Knowledge Context Modeling with Pre-trained Language Models for Contrastive Knowledge Graph Completion

Guangqian Yang[1], **Yi Liu**[2], **Lei Zhang**[1*], **Licheng Zhang**[1]
Hongtao Xie[1], **Zhendong Mao**[1]
[1]University of Science and Technology of China, Hefei, China
[2]People's Daily Online, Beijing, China
yanggq@mail.ustc.edu.cn, gavin1332@gmail.com, leizh23@ustc.edu.cn
zlczlc@mail.ustc.edu.cn, htxie@ustc.edu.cn, zdmao@ustc.edu.cn

Abstract

Text-based knowledge graph completion (KGC) methods utilize pre-trained language models for triple encoding and further fine-tune the model to achieve completion. Despite their excellent performance, they neglect the knowledge context in inferring process. Intuitively, knowledge contexts, which refer to the neighboring triples around the target triples, are important information for triple inferring, since they provide additional detailed information about the entities. To this end, we propose a novel framework named KnowC, which models the knowledge context as additional prompts with pre-trained language models for knowledge graph completion. Given the substantial number of neighbors typically associated with entities, along with the constrained input token capacity of language models, we further devise several strategies to sample the neighbors. We conduct extensive experiments on common datasets FB15k-237, WN18RR and Wikidata5M, experiments show that KnowC achieves state-of-the-art performance.

1 Introduction

Knowledge graphs are collections of facts, which are represented as sets of triples (h, r, t), denoting that the head entity h has a r relation with the tail entity t. Knowledge graphs play important roles in practical applications of many fields, such as question answering (Bauer et al., 2018), recommendation system (Zhang et al., 2016), and so on. In real world scenarios, large-scale knowledge graphs are usually incomplete, which hinders their performance in downstream tasks. To solve this problem, researchers propose many knowledge graph completion methods to infer the missing facts.

To complete the missing links in knowledge graph, many researchers propose to embed entities and relations into vectors using representation techniques. In this way, the distribution of entities and relations could be effectively modeled, which further facilitates the completion task. Generally, KGC methods of this kind could be divided into embedding-based and text-based methods. Among them, embedding-based methods aim to embed the entities and relations into low-dimensional vectors by the structure of the knowledge graph, such as TransE (Bordes et al., 2013), ComplEx (Trouillon et al., 2016), TuckER (Balazevic et al., 2019) etc. On the other hand, text-based methods exploit the available text such as name or description of the entities and relations for inference (Xie et al., 2016). Recently, benefit from the development of pre-trained language models, text-based methods achieve excellent performance in KGC task (Yao et al., 2019; Wang et al., 2022).

However, existing text-based KGC methods mainly focus on the triple text itself for inferring, while neglect the knowledge context of the target triple, where knowledge context refers to the neighboring triples around the target entities. Here the target entity denotes the given entity for relation prediction, and the neighboring triples denote triples directly connected to the target entity. Intuitively, these knowledge contexts are important information for the inference of language model, as they describe detailed supplemental information about the target entities. For example, suppose we need to infer the missing triple (*James Harden, Nationality, ?*). We could only use the knowledge implicitly inherited in the language model if we have no additional information. In contrast, with access of knowledge context (*James Harden, Borns In, Los Angeles*) as additional information, it will be much easier for language models to get the right answer.

To this end, we propose a novel framework named KnowC, which models the knowledge context as additional prompts with pre-trained language models for knowledge graph completion.

*Corresponding author.

Findings of the Association for Computational Linguistics ACL 2024, pages 8619–8630

Specifically, for a target triple to be predicted, we sample the neighbors around the target entities as knowledge context. We then use these knowledge contexts as supplemental prompts, which are concatenated together with targets and fed into the language models for encoding. Consequently, the entities and relationships used for prediction are incorporated with more supplementary information related to them, thereby enhancing the feasibility of arriving at accurate answers.

Despite this, there might be hundreds of neighbors for a target entity in the knowledge graph, whereas the language model has a limit on the number of input tokens. To solve this problem, we introduce several strategies for neighbor sampling: random sampling, kNN sampling, and dynamic sampling. Random sampling randomly samples neighbors connected to the target entity, which consumes least computation resource. kNN sampling uses the target relationship as query and neighbors relationships as keys to model the relevance between query-key pairs and select the most relevant k neighbors (Khandelwal et al., 2019). Dynamic sampling dynamically samples k neighbors from the neighbor set in each epoch to mitigate the overfitting risk to specific neighbor configurations.

After getting the text embeddings of entities and relations, we fine-tune the language model for inference. We optimize the model with InfoNCE contrastive loss, following recent state-of-the-art researches (Wang et al., 2022). To evaluate the effectiveness of KnowC, we conduct extensive experiments on common datasets: WN18RR, FB15k-237, and Wikidata5M, and compare its MRR and Hits@k metrics with several benchmarks. Experimental results show that KnowC gets excellent performance.

Our main contributions can be summarized as follows:

- We propose a novel framework that models the knowledge context of target entities as additional prompts with pre-trained language models for knowledge graph completion.
- We further introduce several strategies for neighbor sampling to deal with the input token number limitation problem, where sampling strategies include random sampling, kNN sampling, and dynamic sampling.
- We conduct extensive experiments on common datasets: WN18RR, FB15k-237, and Wikidata5M to evaluate the effectiveness of KnowC, results show that KnowC gets excellent performance.

