MoCoKGC: Momentum Contrast Entity Encoding for Knowledge Graph Completion

Anonymous ACL submission

Abstract

In recent years, numerous studies have sought to enhance the capabilities of pretrained language models (PLMs) for Knowledge Graph Completion (KGC) tasks by integrating structural information from knowledge graphs. However, existing approaches have not effectively combined the structural attributes of knowledge graphs with the textual descriptions of entities to generate robust entity encodings. To address this issue, this paper proposes MoCoKGC (Momentum Contrast Entity Encoding for Knowledge Graph Completion), which incorporates three primary encoders: the entity-relation encoder, the entity encoder, and the momentum entity encoder. Momentum contrast learning not only provides more negative samples but also allows for the gradual updating of entity encodings. Consequently, we reintroduce the generated entity encodings into the encoder to incorporate the graph's structural information. Additionally, MoCoKGC enhances the inferential capabilities of the entity-relation encoder through deep prompts of relations. On the standard evaluation metric, Mean Reciprocal Rank (MRR), the MoCoKGC model demonstrates superior performance, achieving a 7.1% improvement on the WN18RR dataset and an 11% improvement on the Wikidata5M dataset, while also surpassing the current best model on the FB15k-237 dataset. Through a series of experiments, this paper thoroughly examines the role and contribution of each component and parameter of the model.

1 Introduction

As an important method of knowledge representation, the fundamental building blocks of a knowledge graph are factual triples, such as (Steve Jobs, founded, Apple Inc.). However, the process of constructing knowledge graphs, whether manually or through automation, inevitably leads to the presence of many missing triplets within the knowledge graph. Therefore, the knowledge graph completion task (KGC) aims to complete the missing triples.

Among the numerous methods for KGC, knowledge graph embedding (KGE) techniques are the most classical and widely adopted. The core idea behind these methods is to generate embedding vectors for entities and relationships and use different scoring functions to predict the missing triplets (Bordes et al., 2013; Dettmers et al., 2018; Sun et al., 2019; Balazevic et al., 2019). Building upon this, some studies introduce the structure of knowledge graphs as supplementary information in the reasoning process to improve prediction accuracy (Schlichtkrull et al., 2018; Vashishth et al., 2020; Chen et al., 2021).

In recent years, researchers have begun exploring the integration of Pre-trained Language Models (PLMs) into the task of KGC, aiming to improve accuracy through the textual descriptions of entities and relationships. Models based on pre-trained language encoders can be broadly categorized into three types: (1) Cross-encoder models (Yao et al., 2019) ; (2) Bi-encoder models (Wang et al., 2021a, 2022);(3) Single-encoder models (Liu et al., 2022b; Chen et al., 2023a). Although cross-encoder models fully utilize the semantic information of triplets, their performance is often not as good as other methods due to the high cost of obtaining negative samples during training. Bi-encoder models, by separating the tail entity, not only preserve the textual information of the tail entity but also effectively increase the number of negative samples. Single-encoder models, despite removing the textual information of the tail entity, demonstrate unique advantages by acquiring a large number of negative samples for tail entities and introducing graph information through entity embeddings.

In order to preserve the textual information of entities while flexibly integrating entity encoding into the training and prediction phases of the model, this study adopts the momentum contrastive learning mechanism (He et al., 2020), thereby proposing the MoCoKGC model. This model draws inspiration from the Bi-encoder architecture, equipped with a head entity-relation encoder and an entity encoder. The distinction lies in that, within the MoCoKGC model, the update of entity encoding relies on a momentum entity encoder, rather than directly utilizing the entity encoder, thus achieving a smoothing of the entity encoding update process. Consequently, the MoCoKGC model exhibits two significant features compared to other methods: (1) Entity Queue. This queue is maintained in the order of entity encodings generated by the momentum entity encoder, providing the model with a rich source of negative tail entity samples; (2) Reutilization of entity encoding. Under this mechanism, entity encoding, as a part of integrating the structural information of the knowledge graph, is re-imported into the encoder.

In terms of experimental validation, the MoCoKGC model not only successfully preserved the textual information of entities but also flexibly integrated entity encoding into the model's training and prediction processes. Its performance on standard datasets WN18RR, FB15k-237, and Wikidata5M demonstrates the model's superior capabilities: on the WN18RR dataset, in Mean Reciprocal Rank (MRR), the model achieved a 7.1% improvement, an 11% increase on the Wikidata5M dataset, and surpassed previous models on the FB15k-237 dataset. Furthermore, comparative experimental data reveal that MoCoKGC effectively overcomes the inconsistency issues exhibited by PLMs when dealing with sparse and dense knowledge graphs (Wang et al., 2022).

2 Related Work

Knowledge Graph Completion (KGC) tasks is to predict missing triplet information within a knowledge graph. In the domain of knowledge graph embeddings, a typical method is TransE (Bordes et al., 2013), which utilizes the Euclidean distance between the sum of head entity and relation embeddings and the tail entity embedding as a scoring function. The RotatE (Sun et al., 2019) method is based on the core concept of interpreting relations in the knowledge graph as rotational operations in complex space. Graph-based approaches, such as R-GCN (Schlichtkrull et al., 2018), address the problem of relation learning by introducing weight matrices for different types of relations in graph neural networks, thereby capturing the unique semantics of each relation type.

Methods based on Pre-trained Language Models (PLMs), such as KG-BERT (Yao et al., 2019), concatenate the descriptions of head entities, relations, and tail entities, and directly obtain the triplet score by inputting it into the BERT (Devlin et al., 2019) model. StAR (Wang et al., 2021a) adopts a dual-encoder architecture, which significantly reduces the inference time overhead of language models. SimKGC (Wang et al., 2022) introduces a contrastive learning approach.

Prompt Tuning has emerged as a strategy aimed at significantly improving the performance of PLMs by adding prompt tokens to the input. This approach was initially developed to address the challenge of fine-tuning Large Language Models (LLMs) for downstream tasks (Brown et al., 2020). Recently, in KGC tasks, researchers have improved the performance of PLMs through immediate learning, Lv et al. (2022) adding prompt templates and soft prompts to the input, while Chen et al. (2022) and Liu et al. (2022b) specified different soft prompt tokens for different types of relations. This paper views relations as deep prompt parameters and introduces entity neighborhood prompts, achieving the objective of leveraging both textual descriptions and knowledge graph structural information.

