Robust Embedding with Multi-Level Structures for Link Prediction

Zihan Wang[1,2], **Zhaochun Ren**[3], **Chunyu He**[1,2], **Peng Zhang**[1,2*] and **Yue Hu**[1,2]

[1]Institute of Information Engineering, Chinese Academy of Sciences
[2]School of Cyber Security, University of Chinese Academy of Sciences
[3]Shandong University

wangzihan1995@iie.ac.cn, zhaochun.ren@sdu.edu.cn, {hechunyu, pengzhang, huyue}@iie.ac.cn

Abstract

Knowledge Graph (KG) embedding has become crucial for the task of link prediction. Recent work applies encoder-decoder models to tackle this problem, where an encoder is formulated as a graph neural network (GNN) and a decoder is represented by an embedding method. These approaches enforce embedding techniques with structure information. Unfortunately, existing GNN-based frameworks still confront 3 severe problems: *low representational power*, *stacking in a flat way*, and *poor robustness to noise*. In this work, we propose a novel multi-level graph neural network (M-GNN) to address the above challenges. We first identify an injective aggregate scheme and design a powerful GNN layer using multi-layer perceptrons (MLPs). Then, we define graph coarsening schemes for various kinds of relations, and stack GNN layers on a series of coarsened graphs, so as to model hierarchical structures. Furthermore, attention mechanisms are adopted so that our approach can make predictions accurately even on the noisy knowledge graph. Results on WN18 and FB15k datasets show that our approach is effective in the standard link prediction task, significantly and consistently outperforming competitive baselines. Furthermore, robustness analysis on FB15k-237 dataset demonstrates that our proposed M-GNN is highly robust to sparsity and noise.

1 Introduction

Recently, there has been a surge of interest in knowledge graph construction, with projects such as DBpedia [Auer *et al.*, 2007] and Google's Knowledge Vault [Dong *et al.*, 2014]. KGs present data as multi-relational graphs composed of entities as nodes and relations as different types of edges. The edges (triples) are represented as a triple of the form (s, r, o) (e.g. $s = \textit{John}$, $r = \textit{IsBornIn}$, $o = \textit{Athens}$). These knowledge graphs can store rich factual knowledge and have proven useful for many NLP tasks, including question answering [Bordes *et al.*, 2015], information retrieval [Xiong and Callan, 2015]. Meanwhile, the incompleteness of these KGs stimulates more attention to predicting missing triples, which is the aim of link prediction task.

One of the most crucial techniques for link prediction task is knowledge graph embedding. Its key idea is to encode entities and relations into continuous low dimensional vector spaces, i.e., embeddings. And then these embeddings are inputted to scoring models for predicting new relations. Finally, entity and relation embeddings are obtained by maximizing the total likelihood of observed triples. With the translational assumption of $e_s + e_r \approx e_o$, early works, such as TransE [Bordes *et al.*, 2013], mapped entities and relations into the same vector space. TransH [Wang *et al.*, 2014], TransR [Lin *et al.*, 2015] are then proposed by introducing more complicated relational translation constraints. Besides, DistMult [Chang *et al.*, 2014], ComplEx [Trouillon *et al.*, 2016] utilize multiplicative score functions for calculating probability of triples.

Despite the success of embedding methods, link prediction is, in fact, an inherently graph-formulated task. Embedding methods ignore the neighbor structure of the knowledge graph, where many missing pieces of information reside. In contrast, graph neural network (GNN) can effectively learn node embeddings by recursively aggregating and transforming embeddings of neighbor nodes. To combine the strengths of both methods into a single model, [Schlichtkrull *et al.*, 2018] proposed an encoder-decoder framework (R-GCN). R-GCN contains an encoder of a GNN and a decoder of a tensor factorization embedding method, *DistMult*. By explicit modeling neighborhood structures, R-GCN recovers missing facts and significantly outperforms the direct optimization of the factorization model.

However, R-GCN model still faces 3 challenges: 1) **Low representational power**: R-GCN uses *mean* aggregator on the embeddings of the neighbor nodes. This kind of aggregation scheme is not injective, and easily confused by structures with repeating features. Consider the central node v_1 and v_2, where v_1 and v_2 have the same set of neighbors with distinct feature vectors, but v_1 contains multiple copies of the neighbor set of v_2. The *mean* aggregator takes averages over distinct feature vectors, and maps v_1 and v_2 to the same embeddings. 2) **Stacking in a flat way**: R-GCN model is inherently flat since it can only propagate information along the edges. Thus, it is limited to the 1-hop or 2-hop local neighbors, and not able to model the multi-level structure of the

*Corresponding Author

original graph. 3) **Poor robustness to noise**: For a given central node, R-GCN weighs information from every neighbor node equally and then performs element-wise mean pooling on the aggregated information embeddings. However, the importance of different neighbors is not the same, and information from unimportant or unreliable neighbors may severely mislead embedding learning, especially in noisy and sparse real-world knowledge graphs.

