWA) — the knowledge not in KGs is not false, but unknown. Thus, false positives under CWA shall be removed, as long as we recognize exact true and false triples from unknown. For these unknown triples, we conduct human annotation to check if they are valid.

We further propose a novel PLM-based KGC model, **PKGC**, to better induce the implicit knowledge hidden in the PLM's parameters. Motivated by the prompt-based models (Petroni et al., 2019; Shin et al., 2020), the basic idea is to convert each triple into natural prompt sentences instead of simply splicing their labels. In specific, we manually define the prompt template for each relation type and further introduce soft prompts to better express the semantics of triples. Moreover, benefiting from prompt tuning, PKGC can flexibly consider the contexts of triples, such as definition and attributes by inserting them as the support prompt at the end of the triple prompt.

We conduct experiments on two KGC datasets sampled from Wikidata and Freebase, and re-evaluate the KGE-based and PLM-based KGC models under OWA instead of CWA. According to our experimental results, we find that: (1) OWA provides a more accurate evaluation for KGC, especially for the more knowledgeable PLM-based KGC model and the more open link prediction task. (2) By converting triples and supporting information into natural prompt sentences, our PKGC model can effectively utilize the PLM's knowledge in the KGC task, and thus is less sensitive to the amount of training data. (3) The reason for the good performance of our model is not only that PLMs have seen part of relevant knowledge in massive text, but also that our model has the reasoning ability and can combine knowledge from PLMs and KGs to infer unknown knowledge.

2 Related Work

2.1 Evaluation of KGC

Most exitsing KGC models (Ji et al., 2021) are evaluated under CWA, since the datasets can be constructed automatically. However, CWA is essentially an approximate assumption, which may bring inaccurate evaluation results.

OWA is rarely used to evaluate the performance of KGC models since it requires manual annotation for unseen triples. In recent years, there are two datasets CoDEx (Safavi and Koutra, 2020) and InferWiki (Cao et al., 2021), which provide evaluation datasets for triple classification under OWA. Besides, Safavi et al. (2020) evaluat the calibration of knowledge graph embeddings under OWA. Although these works are partially performed under OWA, they only use OWA as an additional experimental setting. In this work, we first systematically compare the differences between different models and different tasks under CWA and OWA. We find that CWA cannot accurately reflect the real performance of KGC models, which is more evident for PLM-based KGC models and link prediction task.

2.2 KGC Models

KGE models are the early mainstream approach for KGC. KGE models can be divided into three categories: (1) translation-based models (Bordes et al., 2013; Sun et al., 2019); (2) tensor-factorization based models (Balažević et al., 2019; Nickel et al., 2016) and (3) non-linear models (Dettmers et al., 2018; Nguyen et al., 2017). In addition, there are some KGE models that further introduce additional information, such as text (Xie et al., 2016; Veira et al., 2019) and attributes (Lin et al., 2016).

In addition to KGE models, there are some PLM-based models that attempt to obtain knowledge from PLMs, which are detailed in the following.

PLM-based KGC models fine-tune the PLMs on the KGC task to leverage both the implicit knowledge in PLMs and the structured knowledge in KGs. KG-BERT (Yao et al., 2019) is the first model that uses PLMs to perform KGC. It simply splices the labels of entities and relations in triples as the input to PLMs. Based on KG-BERT, Kim et al. (2020) further introduce multi-task learning, and Talukdar et al. (2021) focuses on zero-shot learning setting. Compared with our model, these model all simply splices the labels in triples, which results in incoherent sentences and cannot fully exploit the implicit knowledge in PLMs.

Prompt-based knowledge probing models aims at probing how much knowledge the PLM contains. Therefore, they do not fine-tune the PLM on KGC tasks. LAMA (Petroni et al., 2019) is the first prompt-based knowledge probing work, which converts a triple query into sentences with `[MASK]` and uses the output of `[MASK]` as the predicted entity. Based on LAMA, there are some models (Shin et al., 2020; Zhong et al., 2021; Liu et al., 2021) improved from automatic template generation and adding soft prompts. These models focus on probing and do not use the knowledge already in KGs. In addition, most of them can only predict entities with a single token, so these models cannot be realistically used for KGC yet.

3 Preliminary

Knowledge graph is a network composed of entities and relations. It can be defined as $\mathcal{KG} = \{\mathcal{E}, \mathcal{R}, \mathcal{T}\}$, where $\mathcal{E}$ is the set of entities and $\mathcal{R}$ is the set of relations. $\mathcal{T} = \{(h, r, t)\} \subseteq \mathcal{E} \times \mathcal{R} \times \mathcal{E}$ is the triple set, where h and t are the head and tail entities, and r is the relation between them.

Knowledge graph completion task aims at completing missing triples $(h, r, t) \notin \mathcal{T}$ for the knowledge graph. There are two main methods to do this task, namely link prediction and triple classification, where the former mainly predicts missing entities for triple queries $(h, r, ?)$ or $(?, r, t)$, and the latter aims to determine whether a given triple (h, r, t) is correct or not.

Closed-world assumption (CWA) believes that the triples that do not appear in a given knowledge graph are wrong. This means that if the dataset consists of training/validation/test set, and the model is tested on the test set, only the triples that have appeared in the entire dataset are considered to be correct. We can easily evaluate the performance of models without annotation under CWA. However, CWA is essentially an approximation and cannot guarantee the accuracy of the evaluation results.

Open-world assumption (OWA) believes that the triples contained in the knowledge graph are not complete. Therefore, the evaluation under the open-world assumption is more accurate and closer to the real scenario, but requires additional human annotations to carefully verify whether the completed triples that are not in the knowledge graph are correct or not.

