Knowledge Graph Pooling and Unpooling for Concept Abstraction

Juan Li, Wen Zhang*, Zhiqiang Liu, Mingchen Tu,
Mingyang Chen, Ningyu Zhang, Shijian Li*
Zhejiang University
{lijuan18, zhang.wen, zhiqiangliu, mingchentz,
mingyangchen, zhangningyu, shijianli}@zju.edu.cn

Abstract

Knowledge graph embedding (KGE) aims to embed entities and relations as vectors in a continuous space and has proven to be effective for KG tasks. Recently, graph neural networks (GNN) based KGEs gain much attention due to their strong capability of encoding complex graph structures. However, most GNN-based KGEs are directly optimized based on the instance triples in KGs, ignoring the latent concepts and hierarchies of the entities. Though some works explicitly inject concepts and hierarchies into models, they are limited to predefined concepts and hierarchies, which are missing in a lot of KGs. Thus in this paper, we propose a novel framework with KG Pooling and unpooling and Contrastive Learning (**KGPCL**) to abstract and encode the latent concepts for better KG prediction. Specifically, with an input KG, we first construct a U-KG through KG pooling and unpooling. KG pooling abstracts the input graph to a smaller graph as a pooled graph, and KG unpooling recovers the input graph from the pooled graph. Then we model the U-KG with relational KGEs to get the representations of entities and relations for prediction. Finally, we propose the local and global contrastive loss to jointly enhance the representation of entities. Experimental results show that our models outperform the KGE baselines on link prediction task.

1 Introduction

Knowledge graphs (KGs) are crucial for many knowledge-driven applications, such as recommendation systems (Yang et al., 2022), and knowledge-based question answering (Saxena et al., 2021). They store knowledge in the form of triples (*h,r,t*), i.e., (*head entity, relation, tail entity*) to indicate the relation between the head entity and the tail entity. However, KGs usually suffer from incompleteness, limiting the effectiveness of KG applications. Thus, knowledge graph embedding (KGE) methods are proposed to infer new facts.

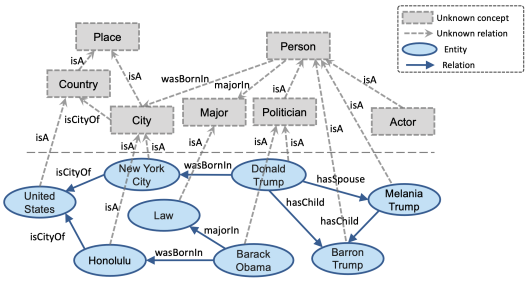

Figure 1: A hypothetical knowledge graph, indicating entities have latent concepts and hierarchies, although they are not predefined.

There are many types of KGEs proposed in the past decade (Wang et al., 2017). Translation-based models (Sun et al., 2019) measure the plausibility of a triple by interpreting the relation as a translation, and semantic matching models (Yang et al., 2015) calculate semantic similarity. Furthermore, GNN-based KGEs (Schlichtkrull et al., 2018) gain much attention for the ability to explicitly model complex graph structures. They use relational GNNs as encoders and conventional KGEs as decoders to learn entity and relation representations, and have achieved promising results on link prediction and triple classification tasks.

Despite the success of the GNN-based KGEs, they stack GNN layers in a flat manner, ignoring the concepts and hierarchies of entities. Although some studies explore injecting explicit concepts into models (Niu et al., 2022), such predefined information is usually missing in lots of KGs. However, the concepts and hierarchies of entities could be reflected by the relations of triples. As shown in Figure 1, the entities *New York City* and *Honolulu* are both the tail entities of the relation *wasBornIn*, suggesting that they may belong to the same concept. Thus we propose the research question **is it possible to learn latent concepts (i.e., do concept abstraction) based on raw triples and enhance**

* Corresponding author

Proceedings of the 31st International Conference on Computational Linguistics, pages 5364–5374
January 19–24, 2025.

the prediction results with the latent concepts?

In this paper, we propose a framework with KG Pooling and unpooling and Contrastive Learning (**KGPCL**) to abstract and encode the latent concepts of entities in KGs. Specifically, we transform the input KG into a sequence of KGs. The input KG is first pooled to a more abstract KG, abbreviated as pooled KG. And then the pooled KG is recovered to the input KG, abbreviated as recovered KG. The sequence of the input KG, pooled KG and recovered KG presents a U-shape , thus we call it U-KG. Then we use relational graph neural networks to model the KGs in the U-KG and get the representations of the entities and relations for prediction. Besides, we adopt local and global contrastive learning (CL) loss as the training objective to constrain entity embeddings. The local CL focuses on entity embeddings on a single graph, and the global CL focuses on both entity and concept embeddings on two graphs across the U-KG. We conduct experiments on four datasets and compare KGPCL with conventional KGEs, especially GNN-based KGEs. Results demonstrate that KGPCL can achieve better performance than baselines on link prediction task. Additionally, our ablation and visualization results verify the effectiveness of U-KG construction and modeling.

2 Related Work

2.1 Knowledge Graph Embedding

From the perspective of learning resources, we divide KGEs into two categories: (1) methods encoding only the raw triples, and (2) methods encoding both the raw triples and the concepts of entities.

Methods encoding only the raw triples mainly consist of translation-based, semantic matching, and neural network models. Translation-based models describe relations as translations, for example, TransE (Bordes et al., 2013) assumes $\boldsymbol{h} + \boldsymbol{r} \approx \boldsymbol{t}$, where $\boldsymbol{h}$, $\boldsymbol{r}$ and $\boldsymbol{t}$ are embeddings of h, r and t. TransH (Wang et al., 2014) and TransD (Ji et al., 2015) extend TransE to handle complex relations. RotatE (Sun et al., 2019) defines relations as rotations in a complex vector space. Semantic matching models calculate the semantic similarity of entities and relations, where DistMult (Yang et al., 2015) defines the score function as $f_r(h,t) = \boldsymbol{h}^\top diag(\boldsymbol{r})\boldsymbol{t}$. ComplEx (Trouillon et al., 2016) extends DistMult to complex space to model asymmetric relations. Neural network models use convolutional neural networks like ConvE (Dettmers et al., 2018) or Graph Neural Networks (Schlichtkrull et al., 2018) (GNN) for KG completion. RGCN (Schlichtkrull et al., 2018) introduces relation-specific transformations for neighbor aggregation, and CompGCN (Vashishth et al., 2020) further introduces composition operations. SEGNN (Li et al., 2022) proposes three semantic evidences from relation, entity, and triple levels to understand the extrapolation ability of KGE. These GNNs are typically used as encoders for capturing global structure, while other KGEs can be served as decoders in our work.

