Generalized Multi-Relational Graph Convolution Network

Donghan Yu, Yiming Yang, Ruohong Zhang, Yuexin Wu
Carnegie Mellon University
{dyu2,yiming,ruohongz,yuexinw}@cs.cmu.edu

Abstract

Graph Convolutional Networks (GCNs) have received increasing attention in recent machine learning. How to effectively leverage the rich structural information in complex graphs, such as knowledge graphs with heterogeneous types of entities and relations, is a primary open challenge in the field. Most GCN methods are either restricted to graphs with a homogeneous type of edges (e.g., citation links only), or focusing on representation learning for nodes only instead of jointly optimizing the embeddings of both nodes and edges for target-driven objectives. This paper addresses these limitations by proposing a novel framework, namely the GEneralized Multi-relational Graph Convolutional Networks (GEM-GCN), which combines the power of GCNs in graph-based belief propagation and the strengths of advanced knowledge-base embedding methods, and goes beyond. Our theoretical analysis shows that GEM-GCN offers an elegant unification of several well-known GCN methods as specific cases, with a new perspective of graph convolution. Experimental results on benchmark datasets show the advantageous performance of GEM-GCN over strong baseline methods in the tasks of knowledge graph alignment and entity classification.

1 Introduction

Graph Convolution Networks (GCNs) have received increasing attention in recent machine learning research due as powerful methods for graph-based node feature induction and belief propagation, and been successfully applied to many real-world problems, including natural language processing [11, 13], computer vision [32, 12], recommender systems [15, 42], epidemiological forecasting [35], and more. Existing GCNs share the same core idea, i.e., using a graph to identify the neighborhood of each node, and to learn the embedding (vector representation) of that node via recursive aggregation of the neighborhood embeddings. In other words, graph convolution plays the central role in smoothing the learned representations (latent vectors) of nodes based on believe propagation over the entire graph. However, early work in GCNs also have a limitation in common, i.e., graph convolution is only used to learn node embedding conditioned on fixed edges [11], instead of jointly learning the optimal embeddings for both nodes and edges. Later efforts move on the direction of learning of wights of edges in the input graph in parallel with the learning of node embedding [30, 43], which is more powerful than early GCNs and more capable in model adaptation for downstream tasks. However, both the early GCNs and the latter GCNs have a constraint in common, that is, the edges in the input graph must be of homogeneous kind, such as the links in a citation graph or the elements in a co-occurrence count matrix. This constraint (or assumption) significantly limit the applicability of GCNs to a broad range of real-world applications where the capability to model *heterogeneous* relations (edges) is crucial for the true utility of graph-based embedding and inference.

As an indirectly related area, methods for Knowledge Graph (KG) completion (a.k.a. knowledge graph embedding) have been intensively studied in recent years. Various algorithms have been developed for joint optimization of the embeddings of both entities and relations, with respect to the

Preprint. Under review.

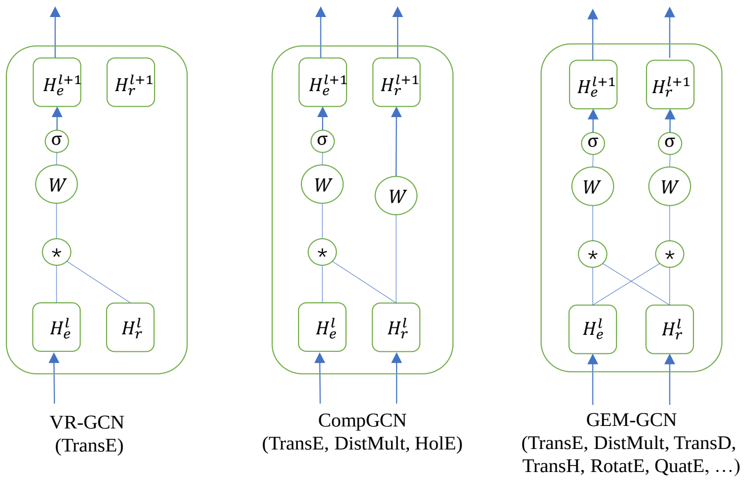

Figure 1: A simple realization of GEM-GCN compared to previous works VR-GCN and CompGCN, where H_e^l and H_r^l means the entity (node) embedding and relation (edge) embedding at layer l respectively. $\star$ denotes the graph convolution operation which aggregates the neighbour information. W is the model parameter of linear transformation and σ is the activation function. The names in the brackets below correspond to the incorporated KG completion models.

task of predicting unknown entity-relation-entity triplets based on observed triplets. Representative approaches include TransE [3], DistMult [40], ComplEx [27], RotatE [26], QuatE [44] and more. The major difference of KG completion methods in contrast to GCNs is that the former do not explicitly leverage the believe propagation power of graph convolution in the representation learning process; instead, entity-relation-entity triplets are treated independently in their objective functions. In other words, those methods lack the ability of using the graph structures to enforce the local/global smoothness in the embedding spaces for entities and relations.

How to jointly leverage the strengths of both GCN models and KG completion methods for task-oriented representation learning of both entities and relations is an open challenge for research, which has not been studied in sufficient depth and is the focus of this paper. Representative works in this direction, or the only methods of this kind so far to our knowledge, are VR-GCN [41] and CompGCN [29]. They use a graph neural network to jointly learn multi-layer latent representations (embeddings) for both entities and relations. However, the *relation embedding* part of the learning process leaves the entity representations out of the picture. In other words, the graph structure is only used to propagate information from embedded nodes plus edges in the optimization of node embedding (which makes sense), but not used to propagate information from embedded nodes in the optimization of edge embedding, which is arguably sub-optimal and a fundamental limitation of those models. Moreover, the existing works did not leverage recent advanced KG completion methods such as RotatE [26] and QuatE [44].

To address the aforementioned open challenge and the limitations of existing approaches, we propose novel framework, namely GEM-GCN (GEneralized Multi-relational Graph Convolution Network). It provides a theoretically sound generalization of existing GCN models, to allow the incorporation of various KG completion methods for task-oriented embeddings of both entities and relations via graph convolution operations. Especially, in order to capture the rich semantics of heterogeneous relations in knowledge graphs, both entity embeddings and relation embeddings in our model are used to enforce optimization of each other in a recursive aggregation process. Figure 1 illustrates the main differences between previous works and our model. A more detailed comparison will be provided in Section 4. Experiment results on benchmark datasets for knowledge graph alignment and entity classification tasks show that GEM-GCN consistently and significantly outperforms other representative baseline methods.