4 Methodology

4.1 Framework

In this paper, we propose a novel PLM-based KGC model named PKGC, which can leverage the implicit knowledge in PLMs and the structured knowledge in KGs to infer new knowledge.

Specifically, on the one hand, we convert a triple into prompt sentences to use the knowledge in PLMs. As shown in Figure 2, given a triple, our model transforms it into triple prompts P^T and support prompts P^S, which are jointly fed into a pre-trained language model. Formally, the final input texts T to the PLM can be defined as T = `[CLS]`P^T P^S`[SEP]` and the output of `[CLS]` in the language model is used to predict the label of the given triple. On the other hand, we feed positive/negative triples to our model for triple classification and use cross-entropy loss for training. In this way, our model can exploit the structural information in KGs.

In the following sections, we will introduce the design strategy of triple prompts(Section 4.2) and the production method of support prompts (Section 4.3) in detail. In addition, we will also explain the training method of our model in Section 4.4.

4.2 Triple Prompts

To better exploit the implicit knowledge in PLMs, we transformed each triple into triple prompts. Motivated by LAMA (Petroni et al., 2019), for every relation $r \in \mathcal{R}$, we manually design a hard template for the relation to represent the semantics of the associated triples. For example, in Figure 2, the hard template for relation *member of sports team* is “[X] plays for [Y].”. By replacing [X] and [Y] with

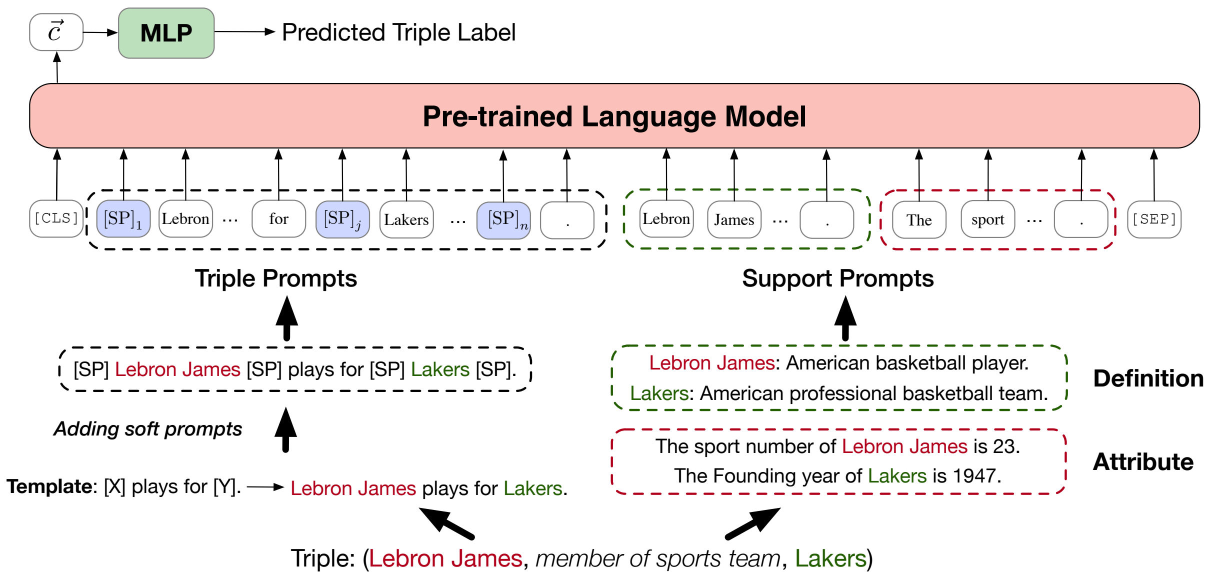
$\vec{c}$
MLP
Predicted Triple Label
Pre-trained Language Model
[CLS]
[SP]$_1$
Lebron
for
[SP]$_j$
Lakers
[SP]$_n$
.
Lebron
James
.
The
sport
.
[SEP]
Triple Prompts
Support Prompts
[SP] Lebron James [SP] plays for [SP] Lakers [SP].
Lebron James: American basketball player.
Lakers: American professional basketball team.
Definition
Adding soft prompts
The sport number of Lebron James is 23.
The Founding year of Lakers is 1947.
Attribute
Template: [X] plays for [Y]. Lebron James plays for Lakers.
Triple: (Lebron James, member of sports team, Lakers)

Figure 2: Illustration of our PKGC model for triple classification. Triples are transformed into triple prompts (left part) and support prompts (right part) to do the classification using PLMs.

the labels of the head and tail entities, we can obtain the preliminary triple prompts P_p^T. In Figure 2, P_p^T is "Lebron James plays for Lakers.".

To make triple prompts more expressive, inspired by Han et al. (2021), we also add some soft prompts to P_p^T to form the final triple prompts P^T. Formally, we have a vector lookup table $\mathbf{P} \in \mathbb{R}^{|\mathcal{R}|\times n\times d}$ for soft prompts, where n is the total number of soft prompts contained in the triple prompts for one triple, and d is the dimension of the word vector corresponding to the language model. As shown in Figure 3, the template and entity label split the triple prompts into six positions and we can insert soft prompts in them respectively. The number of soft prompts at each position is $n_1, n_2, \cdots n_6$. In our model, we have $n = \sum_{i=1}^{6} n_i$. For the k-th soft prompt [SP]$_k$ in the triple prompts, when it is input to the language model, the corresponding word vector will be replaced with a vector from $\mathbf{P}$, i.e., $\mathbf{p}_r^k = \mathbf{P}_{[idx(r),k]} \in \mathbb{R}^d$, where $idx(r)$ is the ranking index of relation r. In other words, $\mathbf{p}_r^k$ is the k-th vector corresponding to the relation r in $\mathbf{P}$. As the training progresses, the vector lookup table $\mathbf{P}$ will be updated so that it can better represent the semantics of the corresponding triples together with the hard templates.