In this paper, we propose a Multi-level Graph Neural Network (M-GNN) framework for link prediction. M-GNN framework consists of an encoder of a multi-level graph neural network, and a decoder of an embedding method (e.g. *DisMult*, ComplEx). To improve the representational power of GNN layers, we first identify the structures confusing R-GCN and then define an injective aggregate scheme using multi-layer perceptrons (MLPs). On this basis, we develop graph coarsening schemes for relations with different mapping properties, and then stack multiple GNNs on these graphs so as to model multi-level structures of the original graph. Furthermore, to perform predictions on the noisy knowledge graphs, attention mechanisms are adopted so that M-GNN can learn the neighbor information adaptively. We show that M-GNN significantly outperforms the standard baselines on the WN18 and FB15k datasets. Meanwhile, the extensive analysis demonstrates that M-GNN maintains robustness even on noisy and sparse datasets.

Our contributions are summarized as follows:

- We point out the graph structures that cannot be distinguished by R-GCN, and define powerful GNN layers with an injective aggregation scheme.
- We develop a graph coarsening scheme and stack multiple GNN layers on a series of coarsened graphs, which helps to learn the multi-level structural information of the original KG.
- We demonstrate the effectiveness of our proposed M-GNN on the standard WN18 and FB15k datasets. Furthermore, experiments on the noisy knowledge graph suggest that our approach is highly robust to noise and sparsity.

2 Related Work

In recent years, knowledge graph embedding has obtained a surge of attention, and become one of the most crucial techniques in link prediction task. Its key idea is to embed the entities and relations of a KG into continuous vector spaces, in order to simplify manipulation while preserving the inherent structure. With the translational assumption of $e_s + e_r \approx e_o$, early works, such as TransE [Bordes *et al.*, 2013], mapped entities and relations into the same vector space. TransH [Wang *et al.*, 2014], TransR [Lin *et al.*, 2015] further extended TransE by introducing more complicate relational translation constraints and projecting entities and relations into different continuous spaces. These models based on translational constraints are also called *additive* models or *translational distance* models. Besides, *DistMult* [Yang *et al.*, 2015] and ComplEx [Trouillon *et al.*, 2016] utilize multiplicative score functions for calculating plausibility of triples, which are known as *multiplicative* models or *tensor factorization* models.

Despite the huge success in embedding methods for making predictions of new facts, link prediction is an inherently graph-formulated task, where connectivity structures are also important. Our proposed model (M-GNN) incorporate previous embedding methods with those neighbor structures by assigning an extension of graph neural network (GNN) as the encoder and an embedding method as the decoder.

GNNs were proposed to efficiently ingest graph structure and perform predictions. Recent attempts to extend GNN can be divided into two classes of approaches. The spectral approaches [Henaff *et al.*, 2015; Defferrard *et al.*, 2016] learn convolution filter using the eigendecomposition of the graph Laplacian. The second class of methods, known as spatial approaches, define convolutions directly on the graph. These approaches maintain the effective parameter-sharing, while are invariant to edge order and node degree, such as learning adaptive weights for different node degrees [Duvenaud *et al.*, 2015], or formulating the problem as message passing [Gilmer *et al.*, 2017].

The most relevant work to ours is R-GCN [Schlichtkrull *et al.*, 2018], which enforces connectivity structures in embedding methods with an encoder of GNN module. However, R-GCN is easily affected by noise since it aggregates all kinds of information from neighbors, cannot distinguish some simple structures due to non-injective aggregation scheme, and ignore the hierarchical structural information in the graph. GCNN [Neil *et al.*, 2018] is a simple extension of R-GCN with a regularized attention mechanism that slightly improves the performances, but still suffers from low representational power, flat network structure and poor robustness problems. In contrast, our proposed approach weighs neighbor structural information adaptively, identifies an injective aggregator using MLPs and stacks multiple GNN layers in a multi-level way to obtain further improvements.

3 Method

We first introduce the following definitions of the knowledge graph (KG): we denote the knowledge graph as a directed and multi-labeled graph $G = (V, \mathcal{E}, \mathcal{R})$, where $v_i \in V$ is a node, $r_i \in \mathcal{R}$ is a relation type and $(v_i, r_j, v_k) \in \mathcal{E}$ is a labeled edge (or a triple).

In this section, we present our approach in detail. We first introduce a basic graph neural network for link prediction (§ 3.1). Then we detail the GNN layers (§ 3.2) with high representational power and multi-level GNN (§ 3.3) with Graph Coarsening.

3.1 A Basic GNN Model

We first begin with a basic GNN framework for link prediction. Following [Xu *et al.*, 2018; Schlichtkrull *et al.*, 2018], an end-to-end GNN link prediction model comprises two components: a GNN-based encoder for embedding entities, and a decoder for calculating the probability of edges.