2 Related Background

2.1 Graph Convolutional Network

Graph Convolutional Networks (GCNs) derive its idea from traditional Convolutional Neural Networks (CNNs) by extending convolutional operations onto non-Euclidean data structures. Early trials focus on spectral transformation over adjacency matrices [4, 7] which requires tremendous computational cost of eigen-decomposition. Recent work [11] saves the computation of decomposition by making first order Chebyshev polynomial approximation and popularizes its use while keeping the strong performance over graph-structured data. This change formulates a uniform Message Passing framework for most followup works. Thereafter, while many extension works tries to improve the effectiveness by reweighting edge weights [30] or making residual links [38], several works [29, 20, 21] try to leverage the power of GCNs on knowledge graph by allowing multi-relation edge types to interact in the overall framework. These methods define new message passing routines by updating node representation using relation-specific transformation or interaction with relation embeddings. However, none of them have a symmetrical updating rule for their relation embedding, thus limiting the representation power of relation embeddings. Our method differs in that both entity embeddings and relation embeddings are utilized to update each other, where relation embeddings are also enhanced by neighbour context.

2.2 KG Embedding

Traditional Knowledge Graph (KG) tasks focus on making link or entity predictions. These tasks mostly reduce to modeling (head entity, relation, tail entity) triplets. Many solutions [3, 40, 16, 31] rely on learning embeddings for each part. Specifically, a scoring function f is defined to measure the plausibility of triplets given the embeddings and help update the representation on the training data composed of positive and sampled negative triplets. By using different types of scoring functions, KG embedding methods can reflect different designing criteria, which include translation relation [3, 33], inner product [40], rotational relation [26, 44] and many others [22, 8].

In our work, we borrow the idea of scoring function by adaptively allowing similar embedding learning design in the new GCN framework. We evaluate the performance of different scoring functions proposed in TransE [3], DistMult [40], TransH [33], TransD [9], RotatE [26], and QuatE [44] through our framework on tasks including knowledge graph alignment and entity classification, which proves the effectiveness of our model design and such incorporation.

3 Proposed Method

3.1 Reformulation of Vanilla GCN

In vanilla GCN [11], the multi-layer node embedding is updated as follows (we omit the normalization coefficient part for brevity):

$$\mathbf{m}_v^{l+1} = \sum_{u \in \mathcal{N}(v)} \mathbf{h}_u^l \tag{1}$$

$$\mathbf{h}_v^{l+1} = \sigma(W^l(\mathbf{m}_v^{l+1} + \mathbf{h}_v^l)) \tag{2}$$

where we denote by $\mathbf{h}_v^l$ the embedding of node v at layer l, by $\mathcal{N}(v)$ the set of immediate neighbours of node v, by $\mathbf{m}_v^{l+1}$ the aggregated representation of those neighbors, by $\sigma(\cdot)$ an activation function (e,g., element-wise sigmoid or ReLU), and by W^l the matrix of model parameters to be learned by the GCN. We can reformulate the vanilla GCN by introducing a scoring function f that measures the plausibility of each edge. Edges observed in the graph tend to have higher scores than those that have not been observed. Specifically, for edge (u, v), if we define f as the inner product of the embeddings of the two connected nodes as $f(\mathbf{h}_u, \mathbf{h}_v) = \mathbf{h}_u^T \mathbf{h}_v$, then Equation 1 have the equivalent forms of:

$$\mathbf{m}_v^{l+1} = \sum_{u \in \mathcal{N}(v)} \frac{\partial f(\mathbf{h}_u^l, \mathbf{h}_v^l)}{\partial \mathbf{h}_v^l} = \frac{\partial (\sum_{u \in \mathcal{N}(v)} f(\mathbf{h}_u^l, \mathbf{h}_v^l))}{\partial \mathbf{h}_v^l} \tag{3}$$

It follows that $\mathbf{h}_v^l + \mathbf{m}_v^{l+1}$ can be regarded as one step gradient ascent to maximize the sum of scoring function $\sum_{u \in \mathcal{N}(v)} f(\mathbf{h}_u^l, \mathbf{h}_v^l)$, with learning rate of 1. Furthermore, Equation 2 can be viewed as a generalized projection onto the embedding space for downstream tasks.

The above reformulation provides an explicit view about *what* the vanilla GCN is optimizing, instead of *how* the updates are executed procedurally. More importantly, it sheds light on how to design a more powerful framework to enable more generalized multi-relational graph convolution over knowledge graphs, which we introduce in the next section.

3.2 The New Framework

Since the relations in knowledge graphs are of heterogeneous types, we need to define the scoring function f to measure the plausibility of entity-relation-entity triplets, instead of entity-entity pairs in vanilla GCN. Samely, triplets observed in the knowlede graph tend to have higher scores than those that have not been observed. For each triplet (u, r, v), where u, r, v denote head entity, relation, and tail entity respectively, the scoring value is calculated by $f(\mathbf{h}_u, \mathbf{h}_r, \mathbf{h}_v)$ using their embedding vectors. Note that most knowledge graph embedding techniques can be used to define f.

Analogous to Equation 3, if we denote $\mathbf{h}_v^l$ the embedding of entity v at layer l, the entity updating rules are:

$$\mathbf{m}_v^{l+1} = \sum_{(u,r) \in \mathcal{N}_{\text{in}}(v)} W_r^l \frac{\partial f_{\text{in}}(\mathbf{h}_u^l, \mathbf{h}_r^l, \mathbf{h}_v^l)}{\partial \mathbf{h}_v^l} + \sum_{(u,r) \in \mathcal{N}_{\text{out}}(v)} W_r^l \frac{\partial f_{\text{out}}(\mathbf{h}_v^l, \mathbf{h}_r^l, \mathbf{h}_u^l)}{\partial \mathbf{h}_v^l} \tag{4}$$

$$\mathbf{h}_v^{l+1} = \sigma_{\text{ent}}(\mathbf{m}_v^{l+1} + W_0^l \mathbf{h}_v^l) \tag{5}$$

where $\mathcal{N}_{\text{in}}(v) = \{(u, r) \mid u \xrightarrow{r} v\}$ is the set of immediate entity-relation neighbours of entity v with an in-link while $\mathcal{N}_{\text{out}}(v) = \{(u, r) \mid u \xleftarrow{r} v\}$ is the set of immediate neighbours with an out-link from v. $\mathbf{h}_r^l$ means the embedding of relation r at layer l. Notice that the linear transformation matrix W_r^l is relation-specific, and the scoring functions f_{in} are for the in-link neighbours and f_{out} are for the out-link neighbours, respectively. $\sigma_{\text{ent}}(\cdot)$ denotes the activation function for entity update.