Methods encoding both the raw triples and the concepts of entities encode explicit or implicit concepts. For explicit concepts, CAKE (Niu et al., 2022) generates commonsense with entity concepts to produce high-quality negative triples in training and filter the candidates in evaluation. TransO (Li et al., 2023) proposes specific constraint strategies for entity types, relations, and hierarchical information. Concept2Box (Huang et al., 2023) embeds ontology-view and instance-view of a KG using dual geometric representations by modeling concepts with box embeddings and entities as vectors. For implicit concepts, TypeDM (Jain et al., 2018) calculates type compatibility scores by introducing type embeddings for entities and relations. AutoETER (Niu et al., 2020) learns latent type embeddings of entities by regarding relations as translations between the types of entities with relation-aware projections. Different from the two implicit methods, our framework models the interactions of implicit concepts to enhance the entity representations via the constraints between the entities and concepts.

2.2 Contrastive learning

Contrastive learning has achieved great success in various domains, such as natural language processing (Das et al., 2022), recommendation (Yang et al., 2022), and KG reasoning (Luo et al., 2022). It defines the positive and negative pairs, and then pulls the positive pairs close and pushes apart the negative pairs. KCL (Fang et al., 2022) builds chemical element KG and proposes a knowledge-enhanced contrastive learning framework for molecular representation learning. KGE-CL (Luo et al., 2022) introduces contrastive learning by constructing the positive pairs for those entities that share the same entity-relation couple and those entity-relation couples that share the same entity, and regarding unrelated entities and couples as negative pairs. KR-

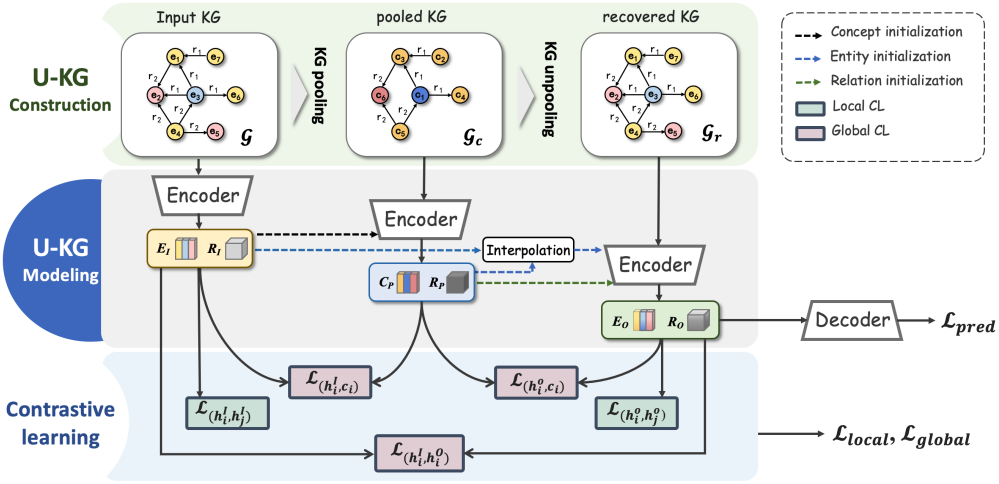

Figure 2: The architecture of our framework KGPCL.

ACL (Tan et al., 2023) proposes contrastive learning loss to alleviate the sparsity of KGs, introducing more negative samples and enriching the feedback to sparse entities for enhancing sparse KG completion. Contrastive learning could help us learn more distinguishable representations of entities both in one KG and across the KGs in the U-KG.

2.3 Graph pooling

Due to the flatness of GNN models, graph pooling approaches are proposed to obtain embeddings with hierarchies. DIFFPOOL (Ying et al., 2018) proposes a differentiable pooling module and learns a differentiable soft cluster assignment for nodes at each layer of a deep GNN. Then it can coarse the input graph and generate hierarchical representations. Graph U-Nets (Gao and Ji, 2019) introduces the graph pooling layer and unpooling layer for node classification. The graph pooling layer adaptively selects a subset of nodes to form a new but smaller graph. SAGPool (Lee et al., 2019) exploits a self-attention mechanism to distinguish between the nodes that should be dropped and retained, considering both node features and graph topology. Different from graph pooling, KG pooling is more challenging since the relationships between entities should also be considered during pooling.

3 Methodology

In this section, we introduce a method that models latent concept information with KG Pooling and unpooling and Contrastive Learning, abbreviated as **KGPCL**. As shown in Figure 2, KGPCL comprises three main components: 1) **U-KG construction** abstracts the input KG $\mathcal{G} = \{\mathcal{E}, \mathcal{R}, \mathcal{T}_e\}$ to the pooled KG $\mathcal{G}_c = \{\mathcal{C}, \mathcal{R}, \mathcal{T}_c\}$ and restores the pooled KG to get the recovered KG $\mathcal{G}_r = \mathcal{G}$, where $\mathcal{E}$, $\mathcal{R}$ and $\mathcal{C}$ indicate entities, relations and fine-grained concepts respectively. $\mathcal{T}_e$ and $\mathcal{T}_c$ indicate triples with entities and concepts as nodes. 2) **U-KG modeling** uses GNN as encoder and conventional KGE score function as decoder on U-KG to obtain embeddings of entities and relations. 3) **Local and global contrastive learning** focuses on constraining entity embeddings on a single graph, and constraining entity and concept embeddings on two graphs across U-KG.

3.1 U-KG Construction

We consider a U-KG with three KGs for simplicity, where the input KG, the pooled KG, and the recovered KG serve as the first, middle, and last ones. Actually, a U-KG can be a sequence of multiple (more than three) KGs at different scales to abstract multi-layer concepts. The construction consists of two steps: KG pooling and KG unpooling.

3.1.1 KG Pooling

KG pooling automatically abstracts the input KG $\mathcal{G}$ into the pooled KG $\mathcal{G}_c$ as depicted in Figure 3. The core is that a relation can reflect the concepts of its head and tail entities. For instance, the head entity can be in the concept *Person* and the tail entity in *Country* for the relation *nationality*. We perform five operations in KG pooling: 1) coarse-grained concept generation, 2) concept splitting, 3) sub-concept merging, 4) entity to concept conversion, and 5) concept triple deduplication. The first three operations dedicate to obtaining fine-grained concepts and establishing entity-concept mapping. The last two translate relational triples to concept triples to output the pooled KG.

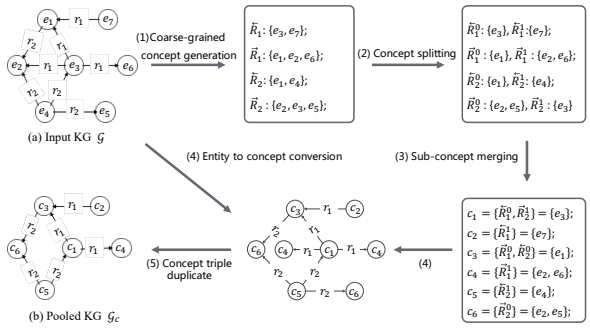

Figure 3: The workflow of KG pooling. It is an example of $k = 2$ and $\delta = 1.0$. $\overleftarrow{R}_i$ and $\overrightarrow{R}_i$ are the head and tail entity sets of the relation r_i, respectively.