2 Related Work

2.1 Pre-trained Language Models

Benefit from the success of Transformer (Vaswani et al., 2017) architecture, recently a diverse array of Transformer-based pre-trained language models have emerged, each contributing to the rapid progress in natural language processing. Among them, BERT (Devlin et al., 2019) introduces bidirectional context by pre-training on masked language modeling tasks, which helps it to understand intricate language context. GPT (Radford et al., 2018), on the other hand, focuses on the autoregressive generation pre-training with large number of parameters, enabling it to produce creative generated texts. T5 (Raffel et al., 2020) unifies text-to-text tasks by casting various problems into a uniform format, exhibiting its versatility in naturalanguage processing. XLNet (Yang et al., 2019b) employs permutation language modeling to capture bidirectionality without compromising on autoregressive synthesis. RoBERTa (Liu et al., 2019) refines BERT's training methodology to enhance performance, while ALBERT (Lan et al., 2020) introduces parameter-sharing techniques for efficiency. There are also many other state-of-the-art pre-trained language models that focus on limited resource scenerios (Sanh et al., 2019; Sun et al., 2020).

2.2 Knowledge graph completion with PLMs

To leverage the remarkble representation capability of these pre-trained language models, many researchers propose to apply them for knowledge graph completion task. Generally, these 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) first uses PLMs to perform KGC, which simply splices the labels of entities and relations in triples as the input to PLMs. Based on KG-BERT, MTL-KGC (Kim et al., 2020) further introduces multi-task learning to learn more relational properties and lexical similarities in KGs. PKGC (Lv et al., 2022) converts triples into natural prompt sentences to mitigate the gap between structured knowledge and natural language, and further introduces soft prompts to better express the semantics

of triples. SimKGC (Wang et al., 2022) proposes several negatives to improve the contrastive learning efficiency, and changes the loss function to InfoNCE to make the model focus on hard negatives. GHN (Qiao et al., 2023) further leverages a sequence-to-sequence architecture to generate high-quality hard negative samples.

2.3 Contrastive Learning

Contrastive Learning effectively enhances representation learning by emphasizing disparities between positives and negatives. In the realm of Natural Language Processing, contrastive learning techniques generally aim to capitalize the difference between sentence pairs from text data in different tasks (Gao et al., 2021; Wang et al., 2021b; Ni et al., 2022), surpassing many non-contrastive methods on semantic similarity evaluation. Contrastive learning is also used to improve the efficacy of dense passage retrieval for open-domain question answering (Karpukhin et al., 2020; Qu et al., 2021; Xiong et al., 2020), where positive passages encapsulate those housing accurate answers. In knowledge graph completion field, contrastive learning is widely used for knowledge graph completion, in which typical methods focus on hard negative mining for better triple representation (Wang et al., 2022; Qiao et al., 2023).

3 Method

3.1 Preliminaries

A Knowledge Graph $\mathcal{G}$ comprises a collection of triples $\mathcal{T} \subseteq \mathcal{E} \times \mathcal{R} \times \mathcal{E}$, where $\mathcal{E}$ signifies a set of entities, and $\mathcal{R}$ signifies a set of relations. Conceptually, a knowledge graph can be visualized as a directed graph, where vertices correspond to entities, and directed edges connecting vertices signify relationships between the corresponding entities. A triple (h, r, t) encodes the presence of a relation r from the head entity h to the tail entity t, where $h, t \in \mathcal{E}$ and $r \in \mathcal{R}$. The n-hop neighboring triples of a target entity h is defined as $\mathcal{E}_n(h)$, where $\mathcal{E}_n(\cdot)$ is the set of n-hop neighbors.

In real-world scenarios, knowledge graphs are always incomplete, leaving a multitude of unknown triples. To address this, knowledge graph completion task aims to identify these missing relationships given an incomplete knowledge graph. Following the prevalent entity ranking evaluation framework, the prediction of tail entities $(h, r, ?)$ needs to rank all entities given a head-relation pair (h, r), and the head entity prediction $(?, r, t)$ is similar. In this study, we follow the settings of (Malaviya et al., 2020), where we construct a inverse counterpart (t, r^{-1}, h) for each triple (h, r, t), with r^{-1} representing the inverse relation of r. This reformulation greatly simplifies the task, making it to focus solely on tail entity prediction while effectively accounting for both directions.

3.2 Model Architecture

The overall framework of KnowC is illustrated in Figure 1. Following SimKGC (Wang et al., 2022), the architecture of our model adopts a bi-encoder design. To be specific, we instantiate two distinct BERT encoders with the same pre-trained language model, where the first encoder BERT_{hr} is used to encode the head-relation pair (h, r), and the second encoder BERT_t is used to encode the tail t. After initialization, the parameters of these two encoders are updated separately.