The relation updating rules can be defined in a similar manner:

$$\mathbf{m}_r^{l+1} = \sum_{(u,v) \in \mathcal{N}(r)} \frac{\partial f_r(\mathbf{h}_u^l, \mathbf{h}_r^l, \mathbf{h}_v^l)}{\partial \mathbf{h}_r^l} \tag{6}$$

$$\mathbf{h}_r^{l+1} = \sigma_{\text{rel}}(W_{\text{rel}}^l(\mathbf{m}_r^{l+1} + \mathbf{h}_r^l)) \tag{7}$$

where $\mathcal{N}(r) = \{(u, v) \mid u \xrightarrow{r} v\}$ means the set of immediate entity neighbours of relation r, where the left side of tuple is head entity and the right side is tail entity. $\sigma_{\text{rel}}(\cdot)$ denotes the activation function for relation update. Notice that our framework is very general and can subsume other representative methods, which will be introduced in the next section.

Moreover, the derivative $\partial f / \partial \mathbf{h}$ in Equation 4 and Equation 6 can be calculated by *Auto Differentiation* (AD) package of many existing libraries including Pytorch [17] and Tensorflow [1], which makes our model easy to implement. Notice that this AD happens in the inference process, instead of backpropagation during the training process.

To apply our model on the downstream tasks, denoting the number of layers as L, we use the output entity embedding $\{\mathbf{h}_v^{(L)}\}$ and relation embedding $\{\mathbf{h}_r^{(L)}\}$ of the final layer to construct loss functions. For example, in entity classification task, we use cross-entropy loss based on entity labels; in knowledge graph alignment, we use the distance between the embedding vectors of two entities from different knowledge graphs for loss function. For details of loss functions, please refer to Appendix B.2 and C.2. The training manners are end-to-end in both tasks. In our experiments for these tasks, we set $f_{\text{in}} = f_{\text{out}} = f_r$ for simplicity and $W_r^l = W^l$ to prevent over-parameterization. Moreover, to avoid numerical instabilities and exploding/vanishing gradients when used in a deep neural network model, GEM-GCN applies the following normalization: we replace $\mathbf{m}_v^{l+1}$ in Equation 5 with $\alpha \mathbf{m}_v^{l+1} / (|\mathcal{N}_{\text{in}}(v)| + |\mathcal{N}_{\text{out}}(v)|)$ and $\mathbf{m}_r^{l+1}$ in Equation 7 with $\alpha \mathbf{m}_v^{l+1} / |\mathcal{N}(r)|$, where $|\cdot|$ means the cardinality of a set and α is a hyperparameter which controls the weight of aggregated neighbour information.

4 Unified View of Representative Methods

In the following we provide a unified view of several representative GCN methods for knowledge graph modeling [20, 21, 29], by showing that they are restricted versions under our framework.

4.1 CompGCN

COMPGCN [29] is most relevant to our method. In the $(l+1)$-th layer of COMPGCN, the embeddings of each entity and relation are updated as:

- Entity update:

$$\mathbf{m}_v^{l+1} = \sum_{(u,r)\in\mathcal{N}_{\text{in}}(v)} W_r^l \phi\left(\mathbf{h}_u^l, \mathbf{h}_r^l\right) + \sum_{(u,r)\in\mathcal{N}_{\text{out}}(v)} W_r^l \phi\left(\mathbf{h}_u^l, \mathbf{h}_r^l\right) \tag{8}$$

$$\mathbf{h}_v^{l+1} = \sigma(\mathbf{m}_v^{l+1} + W_0^l \mathbf{h}_v^l) \tag{9}$$

  where $\phi : \mathbb{R}^{d_l} \times \mathbb{R}^{d_l} \rightarrow \mathbb{R}^{d_l}$ is a composition operator which can be element-wise subtraction, element-wise multiplication or circular-correlation [18].

- Relation update:

$$\mathbf{h}_r^{l+1} = W_{\text{rel}}^l \mathbf{h}_r^l \tag{10}$$

Proposition 1 COMPGCN *can be fully recoverd by GEM-GCN when 1)* $f_{in}(\mathbf{h}_u^l, \mathbf{h}_r^l, \mathbf{h}_v^l) = f_{out}(\mathbf{h}_v^l, \mathbf{h}_r^l, \mathbf{h}_u^l) = \phi\left(\mathbf{h}_u^l, \mathbf{h}_r^l\right)^T \mathbf{h}_v^l$*; and 2)* $f_r = 0$*; and 3)* $\sigma_{rel}(\cdot)$ *is the identity function.*

As shown above, in COMPGCN, the relation embedding is only updated by linear transformation. While in GEM-GCN, the relation representation update process aggregates neighbour entity representations, shown in Equation 6 and 7, to capture the rich semantics of heterogeneous relations and learn better context-based relation embeddings. Additionally, our framework is more general since the scoring functions $f(\mathbf{h}_u, \mathbf{h}_r, \mathbf{h}_v)$ are not restricted to be $\phi\left(\mathbf{h}_u, \mathbf{h}_r\right)^T \mathbf{h}_v$, and other forms of knowledge graph embedding techniques such as TransH [33], TransD [9], MLP [8], and NTN [22] can also be incorporated.

4.2 R-GCN

R-GCN [20] extends vanilla GCN with relation-specific linear transformations, without considering relation representations. The embedding update can be listed as follows:

- Entity update:

$$\mathbf{m}_v^{l+1} = \sum_{(u,r)\in\mathcal{N}_{\text{in}}(v)} W_r^l \mathbf{h}_u^l + \sum_{(u,r)\in\mathcal{N}_{\text{out}}(v)} W_r^l \mathbf{h}_u^l \tag{11}$$

$$\mathbf{h}_v^{l+1} = \sigma(\mathbf{m}_v^{l+1} + W_0^l \mathbf{h}_v^l) \tag{12}$$

- No relation update.