Coarse-grained concept generation regards the head and tail entity sets of relations as coarse-grained concepts. We start by traversing all triples in $\mathcal{G}$ to get the coarse-grained concepts according to the relations. For a specific relation r, the coarse concept set of its head entities and tail entities are represented as $\overleftarrow{R}$ and $\overrightarrow{R}$, where $\overleftarrow{R} = \{h|(h,r,e), e \in \mathcal{E}\}$ and $\overrightarrow{R} = \{t|(e,r,t), e \in \mathcal{E}\}$.

Concept splitting aims to capture fine-grained concepts as the concepts may vary or contain hierarchies. For example, the concept of the head entity can be *Person* or *Film* when the relation is *language*, and can be *Actor* or *Writer* if the concept is *Person*. Thus we adopt the k-means algorithm to cluster entities of each concept into k sub-concepts based on the entity embeddings pre-trained by TransE (Bordes et al., 2013), where k is a hyper-parameter. Take $\overleftarrow{R}$ as an example, we split it into $\{\overleftarrow{R}^0, \overleftarrow{R}^1, ..., \overleftarrow{R}^{k-1}\}$ as sub-concepts. If the number of entities $e \in \overleftarrow{R}$ is insufficient, i.e., $|e| \leq k$, we regard each entity as a sub-concept.

Sub-concept merging merges the potentially similar sub-concepts to avoid redundancy. Specifically, we first get all sub-concept pairs, with each consisting of two sub-concepts from two different coarse-grained concepts. Then we calculate Jaccard similarity and reserve the sub-concept pairs with the similarity score above the given threshold δ. With the reserved sub-concept pairs, we merge the pairs that share at least one concept into a fine-grained concept, and regard each of the reserved sub-concept pairs that are not merged as a fine-grained concept. Besides, we consider each sub-concept that does not appear in the reserved sub-concept pairs as a fine-grained concept. These fine-grained concepts are served as concept nodes in the pooled KG. Finally, we get entity-concept mapping via mapping of the entity to sub-concept and the sub-concept to fine-grained concept.

Entity to concept conversion concentrates on converting the relational triples in $\mathcal{G}$ into concept triples in $\mathcal{G}_c$. Given a triple, we convert the head and tail entities into concepts based on both the entity-concept mapping and the relation. For example, we convert e_2 into c_4 in the triple (e_3, r_1, e_2) and into c_6 in the triple (e_1, r_2, e_2) as shown in Figure 3. To be specific, e_2 in (e_3, r_1, e_2) belongs to the sub-concept of the concept $\overrightarrow{R}_1$ due to the relation is r_1. Since e_2 appears in $\overrightarrow{R}_1^1$ and $\overrightarrow{R}_2^0$, it thus belongs to $\overrightarrow{R}_1^1$ and c_4. In contrast, e_2 in (e_1, r_2, e_2) belongs to the sub-concept of the concept $\overrightarrow{R}_2$, which points to $\overrightarrow{R}_2^0$ and c_6.

Concept triple deduplication removes duplicated concept triples after the conversion from entities to concepts. We output the pooled KG with abstract concepts as nodes and relations as edges.

3.1.2 KG Unpooling

KG unpooling restores the pooled KG $\mathcal{G}_c$ into the original input KG as the recovered KG $\mathcal{G}_r = \mathcal{G}$. $\mathcal{G}_r$ and $\mathcal{G}_c$ use the same entity-concept mapping as $\mathcal{G}$ and $\mathcal{G}_c$. The three graphs share relations.

3.2 U-KG Modeling

The goal of U-KG modeling is to obtain entity and relation embeddings. We first review the GNN encoders used in our paper, and then describe the initialization and update of the entity and relation embeddings in the three KGs of U-KG.

3.2.1 GNN Encoder

As the update of the KGs is achieved by GNN layers, we use three typical GNN encoders including RGCN, CompGCN, and SEGNN to capture graph structure.[1] RGCN (Schlichtkrull et al., 2018) introduces relation-specific linear transformations on neighbor entities to update an entity. CompGCN (Vashishth et al., 2020) updates both entities and relations, and uses composition operators to model the interactions of entities and relations. SEGNN (Li et al., 2022) considers relation, entity, and triple level semantic evidence aggregation.

3.2.2 U-KG Encoding

Given the input KG $\mathcal{G}$, we use a GNN encoder to get the entity and relation embeddings:

$$\boldsymbol{E_I}, \boldsymbol{R_I} = \text{GNN}(\mathcal{G}) \quad (1)$$

[1]The update details of RGCN, CompGCN, and SEGNN are in Appendix A.

where $\boldsymbol{E_I} \in \mathbb{R}^{n_e \times d}$ is the entity embedding, $\boldsymbol{R_I} \in \mathbb{R}^{n_r \times d}$ is the relation embedding. n_e and n_r are the numbers of the entities and relations. d is the dimension.

For the pooled KG $\mathcal{G}_c$, we first transform output entity embeddings to concept embeddings to initialize the concepts:

$$\boldsymbol{C_I} = \boldsymbol{N_{ce}} \cdot (\boldsymbol{M_e} \boldsymbol{E_I}) \tag{2}$$

where $\boldsymbol{M_e} \in \mathbb{R}^{n_c \times n_e}$ is a matrix that indicates the mapping between the concepts and the entities established in KG pooling, n_c is the number of the concepts. $\boldsymbol{M_e}(i, j)$ is 1 or 0 to indicate whether the ith concept contains the jth entity. $\boldsymbol{N_{ce}} \in \mathbb{R}^{n_c \times n_c}$ is a diagonal matrix. $\boldsymbol{N_{ce}}(i, i)$ equals the reciprocal of the sum of the element values in ith row of $\boldsymbol{M_e}$. Then we initialize the concept and relation embeddings with $\boldsymbol{C_I}$ and $\boldsymbol{R_I}$ respectively, and apply another GNN encoder to update the embeddings. The update is defined as:

$$\boldsymbol{C_P}, \boldsymbol{R_P} = \text{GNN}(\mathcal{G}_c, \boldsymbol{C_I}, \boldsymbol{R_I}) \tag{3}$$

where $\boldsymbol{C_P}$ and $\boldsymbol{R_P}$ are the updated concept and relation embeddings.

For the recovered KG $\mathcal{G}_r$, we first transform the concept embeddings into entity embeddings to initialize entities:

$$\boldsymbol{E_P} = \boldsymbol{N_{ec}} \cdot (\boldsymbol{M_c} \boldsymbol{C_P}) \tag{4}$$

where $\boldsymbol{M_c} \in \mathbb{R}^{n_e \times n_c}$ is the transpose of $\boldsymbol{M_e}$. $\boldsymbol{N_{ec}} \in \mathbb{R}^{n_e \times n_e}$ is a diagonal matrix, where the diagonal element $\boldsymbol{N_{ec}}(i, i)$ is the reciprocal of the number of concepts that the ith entity has. Then we initialize the entity embeddings by adopting a simple interpolation calculation to incorporate concept information. The initialization of entities and relations is defined as:

$$\boldsymbol{E_O} = \alpha_e * \boldsymbol{E_I} + \alpha_c * \boldsymbol{E_P}; \quad \boldsymbol{R_O} = \boldsymbol{R_P} \tag{5}$$

where α_e and α_c are the weight of entity embedding from $\mathcal{G}$ and $\mathcal{G}_c$. The update is defined as:

$$\boldsymbol{E_O^*}, \boldsymbol{R_O^*} = \text{GNN}(\mathcal{G}_r, \boldsymbol{E_O}, \boldsymbol{R_O}) \tag{6}$$

To reduce the parameters, we directly fed $\boldsymbol{E_O}$ and $\boldsymbol{R_O}$ into the decoder for prediction. The results of using $\boldsymbol{E_O^*}$ and $\boldsymbol{R_O^*}$ are discussed in model variants of experiments.