For a triple (h, r, t), the first encoder BERT_{hr} generates the joint embedding of the head entity h and the relation r. To achieve this, we concatenate the textual descriptions of entity h and relation r as the input of the encoder. Specifically, to model the knowledge context around the entity, we use the neighboring triples connected to the entity as additional prompts for complement. Suppose there are k neighboring triples directly connected to the entity h, we explicitly use these triples by concatenating them in the form of $h' = h : r_1, t_1; r_2, t_2; \cdots; r_k, t_k$. In this way, we get more knowledge triples related to this entity, which serve as additional prompts that describe the supplemental information of h. By incorporating these neighboring triples as additional prompts, we expect our model to obtain a more comprehensive understanding of the entity's semantic within the broader knowledge graph, which improves the model's capacity for contextualized knowledge representation, potentially contributing to the learning of interdependencies between relations. We then introduce a special [SEP] symbol between the head description h' and the relation r. Subsequently, the concatenated texts are tokenized and further fed into BERT_{hr} for encoding. Rather than directly utilizing the first token's hidden state of the last layer, we use mean pooling followed by L_2 normalization to get the embedding $\mathbf{e}_{hr}$, since it has shown better performance (Reimers and Gurevych, 2019). By merging the entity h and relation r, the encoder BERT_{hr} generates a relation-aware embedding for

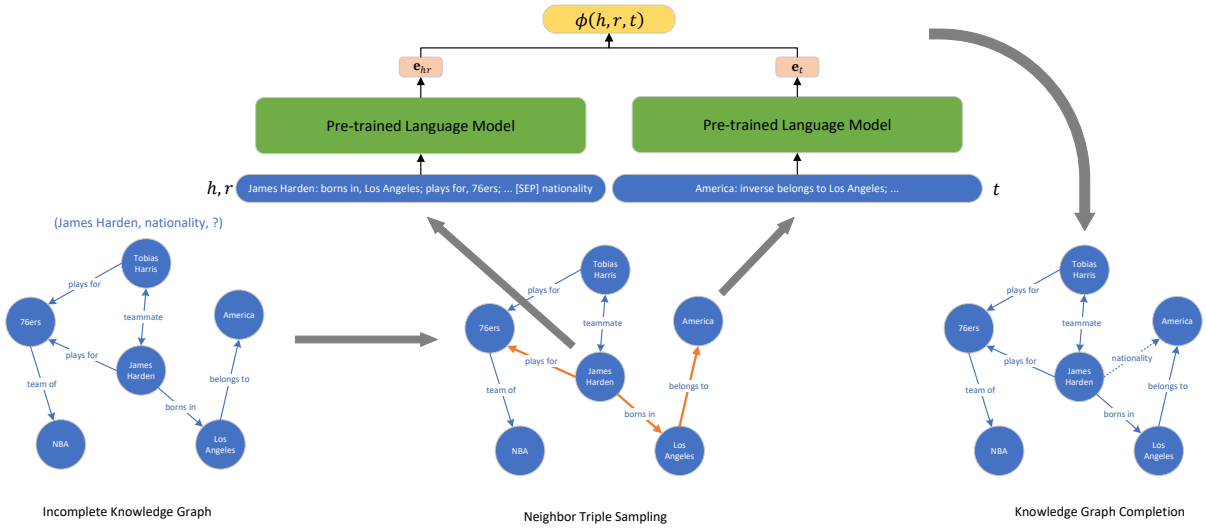

Figure 1: Illustration of the KnowC framework for knowledge graph completion. Given an triple (*James Harden, nationality, ?*) to be predicted, our model first sample the neighbors around the head entity "*James Harden*" and the candidate tail entity "*America*", incorporating them into the original description of these entities. We then feed them into pre-trained language models and encode them into embdddings. Finally, the model predicts the score of each triple and select the triple with largest score as prediction.

the head entity.

The tail entity t is encoded in the similar way with the head entity h. That is, we also concatenate the knowledge contexts of t in the form of $t' = t : r_1, h_1; r_2, h_2; \cdots; r_k, h_k$. The tail texts t' are then fed into BERT_t to compute the embedding $\mathbf{e}_t$ for the tail entity t.

3.3 Neighbor Sampling

As shown in previous part, in KnowC we need to model the knowledge contexts, namely the neighboring triples, as additional prompts to facilitate the completion task. In this paper, we use 1-hop neighboring triples, i.e., triples directly connected to the target entities as implementation of knowledge context. In practical knowledge graphs, however, entities may have a large amount of neighboring triples, while the pre-trained language models have limited input token number. This requires us to sample neighbors so as to restrict the token number as well as the computation complexity. For this purpose, we sample the neighboring triples within a threshold k, and propose several strategies for neighbor sampling.

3.3.1 Random Sampling.

We sample k neighbors at the beginning of the training process, and keep the neighbor set fixed throughout epochs. The fixed neighbor set could ensure that the relationships established between entities remain consistent throughout the iterative learning process. In this way, we aim to provide the model with a stable knowledge context, thereby improves the stability and robustness of the training process.

3.3.2 kNN Sampling.

We use PLMs to measure the similarities between relations by embedding the descriptions of relations into embedding and calculating a cosine similarity matrix for them. Here we neglect the entities for similarity calculation, for incorporating entities together with relation involves a combinational search, leading to exponential computational complexity. We use this precomputed similarity matrix to construct a relation similarity dictionary, and select k nearest neighbors by order of this dictionary from the neighbor set. By this similarity calculation, we expect the target entity and relationship to find the neighboring triples that have the most relevant semantics for inference. In this way, the model could focus on neighboring triples that are more important for the prediction despite the large neighbor set scale.