Proposition 2 *R-GCN can be fully recoverd by GEM-GCN when 1)* $f_{in}(\mathbf{h}_u^l, \mathbf{h}_r^l, \mathbf{h}_v^l) = f_{out}(\mathbf{h}_v^l, \mathbf{h}_r^l, \mathbf{h}_u^l) = (\mathbf{h}_u^l)^T \mathbf{h}_v^l$*; and 2)* $\mathbf{h}_r^l = 0$ *(no relation embedding).*

4.3 W-GCN

W-GCN [21] treats the relation as learnable weights of edges, and applies vanilla GCN on the weighted simple graph. The update process can be written as:

- Entity update:

$$\mathbf{m}_v^{l+1} = \sum_{(u,r)\in\mathcal{N}_{\text{in}}(v)} W^l(\alpha_r^l \mathbf{h}_u^l) + \sum_{(u,r)\in\mathcal{N}_{\text{out}}(v)} W^l(\alpha_r^l \mathbf{h}_u^l) \tag{13}$$

$$\mathbf{h}_v^{l+1} = \sigma(\mathbf{m}_v^{l+1} + W^l \mathbf{h}_v^l) \tag{14}$$

  where $\alpha_r^l \in \mathbb{R}$ is a relation-specific learnable parameter.

- No relation update.

Proposition 3 *W-GCN can be fully recoverd by GEM-GCN when 1)* $f_{in}(\mathbf{h}_u^l, \mathbf{h}_r^l, \mathbf{h}_v^l) = f_{out}(\mathbf{h}_v^l, \mathbf{h}_r^l, \mathbf{h}_u^l) = (\mathbf{h}_u^l)^T\mathbf{h}_v^l$*; and 2)* $W_r^l = W^l\alpha_r^l$*; and 3)* $\mathbf{h}_r^l = 0$ *(no relation embedding).*

5 Experiments

5.1 Basic Settings

In this section, we conduct extensive experiments on two well-known tasks of knowledge graphs, *graph alignment* and *entity classification*, to demonstrate the effectiveness of our model. The computing infrastructure we use is NVIDIA RTX 2080Ti GPU in all the experiments. All the experiments are conducted by 5 independent runs with different random seeds. In this paper, following previous works [20, 29], the input features of entities and relations are random vectors initialized by truncated normal distribution so that our model will rely solely on graph structure. We leave the combination of other features such as text description of entities and relations as future work. Our model is evaluated by combining the following representative knowledge graph embedding methods: TransE [3], DistMult [40], TransH [33], TransD [9] RotatE [26] and QuatE [44], where the details are shown in Appendix A. Note that TransE [3] and DistMult [40] are widely used knowledge graph embedding techniques since they are simple and effective. RotatE [26] and QuatE [44] are recent proposed models and achieves state-of-the-art results in commonly used benchmark datasets for knowledge graph completion. TransH [33] and TransD [9] are based on the extensions of TransE [3], and can not be incorporated by COMPGCN [29]. We leave other embedding methods for future work.

5.2 Knowledge Graph Alignment

Knowledge graph alignment refer to the task which aims to find entities/relations in two different knowledge graphs KG_1 and KG_2 that represent the same real-world entity/relation. To apply our GCN framework on knowledge graph alignment, we follow the previous work [34] which utilize two GCNs with shared parameters to model two knowledege graphs separately, then the output embeddings of each final layer are used for entity/relation alignment. A detailed description can be found in Appendix B.

5.2.1 Datasets and Baselines

We use DBP15K [23] which contains three datasets built from multi-lingual DBpedia, namely $\text{DBP}_{\text{ZH-EN}}$ (Chinese-English), $\text{DBP}_{\text{ZH-EN}}$ (Japanese-English) and $\text{DBP}_{\text{FR-EN}}$ (French-English) for knowledge graph alignment. A summary statistics of these datasets is provided in Appendix B.1. Following previous work [5, 25], we report Hits@1, Hits@10, Mean Reciprocal Rank (MRR) to evaluate the alignment performance. Higher Hits@1, Hits@10 and MRR scores indicate better performance.

To demonstrate the effectiveness of GEM-GCN, we compare it with several representative multi-relational GCN baseline methods including R-GCN [20], W-GCN [21], VR-GCN [41] and COMPGCN [29]. We also include other baseline methods which are designed specifically for knowledge graph alignment task for a comprehensive comparison: MTransE [6], IPTransE [45], JAPE [23], AlignE [24], GCN-Align [34], MuGNN [5], and AliNet [25]. Note that since our model solely relies on structure information, we do not compare with the alignment models which incorporate the surface information of entities into their representations like [39, 36].

5.2.2 Results of Entity Alignment

Table 1 shows the experiment results in DBP15K datasets comparing with all the baseline methods. Our model achieves the best or highly competitive results in all the three datasets, outperforming the baseline methods by a large margin. Specifically, our model outperforms the best baseline methods by 5.7%, 6.5%, and 6.6% in MRR on $\text{DBP}_{\text{ZH-EN}}$, $\text{DBP}_{\text{JA-EN}}$, and $\text{DBP}_{\text{FR-EN}}$ respectively. Note that even equipped with the recent state-of-the-art knowledge graph embedding methods, i.e., RotatE [26] and QuatE [44], COMPGCN [29] still obtains much lower performance than GEM-GCN. The results demonstrate the effectiveness of graph convolution updating for relation embeddings.

Table 1: Experiment results in knowledge graph entity alignment task on DBP15K datasets, where the average results over 5 different runs are reported. * indicate that results are directly taken from [25]. The results of VR-GCN [41] are directly taken from the original paper. The highest score are marked in **bold**. COMPGCN marked with † incorporates the composition operations in RotatE [26] and QuatE [44] while original COMPGCN [29] only contains subtraction, multiplication and circular-correlation operations.

Models	$\text{DBP}_{\text{ZH-EN}}$			$\text{DBP}_{\text{JA-EN}}$			$\text{DBP}_{\text{FR-EN}}$		
	MRR	H@1	H@10	MRR	H@1	H@10	MRR	H@1	H@10
MTransE*[6]	0.364	30.8	61.4	0.349	27.9	57.5	0.335	24.4	55.6
IPTransE*[45]	0.516	40.6	73.5	0.474	36.7	69.3	0.451	33.3	68.5
JAPE*[23]	0.490	41.2	74.5	0.476	36.3	68.5	0.430	32.4	66.7
AlignE*[24]	0.581	47.2	79.2	0.563	44.8	78.9	0.599	48.1	82.4
GCN-Align*[34]	0.549	41.3	74.4	0.546	39.9	74.5	0.532	37.3	74.5
MuGCN*[5]	0.611	49.4	**84.4**	0.621	50.1	**85.7**	0.621	49.5	87.0
AliNet*[25]	0.628	53.9	82.6	0.645	54.9	83.1	0.657	55.2	85.2
R-GCN*[20]	0.564	46.3	73.4	0.571	47.1	75.4	0.570	46.9	75.8
W-GCN [21]	0.553	43.6	73.8	0.554	41.2	74.7	0.541	39.8	74.4
VR-GCN [41]	0.501	38.0	73.3	0.470	35.2	72.2	0.495	36.1	75.1
COMPGCN[29]	0.605	49.4	81.2	0.614	50.4	82.2	0.625	50.5	85.0
COMPGCN†	0.628	52.8	81.1	0.629	52.8	81.5	0.641	52.6	85.4
GEM-GCN	**0.664**	**56.2**	84.2	**0.670**	**57.0**	85.2	**0.683**	**57.2**	**88.5**

Table 2: Knowledge graph entity alignment results over 5 different runs on $\text{DBP}_{\text{ZH-EN}}$ by incorporating different knowledge graph embedding methods into our model. The highest score are marked in **bold**.