Following the previous work (Vashishth et al., 2020), we use ConvE (Dettmers et al., 2018) as the decoder and define the query embedding as:

$$\boldsymbol{q} = \text{ConvE}(\boldsymbol{h}, \boldsymbol{r}) \tag{7}$$

where $\boldsymbol{h}$ and $\boldsymbol{r}$ are taken from $\boldsymbol{E_O}$ and $\boldsymbol{R_O}$ respectively. Finally, we use the standard binary cross entropy as the training target of predicting links:

$$\begin{aligned} \mathcal{L}_p = -\frac{1}{N} \sum_t & p(t) \cdot log(m(\boldsymbol{q}, \boldsymbol{t})) \\ & + (1 - p(t)) \cdot log(1 - m(\boldsymbol{q}, \boldsymbol{t})) \end{aligned} \tag{8}$$

where N is the number of candidate entities, and $\boldsymbol{t}$ is taken from $\boldsymbol{E_O}$. $m(\boldsymbol{q}, \boldsymbol{t}) = \sigma(\boldsymbol{q}^T \boldsymbol{t})$ is the matching score of the query and the candidate entity. $p(t)$=1 (or 0) denotes (h, r, t) is a positive (or negative) triple.

3.3 Local and Global Contrastive Learning

We design local and global contrastive learning loss to jointly optimize and enhance the entity and concept embeddings.

3.3.1 Local Contrastive Learning

The local contrastive learning is applied on a single KG (i.e., $\mathcal{G}$ or $\mathcal{G}_r$) to pull the similar entities close. We consider two entities similar when the Jarcard similarity of their concept sets is higher than a given threshold δ_s. If h_i and h_j are similar entities, we define the local loss as:

$$\mathcal{L}_{(h_i, h_j)} = \frac{-1}{|P_s(i)|} \sum_{h_j \in P(i)} log \frac{e^{s(\boldsymbol{h_i}, \boldsymbol{h_j})/\eta}}{\sum\limits_{h_k \in N_s(i)} e^{s(\boldsymbol{h_i}, \boldsymbol{h_k})/\eta}} \tag{9}$$

where $P_s(i)$ and $N_s(i)$ are the set of all positive and negative pairs in the mini-batch, respectively. $s(\boldsymbol{h_i}, \boldsymbol{h_j})$ is the cosine similarity of the two entities. η is the temperature parameter. Then the overall local CL loss can be defined as:

$$\mathcal{L}_{local} = \mathcal{L}_{(h_i^I, h_j^I)} + \mathcal{L}_{(h_i^O, h_j^O)} \tag{10}$$

where the superscripts I and O indicate entity embeddings from the input and the recovered KG, respectively.

3.3.2 Global Contrastive Learning

The global contrastive learning is applied between two graphs, including $(\mathcal{G}, \mathcal{G}_c)$, $(\mathcal{G}_r, \mathcal{G}_c)$ and $(\mathcal{G}, \mathcal{G}_r)$. $(\mathcal{G}/\mathcal{G}_r, \mathcal{G}_c)$ pulls an entity and its concepts close:

$$\mathcal{L}_{(h_i, c_i)} = \frac{-1}{|P_c(i)|} \sum_{c_i \in P_c(i)} log \frac{e^{s(\boldsymbol{h_i}, \boldsymbol{c_i})/\eta}}{\sum\limits_{c_k \in N_c(i)} e^{s(\boldsymbol{h_i}, \boldsymbol{c_k})/\eta}} \tag{11}$$

where $P_c(i)$ is the positive concept set of the entities in the mini-batch and $N_c(i)$ is the negative

concept set. $(\mathcal{G}, \mathcal{G}_r)$ pulls the same entities in the KGs $\mathcal{G}$ and $\mathcal{G}_r$ close:

$$\mathcal{L}_{(h_i^I, h_i^O)} = \frac{-1}{|P(i)|} \sum_{h_i \in P(i)} log \frac{e^{s(\boldsymbol{h_i^I}, \boldsymbol{h_i^O})/\eta}}{\sum_{h_k \in N(i)} e^{s(\boldsymbol{h_i^I}, \boldsymbol{h_k^O})/\eta}} \tag{12}$$

Then the overall global loss can be defined as:

$$\mathcal{L}_{global} = \mathcal{L}_{(h_i^I, c_i)} + \mathcal{L}_{(h_i^O, c_i)} + \mathcal{L}_{(h_i^I, h_i^O)} \tag{13}$$

where h_i^I and h_i^O indicate entity embeddings of h_i from $\mathcal{G}$ and $\mathcal{G}_r$, and c_i indicates concept embedding of c_i from $\mathcal{G}_c$.

3.4 Training objective

By combining the prediction loss with the contrastive learning loss, we minimize the following function to jointly learn the parameters of GNN encoders and the entity and relation embeddings:

$$\mathcal{L} = \mathcal{L}_p + \lambda_1 \mathcal{L}_{local} + \lambda_2 \mathcal{L}_{global} \tag{14}$$

where λ_1 and λ_2 are hyper-parameters to control the local and global contrastive loss, respectively.

4 Experiments

In this section, we conduct link prediction experiments to explore the following research questions: (RQ1) How does KGPCL perform compared with conventional, especially GNN-based, KGEs? (RQ2) How do KG pooling and unpooling and different contrastive learning components affect KGPCL? (RQ3) Can KGPCL capture concept information in U-KG construction and modeling?

4.1 Experimental Setup

Datasets. The experiments are conducted on four datasets: FB15K237 (Toutanova and Chen, 2015), NELL995 (Xiong et al., 2017), YAGO29K (Hao et al., 2019) and WN18RR (Dettmers et al., 2018). FB15K237 and WN18RR are the subsets of FB15K and WN18. NELL995 is extracted from the 995-th iteration of NELL. YAGO29K is extracted from YAGO. We use the datasets as the entities may reflect relatively obvious concepts. The statistics are listed in Table 1.

Dataset	#Ent	#Rel	#Train	#Valid	#Test
FB15K237	14541	237	272115	17535	20466
NELL995	75492	200	149678	543	3992
WN18RR	40943	11	86835	3034	3134
YAGO29K	26078	34	351664	-	39074

Table 1: Dataset statistics.