Encoder

A GNN-based encoder collects information from the neighborhood structure and then map entities (nodes) to embed-

dings. Modern GNNs follow the neighbor aggregation strategy [Xu *et al.*, 2018], updating the representation of a node by aggregating representations of its neighbors repeatedly. The $k-th$ layer of a GNN can be formulated as follow:

$$a_v^{(k)} = AGGREGATE^k(h_u^{(k-1)} : u \in \mathcal{N}(v)),$$
$$h_v^{(k)} = COMBINE^k(h_v^{(k-1)}, a_v^{(k)}), \quad (1)$$

where h_v^k is the feature vector of node v at the k-*th* layer and $\mathcal{N}(v)$ is the set of nodes adjacent to node v.

In a relational multi-graph (such as KG), we can naturally extend the above framework as follow:

$$a_v^{(k)} = AGGREGATE^k(h_{r,u}^{(k-1)} : u \in \mathcal{N}_v^r),$$
$$h_v^{(k)} = COMBINE^k(h_{r_0,v}^{(k-1)}, a_v^{(k)}), \quad (2)$$

where $h_{r,u}^{(k)} : u \in \mathcal{N}_v^r$ denotes the message passing from the neighbor node u under relation r at the $k-th$ layer, and $h_{r_0,v}^{(k)}$ denotes the self-connection message.

R-GCN and R-GCN+ [Schlichtkrull *et al.*, 2018] set $h_{r,u} = W_r h_u$, $AGGREGATE(\cdot) = MEAN(\cdot)$, and define the following propagation model for the forward-pass update of the entity embeddings:

$$h_i^{(k)} = \sigma(\sum_{r\in\mathcal{R}} \sum_{j\in\mathcal{N}_i^r} \frac{1}{c_{i,r}} W_r^{(k-1)} h_j^{(k-1)} + W_0^{(k-1)} h_i^{(k-1)}), \quad (3)$$

where $\mathcal{N}_i^r$ denotes the set of neighbor indices of node i under relation $r \in \mathcal{R}$. $c_{i,r}$ is a normalization constant ($c_{i,r} = |\mathcal{N}_i^r|$). In [Neil *et al.*, 2018], attention mechanisms are adopted and $c_{i,r}$ is trainable.

Decoder

Link prediction deals with the prediction of new facts (triples), given an incomplete subset of the knowledge graph. The task is to assign scores in order to determine how likely those unseen edges belong to $\mathcal{E}$.

To address the problem, the decoder needs to reconstruct edges of the knowledge graph using the entity embeddings from the encoder, i.e., $e_v = h_v^{(K)}$, for $v \in V$. A variety of algorithms can be adopted, since the choice of decoder is independent of the encoder. R-GCN [Schlichtkrull *et al.*, 2018] mainly focuses on the DisMult decoder:

$$f(e_s, R_r, e_o) = e_s^T R_r e_o, \quad (4)$$

where e_s, e_o are the subject and object embeddings and R_r is a diagonal relaton matrix. In our work, we also consider ComplEX decoder [Trouillon *et al.*, 2016].

In the following sections, we mainly focus on the GNN based encoder, designing GNN layers with high representational power, and extending flat GNN layers to the multi-level form.

3.2 A GNN Layer with High Representational Power

The choice of $AGGREGATE(\cdot)$ and $COMBINE(\cdot)$ in Eq. 2 is crucial to the representation power of the GNN layer. In the previous studies, R-GCN and R-GCN+ [Schlichtkrull *et al.*, 2018] utilize *mean* pooling as their aggregator. However, as Figure 1 shows, such kind GNN can be even confused by some simple graphs. In Figure 1, nodes with different colors have different embeddings. In Figure 1(a),

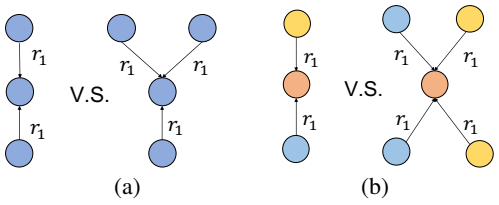


Figure 1: Examples of some simple structures that confuse the *mean* aggregator.

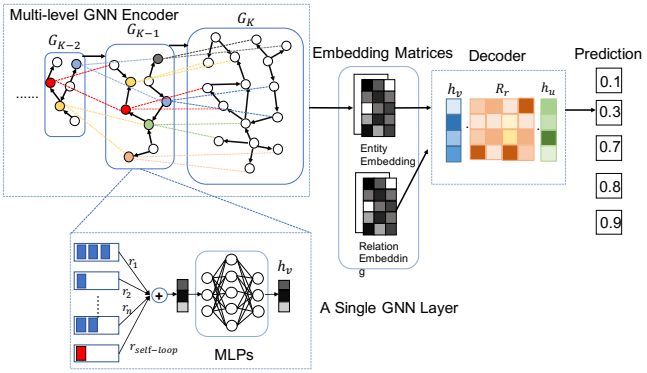


Figure 2: Overview of the M-GNN. M-GNN consists of an encoder of a multi-level GNN and a decoder of an embedding method. The outputs of the encoder, entity embedding matrices, are then inputted to the decoder with relation embeddings for predicting new facts.