4.3 Support Prompts

In addition to the triple information in the knowledge graph itself, there are many support information that can help knowledge graph completion,

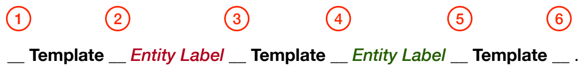
1
2
3
4
5
6
_ Template _ Entity Label _ Template _ Entity Label _ Template _ .

Figure 3: Illustration of inserting soft prompts into triple prompts. The "Template" in the sentence represents the words in the hard template. We can insert several soft prompts in six positions (underlined) at most, and the sum of the numbers of these soft prompts is n.

such as definition and attribute. In previous knowledge graph embedding models (Veira et al., 2019; Lin et al., 2016), it is usually necessary to change the model structure to introduce specific types of additional information, which will bring a lot of additional overhead and is not conducive to the unification of multiple types of support information.

Due to the generality of the language, it is easy to introduce various support information in our model without changing the model structure. As shown in Table 1, we define templates to convert support information into the corresponding sentences. For a triple (h, r, t), there may be more than one corresponding attribute. In order to avoid too complex models, in this work, we use a random strategy to select attributes, i.e., randomly selecting an attribute for each entity in a triple.

It is worth noting that our model does not require all support information to be present. If it does not exist, just do not add the corresponding information. In addition, our model can also support more other types of support information well, just by manually defining the corresponding templates as in Table 1.

Type	Template
Definition	"[***Entity***]: [***Definition Text***]."
Attribute	"The [***Attribute***] of [***Entity***] is [***Value***]."

Table 1: Templates for support information, where [***Entity***], [***Attribute***], [***Value***] denote the label of entity, attribute and value respectively. [***Definition Text***] is the text corresponding to the entity definition.

4.4 Training

Our model is trained on the triple set $\mathcal{D} = \mathcal{T} \cup \mathcal{T}^-$ as triple classification. Specifically, $\mathcal{T}^-$ consists of two types of negative triples: (1) random negative triples $\mathcal{T}^-_{\text{RAN}}$, which is generated by randomly replacing the head or tail entities of the triple in $\mathcal{T}$ with other entity in $\mathcal{E}$. Random negative examples are simple, but can cover most entities. (2) KGE-based negative triples $\mathcal{T}^-_{\text{KGE}}$, which is generated by replacing the head or tail entity with another entity that the KGE model considers to have a high probability of holding. KGE negative triples are more difficult. In our model, there is a hyperparameter α to control the ratio of $\mathcal{T}^-_{\text{RAN}}$ and $\mathcal{T}^-_{\text{KGE}}$, i.e., $\frac{|\mathcal{T}^-_{\text{RAN}}|}{|\mathcal{T}^-_{\text{KGE}}|} = \frac{\alpha}{1-\alpha}$. Besides, we also have a hyperparameter K to control the ratio of positive and negative triples, i.e., $|\mathcal{T}| = K \cdot |\mathcal{T}^-|$. Given a triple $\tau = (h, r, t)$, the classification score for the triple can be defined as:

$$\mathbf{s}_\tau = \text{Softmax}(\mathbf{W}\mathbf{c}), \tag{1}$$

where $\mathbf{c} \in \mathbb{R}^d$ is the output vector of the input token `[CLS]`, $\mathbf{W} \in \mathbb{R}^{2\times d}$ is a linear neural network. We define the following cross-entropy loss for optimization:

$$\mathcal{L} = -\sum_{\tau \in \mathcal{T} \cup \mathcal{T}^-} (y_\tau \log(\mathbf{s}^1_\tau) + (1 - y_\tau)\frac{\log(\mathbf{s}^0_\tau)}{K}), \tag{2}$$

where $y_\tau \in \{0, 1\}$ is the label for triple τ and $\mathbf{s}^0_\tau, \mathbf{s}^1_\tau \in [0, 1]$ are the value of the first two dimensions of $\mathbf{s}_\tau$.

5 Experiments

In experiments, we give the results of models under CWA and OWA. Specifically, the results under CWA are for reference, and the results under OWA can better reflect the real performance of the model.

5.1 Evaluation Protocol

Link Prediction Given a positive triple (h, r, t) in the test set, we convert it into a triple query $(h, r, ?)$ or $(?, r, t)$. The link prediction task requires the model to give a descending order of the probability that each entity is the missing entity. Following previous work (Dettmers et al., 2018), we use two evaluation metrics, i.e., MRR and Hits@N. However, these two metrics are not applicable in link prediction under OWA since we cannot get the true label of all possible triples by manual annotation. For example, given a medium-sized dataset with 10,000 entities and 10,000 triples in the test set, we need to know the true labels of at most 200 million $(2 \times 10,000 \times 10,000)$ triples, which is not possible to get by annotation. Therefore, we use an alternative evaluation. Specifically, we sample triples from test set and fill the missing entity with the top-1 predicted entity. Then, we manually annotate the correct ratio of these triples. This evaluation metric is denoted as CR@1.

Triple Classification Triple classification task aims to judge whether a given triple is correct or not. This is essentially a binary classification task, so we use Accuracy and F1 as the evaluation metrics. In contrast to link prediction, triple classification task enables a low-cost evaluation of the model's performance under OWA, because we only need to ensure a small number of (consistent with the number of positive triples in the test set) negative triples that are really wrong through annotation.