Contrastive Learning by differentiating positive and negative sample features, learns distinctive feature representations and has been successfully applied in multiple domains, including computer vision. MoCo (He et al., 2020) proposed a momentum-based contrastive learning method, effectively solving the problem of sample pair construction in unsupervised learning by building a dynamically changing encoder queue. SimCLR (Chen et al., 2020), as a method of visual representation learning, significantly improved the performance of image recognition tasks through large-scale unsupervised contrastive learning. In KGC tasks, SimKGC (Wang et al., 2022) treats tail entities as positive and negative samples, implementing efficient training through three different simple negative sampling strategies. This paper combines the momentum contrast method of MoCo, utilizing it while dynamically updating entity encodings.

3 Methodology

3.1 Notations

Knowledge Graphs (KGs) represent a crucial data structure for organizing and storing factual relationships in the real world, which can be formally represented as $\mathcal{G} = \{\mathcal{E}, \mathcal{R}, \mathcal{T}\}$. Here, $\mathcal{E}$ and $\mathcal{R}$ denote the sets of entities and relationships, respectively. $\mathcal{T} = \{(h, r, t)\} \subseteq \mathcal{E} \times \mathcal{R} \times \mathcal{E}$ defines a set of triples, each comprising a head entity (h), a relation (r), and a tail entity (t). The task of KGC aims to fill in missing triples within a knowledge graph, with link prediction as its core task. This task focuses on predicting the missing entity part in given triples $(h, r, ?)$ or $(?, r, t)$.

3.2 Neighborhood Prompts

To integrate the structural information of knowledge graphs into pre-trained language models, this study proposes a method that utilizes the neighborhood information of entities as prompts. The neighborhood of an entity is defined as the directly connected entities and their corresponding relations, formally represented as follows:

$$N(e) = \{(e_i, r_i) | (e_i, r_i, e) \in \mathcal{T}\} \tag{1}$$

Given the variation in the neighborhood size of entities within knowledge graphs, this research introduces a parameter—σ—to standardize the dimension of sampled neighborhood information. Specifically, when the neighborhood size of an entity exceeds the set σ, a corresponding number of entity-relation pairs are randomly extracted from this neighborhood to meet the σ; conversely, if the neighborhood size is smaller than the σ, specific padding tokens (pad tokens) are introduced to fill up to the σ. This treatment ensures the dimensional consistency of neighborhood information across all entities, facilitating subsequent processing. To obtain neighborhood prompts, entities and relations within the neighborhood are summed and then processed through a Multilayer Perceptron (MLP):

$$\boldsymbol{p}_{N(e)} = \text{MLP}([\boldsymbol{e_0} + \boldsymbol{r_0}, ..., \boldsymbol{e_\sigma} + \boldsymbol{r_\sigma}]) \tag{2}$$

3.3 Model Architecture

The MoCoKGC model is comprised of three primary components: the entity-relation encoder, the entity encoder, and the momentum entity encoder, as illustrated in Figure 1. The principal duties of these encoders are to generate encodings for the head entity and its relations, update the momentum entity encoder and pseudo-entity encodings, and produce entity encodings, respectively. It is noteworthy that the generation of entity encodings is dependent on the slower-updating momentum entity encoder, rather than the entity encoder. Below, we provide a detailed explanation of these three encoders.

Entity-Relation Encoder, the process initiates by aggregating the encodings of all entities within the vicinity of the head entity and their corresponding relations, followed by processing through a Multilayer Perceptron (MLP) to obtain neighborhood prompt information. Subsequently, the description of the head entity, the relation description, and the neighborhood prompt information are concatenated and inputted into a Transformer encoder. To more effectively amalgamate various types of information, we employ the p-tuning v2 strategy, as referenced in (Liu et al., 2022a), introducing the relation as deep prompt information at every layer of the Transformer encoder. The encoding of the head entity-relation is acquired through a pooling layer followed by normalization. This procedure can be formalized as:

$$\boldsymbol{h_r} = \text{ER\_Encoder}(\boldsymbol{d}(h), \boldsymbol{d}(r), \boldsymbol{p}_{N(h)}, \boldsymbol{p}_r) \tag{3}$$

In formula 3, $\boldsymbol{d}(h)$, $\boldsymbol{d}(r)$, $\boldsymbol{p}N(h)$, and $\boldsymbol{p}r$ respectively represent the description of the head entity, the description of the relation, the neighborhood prompt of the head entity, and the relation prompt. It is important to highlight that the relation encoding within the neighborhood and the relation parameters in the deep prompts are not identical.

Entity Encoder focuses on generating encodings for tail entities without incorporating relation descriptions or cues. This simplified processing distinguishes it from the entity-relation encoder. The absence of relation inputs in this encoder is represented by

$$\bar{\boldsymbol{t}} = \text{E\_Encoder}(\boldsymbol{d}(t), \boldsymbol{p}_{N(t)}) \tag{4}$$

In formula 4, $\boldsymbol{d}(t)$ and $\boldsymbol{p}_{N(t)}$ respectively denote the tail entity description and the neighborhood prompt of the tail entity.