Baselines. We compare KGPCL with the conventional KGEs, including TransE (Bordes et al., 2013), DistMult (Yang et al., 2015), RotatE (Sun et al., 2019), HAKE (Zhang et al., 2020), PairRE (Chao et al., 2021) REP-OTE (Wang et al., 2022), SDFormer (Li et al., 2024), and GATH (Wei et al., 2024), and especially the GNN-based KGEs, including RGCN (Schlichtkrull et al., 2018), CompGCN (Vashishth et al., 2020) and SEGNN (Li et al., 2022).

Evaluation Protocol. We report the metrics Mean Reciprocal Rank (MRR) and Hits@n ($n \in \{1, 3, 10\}$). MRR is the mean of all the reciprocals of predicted ranks. Hits@n is the percentage rate of original test triples ranked at the top n in the candidate list. The higher values of MRR and Hits@n indicate better performance. The metrics are in the filtered setting (Bordes et al., 2013).

Parameter Settings. We search k in {10, 100, 1000} for WN18RR, {10, 20} for FB15K237, {10, 50} for NELL995, {50, 100} for YAGO29, δ in {0.5, 0.7, 0.9} on all datasets, and α_e and α_c in {1.0, 1.5, 2.0}. We employ one GNN layer on each KG of U-KG and use the best parameter settings from the original papers to reduce parameter tuning costs. Regarding contrastive learning, we search η in {0.5, 0.8}, δ_s in {0.6, 0.8}. The weights λ_1 and λ_2 are selected in {1e-05, 1e-06, 1e-07} due to the magnitude difference between the contrastive and task losses. We use label smoothing with the parameter set as 0.1 to lessen overfitting and use Adam (Kingma and Ba, 2015) to optimize the objective function. The number of training epochs is set as 500 for RGCN and CompGCN encoders and 600 or 800 for SEGNN encoder.

4.2 Performance Comparisons (RQ1)

Main results. Table 2 and 3 show the link prediction results. Following are observations: (1) Among datasets, our KGPCL methods outperform or are comparable to corresponding GNN KGEs in most of the metrics, demonstrating that KGPCL is effective in enhancing KG prediction. (2) The improvements on NELL995 are more obvious than on FB15K237, although they have more relations. The reason is that NELL995 has a richer variety of concepts than FB15K237. (3) For the datasets YAGO29K and WN18RR with a smaller number of relations, the improvements on YAGO29K are

Model	FB15K237				NELL995			
	MRR	Hits@10	Hits@3	Hits@1	MRR	Hits@10	Hits@3	Hits@1
TransE (Bordes et al., 2013)	0.294	0.465	-	-	0.401	0.501	0.472	0.344
DistMult (Yang et al., 2015)	0.281	0.446	0.301	0.199	0.485	0.610	0.524	0.401
RotatE (Sun et al., 2019)	0.338	0.533	0.375	0.241	0.483	0.565	0.514	0.435
HAKE (Zhang et al., 2020)	0.346	0.542	0.381	0.250	0.508	0.613	0.557	0.442
PairRE (Chao et al., 2021)	0.351	0.544	0.387	0.256	0.536	0.635	0.580	0.470
REP-OTE (Wang et al., 2022)	0.354	0.540	0.388	0.262	-	-	-	-
SDFormer (Li et al., 2024)	0.356	0.541	0.390	0.264	-	-	-	-
GATH (Wei et al., 2024)	0.344	0.527	0.376	0.253	-	-	-	-
RGCN (Schlichtkrull et al., 2018)	0.349	0.526	0.381	0.260	0.500	0.634	0.550	0.419
RGCN-KGPCL	**0.354**	**0.535**	**0.388**	**0.263**	**0.529**	**0.651**	**0.577**	**0.454**
CompGCN (Vashishth et al., 2020)	0.355	0.536	0.390	0.264	0.517	0.637	0.566	0.441
CompGCN-KGPCL	**0.357**	**0.541**	**0.392**	**0.266**	**0.543**	**0.648**	**0.589**	**0.476**
SEGNN (Li et al., 2022)	0.353	0.539	0.387	**0.262**	0.531	0.637	**0.575**	0.460
SEGNN-KGPCL	**0.355**	**0.542**	**0.391**	0.261	**0.533**	**0.653**	**0.575**	**0.463**

Table 2: Link prediction results on FB15K237 and NELL995. Boldface scores are the better ones between the GNN-based KGEs and corresponding KGPCL methods. Scores in the boxes are the best results among all methods.

Model	YAGO29K				WN18RR			
	MRR	Hits@10	Hits@3	Hits@1	MRR	Hits@10	Hits@3	Hits@1
TransE (Bordes et al., 2013)	0.195	0.345	-	0.141	0.243	0.532	0.441	0.043
DistMult (Yang et al., 2015)	0.253	0.288	-	0.229	0.444	0.504	0.470	0.412
RotatE (Sun et al., 2019)	0.593	0.791	0.662	0.486	0.483	0.565	0.514	0.435
HAKE (Zhang et al., 2020)	0.596	0.788	0.662	0.491	0.508	0.613	0.557	0.442
PairRE (Chao et al., 2021)	0.442	0.644	0.495	0.336	0.454	0.548	0.469	0.411
REP-OTE (Wang et al., 2022)	-	-	-	-	0.488	0.588	0.505	0.439
SDFormer (Li et al., 2024)	-	-	-	-	0.458	0.528	0.471	0.425
GATH (Wei et al., 2024)	-	-	-	-	0.463	0.537	0.475	0.426
RGCN (Schlichtkrull et al., 2018)	0.285	0.492	0.314	0.185	0.465	0.535	0.480	0.427
RGCN-KGPCL	**0.372**	**0.596**	**0.419**	**0.260**	**0.472**	**0.547**	**0.487**	**0.435**
CompGCN (Vashishth et al., 2020)	0.436	0.666	0.492	0.319	0.482	0.550	0.496	**0.446**
CompGCN-KGPCL	**0.494**	**0.718**	**0.559**	**0.378**	**0.485**	**0.558**	**0.498**	0.445
SEGNN (Li et al., 2022)	0.589	0.790	0.663	0.477	**0.485**	0.560	**0.497**	**0.448**
SEGNN-KGPCL	**0.616**	**0.805**	**0.690**	**0.509**	0.482	**0.567**	0.496	0.439

Table 3: Link prediction results on YAGO29K and WN18RR.

Model	FB15K237				NELL995			
	MRR	Hits@10	Hits@3	Hits@1	MRR	Hits@10	Hits@3	Hits@1
RGCN	0.348	0.527	0.382	0.259	0.500	0.634	0.550	0.419
R-KGPCL(FEA+SUM)	0.353	0.534	0.387	0.263	0.523	0.648	0.570	0.448
R-KGPCL(FEA+GNN)	0.350	0.532	0.385	0.258	0.514	0.642	0.562	0.438
R-KGPCL(KGE+GNN)	0.350	0.529	0.383	0.259	0.522	0.651	0.570	0.446
R-KGPCL(KGE+SUM)	0.354	0.535	0.388	0.263	0.529	0.651	0.577	0.454

Table 4: The performance of four model variants of RGCN-KGPCL. RGCN-KGPCL is abbreviated as R-KGPCL.

apparent. It indicates that more triples can make up for the limitations of fewer relations in capturing latent concepts. (4) The pavements are more significant on simple GNN encoders than complex ones, because CompGCN and SEGNN introduce composition operations and multi-level semantic evidence to better model the interactions among entities and relations. For example, RGCN-KGPCL improves Hits@1 by 7.5% compared with RGCN on YAGO29K, but CompGCN-KGPCL (SEGNN-KGPCL) improves 5.9% (3.2%) compared with CompGCN (SEGNN).