when conducting neighborhood aggregation, the *mean* aggregator fails to distinguish the structures and obtains the same information, since $\frac{1}{2}(2 \cdot h_{r_1 blue}) = \frac{1}{3}(3 \cdot h_{r_1 blue})$. Similarlly, the *mean* aggregator fails in Figure 1(b) since $\frac{1}{2}(h_{r_1 blue} + h_{r_1 yellow}) = \frac{1}{4}(2 \cdot h_{r_1 blue} + 2 \cdot h_{r_1 yellow})$. Intuitively, a powerful GNN layer maps two entities to the same location only if they have identical neighborhood structures with identical embeddings on the corresponding entities. This means its aggregation scheme is *injective*, i.e., for each pair $((r_0, v), \{(r, u) : u \in \mathcal{N}_v^r\})$, $h_v = GNN((r_0, v), \{(r, u) : u \in \mathcal{N}_v^r\})$ is unique. According to Corollary 6 in [Xu *et al.*, 2018], we can decomposed any injective $GNN(\cdot)$ as $GNN(\cdot) = \phi((1+\epsilon) \cdot f(r_0, v) + \sum_{u\in N_v^r} f(r, u))$ for some function f, ϕ, and constant ϵ. In practice, we can use multi-layer perceptrons (MLPs) to model f and ϕ, according to the universal approximation theorem [Hornik *et al.*, 1989]:

$$h_v^{(k)} = MLP^{(k)}((1+\epsilon^{(k)}) \cdot h_{r_0,v}^{(k-1)} + \sum_{u\in\mathcal{N}_v^r} h_{r,u}^{(k-1)}), \quad (5)$$

where $\epsilon^{(k)}$ can be a learnable parameter or a fixed scalar. Strictly, we should set $h_{r,v} = MLP_r(h_v)$ for every $r \in \mathcal{R} \cup \{r_0\}$ to make aggregation scheme injective. To reduce complexity, we simply employ the linear transformation here: $h_{r,v} = W_r h_v$.

3.3 Multi-Level GNN Layers with Graph Coarsening

In the previous studies [Schlichtkrull *et al.*, 2018; Neil *et al.*, 2018], the GNN method is inherently flat, i.e., they can only

Algorithm 1 Graph Coarsening.

Require:
Knowledge graph $KG = (V, \mathcal{E}, \mathcal{R})$;
Ensure:
Coarsened graph $G_0, G_1, ..., G_k$;
1: $m \leftarrow 0$
2: $G_0 \leftarrow KG$
3: **while** $|E_m| \geq threshold$ **do**
4: $m \leftarrow m + 1$;
5: $G_m \leftarrow EdgeCoarsen(NeighborCoarsen(G_{m-1}))$
6: **end while**
7: **return** $G_0, G_1, ..., G_k$;

propagate information across edges. These methods are unable to aggregate information of different granularities. As Figure 2 shows, our goal is to stack multiple GNN layer in a multi-level way: we first define a strategy to output a series of coarsened graphs, and then we can build a multi-level GNN operating on these graphs with different scales.

Graph Coarsening

To develop a graph coarsening scheme that preserves structural information at different scales, we consider four types of relations in KGs [Lin *et al.*, 2015]: 1-to-1, 1-to-N, N-to-1 and N-to-N. And each type of relation may contain different structures:

- 1-to-1 relations: only contain the 1-to-1 structure.
- 1-to-n/n-to-1 relations: may contain 1-to-1 and 1-to-n/n-to-1 structures.
- n-to-n relations: may contain 1-to-1, 1-to-n and n-to-1 structures.

Here, n-to-n structure is not taken into consideration since it can be divided into multiple 1-to-n and n-to-1 structures. For 1-to-1 structure, we propose a simple coarsening scheme, namely **edge coarsening**. It first selects a subset $\mathcal{E}' \subset \mathcal{E}$, in which no two edges are incident to the same node. And then, as Figure 3(a) shows, for each $(v_i, r, u_i) \in \mathcal{E}'$, it merges (v_i, r, u_i) into a supernode. For 1-to-n/n-to-1 structure, considering the similarity of nodes that share the same neighborhood, we develop the **neighbor coarsening** scheme. As Figure 3(b) and Figure 3(c) shows, (e_1, e_6), (e_2, e_3), (e_4, e_5) are merged into supernodes since they share same neighbor e_7. The orders for both coarsening schemes are arbitrary since we obtain similar node embeddings with different coarsening orders.

To combine edge and neighbor coarsening scheme, as algorithm 1 shows, in each coarsening step, we first compress the input graph with neighbor coarsening, and then adopt edge coarsening to output a new coarser graph.