Momentum Entity Encoder shares its input format with the entity encoder, aimed at encoding the tail entity.

$$\boldsymbol{t} = \text{ME\_Encoder}(\boldsymbol{d}(t), \boldsymbol{p}_{N(t)}) \tag{5}$$

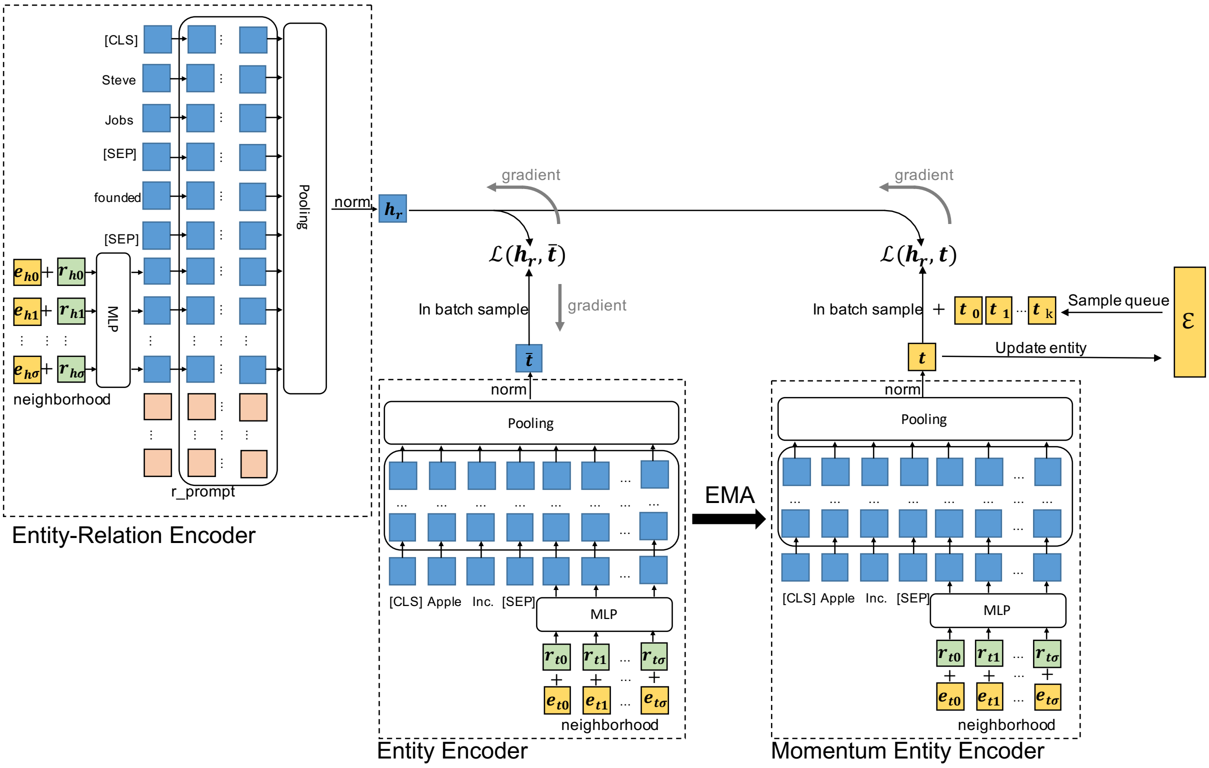

Figure 1: The MoCoKGC framework primarily consists of three encoders: the entity-relation encoder, the entity encoder, and the momentum entity encoder. As illustrated, the momentum entity encoder does not directly participate in the gradient backpropagation process; its parameter updates are based on the Exponential Moving Average (EMA) strategy. MoCoKGC updates entity encodings $\mathcal{E}$ using a momentum entity encoder and augments the number of negative samples by maintaining an entity queue. Importantly, all entity encodings required for generating neighborhood prompts are sourced from $\mathcal{E}$.

However, its distinctive feature lies in how its parameters are updated. Instead of using backpropagation for updates, this encoder's parameters evolve iteratively based on the entity encoder's parameters after each iteration. This method, known as the Exponential Moving Average (EMA):

$$\theta_{ME} = m\theta_{ME} + (1 - m)\theta_E \tag{6}$$

Where θ_{ME} and θ_E denote the parameters of the momentum entity encoder and the entity encoder, respectively. $m \in [0, 1]$ represents the momentum coefficient. This process is also referred to as the Exponential Moving Average (EMA).

It is crucial to highlight that the neighborhood representation of entity encodings, employed by all three encoders, is shared and generated by the momentum entity encoder. Conversely, the relation encoding is unique to each model component and is not shared.

3.4 Negative Sampling

In-batch Negatives like most contrastive learning methods, our study uses tail entities from within the same batch as negative samples. In our approach, we not only utilize entity encodings as positive and negative examples but also integrate them into the representations of their respective neighborhoods.

Entity Queue is maintained by MoCoKGC throughout the training process to generate a larger pool of negative sample entities. Unlike conventional queues, the entity elements in this queue are unique. If an entity that is about to be enqueued is already present in the queue, it is first dequeued and then enqueued again. This mechanism ensures that a greater variety of different entities can be stored while the queue length remains fixed.

3.5 Training and Inference

During the training phase of the model, considering that the neighborhood sampling of the head entity may contain the tail entity, and similarly, the neighborhood sampling of the tail entity may include the head entity, this study adopts a target link dropout strategy after the neighborhood sampling process.

Specifically, the target links existing in the neighborhoods of the head and tail entities are discarded and replaced with a padding token. This measure aims to ensure that training target data is not leaked into the model, thereby affecting the model's generalization capability. In addition to the dropout of target links, to enhance the diversity of neighborhood information, this study also introduces a mechanism to randomly drop entity-relation pairs in the neighborhood with a certain probability.

As illustrated in Figure 1, the process of loss calculation in this study involves multiplying the generated head entity relation encoding $\boldsymbol{h_r}$ with both the pseudo-entity encoding $\bar{\boldsymbol{t}}$ and the actual entity encoding $\boldsymbol{t}$, based on which the loss is calculated. This study employs the same method of calculating the loss function as SimKGC (Wang et al., 2022), use InfoNCE loss with additive margin (Chen et al., 2020; Yang et al., 2019):

$$\mathcal{L}(\boldsymbol{h_r}, \boldsymbol{t}) = -\log \frac{e^{(\boldsymbol{h_r}\boldsymbol{t}^T - \gamma)/\tau}}{e^{(\boldsymbol{h_r}\boldsymbol{t}^T - \gamma)/\tau} + \sum_{i=1}^{|\mathcal{N}|} e^{(\boldsymbol{h_r}\boldsymbol{t_i'}^T)/\tau}} \tag{7}$$

Where γ is the margin coefficient greater than 0, $\tau \in [0, 1]$ is the temperature coefficient and $\mathcal{N}$ is all negative sample entities. Based on the formula 7, the final loss function can be expressed as:

$$loss = \mathcal{L}(\boldsymbol{h_r}, \boldsymbol{t}) + \mathcal{L}(\boldsymbol{h_r}, \bar{\boldsymbol{t}}) \tag{8}$$

To enhance the update frequency of entity encodings, this study not only updates a portion of entity encodings at each iteration but also separately utilizes the momentum entity encoder for inference after a certain number of iterations, to achieve updates of all entities. After completing all training processes, a final update of all entities will be conducted.