3.3.3 Dynamic Sampling.

In each epoch, we dynamically sample k neighbors from the neighbor set. This dynamic sampling process is executed in a random fashion, imparting a distinctive neighbor set within each individual epoch. This dynamic strategy endows our training process with more adaptability and diversity. And

	Entities	Relations	Train	Valid	Test
WN18RR	40,943	11	86,835	3,034	3,134
FB15k-237	14,541	237	272,115	17,535	20,466
Wikidata5M-Trans	4,594,485	822	20,614,279	5,163	5,163
Wikidata5M-Ind	4,579,609	822	20,496,514	6,699	6,894

Table 1: Statistics of the datasets used in this paper. "Wikidata5M-Trans" and "Wikidata5M-Ind" refer to the transductive and inductive settings, respectively.

by introducing randomness, we effectively mitigate the overfitting risk to specific neighbor configurations, fostering more understanding of the underlying relationships within the knowledge graph. As a result, we expect the model to discern meaningful patterns and extracting valuable information from different knowledge contexts, which contributes to its capacity for generalization.

3.4 Training and Inference

Following (Yang et al., 2019a; Chen et al., 2020), we use InfoNCE loss with additive margin during training:

$$\mathcal{L} = -\log \frac{e^{(\phi(h,r,t)-\gamma)/\tau}}{e^{(\phi(h,r,t)-\gamma)/\tau} + \sum_{i=1}^{|\mathcal{N}|} e^{\phi(h,r,t_i')/\tau}} \tag{1}$$

Here $\phi(h,r,t)$ is the function that calculates the score for a candidate triple, which is defined as the cosine similarity, i.e. $\phi(h,r,t) = \cos(\mathbf{e}_{hr}, \mathbf{e}_t) \in [-1,1]$. (h,r,t') are negative samples, here we use three kinds of negatives samples, including In-batch Negatives, Pre-batch Negatives and Self-Negatives introduced in SimKGC (Wang et al., 2022). The additive margin $\gamma > 0$ encourages the model to assign larger scores for positive triples (h,r,t), which explicitly improves the separation between positive and negative embeddings. The temperature τ could scale the loss function, and an appropriate temperature could help the model to focus on hard negatives.

As stated previously, we only need to predict the tail entity given head h and relation r. For tail entity prediction $(h,r,?)$, we compute the similarity between the head-relation embedding $\mathbf{e}_{hr}$ and the tail embedding $\mathbf{e}_t$ to select the entity with largest score as prediction:

$$\arg\max_{t_i} \cos(\mathbf{e}_{hr}, \mathbf{e}_{t_i}), t_i \in \mathcal{E} \tag{2}$$

As spatial locality is a common characteristic observed in knowledge graphs, in which entities located closely tend to exhibit stronger relationships compared to those farther apart, we also use Graph-based Re-ranking (Wang et al., 2022). That is, we explicitly increase the score of n-hop neighbors by $\alpha \geq 0$ in prediction:

$$\arg\max_{t_i} \cos(\mathbf{e}_{hr}, \mathbf{e}_{t_i}) + \alpha \mathbb{1}(t_i \in \mathcal{E}_n(h)) \tag{3}$$

where $\mathcal{E}_n(h)$ is the set of n-hop neighbors of the entity h.

4 Experiments

4.1 Experiment Setup

4.1.1 Datasets

We utilize three distinct datasets for our evaluation process, namely WN18RR (Dettmers et al., 2018), FB15k-237 (Toutanova et al., 2015), and Wikidata5M (Wang et al., 2021c). The detailed statistics are comprehensively presented in Table 1. Compared to WN18RR and FB15k-237 datasets, the Wikidata5M dataset has a significantly larger scale. The Wikidata5M dataset offers two distinctive settings: transductive and inductive, denoted as "Wikidata5M-Trans" and "Wikidata5M-Ind" respectively. In the transductive setting, all entities present in the test set are also presented within the training set. While in the inductive setting, there is no entity overlap between the training and test sets. With regards to textual descriptions, for the WN18RR and FB15k-237 datasets, we leverage the textual data provided by KG-BERT (Yao et al., 2019). As for the Wikidata5M dataset, there are inherent comprehensive descriptions for all entities and relations.

4.1.2 Evaluation Metrics

Following previous researches, our Knowledge Graph Completion model is evaluated via the entity ranking task. Specifically, for each test triple (h,r,t), we need to predict the tail entity t given the head entity h and the relation r, and the head entity predicition could be attained by triple reversing as stated in Method section. For evaluation of