Multi-Level GNN

The outputs of the graph coarsening scheme are a series of coarsened graphs, $G_1, ..., G_K$, where G_1 is the coarsest graph and G_K is the original graph. As Figure 2 shows, our goal here is to stack K GNN layers aggregating structural information from the coarsest graph to the original graph. We denote $S^{(k)} \in R^{n_k \times n_{k+1}}$ as the cluster assignment matrix at layer k. Each column of $S^{(k)}$ refers to one of the n_{k+1} nodes at layer $k+1$, and each row refers to one of the n_k supernodes (clusters) at layer k. $S^{(k)}$ provides the assignment of each node in graph G_{k+1} to the supernode in the coarsened graph G_k. After the embedding matrix $H^{(k-1)}$ for the graph G_{k-1} is learned, we extend it as the initial representations for the graph G_k by ${S^{(k-1)}}^T H^{(k-1)}$. Now we can define $k-th$ layer of our proposed multi-level GNN by extending Eq. 5 as follow (on the graph G_k):

$$
\begin{aligned}
&H'^{(k-1)} = {S^{(k-1)}}^T H^{(k-1)}, \\
&h_{r,v}^{(k-1)} = W_r^{(k-1)} h_v'^{(k-1)}, \\
&h_v^{(k)} = MLP^{(k)}((1+\epsilon^{(k)}) \cdot h_{r_0,v}^{(k-1)} + \sum_{u \in \mathcal{N}_v^{r,(k)}} h_{r,u}^{(k-1)}),
\end{aligned} \tag{6}
$$

where $H'^{(k)}$ is a embedding matrix mapped from the embedding matrix $H^{(k)}$ through assignment matrix $S^{(k)}$ and we randomly initialize the embedding matrix $H^{(0)}$. Following the previous work [Schlichtkrull *et al.*, 2018; Neil *et al.*, 2018], we optimize our model with cross-entropy loss and negative sampling.

Dataset	# Ent	# Rel	# Train /Valid/Test
WN18	40943	18	141442/5000/5000
FB15k	14951	1345	483142/50000/59071
FB15k-237	14541	237	272115/17535/20466

Table 1: Datasets Statistics.

4 Experiments

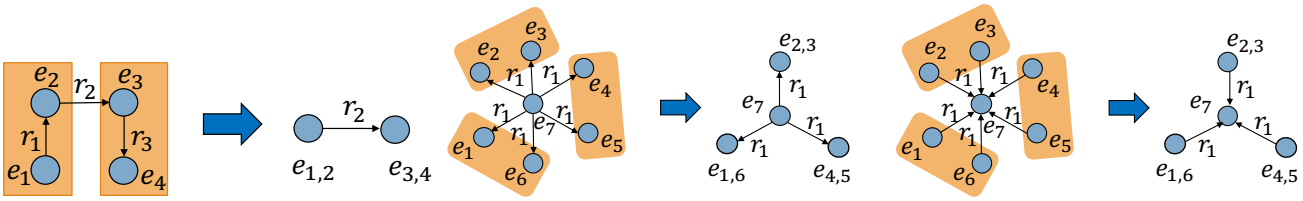
In this section, we present our experimental settings and results. We first introdue the datasets and baselines in our experiment (§ 4.1 and § 4.2). Then, we first evaluate our approach in the standard link prediction task (§ 4.3). After that, by constructing noisy knowledge graphs, we conduct extensive analysis on the robustness of our approach (§ 4.4).

4.1 Datasets

We evaluate our link prediction algorithm on two commonly used datasets: FB15k, a subset of the multi-label knowledge base Freebase and WN18, a subset of WordNet featuring lexical relations between words. Both datasets are released by [Bordes *et al.*, 2013].

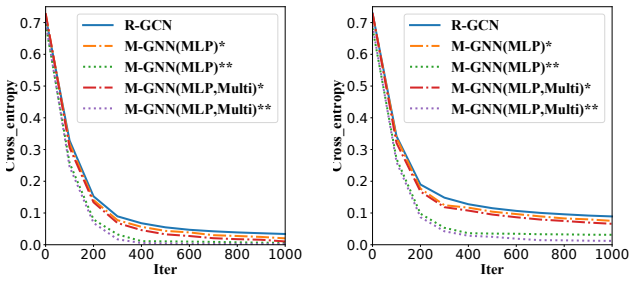
As [Toutanova and Chen, 2015] shows, inverse triplet pairs $t = (e_1, r, e_2)$ and $t' = (e_2, r^{-1}, e_1)$ appears in both datasets with t in the training set and t' in the test set. This inverse triplet pair flaw simplifies a large part of link prediction task for the memorization of inverse triplet pairs. To address this problem, [Toutanova and Chen, 2015] release the dataset FB15k-237 removing all inverse triplet pairs. Thus, We use FB15k-237 dataset for our extensive experiments. Table 1 further summarizes the statistics of the datasets.