For the prediction inference of KGC, it is only necessary to generate the head entity-relation encoding through the entity-relation encoder, and then multiply it with all entity encodings to obtain the predictive scores for all entities.

$$scores = \{\boldsymbol{h_r}\boldsymbol{t_i}^T | t_i \in \mathcal{E}\} \tag{9}$$

In terms of time complexity, the time complexity of this study in the test set is consistent with that of most KGC models, which is $|\mathcal{T}_{\text{test}}|$.

4 Experiments

4.1 Experimental Setup

Dataset Evaluation In this study, three benchmark datasets were utilized to assess the performance of the proposed model, specifically: WN18RR, FB15k-237, and Wikidata5M. Table 1 presents the detailed distribution of these datasets. The WN18RR dataset (Dettmers et al., 2018) is constructed based on the WordNet knowledge base (Miller, 1998), aimed at link prediction tasks, containing entities represented by English phrases and their semantic relationships. The FB15k-237 dataset (Toutanova et al., 2015) is a subset derived from the Freebase knowledge base (Bollacker et al., 2008), encompassing entities in the real world and their interrelations. The Wikidata5M dataset (Wang et al., 2021b) is a large-scale knowledge graph dataset, integrating information from the Wikidata knowledge graph and Wikipedia pages, providing Wikipedia page descriptions for each entity. Compared to WN18RR and FB15k-237, the Wikidata5M dataset surpasses them by two orders of magnitude in both the number of entities and triples, indicating its larger scale and complexity.

Evaluation Metrics In the task of KGC, the assessment of model performance is primarily achieved by measuring the ranking of target triples among all potential triples' scores. This study adopts the commonly used evaluation metrics in previous research, including Hits@1, Hits@3, Hits@10, and MRR. The Hits@k metric measures the frequency with which the target triple appears among the top k triples with the highest scores, while the MRR is the average of the reciprocal ranks of the target triples. To enhance the accuracy and fairness of the evaluation, we employed the filtered ranking setting proposed by (Bordes et al., 2013), which eliminates potential ranking biases by excluding all possible triples $(h, r, ?)$ or $(t, r^{-1}, ?)$ that already exist in the training set. Furthermore, following the random evaluation protocol suggested by Sun et al. (2020), we accurately assess model performance.

Implementation Details To ensure the comparability of the results of this study with existing research, we selected the "bert-base-uncased" version of the BERT model as the Transformer encoder for this research. Utilizing the AdamW

Dataset	# entity	relation	# 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	4,594,485	822	20,614,279	5,163	5,163

Table 1: Summary statistics of benchmark datasets.

	WN18RR				FB15k-237				Wikidata5M			
	MRR	Hits@1	Hits@3	Hits@10	MRR	Hits@1	Hits@3	Hits@10	MRR	Hits@1	Hits@3	Hits@10
Knowledge graph embedding method												
TransE (Bordes et al., 2013)$^{\diamond}$	0.243	0.043	0.441	0.532	0.279	0.198	0.376	0.441	0.253	0.170	0.311	0.392
DistMult (Yang et al., 2015)$^{\diamond}$	0.444	0.412	0.470	0.504	0.281	0.199	0.301	0.446	0.253	0.209	0.278	0.334
ComplEx (Trouillon et al., 2016)$^{\diamond}$	0.449	0.409	0.469	0.530	0.278	0.194	0.297	0.450	0.308	0.255	-	0.398
R-GCN (Schlichtkrull et al., 2018)†	0.123	0.080	0.137	0.207	0.164	0.100	0.181	0.300	-	-	-	-
ConvE (Dettmers et al., 2018)†	0.456	0.419	0.470	0.531	0.312	0.225	0.341	0.497	-	-	-	-
RotatE (Sun et al., 2019)$^{\diamond}$	0.476	0.428	0.492	0.571	0.338	0.241	0.375	0.533	0.290	0.234	0.322	0.390
TuckER (Balazevic et al., 2019)	0.470	0.443	0.482	0.526	0.358	0.266	0.394	0.544	-	-	-	-
CompGCN (Vashishth et al., 2020)	0.479	0.443	0.494	0.546	0.355	0.264	0.390	0.535	-	-	-	-
HittER (Chen et al., 2021)	0.503	0.462	0.516	0.584	0.373	0.279	0.409	0.558	-	-	-	-
N-Former (Liu et al., 2022b)	0.486	0.443	0.501	0.578	0.372	0.277	0.412	0.556	-	-	-	-
PLM-Based method												
KG-BERT (Yao et al., 2019)	0.216	0.041	0.302	0.524	-	-	-	0.420	-	-	-	-
StAR (Wang et al., 2021a)	0.401	0.243	0.491	0.709	0.296	0.205	0.322	0.482	-	-	-	-
KEPLER(Wang et al., 2021b)$^{\diamond}$	-	-	-	-	-	-	-	-	0.210	0.173	0.224	0.277
KG-S2S (Chen et al., 2022)	0.574	0.531	0.595	0.661	0.336	0.257	0.373	0.498	-	-	-	-
N-BERT (Liu et al., 2022b)	0.583	0.529	0.607	0.686	0.381	0.287	0.420	0.562	-	-	-	-
SimKGC (Wang et al., 2022)	0.671	0.585	0.731	0.817	0.333	0.246	0.362	0.510	0.358	0.313	0.376	0.441
CSProm-KG (Chen et al., 2023b)	0.575	0.522	0.596	0.678	0.358	0.269	0.393	0.538	0.380	0.343	0.399	0.446
GHN (Qiao et al., 2023)	0.678	0.596	0.719	0.821	0.339	0.251	0.364	0.518	0.364	0.317	0.380	0.453
Ensemble method												
StAR(Self-Adp) (Wang et al., 2021a)	0.551	0.459	0.594	0.732	0.365	0.266	0.404	0.562	-	-	-	-
CoLE (Liu et al., 2022b)	0.587	0.532	0.608	0.694	0.389	0.294	0.430	0.574	-	-	-	-
MoCoKGC(Ours)	**0.742**	**0.665**	**0.792**	**0.881**	**0.391**	**0.296**	**0.431**	**0.580**	**0.490**	**0.435**	**0.517**	**0.591**