Method	WN18RR				FB15k-237			
	MRR	**Hits@1**	**Hits@3**	**Hits@10**	**MRR**	**Hits@1**	**Hits@3**	**Hits@10**
embedding-based methods								
TransE (2013)	24.3	4.3	44.1	53.2	27.9	19.8	37.6	44.1
DistMult (2014)	44.4	41.2	47.0	50.4	28.1	19.9	30.1	44.6
RotatE (2018)	47.6	42.8	49.2	57.1	33.8	24.1	37.5	53.3
TuckER (2019)	47.0	44.3	48.2	52.6	**35.8**	**26.6**	**39.4**	**54.4**
text-based methods								
KG-BERT (2019)	21.6	4.1	30.2	52.4	-	-	-	42.0
MTL-KGC (2020)	33.1	20.3	38.3	59.7	26.7	17.2	29.8	45.8
StAR (2021a)	40.1	24.3	49.1	70.9	29.6	20.5	32.2	48.2
SimKGC (2022)	66.6	58.7	71.7	80.0	33.6	24.9	36.2	51.1
GHN (2023)	67.8	59.6	71.9	82.1	33.9	25.1	36.4	51.8
KnowC$_{rand}$	68.2	60.4	73.1	81.9	34.2	25.4	37.1	51.8
KnowC$_{k\text{NN}}$	68.7	60.5	74.1	83.2	**34.4**	**25.6**	**37.3**	52.2
KnowC$_{dyn}$	**69.9**	**62.3**	**74.9**	**83.9**	34.0	25.0	37.2	**52.3**

Table 2: Experiment results on WN18RR and FB15k-237 dataset. KnowC$_{rand}$, KnowC$_{k\text{NN}}$, and KnowC$_{dyn}$ refer to random sampling, kNN sampling, and dynamic sampling, respectively. The bold represents the best performance.

the prediction performance, we use four automated evaluation metrics following most researchers: the mean reciprocal rank (MRR), as well as Hits@k (H@k), where $k \in \{1, 3, 10\}$. MRR quantifies the average reciprocal rank of test triples, while H@k quantifies the percentage of accurate entity predictions within the top-k ranks. Note that both MRR and H@k are computed under the filtered setting (Bordes et al., 2013), which ignores the scores attributed to all known true triples within the training, validation, and test datasets. To ensure comprehensive analysis, we average the prediction metrics of head entity and tail entity.

4.1.3 Implementation Details

Our experiment setting mainly follows SimKGC (Wang et al., 2022): The encoders are initialized with the bert-base-uncased model. The encoder model could be easily replaced by any other advanced re-trained language models to enhance performance. To ensure a consistent evaluation, most hyperparameters, except for the learning rate and training epochs, remain identical across all datasets, thereby avoiding dataset-specific tuning. In optimization process, we use grid search for the ideal learning rate. For neighboring triple sampling, we tune the sampled neighbor size from 1 to 10. For target entity descriptions and neighboring triples, we impose a truncation limit of 100 tokens. The temperature parameter τ is initialized with 0.05, and the additive margin γ for the InfoNCE loss is set to 0.02. The weight of re-ranking operation during inference is set to 0.05. For optimization, we employ the AdamW optimizer with linear learning rate decay. The batch size for model training is set to 1024, and we train the model using distributed training across 4 NVIDIA A40 GPUs. For the WN18RR, FB15k-237, and Wikidata5M datasets, we train for 50, 10, and 1 epochs, respectively.

4.2 Experiment results

4.2.1 Baselines

We choose both embedding-based methods and text-based methods as baselines for performance comparison. For embedding-based methods, we select four classical methods, including TransE (Bordes et al., 2013), DistMult (Yang et al., 2014), RotatE (Sun et al., 2018) and TuckER (Balazevic et al., 2019). For text-based methods, we select several recent representative methods that use PLMs, including KG-BERT (Yao et al., 2019), MTL-KGC (Kim et al., 2020), KEPLER (Wang et al., 2021c), StAR (Wang et al., 2021a), SimKGC (Wang et al., 2022) and GHN (Qiao et al., 2023). For performance comparison, we directly reuse the numbers reported by SimKGC (Wang et al., 2022) and GHN (Qiao et al., 2023).

4.2.2 Main Results

The experiment results on WN18RR and FB15k-237 datasets are in Table 2, and the results on large scale knowledge graph Wikidata5M are in Table 3. We denote KnowC with three variant sampling strategies by KnowC$_{rand}$, KnowC$_{k\text{NN}}$, and

Method	Wikidata5M-Trans				Wikidata5M-Ind			
	MRR	**Hits@1**	**Hits@3**	**Hits@10**	**MRR**	**Hits@1**	**Hits@3**	**Hits@10**
embedding-based methods								
TransE (2013)	25.3	17.0	31.1	39.2	-	-	-	-
RotatE (2018)	29.0	23.4	32.2	39.0	-	-	-	-
text-based methods								
KEPLER (2021c)	21.0	17.3	22.4	27.7	40.2	22.2	51.4	73.0
SimKGC (2022)	35.8	31.3	37.6	44.1	71.4	60.9	78.5	91.7
GHN (2023)	36.4	31.7	38.0	45.3	-	-	-	-
KnowC$_{rand}$	42.0	37.2	43.8	50.8	73.0	62.9	80.1	92.4
KnowC$_{k\text{NN}}$	**42.6**	**37.3**	**44.7**	**52.5**	**73.2**	**63.1**	**80.3**	**92.5**

Table 3: Experiment results on Wikidata5M with transductive and inductive setttings. KnowC$_{rand}$ and KnowC$_{k\text{NN}}$ refer to random sampling and kNN sampling, respectively. Since we only train one epoch on these two datasets, we omit the dynamic sampling in this table. The bold represents the best performance.