All the datasets only provide positive triples. Following [Bordes *et al.*, 2013], we adopt the *local closed world assumption* to generate negatives. Specifically, given a triple, we randomly corrupt the subject or the object to generate a negative example.



(a) 1-to-1 structure (b) 1-to-n structure (c) n-to-1 structure

Figure 3: Examples of graph coarsening scheme. 3(a): edge coarsening scheme. 3(b) and 3(c): common neighbor coarsening scheme.



(a) WN18 (b) FB15k

Figure 4: Cross-entropy loss on training sets.

4.2 Baselines

DistMult [Yang *et al.*, 2015] and ComplEx [Trouillon *et al.*, 2016] are the common baselines. These factorization methods not only perform well on standard datasets, but are the decoders of our proposed model. Besides, we also consider R-GCN and R-GCN+ [Schlichtkrull *et al.*, 2018], the current GNN-based methods for link prediction, and GCNN [Neil *et al.*, 2018], which is an extension of R-GCN in a way that weighs the different connections of a central node adaptively. We further compare with the classic algorithm TransE [Bordes *et al.*, 2013].

4.3 Link Prediction Task

In this section, we combine our multi-level GNN model with decoders to predict new facts in knowledge graphs. Multiple decoders are adopted including *DistMult* [Yang *et al.*, 2015] and *ComplEx* [Trouillon *et al.*, 2016]. We select commonly used WN18 and FB15k for evaluating our link prediction algorithms.

Results

For the evaluation of link prediction results, we measure the quality of every test triple among all possible subjects and objects substitution. In our experiment, we use two widely used evaluation metrics for link prediction: *mean reciprocal rank* (MRR) and *Hits at n* (Hits@n). We report both filtered and raw MRR (following [Bordes *et al.*, 2013]), and filtered Hits at 1, 3 and 10.

In addition to baselines, we evaluate several variants of our model: **M-GNN(MLP)** employs Eq.5 without graph coarsening scheme; **M-GNN(MLP, Multi)** uses both MLPs and graph coarsening scheme, i.e., Eq.6. Considering complementarity between models, we also employ ensemble scheme as [Schlichtkrull *et al.*, 2018], which we refer to as **M-GNN+(MLP, Multi)**. To avoid the influence of the choices of decoders, we consider both *DistMult* (with "*") and ComplEx (with "**") decoders for both M-GNN and R-GCN.

The hyperparameters in M-GNN are determined by the grid search on the validation set. The ranges of the hyperparameters are manually set as follow: learning rate {0.01, 0.005, 0.003, 0.001}, dropout rate {0, 0.1, 0.2, 0.3,...,0.9}, embeddings size {100, 150, 200, 300}, regularization coefficient {0.01,0.05,0.1,0.5,1.0}, the number of negative samples {1,3,5,10} and $\epsilon = 0$. For both FB15k and WN18 datasets, we use M-GNN with three GNN layers and all MLPs have two layers with the hidden unit number $\in$ {10,50,100,200}. For the ComplEx encoder, we treat complex vector C^d as real vector $R^{d\times 2}$ in the encoder. We train the models with Adam optimizer [Kingma and Ba, 2015].

Results are shown in Table 2. We first compare our model M-GNN with the first five baselines. M-GNN consistently outperforms those methods without graph neural network. Thus, we can conclude that GNN takes the neighborhood structures into consideration, which significantly facilitates the learning process of entity and relation embeddings.

Then, we compare M-GNN to R-GCN with the same decoder *DistMult* or ComplEx. M-GNN(MLP) consistently outperforms R-GCN, while M-GNN(MLP, Multi) consistently outperforms both M-GNN(MLP) and R-GCN. It demonstrates that MLP layers and multi-level structural information are both beneficial for link prediction. Additionally, we show the cross-entropy loss on training sets in Figure 4. We observe that our approach can fit the training sets better and its convergence process is significantly faster, which contributes to high representational power and multi-level structural information.

Finally, we compare M-GNN** with M-GNN* and R-GCN. M-GNN** significantly outperforms M-GNN* and R-GCN for the most of situations. This suggests that combined with the decoder that achieves higher results, M-GNN can predict new facts more accurately.

In summary, our proposed model M-GNN can accurately predict new facts, and each component of our model is helpful for entity and relation representation learning.