Table 2: Experimental results for various baseline methods on WN18RR, FB15k-237, and Wikidata5M datasets. †: Results are sourced from Wang et al. (2021a); $^{\diamond}$: Results are sourced from Chen et al. (2023b). The best methods are highlighted in bold, with the most effective methods in each category underscored for emphasis. The results for the MoCoKGC model are reported as the average of three experimental runs.

optimizer for model training. The learning rate was set to 5×10^{-5}. The batch size was selected from the set $\{256, 512, 1024\}$. The range of the momentum coefficient m was chosen from $\{0, 0.5, 0.9, 0.99, 0.999\}$. The neighborhood sampling size σ was selected from the set $\{256, 512, 1024\}$. The length of the maintained entity queue was chosen from the set $\{512, 1024, 2048, 4096, 8192, 16384, 32768\}$. For further details, please refer to Appendix A.

4.2 Main Results

On the WN18RR, FB15k-237, and Wikidata5M datasets, we compared the MoCoKGC model with other leading models, as shown in Table 2. The experimental results demonstrate that MoCoKGC achieved state-of-the-art performance across all evaluation metrics. Notably, on the WN18RR and Wikidata5M datasets, MoCoKGC realized significant improvements of 7.1% (from 0.671 to 0.742) and 11% (from 0.343 to 0.399), respectively. As a method based on pre-trained language models (PLM-Based), MoCoKGC also achieved a 1.0% performance improvement (from 0.381 to 0.391) on the FB15k-237 dataset, surpassing the previous best ensemble learning approach.

Furthermore, we conducted a separate analysis on the MRR values of models that performed well on the WN18RR and FB15k-237 datasets, as depicted in Figure 2. The analysis revealed that knowledge graph embedding methods exhibited relatively balanced performance on these two datasets (i.e., models that performed well on WN18RR also excelled on FB15k-237). In PLM-based models, SimKGC and GHN exhibit significant performance improvements on the WN18RR dataset, yet they lag on the FB15k-237 dataset. We attribute this phenomenon to SimKGC's use of entity descriptions, generating entity encodings through an entity encoder, and the absence of knowledge graph structural information during inference. MoCoKGC successfully addressed the inconsistency in performance of PLM-based models on these two datasets.

On the larger Wikidata5M dataset, the performance improvement of MoCoKGC was especially pronounced, which is closely related to the rich entity textual descriptions and significant knowledge graph structure within the Wikidata5M dataset. Our proposed MoCoKGC model, as a PLM-based method, not only integrates the entity encoder from SimKGC (Wang et al., 2022) but also, like models such as CoLE (Liu et al., 2022b) and CSProm-KG (Chen et al., 2023b), incorporates the structure of knowledge graphs (e.g., relation and neighborhood prompts) into the model. This effectively combines the advantages of textual descriptions with the knowledge graph structure.

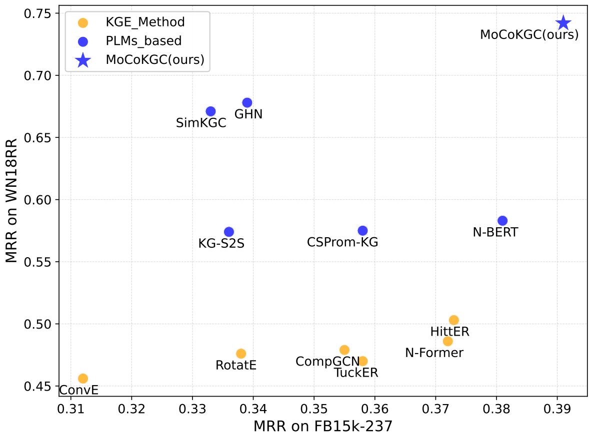

Figure 2: MRR performance of different models on WN18RR and FB15k-237 datasets.

4.3 Ablation Studies

Model	MRR	Hits@1	Hits@10
MoCoKGC w/o momentum entity encoder	0.727	0.645	0.875
MoCoKGC w/o entity queue	0.735	0.657	0.877
MoCoKGC w/o neighborhood prompt	0.696	0.614	0.845
MoCoKGC w/o relation prompt	0.597	0.476	0.818
MoCoKGC	**0.742**	**0.665**	**0.881**

Table 3: Ablation Study regarding important components in MoCoKGC on the benchmark of WN18RR.

Model	MRR	H@1	H@10
MoCoKGC w/o momentum entity encoder	0.369	0.280	0.548
MoCoKGC w/o entity queue	0.379	0.284	0.569
MoCoKGC w/o neighborhood prompt	0.385	0.292	0.570
MoCoKGC w/o relation prompt	0.327	0.242	0.496
MoCoKGC	**0.391**	**0.296**	**0.580**

Table 4: Ablation Study regarding important components in MoCoKGC on the benchmark of FB15k-237.

Structural component. In the WN18RR and FB15k-237 datasets, we conducted an analysis to understand the roles of different components within MoCoKGC, as shown in Tables 3 and 4.

In the experiment, I separately removed the momentum entity encoder (using the entity encoder instead to generate entities) and the entity queue. It was observed that there was a decrease in performance on both datasets, and these two components exhibited similar behaviors on both datasets. This aligns with expectations, as they correspond to smoother entity updates and an increase in the number of negative samples, both of which are related to entity generation. Removing both components had a more significant impact in FB15k-237, suggesting that learning entities in dense knowledge graphs is more challenging.