Dataset	$\lvert\mathcal{E}\rvert$	$\lvert\mathcal{R}\rvert$	**# Train**	**# Valid**	**# Test**
Wiki27K	27,122	62	74,793	20,242/1,994	20,244/1,994
FB15K-237-N	13,104	93	87,282	14,082/2,046	16,452/2,048
FB15K-237-NH	13,104	93	87,282	14,082/1,006	16,452/1,004

Table 2: Statistics of the datasets we use, where # Train , # Valid and # Test denote the number of triples in the training, validation and test sets, respectively. For the right two columns, the front and back of the slash represent the number of triples used for evaluation under CWA and OWA, respectively.

5.2 Datasets

We use two main datasets sampled from Wikidata and Freebase in our experiments.

As we introduce in Section 2, CoDEx and InferWiki provide evaluation datasets for triple classification under OWA. However, they have some problems that do not apply to our task. For example, the distribution of relations for the negative triples of CoDEx differs significantly from the training set, which violates the assumption of consistent distribution. InferWiki is mainly concerned with the

	Model	Wiki27K					FB15K-237-N				
		MRR	@1	@3	@10	**CR@1**	MRR	@1	@3	@10	**CR@1**
KGE Models	TransE (Bordes et al., 2013)	15.5	3.2	22.8	37.8	16.0	25.5	15.2	30.1	45.9	42.0
	TransC (Lv et al., 2018)	17.5	12.4	21.5	33.9	20.0	23.3	12.9	29.8	39.5	44.0
	ConvE (Dettmers et al., 2018)	22.6	16.4	24.4	35.4	21.5	27.3	19.2	30.5	42.9	48.5
	WWV (Veira et al., 2019)	19.8	15.7	23.7	36.5	22.5	26.9	13.7	28.7	44.3	40.5
	TuckER (Balažević et al., 2019)	24.6	18.3	26.5	38.2	33.0	31.2	22.8	**34.6**	48.6	51.0
	RotatE (Sun et al., 2019)	21.6	12.3	25.6	39.4	30.5	27.9	17.7	32.0	48.1	53.0
PLM-based	KG-BERT (Yao et al., 2019)	19.2	11.9	21.9	35.2	35.5	20.3	13.9	20.1	40.3	47.5
	LP-RP-RR (Kim et al., 2020)	21.7	13.8	23.5	37.9	38.0	24.8	15.5	25.6	43.6	52.5
	PKGC	25.2	18.9	28.5	39.0	44.0	30.7	23.2	32.8	47.1	58.5
	PKGC w/ attribute	25.5	19.1	28.8	39.4	44.0	31.1	23.5	32.9	47.7	58.5
	PKGC w/ definition	**28.5**	**23.0**	**30.5**	**40.9**	**47.5**	**33.2**	**26.1**	**34.6**	**48.7**	**62.5**

Table 3: Link prediction results on two datasets. @X denotes Hits@X. CR@1 is the evaluation metric for OWA in Section 5.1. All metrics are multiplied by 100. The best score is in **bold**.

triples that can be inferred from rules. Therefore, we construct a new dataset named Wiki27K based on Wikidata and manually annotate real negative triples. Due to space limitations, we put the detailed steps of dataset construction in Appendix A.

As reported by (Akrami et al., 2020), there are many mediator (CVT) nodes in Freebase, which will bring Cartesian production relations. (Akrami et al., 2020) confirm that the prediction tasks corresponding to these relations are not meaningful and would improperly improve the model accuracy. In order to increase the difficulty of the task and to be closer to the KGC task in real scenarios, we obtain a dataset FB15K237-N by removing the relations containing mediator nodes in FB15K-237. Besides, to make the triple classification harder, we also construct a dataset FB15K-237-NH based on FB15K-237-N by only modifying the negative triples. It is only used for triple classification. Specifically, for every positive triple (h, r, t) in validation and test set, we use TransE (Bordes et al., 2013) to do link prediction and use the highest probability non-answer entity to replace the missing entity to generate a hard negative triple. The statistics of our datasets [1] are listed in Table 2.

5.3 Experiment Setup

Baseline Models In experiments, we choose six KGE models as comparisons, namely TransE (Bordes et al., 2013), ConvE (Dettmers et al., 2018), TuckER (Balažević et al., 2019), RotatE (Sun et al., 2019), TransC (Lv et al., 2018) and WWV (Veira et al., 2019), the last two of which use concept and definition information, respectively. In addition, we also compare with two PLM-based models KG-BERT (Yao et al., 2019) and LP-RP-RR (Kim et al., 2020). For our model, we have a base model PKGC that does not use any support information and two variants that use two support information, definition and attribute, respectively.

Implementation Details In our implementation, we use RoBERTa-Large as the PLM. For the parameters n, α, K, we choose them from {0, 1, 2, 3, 6}, {0.0, 0.3, 0.5, 0.7, 1.0} and {10, 30, 50, 100}, respectively. More parameter selections are placed in the Appendix F. We use TuckER (Balažević et al., 2019) to generate KGE negative triples for our model. The details of the manual annotation in the experiments are placed in the Appendix E. For TransE and RotatE, we use the codes implemented by OpenKE (Han et al., 2018). For other baseline models, we use the codes released by the authors for re-implementation.