Model variants. We introduce model variants, exploring the comparison of embedding used for k-means algorithm (i.e., KGE/FEA), and the modeling of the recovered KG in U-KG (i.e., SUM/GNN). KGE and FEA indicate the embeddings used to split the concepts are derived from TransE and feature vectors generated by statistics[2]. SUM and GNN indicate equation 5 (i.e., $\boldsymbol{E_O}$ and $\boldsymbol{R_O}$) and equation 6 (i.e., $\boldsymbol{E_O^*}$ and $\boldsymbol{R_O^*}$) are the output of $\mathcal{G}_r$, respectively. From Table 4, the observations are: (1) Compared with the GNN model RGCN, the model variants show performance improvement, and R-KGPCL(KGE+SUM) achieves the best performance. (2) For the comparison of SUM and GNN, R-KGPCL(FEA/KGE+SUM) shows better

[2]The details of the generation are in Appendix B

Model	NELL995			
	MRR	Hits@10	Hits@3	Hits@1
RGCN	0.500	0.634	0.550	0.419
R-KGP	0.518	0.633	0.564	0.448
R-KGP (+ig)	0.522	0.651	0.571	0.447
R-KGP (+rg)	0.523	0.651	0.571	0.446
R-KGP (+Local)	0.522	0.639	0.571	0.449
R-KGP (+ig-rg)	0.521	0.643	0.570	0.445
R-KGP (+ig-pg)	0.527	0.648	0.580	0.453
R-KGP (+rg-pg)	0.520	0.632	0.568	0.451
R-KGP (+Global)	0.525	0.645	0.576	0.450
R-KGPCL	0.529	0.651	0.577	0.454

Table 5: Ablation study of RGCN-KGPCL on NELL995 dataset. R-KGP is the abbreviation for RGCN-KGP.

performance than R-KGPCL(FEA/KGE+GNN), indicating that the interpolation operation can incorporate latent concepts to enhance entity embeddings. (3) For the comparison of FEA and KGE, KGE is more efficient in most cases regardless of using interpolation or GNN encoder. It shows that the pre-training embeddings can capture concept information effectively. In summary, the good performance of the four configurations shows that modular modifications to KGPCL can still achieve better results than RGCN, verifying the effectiveness of KGPCL as a framework.

4.3 Abalation study (RQ2)

We conduct ablation studies on RGCN to investigate the contributions of KG pooling and unpooling (-KGP), and local and global contrastive learning (+Local/Global) components where ig, rg and pg respectively indicate the embeddings used for CL from the input KG, recovered KG and pooled KG. The results are demonstrated in Table 5. We can observe that the performance increases with the introduction of KG pooling and unpooling, as well as local and global contrastive learning. For the local contrastive learning loss, the improvements on the input and recovered KG are almost the same. For the global contrastive learning loss, the improvements on ig-pg are better than on ig-rg and rg-pg. The reason is that the input KG and pooled KG are directly connected. Although the pooled KG and recovered KG are connected, the output of the recovered KG is affected by the embeddings of both the input KG and pooled KG.

4.4 Visualization (RQ3)

We utilize t-SNE to perform dimension reduction of entity embeddings for visualization. Specifically, we randomly select 350 entities that belong to only one concept in entity-concept mapping established by KG pooling. Then two label settings KGP_Con and DATA_Con are used to indicate the labels from KGPCL and predefined concepts (Niu et al., 2022), and two embedding settings RGCN and RGCN-KGPCL indicate embeddings learned by the corresponding models. Results are shown in Figure 4. From (a) and (b), RGCN-KGPCL+KGP_Con shows better compactness than RGCN+KGP_Con and can make different concepts more distinguishable. From (b) and (c), we observe that entity embeddings learned by KGPCL show clustering with predefined concepts. They demonstrate the effectiveness of U-KG modeling. Furthermore, Figure (c) shows that concepts in DATA_Con are imbalanced compared with KGP_Con. For example, the number of entities in the concept with blue color is much more than others. It confirms the effectiveness of U-KG construction.

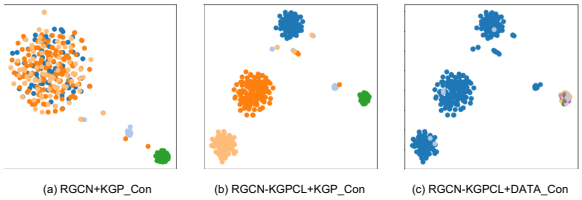

Figure 4: Entity embedding visualization on NELL995.

5 Conclusion

In this paper, we propose a novel framework KGPCL to abstract and model the latent concepts of entities. Specifically, we construct U-KG through KG pooling and unpooling, which abstracts the input KG into a pooled KG and recovers the pooled KG from the input KG. With the U-KG, we use GNN KGEs as the encoder and conventional KGEs as the decoder to learn the embeddings of entities and relations for KG prediction. In addition, we design local and global contrastive learning to jointly optimize the model. Extensive experiments show the effectiveness of our KGPCL.

Limitations

Our method aims to encode implicit concept information of entities to enhance KG prediction. The limitation is that we only consider concepts of entities ignoring other implicit information, such as the properties of relations, and conduct experiments on link prediction task. Thus future directions include encoding more implicit information and enhancing more KG tasks.

Acknowledgments

This work is founded by National Natural Science Foundation of China (NSFC62306276/NSFCU23B2055/NSFCU19B2027), Zhejiang Provincial Natural Science Foundation of China (No. LQ23F020017), Yongjiang Talent Introduction Programme (2022A-238-G), and Fundamental Research Funds for the Central Universities (226-2023-00138). This work was supported by AntGroup.

References

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Advances in neural information processing systems*, pages 2787–2795.

Linlin Chao, Jianshan He, Taifeng Wang, and Wei Chu. 2021. Pairre: Knowledge graph embeddings via paired relation vectors. In *ACL/IJCNLP (1)*, pages 4360–4369. Association for Computational Linguistics.

Sarkar Snigdha Sarathi Das, Arzoo Katiyar, Rebecca J. Passonneau, and Rui Zhang. 2022. Container: Few-shot named entity recognition via contrastive learning. In *ACL (1)*, pages 6338–6353. Association for Computational Linguistics.

Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. 2018. Convolutional 2d knowledge graph embeddings. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Yin Fang, Qiang Zhang, Haihong Yang, Xiang Zhuang, Shumin Deng, Wen Zhang, Ming Qin, Zhuo Chen, Xiaohui Fan, and Huajun Chen. 2022. Molecular contrastive learning with chemical element knowledge graph. In *AAAI*, pages 3968–3976. AAAI Press.