Additionally, the absence of neighborhood prompts also resulted in a performance decline, particularly in WN18RR, where performance dropped by 4.9% (from 0.745 to 0.696). The impact of lacking neighborhood prompts was greater than that of removing the momentum entity encoder and entity queue. This indicates that the neighborhood structure in WN18RR, as opposed to FB15k-237, can be effectively utilized. This may relate to the intrinsic properties of the two knowledge graphs, where WN18RR's structure describes English phrases and their semantic relationships, whereas FB15k-237, as a real-world knowledge graph, has a more random neighborhood structure.

More notably, the removal of relation prompts led to a substantial performance decline of 14.5% (from 0.745 to 0.597) and 6.4% (from 0.391 to 0.327) on the WN18RR and FB15k-237 datasets, respectively. This phenomenon suggests that the simple reuse of entity encodings might interfere with the encoder's effective capture of deep semantic information about entities and their relations. To overcome this issue, the introduction of relation prompts is crucial for restoring and enhancing the synergistic effect of textual semantics and knowledge graph structural information within PLMs.

m	0	0.5	0.9	0.99	0.999
MRR (WN18RR)	0.727	0.728	0.728	0.733	0.742
MRR (FB15k-237)	0.369	0.367	0.375	0.380	0.391

Table 5: Demonstrates the MRR of MoCoKGC on the datasets WN18RR and FB15k-237, with varying momentum coefficient m used during training.

Momentum coefficient. In Table 5, we present the results of the MRR for models trained with different momentum coefficients m on the WN18RR and FB15k-237 datasets. Analysis indicates that higher momentum coefficients m can stably enhance model performance, whereas lower momentum coefficients m have not shown significant improvement in performance. This experimental outcome aligns with our initial rationale for employing a momentum entity encoder, which is to introduce a steady yet gradual entity encoding update mechanism, in the hope of achieving performance improvement.

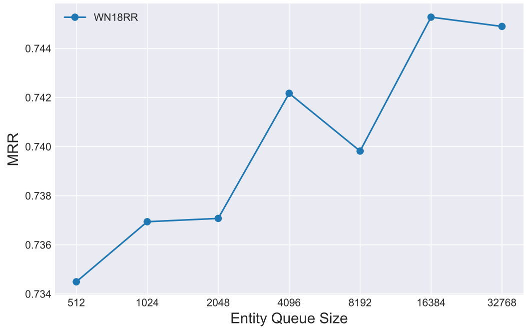

Figure 3: Variation of MRR with entity queue size on WN18RR in MoCoKGC

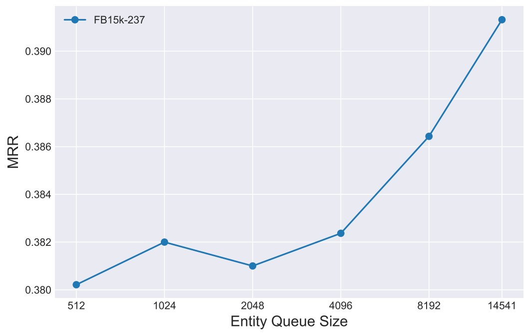

Figure 4: Variation of MRR with entity queue size on FB15k-237 in MoCoKGC

Entity queue size. In the training framework of MoCoKGC, a pivotal component is the maintenance of a dynamic entity queue, aimed at accumulating and leveraging a broader spectrum of negative tail entity samples throughout the training process. To investigate the impact of the entity queue, we examined how variations in the size of the entity queue influence model performance. As illustrated in Figures 3 and 4, the Mean Reciprocal Rank (MRR) on WN18RR and FB15k-237 varies with different entity queue sizes. The results demonstrate a consistent upward trend in the MRR metric as the size of the entity queue increases. This indicates that expanding the entity queue significantly augments the quantity of effective negative entity samples, thereby exerting a positive impact on model performance.

Impact of Training Set Size. During the training process of the MoCoKGC model, we observed that the model could achieve commendable performance even with a limited amount of training

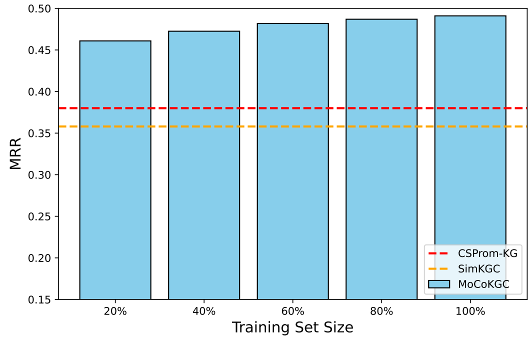

Figure 5: Variation of MRR with training set size on Wikidata5M in MoCoKGC, with comparative final MRR sesults from SimKGC and CSProm-KG on the entire training set.

data. As depicted in Figure 5, for comparison purposes, we presented the training outcomes of the SimKGC and CSProm-KG models on the complete training set in dashed lines. Notably, when utilizing only 20% of the training data, the MRR of the MoCoKGC model could reach 0.460. This result significantly surpasses the final performance of the other two methods. This finding underscores the exceptional generalization capability of MoCoKGC in scenarios of data scarcity.

Furthermore, we have added separate studies on sampling size and model dimensions in Appendix A. It is worth noting that in the WN18RR dataset, we surpassed previous methods using only 26.4% of the model size.

5 Conclusion

This study proposes MoCoKGC, a novel KGC model that leverages momentum contrastive learning in conjunction with PMLs. By expanding the pool of negative samples, it further enhances KGC through the aggregation of entity textual descriptions and their structural information. The MoCoKGC model demonstrated superior performance across multiple datasets. Furthermore, we further validated the critical role of its constituent components and parameter configurations. Future work will focus on adapting MoCoKGC for open knowledge graphs to better manage the emergence of new entities.

6 Limitations

The MoCoKGC model relies on pre-trained language models to integrate textual representations with the structure of knowledge graphs. This results in an increase in training time and memory consumption as the length of the structure input into the model increases. In response, MoCoKGC opts for a compromise by sampling the neighborhoods of entities, rather than aggregating the entire knowledge graph structure as done by R-GCN (Schlichtkrull et al., 2018) and CompGCN (Vashishth et al., 2020). Moreover, the random sampling does not take into account the varying importance of different links within the neighborhood. This leads to the model predictions being more focused on the features within the sampled neighborhoods. In the future, we plan to address this issue.