5.4 Link Prediction Results

The experimental results on link prediction are shown in Table 3, where Hits@1 and CR@1 can evaluate the accuracy of the model to predict the entity with the highest probability under CWA and OWA, respectively. From the table, we can learn that most models have a large difference in performance under Hits@1 and CR@1. This performance gap is more evident in PLM-based models. For example, on Wiki27K, although KG-BERT and LP-RP-RR are lower than almost all KGE models on Hit@1, they both outperform them on CR@1. For KGE models, the performance gap cannot be ignored as well. We can see that the models do not have the same ranking of performance under CWA and OWA, which illustrates the inability of CWA to

[1]Our codes and datasets can be obtained from `https://github.com/THU-KEG/PKGC`.

	Model	Wiki27K		FB15K-237-N		FB15K-237-NH	
		Acc.	F1	Acc.	F1	Acc.	F1
KGE Models	TransE (Bordes et al., 2013)	65.5/64.2	72.3/71.5	66.2/64.0	71.1/70.4	50.3/49.5	66.2/62.1
	TransC (Lv et al., 2018)	68.7/68.4	71.5/71.2	66.4/64.6	71.3/70.8	51.2/50.4	67.7/64.0
	ConvE (Dettmers et al., 2018)	70.7/68.8	73.5/73.5	67.3/67.3	71.8/73.7	54.6/55.3	67.3/67.1
	WWV (Veira et al., 2019)	69.9/68.0	72.8/72.5	65.2/65.7	70.8/70.1	50.5/49.6	66.8/62.1
	TuckER (Balažević et al., 2019)	70.0/69.5	73.1/73.8	68.3/71.0	71.9/74.3	54.3/55.4	67.4/67.3
	RotatE (Sun et al., 2019)	72.3/64.0	75.1/71.3	67.9/63.2	72.3/69.9	51.7/51.9	66.8/64.8
PLM-based	KG-BERT (Yao et al., 2019)	83.7/82.4	84.3/83.1	71.8/72.7	72.8/73.6	56.4/57.6	63.3/63.6
	LP-RP-RR (Kim et al., 2020)	84.3/83.6	85.1/84.4	73.8/74.4	73.0/74.5	58.3/59.1	65.1/65.7
	PKGC	87.0/87.8	87.1/88.0	79.6/81.4	79.5/81.2	63.8/64.8	68.7/68.7
	PKGC w/ attribute	87.6/87.8	87.5/87.9	79.5/81.2	79.5/81.4	64.1/65.0	68.7/69.6
	PKGC w/ definition	**90.0/90.0**	**90.1/90.2**	**82.5/84.4**	**83.0/84.7**	**65.7/66.9**	**70.5/71.3**

Table 4: Triple classification results on three datasets. The values before and after the slash are the results under CWA and OWA, respectively. All metrics are multiplied by 100. The best score is in **bold**.

bring accurate evaluation results on the link prediction task. This work is only a preliminary discovery of the huge performance difference between KGC models under CWA and OWA. We think that the existing KGC models should be systematically and comprehensively re-evaluated under OWA, and we leave it for future work.

By comparing the results of our model with baseline models, we find that although our model does not have a significant performance advantage under CWA, it significantly outperforms previous models (both KEG and PLM-based models) under OWA. This suggests that the approach of converting triples into sentences in our model can make better use of the implicit knowledge in PLMs. For our model, adding support information can achieve performance improvements, of which definition brings better obvious improvement. The possible reason is that the definition is unique and does not need to be randomly selected like attributes. Therefore, it introduces less noise and is more accurate.

5.5 Triple Classification Results

In Table 4, we give the experimental results of all models on the triple classification task. Specifically, we give the results under both CWA and OWA. By comparing the performance of the model under CWA and OWA side-by-side, we can find that most models have a small performance gap. This is probably explained by the small proportion of false negative triples in the triple classification task. Specifically, in triple classification, there are about 5% of false negative triples on average, and for the Hits@1 of the link prediction, there are on average more than 30% of false negative triples.

From the table, we can know that our model significantly outperforms baseline models under both assumptions. Specifically, compared to the KGE models, both our model and other PLM-based KGC models achieve better results, which indicates that the introduction of PLMs can help the model to better determine whether the triple is correct or not. There may also be a reason that the PLM-based KGC models are trained using the classification loss and may be better suited for the triple classification task. Comparing all variants of our model, the definition brings better results, which is consistent with the performance in link prediction, and the reasons should be similar.

5.6 Analysis

In order to further analyze what benefits the PLM can bring to our model and why it can bring these benefits, we conduct some triple classification experiments under OWA. Our analysis can be divided into the following three questions. Due to space limitations, we put more analysis in the Appendix.

Q1: PLMs have seen many facts in the massive texts. Is it because they remember these facts to help our model achieve better results?

A1: Partially yes. It is worth noting that it is non-trivial to answer this question rigorously. Therefore, we do an approximate experiment based on distant supervision. Specifically, for a triple (h, r, t), if h and t appear in a sentence in Wikipedia [2], we consider this sentence to imply the fact of (h, r, t). In our experiments, for each triple (h, r, t) in the validation and test set, we count the number of sentences in Wikipedia that contain both h and t. For BERT, which is mainly pre-trained on Wikipedia texts, we can assume that the number of

[2] https://www.wikipedia.org/

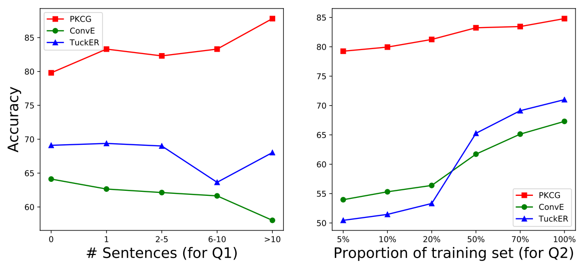

Figure 4: The experimental results on FB15K-237-N corresponding to Q1 (left) and Q2 (right). The horizontal coordinates of the left and right figures are the number of sentences corresponding to the triples in the test set and the proportion of the training set used for training, respectively.

sentences corresponding to the triple can represent the number of times BERT has seen this fact.