Hongyang Gao and Shuiwang Ji. 2019. Graph u-nets. In *ICML*, volume 97 of *Proceedings of Machine Learning Research*, pages 2083–2092. PMLR.

Junheng Hao, Muhao Chen, Wenchao Yu, Yizhou Sun, and Wei Wang. 2019. Universal representation learning of knowledge bases by jointly embedding instances and ontological concepts. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1709–1719.

Zijie Huang, Daheng Wang, Binxuan Huang, Chenwei Zhang, Jingbo Shang, Yan Liang, Zhengyang Wang, Xian Li, Christos Faloutsos, Yizhou Sun, and Wei Wang. 2023. Concept2box: Joint geometric embeddings for learning two-view knowledge graphs. In *ACL (Findings)*, pages 10105–10118. Association for Computational Linguistics.

Prachi Jain, Pankaj Kumar, Mausam, and Soumen Chakrabarti. 2018. Type-sensitive knowledge base inference without explicit type supervision. In *ACL (2)*, pages 75–80. Association for Computational Linguistics.

Guoliang Ji, Shizhu He, Liheng Xu, Kang Liu, and Jun Zhao. 2015. Knowledge graph embedding via dynamic mapping matrix. In *ACL (1)*, pages 687–696. The Association for Computer Linguistics.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *ICLR (Poster)*.

Junhyun Lee, Inyeop Lee, and Jaewoo Kang. 2019. Self-attention graph pooling. In *ICML*, volume 97 of *Proceedings of Machine Learning Research*, pages 3734–3743. PMLR.

Duantengchuan Li, Tao Xia, Jing Wang, Fobo Shi, Qi Zhang, Bing Li, and Yu Xiong. 2024. Sdformer: A shallow-to-deep feature interaction for knowledge graph embedding. *Knowl. Based Syst.*, 284:111253.

Ren Li, Yanan Cao, Qiannan Zhu, Guanqun Bi, Fang Fang, Yi Liu, and Qian Li. 2022. How does knowledge graph embedding extrapolate to unseen data: A semantic evidence view. In *AAAI*, pages 5781–5791. AAAI Press.

Zhao Li, Xin Liu, Xin Wang, Pengkai Liu, and Yuxin Shen. 2023. Transo: a knowledge-driven representation learning method with ontology information constraints. *World Wide Web (WWW)*, 26(1):297–319.

Zhiping Luo, Wentao Xu, Weiqing Liu, Jiang Bian, Jian Yin, and Tie-Yan Liu. 2022. KGE-CL: contrastive learning of tensor decomposition based knowledge graph embeddings. In *COLING*, pages 2598–2607. International Committee on Computational Linguistics.

Guanglin Niu, Bo Li, Yongfei Zhang, and Shiliang Pu. 2022. CAKE: A scalable commonsense-aware framework for multi-view knowledge graph completion. In *ACL (1)*, pages 2867–2877. Association for Computational Linguistics.

Guanglin Niu, Bo Li, Yongfei Zhang, Shiliang Pu, and Jingyang Li. 2020. Autoeter: Automated entity type representation with relation-aware attention for knowledge graph embedding. In *EMNLP (Findings)*, volume EMNLP 2020 of *Findings of ACL*, pages 1172–1181. Association for Computational Linguistics.

Apoorv Saxena, Soumen Chakrabarti, and Partha P. Talukdar. 2021. Question answering over temporal knowledge graphs. In *ACL/IJCNLP (1)*, pages 6663–6676. Association for Computational Linguistics.

Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. 2018. Modeling relational data with graph convolutional networks. In *European semantic web conference*, 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*.

Zhaoxuan Tan, Zilong Chen, Shangbin Feng, Qingyue Zhang, Qinghua Zheng, Jundong Li, and Minnan Luo. 2023. KRACL: contrastive learning with graph context modeling for sparse knowledge graph completion. In *WWW*, pages 2548–2559. ACM.

Kristina Toutanova and Danqi Chen. 2015. Observed versus latent features for knowledge base and text inference. In *Proceedings of the 3rd workshop on continuous vector space models and their compositionality*, pages 57–66.

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.

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*.

Huijuan Wang, Siming Dai, Weiyue Su, Hui Zhong, Zeyang Fang, Zhengjie Huang, Shikun Feng, Zeyu Chen, Yu Sun, and Dianhai Yu. 2022. Simple and effective relation-based embedding propagation for knowledge representation learning. In *IJCAI*, pages 2755–2761. ijcai.org.

Quan Wang, Zhendong Mao, Bin Wang, and Li Guo. 2017. Knowledge graph embedding: A survey of approaches and applications. *IEEE Transactions on Knowledge and Data Engineering*, 29(12):2724–2743.

Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014. Knowledge graph embedding by translating on hyperplanes. In *AAAI*, pages 1112–1119. AAAI Press.

Wanxu Wei, Yitong Song, and Bin Yao. 2024. Enhancing heterogeneous knowledge graph completion with a novel gat-based approach. *ACM Trans. Knowl. Discov. Data*, 18(4):104:1–104:20.

Wenhan Xiong, Thien Hoang, and William Yang Wang. 2017. Deeppath: A reinforcement learning method for knowledge graph reasoning. In *EMNLP*, pages 564–573. Association for Computational Linguistics.

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

Yuhao Yang, Chao Huang, Lianghao Xia, and Chenliang Li. 2022. Knowledge graph contrastive learning for recommendation. In *SIGIR*, pages 1434–1443. ACM.

Zhitao Ying, Jiaxuan You, Christopher Morris, Xiang Ren, William L. Hamilton, and Jure Leskovec. 2018. Hierarchical graph representation learning with differentiable pooling. In *NeurIPS*, pages 4805–4815.

Zhanqiu Zhang, Jianyu Cai, Yongdong Zhang, and Jie Wang. 2020. Learning hierarchy-aware knowledge graph embeddings for link prediction. In *AAAI*, pages 3065–3072. AAAI Press.

A The details of GNN encoders

The update of entity and relation embeddings in the GNN encoders are summarized in Table 6.

In RGCN and CompGCN, the output embeddings of entity and relation are $\boldsymbol{e^{out}} = \boldsymbol{e^L}$ and $\boldsymbol{r^{out}} = \boldsymbol{r^L}$, where L is the number of aggregation layers. In SEGNN, it concats all the $\boldsymbol{r^l}$ and merges them by a transform matrix to get the relation embeddings. Then the output embeddings of entity and relation are $\boldsymbol{e^{out}} = \boldsymbol{e^L}$ and $\boldsymbol{r^{out}} = \boldsymbol{W^{out}} Concat(\{\boldsymbol{r^l} | l = 1, ..., L\})$.