4.4 Extensive Analysis on the Robustness

In the previous sections, we can conclude that our proposed method is effective for the standard link prediction task. To

Method	FB15k					WN18				
	MRR		Hits@			MRR		Hits@		
	Raw	Filter	1	3	10	Raw	Filter	1	3	10
TransE	0.221	0.380	0.231	0.472	0.641	0.335	0.454	0.089	0.823	0.934
DistMult	0.248	0.634	0.522	0.718	0.814	0.526	0.813	0.701	0.921	0.943
ComplEx	0.242	0.692	0.599	0.759	0.840	0.587	0.941	0.936	0.945	0.947
R-GCN	0.251	0.651	0.541	0.736	0.825	0.553	0.814	0.686	0.928	0.955
R-GCN+	0.262	0.696	0.601	0.760	0.842	0.561	0.819	0.697	0.929	0.964
R-GCN**	0.247	0.700	0.603	0.763	0.844	0.588	0.942	0.933	0.945	0.953
R-GCN+**	0.250	0.706	0.615	0.769	0.849	0.589	0.944	0.936	0.947	0.956
M-GNN(MLP)*	0.257	0.658	0.550	0.742	0.833	0.560	0.824	0.702	0.934	0.966
M-GNN(MLP, Multi)*	0.271	0.667	0.563	0.756	0.841	0.568	0.830	0.705	0.946	0.965
M-GNN+(MLP, Multi)*	**0.279**	0.696	0.618	0.772	0.853	0.572	0.835	0.710	0.947	0.966
M-GNN(MLP)**	0.263	0.708	0.611	0.765	0.845	0.590	0.941	0.936	0.945	**0.967**
M-GNN(MLP, Multi)**	0.269	0.717	0.636	0.774	0.857	0.595	0.943	0.939	0.947	**0.967**
M-GNN+(MLP, Multi)**	0.276	**0.747**	**0.671**	**0.795**	**0.876**	**0.597**	**0.948**	**0.940**	**0.950**	**0.967**

Table 2: Performances on FB15k and WN18 Datasets.

Method	Hits@10				MRR			
	100%	50%	Skip	Noised	100%	50%	Skip	Noised
DistMult	0.432	0.202	-	0.206	0.239	0.087	-	0.089
ComplEx	0.441	0.241	-	0.243	0.259	0.109	-	0.110
GCNN	0.475	0.332	0.258	0.214	0.272	0.168	0.133	0.111
GCNN w/att	0.482	0.347	0.340	0.356	0.283	0.185	0.188	0.191
M-GNN(MLP)*	0.456	0.268	0.234	0.210	0.255	0.108	0.115	0.103
M-GNN(MLP, Multi)*	0.465	0.286	0.271	0.266	0.273	0.146	0.167	0.185
M-GNN(MLP)**	0.483	0.341	0.263	0.244	0.278	0.173	0.144	0.121
M-GNN(MLP, Multi)**	**0.506**	**0.358**	**0.356**	**0.367**	**0.327**	**0.198**	**0.205**	**0.209**

Table 3: Performances on FB15k-237 Dataset.

further analyze the robustness to noise and sparsity, we adopt experimental settings in [Neil *et al.*, 2018] on the FB15k-237 dataset, and four different training conditions are taken into consideration.

Results

Following [Neil *et al.*, 2018], the training set is split evenly so that half of the triples are in g_{true} and half in g_{add}. First, to establish a baseline, methods are trained normally on the full datasets ($g_{true} \cup g_{add}$), which refers to "100%" condition. To explore the impact of sparsity, we establish "50%" condition, which refers to training only on g_{true}. To generate noisy triples, we follow [Pujara *et al.*, 2017] and corrupt one of a subject, relation, or object entry for the triples in g_{add} with equal probability, such that $|g_{noise}| = |g_{add}|$. For "Skip" condition, the adjacency matrix of the GNN model contains all the edges from $g_{all} = g_{true} \cup g_{add} \cup g_{noise}$, while only edges in g_{gold} are used for training. For "Noised" condition, methods are trained upon g_{all}.

For predicting new facts on the noisy graphs, we follow [Neil *et al.*, 2018] and adopt the attention mechanism on our proposed GNN layer: $h_v^{(k)} = MLP^{(k)}((1+\epsilon^{(k)}) \cdot h_{r_0,v}^{(k-1)} + \sum_{u \in \mathcal{N}_v^{r,(k)}} \alpha_{v,r,u} h_{r,u}^{(k-1)})$, where $\alpha_{v,r,u} \in [0,1]$ is trainable. For the FB15k-237 datasets of the four training settings, we use M-GNN with two GNN layers and all MLPs have two layers. Other ranges of hyperparameters are the same in the link prediction task. Note that since M-GNN+ and M-GNN show very similar results, we only report the results of the M-GNN model here.

In the experiment, we evaluate link prediction results and report filtered MRR and Hits@10. The results are shown in Table 3. The results of baselines are taken from [Neil *et al.*, 2018], which compare favorably with those reported in [Schlichtkrull *et al.*, 2018]. For the "100%" training condition, our model consistently outperforms other baselines including R-GCN (MRR:0.158, Hits@10:0.414) and R-GCN+ (MRR:0.156,Hits@10:0.417). For the noisy or sparse knowledge graph, all the performances are dramatically reduced, while M-GNN can still achieve the best results. Thus, GNN layers with higher representational power and multi-level structural information are more robust to sparsity and noise.