In experiments, we divide the test set into several disjoint parts based on the number of sentences corresponding to each triple and obtain the performance of our model (with BERT-Large), ConvE and TuckER on them. The experimental result is shown in the left part of Figure 4. From the figure, we know that there is an increase in the performance of our model as the number of sentences corresponding to the triple grows, while ConvE and TuckER are essentially constant or slightly decreasing. This indicates that our model does perform better on the triples that PLMs have seen more times. In addition, it is worth noting that even on the test set with zero relevant sentence, our model still outperforms both KGE models, which indicates that our model also has the ability to reason and can fuse the knowledge from PLMs and KGs to infer new knowledge.

Q2: Can the introduced PLMs make our models less sensitive to the amount of training data?

A2: Yes. Unlike KGE models that require training all entity and relation vectors from scratch, our model is based on PLMs that have been well pre-trained. Therefore, we conjecture that our model is insensitive to the amount of training data. To validate it, we train models using different proportions of the training set and get the performance.

The experimental results are shown in the right part of Figure 4. From the figure, we can see that the performance of our model only decreases slightly as the amount of data used for training decreases. As a comparison, the performance of both KGE models, ConvE and TuckER, decreases significantly. This indicates that our model is less sensitive to the amount of training data compared

Model	Wiki27K		FB15K-237-N	
	Acc.	F1	Acc.	F1
PKGC (w/ BERT-base)	**86.2**	**86.4**	80.9	80.7
PKGC (w/ RoBERTa-base)	85.5	85.7	77.0	78.5
PKGC (w/ KEPLER)	85.7	85.9	77.3	78.8
PKGC (w/ LUKE-base)	86.1	86.3	**82.0**	**82.7**

Table 5: Triple classification results with different pre-trained language models.

to the KGE models and has the potential to be used for sparse knowledge graph completion.

Q3: In recent years there have been some PLMs containing knowledge. Can using them give better results for our model?

A3: Partially yes. We compare the performance of our model using different PLMs. Specifically, we choose two PLMs containing knowledge, KEPLER (Wang et al., 2021) and LUKE (Yamada et al., 2020). KEPLER uses the RoBERTa-base architecture and jointly optimizes the knowledge embedding and language modeling objectives. LUKE continued to pre-train on the Wikipedia corpus with 200K steps based on RoBERTa. For a fair comparison, we choose the base version for every PLM.

We conduct experiments on FB15K-237-N and Wiki27K. The experimental results are shown in Table 5. As we can see from the table, compared to RoBERTa-base, LUKE-base can bring more performance gains than KEPLER. This is probably because LUKE needs to specify the label and position of the entity in the input, which makes it easier to use the entity information in the PLM. However, both LUKE-base and KEPLER perform worse than BERT-base on Wiki27K. One possible reason is that these two PLMs are trained on RoBERTa instead of BERT. And from the table, we can see that BERT-base performs better than RoBERTa-base. A similar phenomenon is reported by Shin et al. (2020). The possible reason is that BERT is mainly trained on Wikipedia corpus and contains more factual knowledge.

6 Conclusion and Future Work

With the rapid development of pre-trained language models, some PLM-based KGC models are proposed. However, there is still a performance gap between these models and SOTA KGE models. In this work, we find two main reasons for the weak performance: (1) Inaccurate evaluation setting. (2) Inappropriate utilization of PLMs. To alleviate these problems, we highlight a more accurate evaluation setting OWA and propose a novel PLM-based KGC model. In our experiments, we verify that CWA cannot bring accurate evaluation results. Moreover, the experimental results show that our model can achieve better results than the previous method. In our future work, we plan to comprehensively and systematically re-evaluate the existing KGC models to reveal their real performance.

Acknowledgments

This work is supported by the National Key Research and Development Program of China (2020AAA0106501), the grants from the Institute for Guo Qiang, Tsinghua University (2019GQB0003) and Beijing Academy of Artificial Intelligence, and the NSFC Youth Project (62006136). Xin Lv is supported by 2021 Tencent Rhino-Bird Elite Training Program. Xin Lv proposed the main idea of the paper and wrote codes. Yankai Lin and Yixin Cao participated in most of the discussions and made major revisions to the paper. Lei Hou, Juanzi Li, Zhiyuan Liu and Peng Li participated in some of the discussions and made minor revisions to the paper. Jie Zhou performed a proof reading before submission.

References

Farahnaz Akrami, Mohammed Samiul Saeef, Qingheng Zhang, Wei Hu, and Chengkai Li. 2020. Realistic re-evaluation of knowledge graph completion methods: An experimental study. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, pages 1995–2010.

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.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *NeurIPS*.

Yixin Cao, Xiang Ji, Xin Lv, Juanzi Li, Yonggang Wen, and Hanwang Zhang. 2021. Are missing links predictable? an inferential benchmark for knowledge graph completion. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 6855–6865.

Wanyun Cui, Yanghua Xiao, Haixun Wang, Yangqiu Song, Seung-won Hwang, and Wei Wang. 2017. Kbqa: learning question answering over qa corpora and knowledge bases. *Proceedings of the VLDB Endowment*, 10(5):565–576.

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

Xu Han, Shulin Cao, Xin Lv, Yankai Lin, Zhiyuan Liu, Maosong Sun, and Juanzi Li. 2018. Openke: An open toolkit for knowledge embedding. In *Proceedings of the 2018 conference on empirical methods in natural language processing: system demonstrations*, pages 139–144.