KnowC$_{dyn}$, respectively. Note that we only run one training epoch on Wikidata5M datasets in both transductive and inductive settings, therefore there is no dynamic sampling on these two datasets. As shown in these two tables, our proposed Model KnowC outperforms all the state-of-the-art text-based methods on all datasets. And compared to the embedding-based methods, KnowC gets the best performance on WN18RR, Wikidata5M-Trans and Wikidata5M-Ind datasets, but performs slightly poorer than embedding-based methods. One possible explanation could be that the graph of FB15k-237 dataset is much denser, and contains fewer entities. Due to the limited entity number, text-based methods are difficult to exert their comprehensive semantic understanding ability. On the contrary, embedding-based methods are able to learn complex structures, especially in this transductive setting with small-scale dataset.

We then report the results for three variants of KnowC on datasets of different scale. As Table 2 shows: Generally, KnowC$_{rand}$ always performs worse compared to the other two variants, which is in accordance with expectation since random fixed sample set is more likely to have less semantic-relevant triple prompts that are potentially helpful for target triple prediction. Specifically, the variant KnowC$_{dyn}$ gets the best on WN18RR dataset compared to other two variant, while on FB15k-237 it's the variant KnowC$_{k\text{NN}}$ that performs better. One possible reason could be that the graph for the FB15k-237 dataset is much denser compared to WN18RR, in which the kNN sampling is more helpful since it gives us a heuristic algorithm to find more relevant neighboring triples. Besides, it has been shown that there are unpredictable links based on the available information in the FB15k-237 (Cao et al., 2021). Therefore, adding neighboring triples as knowledge context may not improve the performance on FB15k-237 dataset so much compared to WN18RR.

As Table 3 shows: On the large scale dataset Wikidata5M, KnowC gets the best performance above any other state-of-the-art methods, especially with the transductive setting. On Wikidata5M-Trans dataset, KnowC outperforms the second best method by a large margin, with MRR, Hits@1, Hits@3 and Hits@10 rising for 6.2, 5.6, 6.7 and 7.2, respectively. And on Wikidata5M-Ind dataset, KnowC also obtains the best results on all metrics. Here embedding-based methods are inherently unable to perform inductive KGC. Text-based methods, in contrast, exhibit its superiority in this setting. In inductive setting, text-based methods could still run inference, since the language models could match text descriptions and assign semantics even if there are unseen entities in test set during training.

4.3 Ablation Study

batch size	**MRR**	**Hits@1**	**Hits@3**	**Hits@10**
256	69.1	60.8	74.6	83.3
512	**69.9**	**62.3**	74.6	83.3
1024	**69.9**	**62.3**	**74.9**	**83.9**

Table 4: Experiment results on WN18RR with different batch size.

We conduct experiments to study the effect of batch size, which is shown in Table 4. In this part, we evaluate a diverse range of batch sizes. As shown in this table, larger batch size helps the

triple	(David H. Riddle, child, Matthew Riddle (biblical scholar))
description	David Hunter Riddle (April 14, 1805 – 1888) was the ninth and last president of Jefferson College ...
head neighbors	...
tail neighbors	(Matthew Riddle (biblical scholar), father, David H. Riddle) ...
SimKGC	harriet eaton stanton blatch
KnowC	Matthew Riddle (biblical scholar)
triple	(Ian Underwood, spouse, Ruth Underwood)
description	Ian Robertson Underwood (born May 22, 1939) is a woodwind and keyboards player...
head neighbors	...
tail neighbors	(Ruth Underwood, spouse, Ian Underwood) ...
SimKGC	gail zappa
KnowC	Ruth Underwood
triple	(Slavyanovo, Targovishte Province, located in the administrative territorial entity, targovishte district)
description	Slavyanovo is a village ... located in Popovo Municipality of the Targovishte Province.
head neighbors	(Slavyanovo, Targovishte Province, located in the administrative territorial entity, popovo municipality) ...
tail neighbors	(targovishte district, contains administrative territorial entity, popovo municipality) ...
SimKGC	popovo, bulgaria
KnowC	targovishte district

Table 5: Examples of KnowC prediction results on the test set of the Wikidata5M-Trans dataset.

model to improve its prediction ability to some extent. This could be attributed to the increased negative sample number with larger batch size.

4.4 Analysis

We further choose several KnowC prediction results on the test set of the Wikidata5M-Trans dataset for analysis. Table 5 shows several examples that KnowC successfully predicts while SimKGC fails. We find that the model could to some extent understand the relationship patterns, including inversion, symmetry, and composition.

4.4.1 Inversion

In the first case, the model needs to predict the tail entity given the head entity "*David H. Riddle*" and the relation "*child*". The description of "*David H. Riddle*" is "*David Hunter Riddle (April 14, 1805 – 1888) was the ninth and last president of Jefferson College from 1862 until its union with Washington College to form Washington Jefferson*", which contains no information about the relationship of "*David H. Riddle*". However, by incorporating the neighbor information (*Matthew Riddle (biblical scholar), father, David H. Riddle*), the model successfully predicts the right answer. Besides, this prediction indicates that the model effectively handles the inversion relationship pattern between "*father*" and "*child*".

4.4.2 Symmetry

Similarly, in the second case, the model needs to predict the missing triple (*Ian Underwood, spouse, ?*). These is also no information about his marry information, therefore it's impossible to infer the answer solely based on this description. While in the neighbor sets, there is a triple (*Ruth Underwood, spouse, Ian Underwood*) that denotes Ian Underwood is the spouse of Ruth Underwood, by which the model could easily get the right answer. This also shows that the model could understand that "*spouse*" is an symmetry relationship pattern.