Model	MRR	H@1	H@10
GEM-GCN (TransE)	0.648 ± 0.003	54.3 ± 0.3	83.4 ± 0.3
GEM-GCN (TransH)	0.650 ± 0.003	54.3 ± 0.4	**84.4 ± 0.3**
GEM-GCN (DistMult)	0.621 ± 0.003	52.0 ± 0.4	80.3 ± 0.4
GEM-GCN (TransD)	0.635 ± 0.003	53.1 ± 0.3	82.7 ± 0.4
GEM-GCN (RotatE)	0.653 ± 0.004	54.9 ± 0.4	83.8 ± 0.4
GEM-GCN (QuatE)	**0.664 ± 0.004**	**56.2 ± 0.4**	84.2 ± 0.4

Table 3: Knowledge graph relation alignment results over 5 different runs on DBP15K datasets. All the models are incorporated with the same KG completion method TransE [3].The highest score are marked in **bold**.

Model	$\text{DBP}_{\text{ZH-EN}}$	$\text{DBP}_{\text{JA-EN}}$	$\text{DBP}_{\text{FR-EN}}$
VR-GCN [41]	0.352 ± 0.006	0.335 ± 0.008	0.280 ± 0.017
COMPGCN [29]	0.366 ± 0.007	0.347 ± 0.009	0.284 ± 0.015
GEM-GCN	**0.514 ± 0.006**	**0.466 ± 0.011**	**0.412 ± 0.021**

We also report the performance of our proposed model with different knowledge graph embedding techniques, including TransE [3], DistMult [40], TransH [33], TransD [9], RotatE [26], and QuatE [44], as shown in Table 2. Note that the choice of embedding techniques does have a large impact on the performance, and QuatE [44] achieves the best results, which is reasonable since it also outperforms other methods in knowledge graph completion task, and satisfies the essential of relational representation learning (i.e., modeling symmetry, anti-symmetry and inversion relation).

5.2.3 Results of Relation Alignment

To directly test the effect of our proposed relation embedding update process, we use the relation embedding for the relation alignment task mentioned in Appendix B. The results over 5 different runs are shown in Table 3, where we see that our model significantly outperforms COMPGCN [29] and VR-GCN [41]. This demonstrate the importance of incorporating entity representation into the update of relation embeddings. Note that we do not compare with R-GCN [20] and W-GCN [21] since relation embeddings are not involved in their models.

Table 4: The mean and standard deviation over 5 different runs of classification accuracy on AM and WN datasets for multi-class classification task and Precision@1 (P@1), Precision@5 (P@5), NDCG@5 (N@5) on FB15K dataset for multi-label classification task respectively. * indicates the results that are directed taken from [29].

Models	AM	WN	FB15K		
			P@1	P@5	N@5
GCN	86.2 ± 1.4	53.4 ± 0.2	86.1 ± 0.3	69.0 ± 0.3	82.7 ± 0.2
R-GCN	89.3*	55.1 ± 0.6	91.7 ± 0.6	73.0 ± 0.4	89.5 ± 0.6
W-GCN	90.2 ± 0.9*	54.2 ± 0.5	91.2 ± 0.6	72.8 ± 0.3	88.6 ± 0.5
COMPGCN	90.6 ± 0.2*	55.9 ± 0.4	92.5 ± 0.1	74.0 ± 0.3	90.1 ± 0.2
GEM-GCN	**91.2 ± 0.2**	**57.8 ± 0.5**	**94.3 ± 0.2**	**74.7 ± 0.2**	**91.6 ± 0.2**

5.3 Knowledge Graph Entity Classification

Entity Classification is the task of predicting the labels of entities in a given knowledge graph. We follow previous work [20, 29] to use the entity output of the last layer in GEM-GCN for label classification.

5.3.1 Datasets and Baselines

We conduct experiments on the following datasets: AM [19], WN [3, 28], and FB15K [3, 37]. The details and statistics of these datasets are shown in Appendix C.1. Each entity in AM and WN datasets has at most one label while entities in FB15K can have multiple labels[1]. In AM and WN, Accuracy are reported to evaluate the entity classification performance. While in FB15K, we report Precision@1 (P@1), Precision@5 (P@5) and NDCG@5 (N@5).

We compare GEM-GCN with vanilla GCN [11] and the relation-based GCN models including R-GCN [20], W-GCN [21] and COMPGCN [29]. For vanilla GCN, we transform the multi-relational graphs to simple graphs with only entities, by setting the weight of edge between two entities as the number of their relations.

5.3.2 Results

The experiment results over 5 different runs are shown in Table 4, where the classification accuracy is reported on AM and WN datasets, and P@1, P@5, N@5 are reported on FB15K dataset. From these results, GEM-GCN outperforms all the baseline GCN methods, which demonstrate the effectiveness of our proposed model in entity classification task, including multi-class classification and multi-label classification. We report the best results of COMPGCN [29] and GEM-GCN incorporated with different knowledge graph embedding methods, and surprisingly, combining TransE [3] achieves the highest performance. Additionally, we see that vanilla GCN performs worse than any relational GCNs in all the datasets, which demonstrates that relation modelling is significant for entity classification.

6 Conclusion

This paper presents a novel framework which leverages various knowledge graph embedding methods into GCNs for multi-relational graph modelling, and more importantly, update both entity and relation representation using graph convolution operation. We show that our model originates from a new intuition behind graph convolution in the view of generalized projected gradient ascent, and subsumes other representative methods as its special and restricted cases. Experiments show that the proposed model obtains the state-of-the-art results in benchmark datasets of two well-known tasks: knowledge graph alignment and entity classification. In the future, we plan to apply our framework into a broader range of domains containing knowledge graphs, including Q&A, recommender system, computer vision and time series analysis. It's also worth exploring to go beyond triplets and extend our framework for knowledge hypergraphs.