References

Ivana Balazevic, Carl Allen, and Timothy M. 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 2019, Hong Kong, China, November 3-7, 2019, pages 5184–5193. Association for Computational Linguistics.

Kurt D. Bollacker, Colin Evans, Praveen K. Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2008, Vancouver, BC, Canada, June 10-12, 2008, pages 1247–1250. ACM.

Antoine Bordes, Nicolas Usunier, Alberto García-Durán, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States, pages 2787–2795.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual.

Chen Chen, Yufei Wang, Bing Li, and Kwok-Yan Lam. 2022. Knowledge is flat: A seq2seq generative framework for various knowledge graph completion. In Proceedings of the 29th International Conference on Computational Linguistics, COLING 2022, Gyeongju, Republic of Korea, October 12-17, 2022, pages 4005–4017. International Committee on Computational Linguistics.

Chen Chen, Yufei Wang, Aixin Sun, Bing Li, and Kwok-Yan Lam. 2023a. Dipping plms sauce: Bridging structure and text for effective knowledge graph completion via conditional soft prompting. In Findings of the Association for Computational Linguistics: ACL 2023, Toronto, Canada, July 9-14, 2023, pages 11489–11503. Association for Computational Linguistics.

Chen Chen, Yufei Wang, Aixin Sun, Bing Li, and Kwok-Yan Lam. 2023b. Dipping plms sauce: Bridging structure and text for effective knowledge graph completion via conditional soft prompting. In Findings of the Association for Computational Linguistics: ACL 2023, Toronto, Canada, July 9-14, 2023, pages 11489–11503. Association for Computational Linguistics.

Sanxing Chen, Xiaodong Liu, Jianfeng Gao, Jian Jiao, Ruofei Zhang, and Yangfeng Ji. 2021. Hitter: Hierarchical transformers for knowledge graph embeddings. In Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7-11 November, 2021, pages 10395–10407. Association for Computational Linguistics.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey E. Hinton. 2020. A simple framework for contrastive learning of visual representations. In Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event, volume 119 of Proceedings of Machine Learning Research, pages 1597–1607. PMLR.

Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. 2018. Convolutional 2d knowledge graph embeddings. In Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018, pages 1811–1818. AAAI Press.

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, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers), pages 4171–4186. Association for Computational Linguistics.

Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross B. Girshick. 2020. Momentum contrast for unsupervised visual representation learning. In 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020, pages 9726–9735. Computer Vision Foundation / IEEE.

Xiao Liu, Kaixuan Ji, Yicheng Fu, Weng Tam, Zhengxiao Du, Zhilin Yang, and Jie Tang. 2022a. P-tuning: Prompt tuning can be comparable to fine-tuning across scales and tasks. In Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), pages 61–68, Dublin, Ireland. Association for Computational Linguistics.

Yang Liu, Zequn Sun, Guangyao Li, and Wei Hu. 2022b. I know what you do not know: Knowledge graph embedding via co-distillation learning. In Proceedings of the 31st ACM International Conference on Information & Knowledge Management, Atlanta, GA, USA, October 17-21, 2022, pages 1329–1338. ACM.

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, Dublin, Ireland, May 22-27, 2022, pages 3570–3581. Association for Computational Linguistics.

George A Miller. 1998. WordNet: An electronic lexical database. MIT press.

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, Toronto, Canada. Association for Computational Linguistics.

Michael Sejr Schlichtkrull, Thomas N. Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. 2018. Modeling relational data with graph convolutional networks. In The Semantic Web - 15th International Conference, ESWC 2018, Heraklion, Crete, Greece, June 3-7, 2018, Proceedings, volume 10843 of Lecture Notes in Computer Science, pages 593–607. Springer.

Zhiqing Sun, Zhi-Hong Deng, Jian-Yun Nie, and Jian Tang. 2019. Rotate: Knowledge graph embedding by relational rotation in complex space. In 7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019. OpenReview.net.

Zhiqing Sun, Shikhar Vashishth, Soumya Sanyal, Partha P. Talukdar, and Yiming Yang. 2020. A re-evaluation of knowledge graph completion methods. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020, pages 5516–5522. Association for Computational Linguistics.

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, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015, pages 1499–1509. The Association for Computational Linguistics.

Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. 2016. Complex embeddings for simple link prediction. In Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016, volume 48 of JMLR Workshop and Conference Proceedings, pages 2071–2080. JMLR.org.

Shikhar Vashishth, Soumya Sanyal, Vikram Nitin, and Partha P. Talukdar. 2020. Composition-based multi-relational graph convolutional networks. In 8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020. OpenReview.net.

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 WWW '21: The Web Conference 2021, Virtual Event / Ljubljana, Slovenia, April 19-23, 2021, pages 1737–1748. ACM / IW3C2.

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), ACL 2022, Dublin, Ireland, May 22-27, 2022, pages 4281–4294. Association for Computational Linguistics.

Xiaozhi Wang, Tianyu Gao, Zhaocheng Zhu, Zhengyan Zhang, Zhiyuan Liu, Juanzi Li, and Jian Tang. 2021b. KEPLER: A unified model for knowledge embedding and pre-trained language representation. Trans. Assoc. Comput. Linguistics, 9:176–194.

Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. 2015. Embedding entities and relations for learning and inference in knowledge bases. In 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings.

Yinfei Yang, Gustavo Hernández Ábrego, Steve Yuan, Mandy Guo, Qinlan Shen, Daniel Cer, Yun-Hsuan Sung, Brian Strope, and Ray Kurzweil.

2019. Improving multilingual sentence embedding using bi-directional dual encoder with additive margin softmax. In Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019, pages 5370–5378. ijcai.org.

Liang Yao, Chengsheng Mao, and Yuan Luo. 2019. KG-BERT: BERT for knowledge graph completion. CoRR, abs/1909.03193.