Xu Han, Weilin Zhao, Ning Ding, Zhiyuan Liu, and Maosong Sun. 2021. Ptr: Prompt tuning with rules for text classification. *arXiv preprint arXiv:2105.11259*.

Shaoxiong Ji, Shirui Pan, Erik Cambria, Pekka Marttinen, and S Yu Philip. 2021. A survey on knowledge graphs: Representation, acquisition, and applications. *IEEE Transactions on Neural Networks and Learning Systems*.

Bosung Kim, Taesuk Hong, Youngjoong Ko, and Jungyun Seo. 2020. Multi-task learning for knowledge graph completion with pre-trained language models. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 1737–1743.

Yankai Lin, Zhiyuan Liu, and Maosong Sun. 2016. Knowledge representation learning with entities, attributes and relations. In *IJCAI*, pages 2866–2872.

Xiao Liu, Yanan Zheng, Zhengxiao Du, Ming Ding, Yujie Qian, Zhilin Yang, and Jie Tang. 2021. Gpt understands, too. *arXiv preprint arXiv:2103.10385*.

Xin Lv, Lei Hou, Juanzi Li, and Zhiyuan Liu. 2018. Differentiating concepts and instances for knowledge graph embedding. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1971–1979.

Dai Quoc Nguyen, Tu Dinh Nguyen, Dat Quoc Nguyen, and Dinh Phung. 2017. A novel embedding model for knowledge base completion based on convolutional neural network. *arXiv preprint arXiv:1712.02121*.

Maximilian Nickel, Lorenzo Rosasco, and Tomaso Poggio. 2016. Holographic embeddings of knowledge graphs. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, pages 1955–1961.

Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander Miller. 2019. Language models as knowledge bases? 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 2463–2473.

Tara Safavi and Danai Koutra. 2020. Codex: A comprehensive knowledge graph completion benchmark. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 8328–8350.

Tara Safavi, Danai Koutra, and Edgar Meij. 2020. Evaluating the calibration of knowledge graph embeddings for trustworthy link prediction. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 8308–8321.

Taylor Shin, Yasaman Razeghi, Robert L Logan IV, Eric Wallace, and Sameer Singh. 2020. Autoprompt: Eliciting knowledge from language models with automatically generated prompts. *arXiv preprint arXiv:2010.15980*.

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

Partha Pratim Talukdar et al. 2021. Okgit: Open knowledge graph link prediction with implicit types. *arXiv preprint arXiv:2106.12806*.

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.

Neil Veira, Brian Keng, Kanchana Padmanabhan, and Andreas Veneris. 2019. Unsupervised embedding enhancements of knowledge graphs using textual associations. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, pages 5218–5225. International Joint Conferences on Artificial Intelligence Organization.

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

Ruobing Xie, Zhiyuan Liu, Jia Jia, Huanbo Luan, and Maosong Sun. 2016. Representation learning of knowledge graphs with entity descriptions. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, pages 2659–2665.

Ikuya Yamada, Akari Asai, Hiroyuki Shindo, Hideaki Takeda, and Yuji Matsumoto. 2020. Luke: deep contextualized entity representations with entity-aware self-attention. *arXiv preprint arXiv:2010.01057*.

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

Zexuan Zhong, Dan Friedman, and Danqi Chen. 2021. Factual probing is [mask]: Learning vs. learning to recall. *arXiv preprint arXiv:2104.05240*.

Hao Zhou, Tom Young, Minlie Huang, Haizhou Zhao, Jingfang Xu, and Xiaoyan Zhu. 2018. Commonsense knowledge aware conversation generation with graph attention. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, pages 4623–4629.

A Dataset Construction

Our Wiki27K is built on Wikidata [3]. The detailed steps for building Wiki27K are:

(1) For all entities in Wikidata, we score the entity in four areas: frequency of the entity, whether the entity has English Wikipedia links, whether the entity has non-English Wikipedia links, and whether the entity has Freebase links. We normalize the entity frequencies of all entities to a continuous value from 0 to 1. For the last three metrics, the score is 1 if the corresponding link is present; otherwise, it is 0. The final score for each entity is obtained by summing the above four items. We randomly select 27,122 entities among the top 30,000 entities in the score ranking to form our entity set $\mathcal{E}$.

(2) For each relation r in Wikidata, we define its frequency as the size of corresponding triple set, i.e., $|\{(h,r,t)|h \in \mathcal{E} \wedge t \in \mathcal{E}\}|$. We sort the relations in descending order by their frequency and select the top 200 relations to form the set of relations $\mathcal{R}_w$. Besides, we also use the set of relations from CoDEx (Safavi and Koutra, 2020) and LAMA (Petroni et al., 2019), denoted as $\mathcal{R}_c$ and $\mathcal{R}_l$, respectively. The final relation set is defined as $\mathcal{R} = \mathcal{R}_w \cap (\mathcal{R}_c \cup \mathcal{R}_l)$.

(3) We select (h,r,t) whose $h, r \in \mathcal{E}$ and $r \in \mathcal{R}$ from Wikidata to form our triple set $\mathcal{T}$.

(4) We randomly shuffle the triple set and compose the training/validation/test set at a ratio of 8:1:1.

(5) There exists some symmetry relation r in $\mathcal{R}$, i.e., if (h,r,t) holds, then (t,r,h) also holds. If (h,r,t) is present in the training set and (t,r,h) exists in the validation set or the test set, the model is able to make predictions easily. To avoid this information leakage and to make the dataset more difficult, inspired by FB15K-237 (Toutanova et al., 2015), for each symmetric relation r, we remove (h,r,t) from the training set if (t,r,h) is in the validation or test set. In our dataset, the symmetric relations being processed include *shares border with* and *twinned administrative body*.