[1]Please refer to [28] and [37] for details of label collection in WN and FB15K datasets respectively.

Broader Impact

We studied the problem of applying graph convolutional networks into knowledge graphs. Knowledge graphs, which are more general and complicated than homogeneous graphs, widely exist in real-world scenarios and have been effectively used in many NLP applications including Language Modeling and Question Answering. This paper aims to obtain better representation of knowledge graphs and can potentially benefits any KG-involved applications. However, chances of privacy concern may also exist like many other machine learning models if not properly framed with privacy protected techniques.

References

[1] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: A system for large-scale machine learning. In *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)*, pages 265–283, 2016.

[2] Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 1247–1250, 2008.

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

[4] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. Spectral networks and locally connected networks on graphs. *arXiv preprint arXiv:1312.6203*, 2013.

[5] Yixin Cao, Zhiyuan Liu, Chengjiang Li, Juanzi Li, and Tat-Seng Chua. Multi-channel graph neural network for entity alignment. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1452–1461, 2019.

[6] Muhao Chen, Yingtao Tian, Mohan Yang, and Carlo Zaniolo. Multilingual knowledge graph embeddings for cross-lingual knowledge alignment. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, pages 1511–1517, 2017.

[7] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in neural information processing systems*, pages 3844–3852, 2016.

[8] Xin Dong, Evgeniy Gabrilovich, Geremy Heitz, Wilko Horn, Ni Lao, Kevin Murphy, Thomas Strohmann, Shaohua Sun, and Wei Zhang. Knowledge vault: A web-scale approach to probabilistic knowledge fusion. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 601–610, 2014.

[9] Guoliang Ji, Shizhu He, Liheng Xu, Kang Liu, and Jun Zhao. Knowledge graph embedding via dynamic mapping matrix. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 687–696, 2015.

[10] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[11] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[12] Loic Landrieu and Martin Simonovsky. Large-scale point cloud semantic segmentation with superpoint graphs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4558–4567, 2018.

[13] Diego Marcheggiani and Ivan Titov. Encoding sentences with graph convolutional networks for semantic role labeling. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1506–1515, 2017.

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

[15] Federico Monti, Davide Boscaini, Jonathan Masci, Emanuele Rodola, Jan Svoboda, and Michael M Bronstein. Geometric deep learning on graphs and manifolds using mixture model cnns. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5115–5124, 2017.

[16] Maximilian Nickel, Lorenzo Rosasco, and Tomaso Poggio. Holographic embeddings of knowledge graphs. In *Thirtieth Aaai conference on artificial intelligence*, 2016.

[17] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.

[18] Lawrence R Rabiner and Bernard Gold. Theory and application of digital signal processing. *Englewood Cliffs, NJ, Prentice-Hall, Inc., 1975. 777 p.*, 1975.

[19] Petar Ristoski, Gerben Klaas Dirk De Vries, and Heiko Paulheim. A collection of benchmark datasets for systematic evaluations of machine learning on the semantic web. In *International Semantic Web Conference*, pages 186–194. Springer, 2016.

[20] Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne Van Den Berg, Ivan Titov, and Max Welling. Modeling relational data with graph convolutional networks. In *European Semantic Web Conference*, pages 593–607. Springer, 2018.

[21] Chao Shang, Yun Tang, Jing Huang, Jinbo Bi, Xiaodong He, and Bowen Zhou. End-to-end structure-aware convolutional networks for knowledge base completion. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 3060–3067, 2019.

[22] Richard Socher, Danqi Chen, Christopher D Manning, and Andrew Ng. Reasoning with neural tensor networks for knowledge base completion. In *Advances in neural information processing systems*, pages 926–934, 2013.

[23] Zequn Sun, Wei Hu, and Chengkai Li. Cross-lingual entity alignment via joint attribute-preserving embedding. In *International Semantic Web Conference*, pages 628–644. Springer, 2017.

[24] Zequn Sun, Wei Hu, Qingheng Zhang, and Yuzhong Qu. Bootstrapping entity alignment with knowledge graph embedding. In *IJCAI*, pages 4396–4402, 2018.

[25] Zequn Sun, Chengming Wang, Wei Hu, Muhao Chen, Jian Dai, Wei Zhang, and Yuzhong Qu. Knowledge graph alignment network with gated multi-hop neighborhood aggregation. *arXiv preprint arXiv:1911.08936*, 2019.

[26] Zhiqing Sun, Zhi-Hong Deng, Jian-Yun Nie, and Jian Tang. Rotate: Knowledge graph embedding by relational rotation in complex space. *arXiv preprint arXiv:1902.10197*, 2019.

[27] Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. Complex embeddings for simple link prediction. In *International Conference on Machine Learning*, pages 2071–2080, 2016.

[28] Ke Tu, Peng Cui, Xiao Wang, Fei Wang, and Wenwu Zhu. Structural deep embedding for hyper-networks. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[29] Shikhar Vashishth, Soumya Sanyal, Vikram Nitin, and Partha Talukdar. Composition-based multi-relational graph convolutional networks. *arXiv preprint arXiv:1911.03082*, 2019.

[30] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

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

[32] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E Sarma, Michael M Bronstein, and Justin M Solomon. Dynamic graph cnn for learning on point clouds. *arXiv preprint arXiv:1801.07829*, 2018.

[33] Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. Knowledge graph embedding by translating on hyperplanes. In *Twenty-Eighth AAAI conference on artificial intelligence*, 2014.

[34] Zhichun Wang, Qingsong Lv, Xiaohan Lan, and Yu Zhang. Cross-lingual knowledge graph alignment via graph convolutional networks. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 349–357, 2018.

[35] Yuexin Wu, Yiming Yang, Hiroshi Nishiura, and Masaya Saitoh. Deep learning for epidemiological predictions. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, pages 1085–1088. ACM, 2018.

[36] Yuting Wu, Xiao Liu, Yansong Feng, Zheng Wang, Rui Yan, and Dongyan Zhao. Relation-aware entity alignment for heterogeneous knowledge graphs. *arXiv preprint arXiv:1908.08210*, 2019.

[37] Ruobing Xie, Zhiyuan Liu, Jia Jia, Huanbo Luan, and Maosong Sun. Representation learning of knowledge graphs with entity descriptions. In *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

[38] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826*, 2018.