Considering the details of SEGNN, it obtains the representations of s_i^{rel}, s_i^{ent} and s_i^{tri} through multi-layer aggregation with attention mechanism. Take one GNN layer as an example. For the relation level SE, the calculation of the representation and attention weight is defined as:

$$s_i^{rel} = \sigma\Big(\sum_{(e_j,r_j)\in\mathcal{N}_i} \alpha_{ij}^{rel} W^{rel} \boldsymbol{r}_j\Big) \tag{15}$$

$$\alpha_{ij}^{rel} = \frac{exp(\boldsymbol{r}_j^T \boldsymbol{e}_i)}{\sum_{(e_k,r_k)\in\mathcal{N}_i} exp(\boldsymbol{r}_k^T \boldsymbol{e}_i)} \tag{16}$$

For the entity level SE, the calculations are:

$$s_i^{ent} = \sigma\Big(\sum_{(e_j,r_j)\in\mathcal{N}_i} \alpha_{ij}^{ent} W^{ent} \boldsymbol{e}_j\Big) \tag{17}$$

$$\alpha_{ij}^{ent} = \frac{exp(\boldsymbol{e}_j^T \boldsymbol{e}_i)}{\sum_{(e_k,r_k)\in\mathcal{N}_i} exp(\boldsymbol{e}_k^T \boldsymbol{e}_i)} \tag{18}$$

For the triple level SE, the aggregation function and attention weight computation are:

$$s_i^{tri} = \sigma\Big(\sum_{(e_j,r_j)\in\mathcal{N}_i} \alpha_{ij}^{tri} W^{tri} \phi(\boldsymbol{e}_j, \boldsymbol{r}_j)\Big) \tag{19}$$

$$\alpha_{ij}^{tri} = \frac{exp(\phi(\boldsymbol{e}_j, \boldsymbol{r}_j)^T \boldsymbol{e}_i)}{\sum_{(e_k,r_k)\in\mathcal{N}_i} exp(\phi(\boldsymbol{e}_k, \boldsymbol{r}_k)^T \boldsymbol{e}_i)} \tag{20}$$

where $\mathcal{N}_i$ is the neighbor (head entity, relation) pairs that are connected to e_i. W_{rel}, W_{ent} and W_{tri} are transformation matrices. $\boldsymbol{e}_i, \boldsymbol{e}_j, \boldsymbol{r}_j, \boldsymbol{e}_k$ and $\boldsymbol{r}_k$ are respectively vectors of e_i, e_j, r_j, e_k and r_k. $\phi(\boldsymbol{e}, \boldsymbol{r}) = \boldsymbol{e} * \boldsymbol{r}$ is a composition function. The

	Entity Update	Relation Update
RGCN	$\boldsymbol{e^{l+1}} = \sigma(\sum_{(h,r)\in\mathcal{N}_{in}(e)} \boldsymbol{W_r^l h^l} + \sum_{(r,t)\in\mathcal{N}_{out}(e)} \boldsymbol{W_r^l t^l} + \boldsymbol{W_0^l e^l})$	-
CompGCN	$\boldsymbol{e^{l+1}} = \sigma(\sum_{(h,r)\in\mathcal{N}_{in}(e)} \boldsymbol{W_r^l}\phi(\boldsymbol{h^l}, \boldsymbol{r^l}) + \sum_{(r,t)\in\mathcal{N}_{out}(e)} \boldsymbol{W_r^l}\phi(\boldsymbol{t^l}, \boldsymbol{r^l}) + \boldsymbol{W_0^l e^l})$	$\boldsymbol{r^{l+1}} = \boldsymbol{W_{rel}^l r^l}$
SEGNN	$\boldsymbol{e^{l+1}} = \boldsymbol{e^l} + \boldsymbol{s_{rel}^l} + \boldsymbol{s_{ent}^l} + \boldsymbol{s_{tri}^l}$	-

Table 6: The update of three popular GNN encoders. For the entity e, $\mathcal{N}_{in}(e)$ is the set of the head entity and relation pairs of e, and $\mathcal{N}_{out}(e)$ is the set of relation and tail entity pairs of e. W_r^l is regularized by block-diagonal-decomposition. ϕ is multiplication composition operator defined as $\phi(e_s, e_r) = e_s * e_r$. For SEGNN, $\boldsymbol{s_{rel}^l}$, $\boldsymbol{s_{ent}^l}$ and $\boldsymbol{s_{tri}^l}$ are respectively obtained by aggregating all the connected relations, neighbor entities, and neighbor entity-relation couples.

multi-layer vision of triple level SE is computed as:

$$(s_i^{tri})^l = \sigma(\sum_{(e_j,r_j)\in\mathcal{N}_i} (\alpha_{ij}^{tri})^l (W^{tri})^l \phi(\boldsymbol{e_j^l}, \boldsymbol{r_j^l}) \tag{21}$$

$(s_i^{rel})^l$ and $(s_i^{ent})^l$ are calculated in a similar way. To reduce the parameters, we use one GNN layer to obtain the three level SE representations in our experiments. It is worth noting that the initialization and update of the entity embedding in SEGNN are the same as RGCN and CompGCN. The difference is the setting of relation embeddings. We learn separate relation embeddings for the input KG and pooled KG, denoted as R_S^I and R_S^P. Then output relation embedding of the recovered KG is calculated as $R_S^O = \frac{R_S^I + R_S^P}{2}$.

B Feature embedding generation

Regarding the statistical method, we generate embeddings for the head entities $h_i \in \overleftarrow{R}$ based on two aspects: (a) the relations connected by the head entity, which is $\{r_j|(h_i, r_j, e) \in \mathcal{G}, e \in \mathcal{E}\}$, and (b) the tail entities that connected by the head entity and the current relation $\{t_j|(h_i, r, t_j) \in \mathcal{G}\}$. As to (a), we use a vector $\boldsymbol{h_i^r} \in \mathbb{R}^n$ to represent h_i, where n is the number of relations in $\mathcal{G}$. Each dimension is set as the number of times the head entity is connected to the corresponding relation. As to (b), we use a vector $\boldsymbol{h_i^t} \in \mathbb{R}^m$ to represent h_i, where m is a hyper-parameter. Specially, we first iterate over all entities in the coarse-grained concept $\overleftarrow{R}$ and get the tail entity set $\overleftarrow{R}_T = \{t_j|(h, r, t_j) \in \mathcal{G}, h \in \overleftarrow{R}\}$. Then we count the tail entities in $\overleftarrow{R}_T$ and sort the occurrence number in descending order. To preserve the more informative tail entities, we prune the tail entities with the occurrence number $|\overleftarrow{R}_T|$ or 1. $|\overleftarrow{R}_T|$ equals the number of entities in $\overleftarrow{R}_T$. Next, we select the top m tail entities in the pruned entity set as features. All tail entities are selected when the number of the pruned tail entities is less than m. We set m as 1000 in experiments. Finally, we map each selected tail entity to each dimension of $\boldsymbol{h_i^t}$, where the value of each dimension corresponds to the number of occurrences of the tail entity counted before. After generating the two feature vectors, we define the entity embedding of h_i from the statistical method as $\boldsymbol{h_i}$=[$\boldsymbol{h_i^t}$; $\boldsymbol{h_i^r}$], where ; is the concatenation operation.