4.4.3 Composition

In the third case, the missing triple to be predicted is (*Slavyanovo, Targovishte Province, located in the administrative territorial entity, ?*). By the description "*located in Popovo Municipality of the Targovishte Province*", SimKGC simply predicts the target as "*popovo, bulgaria*" by word matching. In contrast, KnowC could combine the information of both description and neighboring triples (*Slavyanovo, Targovishte Province, located in the administrative territorial entity, popovo municipality*) and (*targovishte district, contains administrative territorial entity, popovo municipality*), then composes these relationships to get the right anwser "*targovishte district*".

5 Conclusion

In this paper, we propose a novel text-based knowledge graph completion method. We use neighboring triples as supplementary prompts as additional detailed information about the entities to benefit the completion task. We also design several sampling strategies to limit the input token as well as to help the model improve generalization ability. We conduct extensive experiments to evaluate the

excellent performance of our model on knowledge graph completion task.

6 Limitations

Our method exploits the knowledge context in knowledge graphs as information compensation for knowledge graph completion, which inevitably increases the input token length compared to SimKGC. However, Despite the transformers have a quadratic complexity with respect to token length, the inference time with different input token length actually exhibit little difference owing to the parallel computation of GPU for matrix multiplication. Besides, recent studies show that there are better ways to linearize the triples, such as using natural language templates to replace the original relationships, which could be a direction for further researching.

References

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

Lisa Bauer, Yicheng Wang, and Mohit Bansal. 2018. Commonsense for generative multi-hop question answering tasks. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4220–4230.

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.

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.

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.

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

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.

Tianyu Gao, Xingcheng Yao, and Danqi Chen. 2021. Simcse: Simple contrastive learning of sentence embeddings. In *2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021*, pages 6894–6910. Association for Computational Linguistics (ACL).

Vladimir Karpukhin, Barlas Oğuz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen Tau Yih. 2020. Dense passage retrieval for open-domain question answering. In *2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020*, pages 6769–6781. Association for Computational Linguistics (ACL).

Urvashi Khandelwal, Omer Levy, Dan Jurafsky, Luke Zettlemoyer, and Mike Lewis. 2019. Generalization through memorization: Nearest neighbor language models. In *International Conference on Learning Representations*.

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.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2020. Albert: A lite bert for self-supervised learning of language representations. In *International Conference on Learning Representations*.

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 pre-trained models benefit knowledge graph completion? a reliable evaluation and a reasonable approach. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 3570–3581.

Chaitanya Malaviya, Chandra Bhagavatula, Antoine Bosselut, and Yejin Choi. 2020. Commonsense knowledge base completion with structural and semantic context. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 2925–2933.

Jianmo Ni, Gustavo Hernandez Abrego, Noah Constant, Ji Ma, Keith Hall, Daniel Cer, and Yinfei Yang. 2022.

Sentence-t5: Scalable sentence encoders from pretrained text-to-text models. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 1864–1874.

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

Yingqi Qu, Yuchen Ding, Jing Liu, Kai Liu, Ruiyang Ren, Wayne Xin Zhao, Daxiang Dong, Hua Wu, and Haifeng Wang. 2021. Rocketqa: An optimized training approach to dense passage retrieval for open-domain question answering. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5835–5847.

Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. 2018. Improving language understanding by generative pre-training.

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. *The Journal of Machine Learning Research*, 21(1):5485–5551.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. 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 3982–3992.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*.

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*.

Zhiqing Sun, Hongkun Yu, Xiaodan Song, Renjie Liu, Yiming Yang, and Denny Zhou. 2020. Mobilebert: a compact task-agnostic bert for resource-limited devices. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2158–2170.

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

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

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

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

Liang Wang, Wei Zhao, and Jingming Liu. 2021b. Aligning cross-lingual sentence representations with dual momentum contrast. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3807–3815.

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.

Xiaozhi Wang, Tianyu Gao, Zhaocheng Zhu, Zhengyan Zhang, Zhiyuan Liu, Juanzi Li, and Jian Tang. 2021c. 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 AAAI conference on artificial intelligence*, volume 30.

Lee Xiong, Chenyan Xiong, Ye Li, Kwok-Fung Tang, Jialin Liu, Paul N Bennett, Junaid Ahmed, and Arnold Overwijk. 2020. Approximate nearest neighbor negative contrastive learning for dense text retrieval. In *International Conference on Learning Representations*.

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

Yinfei Yang, Gustavo Hernandez Abrego, Steve Yuan, Mandy Guo, Qinlan Shen, Daniel Cer, Yun-Hsuan Sung, Brian Strope, and Ray Kurzweil. 2019a. Improving multilingual sentence embedding using bi-directional dual encoder with additive margin softmax. *arXiv preprint arXiv:1902.08564*.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019b. Xlnet: Generalized autoregressive pretraining for language understanding. *Advances in neural information processing systems*, 32.

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

Fuzheng Zhang, Nicholas Jing Yuan, Defu Lian, Xing Xie, and Wei-Ying Ma. 2016. Collaborative knowledge base embedding for recommender systems. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 353–362.