[39] Kun Xu, Liwei Wang, Mo Yu, Yansong Feng, Yan Song, Zhiguo Wang, and Dong Yu. Cross-lingual knowledge graph alignment via graph matching neural network. *arXiv preprint arXiv:1905.11605*, 2019.

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

[41] Rui Ye, Xin Li, Yujie Fang, Hongyu Zang, and Mingzhong Wang. A vectorized relational graph convolutional network for multi-relational network alignment. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, pages 4135–4141, 2019.

[42] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L Hamilton, and Jure Leskovec. Graph convolutional neural networks for web-scale recommender systems. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 974–983. ACM, 2018.

[43] Donghan Yu, Ruohong Zhang, Zhengbao Jiang, Yuexin Wu, and Yiming Yang. Graph-revised convolutional network. *arXiv preprint arXiv:1911.07123*, 2019.

[44] Shuai Zhang, Yi Tay, Lina Yao, and Qi Liu. Quaternion knowledge graph embeddings. In *Advances in Neural Information Processing Systems*, pages 2731–2741, 2019.

[45] Hao Zhu, Ruobing Xie, Zhiyuan Liu, and Maosong Sun. Iterative entity alignment via joint knowledge embeddings. In *IJCAI*, pages 4258–4264, 2017.

A Scoring Functions of Knowledge Graph Embedding Methods

Our model is evaluated by combining the following representative knowledge graph embedding methods, with embedding of head entity, relation, and tail entity denoted as $\mathbf{h}_u$, $\mathbf{h}_r$, and $\mathbf{h}_v$ respectively. For each method, we show the corresponding scoring function in GEM-GCN, which may be slightly different with the original scoring function.

- TransE [3]: For $\mathbf{h}_u, \mathbf{h}_r, \mathbf{h}_v \in \mathbb{R}^d$,
$$f(\mathbf{h}_u, \mathbf{h}_r, \mathbf{h}_v) = -\|\mathbf{h}_u + \mathbf{h}_r - \mathbf{h}_v\|_2^2. \tag{15}$$
- DistMult [40]: For $\mathbf{h}_u, \mathbf{h}_r, \mathbf{h}_v \in \mathbb{R}^d$.
$$f(\mathbf{h}_u, \mathbf{h}_r, \mathbf{h}_v) = \mathbf{h}_u^T \operatorname{diag}(\mathbf{h}_r)\mathbf{h}_v. \tag{16}$$
- TransH [33]: For $\mathbf{h}_u, \mathbf{h}_v \in \mathbb{R}^d, \mathbf{h}_r \in \mathbb{R}^{2d}$, and $\mathbf{h}_{r1}, \mathbf{h}_{r2} \in \mathbb{R}^d$,
$$f(\mathbf{h}_u, \mathbf{h}_r, \mathbf{h}_v) = -\|\mathbf{h}'_u + \mathbf{h}_{r2} - \mathbf{h}'_v\|_2^2, \tag{17}$$
$$\mathbf{h}'_u = \mathbf{h}_u - \mathbf{h}_{r1}^T \mathbf{h}_u \mathbf{h}_{r1}, \tag{18}$$
$$\mathbf{h}'_v = \mathbf{h}_v - \mathbf{h}_{r1}^T \mathbf{h}_v \mathbf{h}_{r1}, \tag{19}$$
$$\mathbf{h}_r = [\mathbf{h}_{r1}; \mathbf{h}_{r2}], \tag{20}$$
where $[\cdot;\cdot]$ means concatenation of two vectors.
- TransD [9]: For $\mathbf{h}_u, \mathbf{h}_v, \mathbf{h}_r \in \mathbb{R}^{2d}$, and $\mathbf{h}_{u1}, \mathbf{h}_{u2}, \mathbf{h}_{v1}, \mathbf{h}_{v2}, \mathbf{h}_{r1}, \mathbf{h}_{r2} \in \mathbb{R}^d$,
$$f(\mathbf{h}_u, \mathbf{h}_r, \mathbf{h}_v) = -\|\mathbf{h}'_u + \mathbf{h}_{r2} - \mathbf{h}'_v\|_2^2, \tag{21}$$
$$\mathbf{h}'_u = \mathbf{h}_{u1} + \mathbf{h}_{u2}^T \mathbf{h}_{u1} \mathbf{h}_{r1}, \tag{22}$$
$$\mathbf{h}'_v = \mathbf{h}_{v1} - \mathbf{h}_{v2}^T \mathbf{h}_{v1} \mathbf{h}_{r1}, \tag{23}$$
$$\mathbf{h}_u = [\mathbf{h}_{u1}; \mathbf{h}_{u2}], \mathbf{h}_v = [\mathbf{h}_{v1}; \mathbf{h}_{v2}], \mathbf{h}_r = [\mathbf{h}_{r1}; \mathbf{h}_{r2}], \tag{24}$$
where $[\cdot;\cdot]$ means concatenation of two vectors.
- RotatE [26]: For $\mathbf{h}_u, \mathbf{h}_r, \mathbf{h}_v \in \mathbb{C}^d$,
$$f(\mathbf{h}_u, \mathbf{h}_r, \mathbf{h}_v) = -\|\mathbf{h}_u \circ \mathbf{h}_r - \mathbf{h}_v\|_2^2, \tag{25}$$
where $\circ$ denotes element-wise product and the modulus of any element in $\mathbf{h}_r$ is 1, i.e. $|\mathbf{h}_r[i]| = 1 \ \forall i \in \{1, 2, \cdots, d\}$. The norm of complex vector is defined as $\|\mathbf{v}\|_p = \sqrt[p]{\sum |\mathbf{v}_i|^p}$.
- QuatE [44]: For $\mathbf{h}_u, \mathbf{h}_r, \mathbf{h}_v \in \mathbb{H}^d$,
$$f(\mathbf{h}_u, \mathbf{h}_r, \mathbf{h}_v) = \mathbf{h}_u \otimes \mathbf{h}_r \cdot \mathbf{h}_v, \tag{26}$$
where $\otimes$ and $\cdot$ denote Hamilton Product and Inner Product in the hypercomplex space respectively.