5 Conclusions

In this paper, we proposed a novel multi-level GNN-based framework (M-GNN) with high representational power and demonstrated its effectiveness and robustness on multiple datasets under various training conditions. We first pointed out the examples that confused *mean* aggregator, which was widely used by the previous GNN-based models. Then, we identified an injective aggregation scheme, and further designed a powerful GNN layer using MLPs. On the basis of that, we coarsened the knowledge graph into a series of graphs and stacked GNN layers in a multi-level way, to uncover hierarchical structure information. Experimental results show that our approach is both effective for identifying new facts and robust to noise and sparsity.

Acknowledgements

The research work is supported by the National Key R&D Program with No.2016QY03D0503,2016YFB081304, Strategic Priority Research Program of Chinese Academy of Sciences, Grant No.XDC02040400, National Natural Science Foundation of China (No.61602474, No.61602467, No.61702552).

References

[Auer *et al.*, 2007] Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary G. Ives. Dbpedia: A nucleus for a web of open data. In *6th International Semantic Web Conference*, pages 722–735, Busan, Korea, November 2007.

[Bordes *et al.*, 2013] Antoine Bordes, Nicolas Usunier, Alberto García-Durán, Jason Weston, and Oksana Yakhnenko. Translating embeddings for modeling multi-relational data. In *Advances in neural information processing systems*, pages 2787–2795, Nevada, United States, December 2013.

[Bordes *et al.*, 2015] Antoine Bordes, Nicolas Usunier, Sumit Chopra, and Jason Weston. Large-scale simple question answering with memory networks. *arXiv preprint arXiv:1506.02075*, 2015.

[Chang *et al.*, 2014] Kai-Wei Chang, Wen-tau Yih, Bishan Yang, and Christopher Meek. Typed tensor decomposition of knowledge bases for relation extraction. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 1568–1579, Doha, Qatar, October 2014.

[Defferrard *et al.*, 2016] 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 3837–3845, Barcelona, Spain, December 2016.

[Dong *et al.*, 2014] 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 *The 20th International Conference on Knowledge Discovery and Data Mining*, pages 601–610, New York, United States, August 2014.

[Duvenaud *et al.*, 2015] David K. Duvenaud, Dougal Maclaurin, Jorge Aguilera-Iparraguirre, Rafael Gómez-Bombarelli, Timothy Hirzel, Alán Aspuru-Guzik, and Ryan P. Adams. Convolutional networks on graphs for learning molecular fingerprints. In *Advances in neural information processing systems*, pages 2224–2232, Quebec, Canada, December 2015.

[Gilmer *et al.*, 2017] Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. Neural message passing for quantum chemistry. In *Proceedings of the 34th International Conference on Machine Learning*, pages 1263–1272, Sydney, Australia, August 2017.

[Henaff *et al.*, 2015] Mikael Henaff, Joan Bruna, and Yann LeCun. Deep convolutional networks on graph-structured data. *arXiv preprint arXiv:1506.05163*, 2015.

[Hornik *et al.*, 1989] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.

[Kingma and Ba, 2015] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations*, San Diego, United States, May 2015.

[Lin *et al.*, 2015] Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. Learning entity and relation embeddings for knowledge graph completion. In *Proceedings of the 29th AAAI Conference on Artificial Intelligence*, pages 2181–2187, Texas, United States, January 2015.

[Neil *et al.*, 2018] Daniel Neil, Joss Briody, Alix Lacoste, Aaron Sim, Paidi Creed, and Amir Saffari. Interpretable graph convolutional neural networks for inference on noisy knowledge graphs. *arXiv preprint arXiv:1812.00279*, 2018.

[Pujara *et al.*, 2017] Jay Pujara, Eriq Augustine, and Lise Getoor. Sparsity and noise: Where knowledge graph embeddings fall short. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1751–1756, Copenhagen, Denmark, September 2017.

[Schlichtkrull *et al.*, 2018] Michael Sejr Schlichtkrull, Thomas N. Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. Modeling relational data with graph convolutional networks. In *The Semantic Web - 15th International Conference*, pages 593–607, Heraklion, Greece, June 2018.

[Toutanova and Chen, 2015] Kristina Toutanova and Danqi Chen. 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, 2015.

[Trouillon *et al.*, 2016] 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.

[Wang *et al.*, 2014] Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. Knowledge graph embedding by translating on hyperplanes. In *Proceedings of the 28th AAAI Conference on Artificial Intelligence*, pages 1112–1119, Québec, Canada, July 2014.

[Xiong and Callan, 2015] Chenyan Xiong and Jamie Callan. Esdrank: Connecting query and documents through external semi-structured data. In *Proceedings of the 24th ACM International Conference on Information and Knowledge Management*, pages 951–960, Melbourne,Australia, October 2015.

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

[Yang *et al.*, 2015] Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. Embedding entities and relations for learning and inference in knowledge bases. In *3rd International Conference on Learning Representations,*, San Diego, United States, May 2015.