A Details on Implementation

# of GPUs	1
learning rate	5×10^{-5}
initial temperature τ	0.05
gradient clip	10
warmup steps	400
dropout	0.1
neighborhood dropout	0.1
weight decay	10^{-4}
InfoNCE margin	0.02
momentum coefficient m	0.999
pooling	mean

Table 6: Shared hyperparameters for MoCoKGC.

	WN18RR	FB15k-237	Wikidata5M
batch size	1024	256	1024
additional entity size	512	1024	512
max # of word tokens	64	128	64
neighborhood sampling size σ	16	128	32
entity queue size	16384	14541	16384
epochs	30	3	1

Table 7: Hyperparameters of the MoCoKGC model that are not shared across different datasets.

In this study, the hyperparameter settings were primarily aligned with the configuration strategy of SimKGC (Wang et al., 2022). As demonstrated in Table 6, we have listed the hyperparameter settings shared across all datasets. Concurrently, Table 7 showcases the specific hyperparameter configurations for the MoCoKGC model across different datasets.

Given that the experiments were conducted using a single GeForce RTX 4090 graphics card, and faced with memory capacity limitations, we employed gradient accumulation techniques to enable larger batch sizes. It is noteworthy that, due to the infeasibility of directly applying conventional gradient accumulation methods in the contrastive learning process, we first generate all necessary contrastive encodings for the three encoders using smaller batch sizes and disabling gradient saving during each accumulation step. Subsequently, we update the entity-relation encoder and the entity encoder using gradient accumulation techniques. To eliminate the potential randomness introduced by dropout operations, a random number is generated and recorded as the random seed during each gradient accumulation, and this seed is set every time an encoder is invoked. In addition to gradient accumulation, in the experiments on Wikidata5M, we stored the entity encodings in CPU memory rather than in GPU memory to reduce the usage of GPU memory.

During each training epoch, the MoCoKGC model runs on a single GeForce RTX 4090 graphics card, utilizing a configuration that includes four workers for data loading. The runtime varies depending on the dataset: it takes approximately 7 minutes for the WN18RR dataset, about 5 hours for the FB15k-237 dataset, and roughly 65 hours for the Wikidata5M dataset.

Furthermore, drawing from the practices of SimKGC, we made the following adjustments to the textual descriptions of entities: (1) the names of neighboring entities in the training set are concatenated to the description of the entity, and the correct entities are dynamically excluded from the input text during the training process; (2) the descriptions of inverse relations are formed by appending the term "inverse" to the beginning of the original relation descriptions.

Our implementation is based on open-source project *transformers* [1].

B More Experiments

s

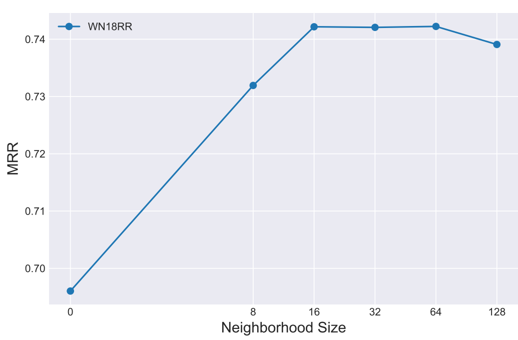

Figure 6: Variation of MRR with neighborhood sampling size on WN18RR in MoCoKGC

The effects of the neighborhood prompt on the performance of MoCoKGC are presented in Tables 3 and 4. Further analysis on the impact of the

[1] https://github.com/huggingface/transformers

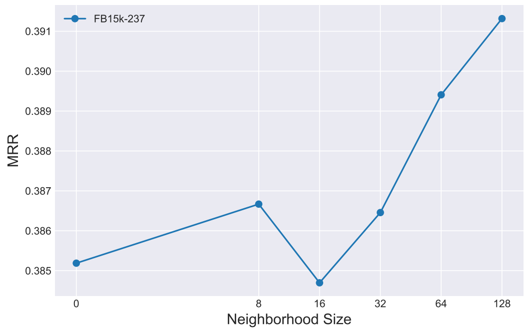

Figure 7: Variation of MRR with neighborhood sampling size on FB15k-237 in MoCoKGC

neighborhood prompt's length and the neighborhood sampling size, σ, is conducted. As illustrated in Figures 6 and 7, an ascending trend in MRR is observed with an increase in σ.

For WN18RR, given the graph's relative sparsity where each entity in the training dataset is connected to an average of 2.12 links, an increase in σ beyond a certain point results in the majority of the entity neighborhoods being smaller than σ. Hence, further increments in σ would only benefit a minority of entities, rendering limited overall improvements. Conversely, the graph for FB15k-237 is comparatively dense, with each entity in the training dataset having an average of 18.71 links. Thus, improvements can still be observed with σ increased to 128.

Additionally, it is evident that for the sparser WN18RR, a neighborhood prompt length of just 16 can enhance the MRR by 4.6%. In contrast, the denser FB15k-237 requires a greater length of neighborhood prompts for noticeable improvements.

PLM	parameters	MRR	Hits@1	Hits@10
bert-large	340M	0.740	**0.667**	0.876
bert-base	110M	**0.742**	0.665	**0.881**
bert-medium	42M	0.718	0.633	0.874
bert-small	29M	0.706	0.620	0.862
bert-tiny	4M	0.644	0.564	0.793

Table 8: Performance of MoCoKGC with PLMs of different sizes on the WN18RR Dataset.

In Table 2, the bert-base is utilized as the Pre-trained Language Models (PLMs) for comparison with other relevant models. To investigate the impact of PLMs of different sizes on MoCoKGC, we conducted experiments using BERT models of varying sizes on WN18RR, as shown in Table 8.

It was observed that the use of a smaller BERT (bert-small) yielded results on WN18RR reaching 0.706, surpassing other models listed in Table 2 while only utilizing 26.4% of the base model.

Overall, performance tends to improve as the size of the PLMs increases, indicating a positive correlation between the size of the PLMs and the performance of MoCoKGC. However, further increases with the bert-large model do not continue to enhance MoCoKGC's performance, suggesting that there is a bottleneck in the textual features utilized by MoCoKGC when the PLMs become excessively large.