A Details of Implementation

A.1 Hyperparameters

We thoroughly list the hyperparameters used for each dataset in Table 6.

Hyperparameter	WN18RR	FB15k-237	Wikidata5M
number of GPUs	4	4	4
initial temperature τ	0.05	0.05	0.05
gradient clip	10	10	10
warmup steps	400	400	400
batch size	1024	1024	1024
max token number	100	100	100
re-ranking weight α	0.05	0.05	0.05
re-ranking hop	5	2	2
dropout	0.1	0.1	0.1
weight decay	10^{-4}	10^{-4}	10^{-4}
InfoNCE margin	0.02	0.02	0.02
pooling method	mean	mean	mean
learning rate	5×10^{-5}	10^{-5}	3×10^{-5}

Table 6: Experiment results on Wikidata5M-Trans with different batch size.

A.2 More Implementation Details

For inverse relation r^{-1}, we add a prefix word "inverse" to the text description of relation r. For example, if the text description of relation r is "member of", the inverse relation r^{-1} is "inverse member of".

To sample neighboring triples, We need to construct graph structure for knowledge graphs in experiments. Specifically, for WN18RR and FB15k-237 datasets, we use undirected graph structure; for Wikidata5M-Trans and Wikidata5M-Ind datasets, we use directed graph structure.

B More Experiment Results

B.1 Effect of Context Triple Number

To study the effect of context triples, we conduct experiments for neighboring triples k with different neighbor numbers, in which we use kNN to sample neighbors. We draw a line chart to make the variation trend much clearer, which is shown in Figure 2.

As shown in this figure, the performance of Hits@k improves steadily with the increase of neighbor numbers, which indicates that increasing the neighbor sampling number could potentially

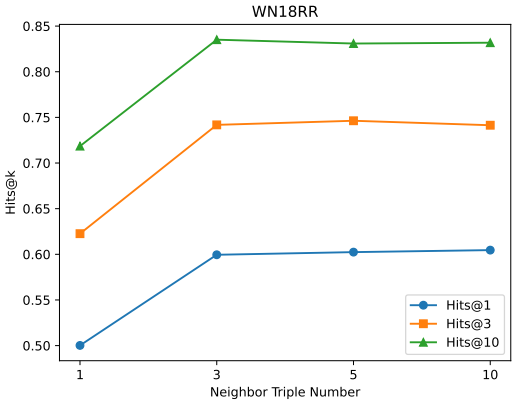

Figure 2: Experiment results on WN18RR with different number of neighboring triples.

increase the performance of the model. We find that the model performs slightly worse with $k = 1$ compared to the model without neighbor sampling. We further check the neighbors used for inference in WN18RR, and find that there are many 1-n relations in WN18RR dataset. Due to the existence of these relationships, the entities tend to choose triple with the same relation with current head relation pair for target entity prediction, since the same relation have the largest similarity score. Under this circumstance, the model is more easily to abuse this neighboring triple for inference, which is likely to lead to overfitting problem. In contrast, with the increase of neighbor number, the entity could get more comprehensive information around it to implement inference, thereby improve the performance.

B.2 Effect of Batch Size

We conduct experiments to study the effect of batch size on large-scale dataset Wikidata5M-Trans, where the results are shown in Table 7. As

batch size	**MRR**	**Hits@1**	**Hits@3**	**Hits@10**
256	41.5	36.2	43.4	51.4
512	42.1	36.8	44.4	51.9
1024	**42.6**	**37.3**	**44.7**	**52.5**

Table 7: Experiment results on Wikidata5M-Trans with different batch size.

shown in this table, larger batch size improves the model's capacity steadily. Compare the performance of batch size 1024 with 256, we could find that the larger batch size improves the MRR, Hits@1, Hits@3, Hits@10 by 0.9, 1.1, 1.3, and 1.1, respectively, which illustrates its great influence.

In this paper we do not use batch size larger than 1024 for experiments.

B.3 Effect of Graph Structure

We conduct experiments to study the effect of graph structure, including directed graph and undirected graph, for neighbor sampling. The experiment results on WN18RR and Wikidata5M are shown in Table 8 and Table 9, respectively.

graph structure	**MRR**	**Hits@1**	**Hits@3**	**Hits@10**
Directed	67.6	59.4	73.0	81.6
Undirected	**69.9**	**62.3**	**74.9**	**83.9**

Table 8: Experiment results on WN18RR with both directed and undirected graph structure.

As shown in Table 8, compared to the directed graph, undirected graph seems to get better performance on WN18RR dataset. One possible explanation could be that the structure of WN18RR is much sparser, in which entities may not have enough neighbors that could serve as supplemental information. Under this circumstance, undirected graph has more neighboring triples for complement, thereby improves the prediction performance.

graph structure	**MRR**	**Hits@1**	**Hits@3**	**Hits@10**
Directed	**42.6**	**37.3**	**44.7**	52.5
Undirected	42.4	37.0	44.6	**52.6**

Table 9: Experiment results on Wikidata5M-Trans with both directed and undirected graph structure.

In contrast, as Table 9 shows, directed graph seems to have better overall performance on Wikidata5M-Trans dataset. This could be due to that the structure of Wikidata5M-Trans is denser, which has more neighboring triples as additional information. Besides, the direction of relations might be more important in this dataset.