B Recall and Re-ranking Framework

For a triple query $(h,r,?)$ in link prediction, the KGC models need to replace the tail entity with each entity in the entity set and then calculate the score. After that, the model can give the ranking of the tail entity according to the score ranking. Therefore, link prediction requires a large amount of computation. For the traditional KGE models, most of them run efficiently and can complete the evaluation quickly. However, due to the introduction of PLMs, PLM-based models run much less efficient compared to the KGE models. This performance inefficiency can greatly increase the evaluation time of link prediction. Take KG-BERT as an example, it takes nearly one month to get the evaluation result of link prediction on a dataset, which is obviously unacceptable.

To alleviate this problem, in this work, we use a recall and re-ranking framework. Specifically, for a triplet query $(h,r,?)$, we first use a KGE model (TuckER is used in experiments) to get the ranking of the tail entities. After that, we select the top X ranked entities and use a PLM-based KGC model to recalculate the scores. Based on these scores, we can re-rank the top X entities.

[3]We use the 20210414 snapshot of Wikidata.

Model	Wiki27K		FB15K-237-N	
	Acc.	F1	Acc.	F1
KG-BERT	82.4	83.1	72.7	73.6
LP-RP-RR	83.6	84.4	74.4	74.5
PKGC (w/o soft prompts)	87.1	87.2	80.5	80.6
PKGC	**87.8**	**88.0**	**81.4**	**81.2**

Table 6: Ablation study on soft prompts. PKGC (w/o soft prompts) denotes our model without soft prompts, i.e., the hyper-parameter $n = 0$.

C Ablation Study on Soft Prompts

We do an ablation study to verify the effectiveness of soft prompts, and the experimental results are shown in Table 6. For PKCG, we set the hyper-parameter n to 6, i.e., $n1, ..., n6$ are all 1 (refer to Table 9). From Table 6, we can know that our model without soft prompts has a small drop in performance. However, it still performs better than the previous PLM-based KGC models. This shows that soft prompts can indeed enhance the expressiveness of triple prompts. Besides, even without soft prompts, our model is able to utilize the implicit knowledge in PLMs better than the previous model.

D Analysis on Relation

We provide in Table 7 the five relations that are most affected by the CWA on the Wiki27K and FB15K-237-N, respectively. In other words, they are the five relations with the highest number of

	Wiki27K	FB15K-237-N
1	*country*	*/people/person/profession*
2	*headquarters location*	*/location/location/contains*
3	*diplomatic relation*	*/education/educational_institution/school_type*
4	*located in the administrative territorial entity*	*/people/person/nationality*
5	*time period*	*/olympics/olympic_games/participating_countries*

Table 7: The five relations that are most affected by the CWA on two datasets.

Dataset	α	K	X	n	$n_1, n_2, n_3, n_4, n_5, n_6$
Wiki27K	0.5	30	30	6	1, 1, 1, 1, 1, 1
FB15K-237-N	0.5	30	30	6	1, 1, 1, 1, 1, 1
FB15K-237-NH	0.5	30	30	6	1, 1, 1, 1, 1, 1

Table 8: The best hyper-parameters on different datasets.

false negative triples. From the table, we can see that most of the relations that are strongly influenced by CWA are 1-N, N-1 or N-N relations. For these relations, it is difficult for the knowledge graph to cover all the correct tail or head entities, which results in incomplete knowledge. For example, for a head entity England and a relation */location/location/contains*, many entities can be used as correct tail entities, because England contains a large number of geographic locations. However, it is difficult for the existing knowledge graph to cover all the correct entities, which makes this kind of relation more influenced by CWA.

E Manual Annotation

For CR@1 in link prediction, we randomly sample 200 triples from the test set for evaluation. After that, we get the triples with the highest prediction probability for each model and merge them into a triple set by breaking them up. By doing so, the annotators do not know which model the triples originate from at the time of annotation, which ensures fairness. For triple classification, we ensure that the distribution of relations (for negative triples) in the validation/test set are consistent with that in the training set.

In the specific annotation, we invited three college students to determine the correctness of each triple, and only the triples with the same opinion will be directly retained. The rest triples need to be discussed to determine and then get a unified opinion.

n	$n_1, n_2, n_3, n_4, n_5, n_6$
0	0, 0, 0, 0, 0, 0
1	1, 0, 0, 0, 0, 0
	0, 1, 0, 0, 0, 0
	0, 0, 1, 0, 0, 0
	0, 0, 0, 1, 0, 0
	0, 0, 0, 0, 1, 0
	0, 0, 0, 0, 0, 1
2	1, 1, 0, 0, 0, 0
	0, 0, 1, 1, 0, 0
	0, 0, 0, 0, 1, 1
3	1, 1, 1, 0, 0, 0
	0, 0, 0, 1, 1, 1
6	1, 1, 1, 1, 1, 1

Table 9: The combinations of $n_1, n_2, \cdots n_6$ for every n.

F Hyper-parameters Selection

As introduced in Section 5.3, the parameter n is selected from $\{0, 1, 2, 3, 6\}$, where n is composed of $n_1, n_2, \cdots n_6$. For every n, we choose some combinations of $n_1, n_2, \cdots n_6$ to form it. We detail the combinations in Table 9. For the parameter X mentioned above, we select it from $\{20, 30, 50, 100, 200\}$. We select the best hyper-parameters using the Hits@1 metric and the final results are shown in Table 8.