B Details of Knowledge Graph Alignment Task

To align from one knowledge graph KG_1 to another knowledge graph KG_2, for a specific entity/relation u in KG_1, we compute the L1-distance between u and each entity/relation v in KG_2 using their embeddings output from each final layer $\mathbf{h}_u$ and $\mathbf{h}_v$, and returns a ranked list of entities/relations as candidate alignments based on the distances. The alignment can be also performed from KG_2 to KG_1. In the experiments, we report the averaged results of both directions of KG alignment. Specifically,

1) For entity alignment task, following previous work [34, 5, 25], we randomly split 30% of aligned entities for training and the rest for testing. We further set aside 30% of the training set as a validation set for hyperparameter tuning, and retrain the model on the whole training set to obtain test performance.

2) We also test on relation alignment task to demonstrate the importance of our proposed relation embedding update process. Since the number of reference aligned relations is very small, we train the model on the entity alignment task mentioned above and directly use the trained relation embedding for relation alignment (as zero-shot evaluation).

B.1 Data Statistics

DBP15K [23] contains three datasets built from multi-lingual DBpedia, namely DBP$_{\text{ZH-EN}}$ (Chinese-English), DBP$_{\text{ZH-EN}}$ (Japanese-English) and DBP$_{\text{FR-EN}}$ (French-English) for knowledge graph alignment. A summary statistics of these datasets is shown in Table 5.

Table 5: Dataset statistics of DBP15K for knowledge graph alignment task, including number of entities, relations, triplets of each knowledge graph and the number of aligned entities and relations.

Datasets		#Entities	#Relations	#Triplets	#Aligned Entities	#Aligned Relations
DBP$_{\text{ZH-EN}}$	Chinese	66,469	2,830	153,929	15,000	891
	English	98,125	2,317	237,674		
DBP$_{\text{JA-EN}}$	Japanese	65,744	2,043	164,373	15,000	582
	English	95,680	2,096	233,319		
DBP$_{\text{FR-EN}}$	French	66,858	1,379	192,191	15,000	75
	English	105,889	2,209	278,590		

B.2 Loss Function

We denote the training entity alignment set as $S = \{(u, v)\}$, where u is the entity in KG_1 while entity v belongs to KG_2 and they refer to the same real-world entity. For the loss function, we follow previous work [34] to use margin-based ranking loss:

$$\mathcal{L} = \sum_{(u,v)\in S} \sum_{(u',v')\in S'_{(u,v)}} l(u, v, u', v') \tag{27}$$

$$l(u, v, u', v') = [\|\mathbf{h}_u - \mathbf{h}_v\|_1 + \gamma - \|\mathbf{h}_{u'} - \mathbf{h}_{v'}\|_1]_+ \tag{28}$$

where $[x]_+ = \max\{0, x\}$. $S'_{(u,v)}$ denotes the set of negative entity alignments constructed by corrupting (u, v), i.e. replacing u or v with a randomly chosen entity in KG_1 or KG_2. γ means the margin hyper-parameter separating positive and negative entity alignments. Suppose for each positive entity alignment, we randomly choose k negtive alignments. Following [34], we set $\gamma = 3$ and $k = 5$.

B.3 Implementation Details

We perform grid search for hyperparameters in the following values: the learning rate l_r in $\{0.001, 0.005, 0.01, 0.05\}$, α in $\{0.1, 0.2, \cdots, 0.9\}$, the hidden dimension d for entity and relation in $\{50, 100, 200, 300\}$, the number of layers L in $\{1, 2, 3, 4, 5\}$. The final selected setting is $l_r = 0.01$, $\alpha = 0.3$, $d = 200$, and $L = 4$. The activation function is set as ReLU. We train our model in full-batch setting using Adam [10].

C Details of Knowledge Graph Entity Classification Task

C.1 Data Statistics

AM [19] contains relationship between different artifacts in Amsterdam Museum. WN [3, 28] consists of a collection of triplets (synset, relation, synset) extracted from WordNet 3.0 [14]. FB15K [3, 37] is extracted from a typical large-scale knowledge graph Freebase [2] . The statistics of these datasets are shown in Table 6.

For AM dataset, we follow the train/test split convention as [19, 20]. As for WN and FB15K datasets, we randomly split the labeled entities into train/valid/test by the ratio of 10%/10%/80% for semi-supervised learning.

C.2 Loss Function

In this paper we conduct experiments on both multi-class classification and multi-label classification. Denoting N as the number of entities, C as the number of classes, and $\mathcal{Y}_L$ as the set of entity indices

Table 6: Number of entities, relations, edges and classes along with the number of labeled entities for each dataset in entity classification task. Labeled denotes the subset of entities that have labels and that are to be classified. Classes denotes the total number of categories of labels.

Datasets	AM	WN	FB15K
#Entities	1,666,764	40,551	14,904
#Relations	133	18	1,341
#Triplets	5,988,321	145,966	579,654
#Labeled	1,000	31,943	13,445
#Classes	11	24	50

that have labels. For multi-class classification, we use the following loss:

$$\mathcal{L} = -\sum_{u \in \mathcal{Y}_L} \sum_{c=1}^{C} Y_{uc} \ln \widehat{Y}_{uc} \tag{29}$$

where $Y_{uc} = 1$ means that the true class of entity u is c, otherwise $Y_{uc} = 0$. $\widehat{Y} \in \mathbb{R}^{N \times C}$ is the output of GCN model, which comes after a row-wise softmax function.

For multi-label classification, we use:

$$\mathcal{L} = -\sum_{u \in \mathcal{Y}_L} \sum_{c=1}^{C} \Big[Y_{uc} \ln \widehat{Y}_{uc} \; + (1 - Y_{uc}) \ln(1 - \widehat{Y}_{uc}) \Big] \tag{30}$$

$Y_{uc} = 1$ means that entity u contains label c, otherwise $Y_{uc} = 0$. $\widehat{Y} \in \mathbb{R}^{N \times C}$ is the output of GCN model, which comes after an element-wise sigmoid function.

C.3 Implementation Details

Grid search are conducted for hyperparameters in the following values: the learning rate l_r in $\{0.001, 0.005, 0.01, 0.05\}$, α in $\{0.1, 0.2, \cdots, 0.9\}$, the hidden dimension d for entity and relation in $\{8, 16, 32, 64\}$, the number of layers L in $\{1, 2, 3, 4, 5\}$. The final selected settings are: for AM dataset, we choose $l_r = 0.01$, $\alpha = 0.3$, $d = 32$, and $L = 4$; for WN dataset, we choose $l_r = 0.01$, $\alpha = 0.5$, $d = 32$, and $L = 4$; for FB15k dataset, we choose $l_r = 0.01$, $\alpha = 0.5$, $d = 32$, and $L = 2$. The activation function is set as ReLU. We train our model in full-batch setting using Adam [10].