Double-Branch Multi-Attention based Graph Neural Network for Knowledge Graph Completion

Hongcai Xu, Junpeng Bao*, Wenbo Liu
Xi'an Jiaotong University
xajt1822@stu.xjtu.edu.cn, baojp@mail.xjtu.edu.cn, stu_Lwb@stu.xjtu.edu.cn

Abstract

Graph neural networks (GNNs), which effectively use topological structures in the knowledge graphs (KG) to embed entities and relations in low-dimensional spaces, have shown great power in knowledge graph completion (KGC). KG has abundant global and local structural information, however, many GNN-based KGC models cannot capture these two types of information about the graph structure by designing complex aggregation schemes and are not designed well to learn representations of seen entities with sparse neighborhoods in isolated subgraphs. In this paper, we find that a simple attention-based method can outperform a general GNN-based approach for KGC. We then propose a double-branch multi-attention-based graph neural network (MA-GNN) to learn more expressive entity representations that contain rich global-local structural information. Specifically, we first explore the graph attention network-based local aggregator to learn entity representations. Furthermore, we propose a snowball local attention mechanism by leveraging the semantic similarity between two-hop neighbors to enrich the entity embedding. Finally, we use Transformer-based self-attention to learn long-range dependence between entities to obtain richer representations with the global graph structure and entity features. Experimental results on five benchmark datasets show that MA-GNN achieves significant improvements over strong baselines for inductive KGC.

1 Introduction

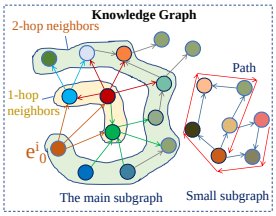
Knowledge graphs (KGs), which are collections of structured knowledge represented by factual triples (head entity, relation, tail entity), are crucial for many applications, including semantic search (Xiong et al., 2017), question-answering (Kaiser et al., 2021), recommendation systems (Wang et al., 2021), etc. However, even large-scale KGs (e.g., Freebase and DBpedia) that have billions of triples, are inevitably incomplete, limiting their real-world applications. Therefore, knowledge graph completion (KGC) recently attracted extensive attention and attempts to automatically predict the missing head (tail) given the relation and tail (head) in a factual triple.

To address these challenges, knowledge graph embedding (KGE) methods have been recently introduced as an effective solution to solve the incompletion problem (Bordes et al., 2013; Dettmers et al., 2018; Sun et al., 2019; Trouillon et al., 2016). KGE aims to define a scoring function and embeds entities and relations into a low-dimensional vector space to assess the plausibility of triplets based on the observed triples. However, it is significantly reliant on the pre-defined scoring function and rather challenging to encode structural information about an entity into a single vector.

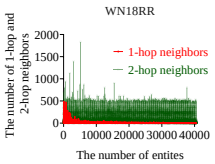
To this end, graph neural networks (GNNs) (Defferrard et al., 2016; Kipf and Welling, 2017) have recently been proposed for inductive KGC, due to the intrinsic graph structure of KGs, which learn the hidden representation of each entity by aggregating its corresponding local neighbors' information (Hamaguchi et al., 2017; Albooyeh et al., 2020). These methods still suffer from restricted expressiveness and problems because of the shallow network architecture. First, they can only capture local information within the neighborhood of individual entities, lacking the capability to exploit global information. Second, a large knowledge graph consists of multiple isolated and small subgraphs that are not connected to the main subgraph, as shown in Figure 1(a). Existing GNN methods often suffer from over-smoothing or over-squashing (Chen et al., 2020) with the increase of GNN layers and limited entities and relations in isolated subgraphs. In contrast, attention-based methods are effective in aggregating multi-hop neighbor information, leading to richer entity representation

*The corresponding author.

Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics
Volume 1: Long Papers, pages 15257–15271



(a) The neighborhood structure for the target entity e_0^i



(b) One-hop and two-hop neighboring entity distributions in real-world KGs (WN18RR).

Figure 1: Our MA-GNN not only aggregates one-hop but also captures the information between two-hop neighborhood entities.

(Nathani et al., 2019; Zhao et al., 2022; Li et al., 2022b). Despite the success of these attention-based GNN methods, most of them concentrate on encoding high-order topological information (a path or random walking sequence) while ignoring the rich structural information from neighborhood entities (Li et al., 2018; Nathani et al., 2019). Besides they have not paid enough attention to the integration of local and global information, which is important for encoding complex information. As shown in Figure 1(a), three one-hop neighbors are connected to e_0^i, and the three one-hop neighboring entities connect the diverging two-hop neighboring entities. Besides, we also note that the number of two-hop neighboring entities is much higher than the number of one-hop neighborhood entities, as shown in Figure 1(b). It is unreasonable to fuse two-hop neighborhood entities for each target entity directly. Thus, it is meaningful for a target entity to pay more attention to its multi-hop neighbors and incorporate both global and local structural information in order to learn effective knowledge representations.

In this paper, we propose a Double-Branch Multi-Attention Graph neural network (MA-GNN) for KGC in order to preserve global-local structural information. MA-GNN is an encoder-decoder model where three types of attention mechanisms (Graph attention network (GAT), Snowball local attention-based mechanism, and Transformer-based self-attention module) and ConvE (Dettmers et al., 2018) play the roles of an encoder and decoder, respectively. Furthermore, three types of attention mechanisms are utilized in the two branches to extract global and local features due to the differences in these two branches' characteristics. Specifically, we first employ GAT and Transformer-based self-attention to learn entity representation in order to capture global-local structure information. Second, a snowball local attention module is proposed to compute the local semantic similarity between two-hop neighborhood entities.

In summary, we make the following contributions:

- We propose a double-branch multi-attention graph neural network for KGC. The network consists of two parallel branches, i.e., the global branch and the local branch. Compared with other attention-based GNN methods, our method can capture local information and long-term dependence between entities through GAT and Transformer based self-attention.
- To extract more discriminative features, we design a snowball local attention mechanism that can learn the entity similarity between 2-hop neighborhood entities of the target entity and encode more information like a snowball.
- We compare our method with previous KGC methods on five benchmark datasets. Experiments demonstrate that MA-GNN leads to significant improvement, achieving Hits@10 scores of 0.679 on the WN18RR dataset, 0.823 on the NELL-995 dataset, and 0.932 on the FB15K dataset, outperforming state-of-the-art methods by **12.7%**, **4.3%** and **15.1%**, respectively.

2 Methods

An encoder-decoder framework is used in the MA-GNN model. The framework of MA-GNN is shown in Figure 2. We only need to stack the encoder into several layers to obtain entity representation. MA-GNN presents four main modules: GAT(Encoder), Transformer-based self-attention module (Encoder), Snowball local attention module (Encoder), and Knowledge graph completion module (Decoder).

2.1 Graph attention network module

Assuming that each entity e_i in the knowledge graph $G = (E, R)$ has an initial feature vector $z_0 \in R^d$, we employ a graph attention network (GAT) to gather local information from the target entity's neighbors, which calculates the attention score $\alpha_{i,j}^l$ in the l-th layer given the entity e_i and

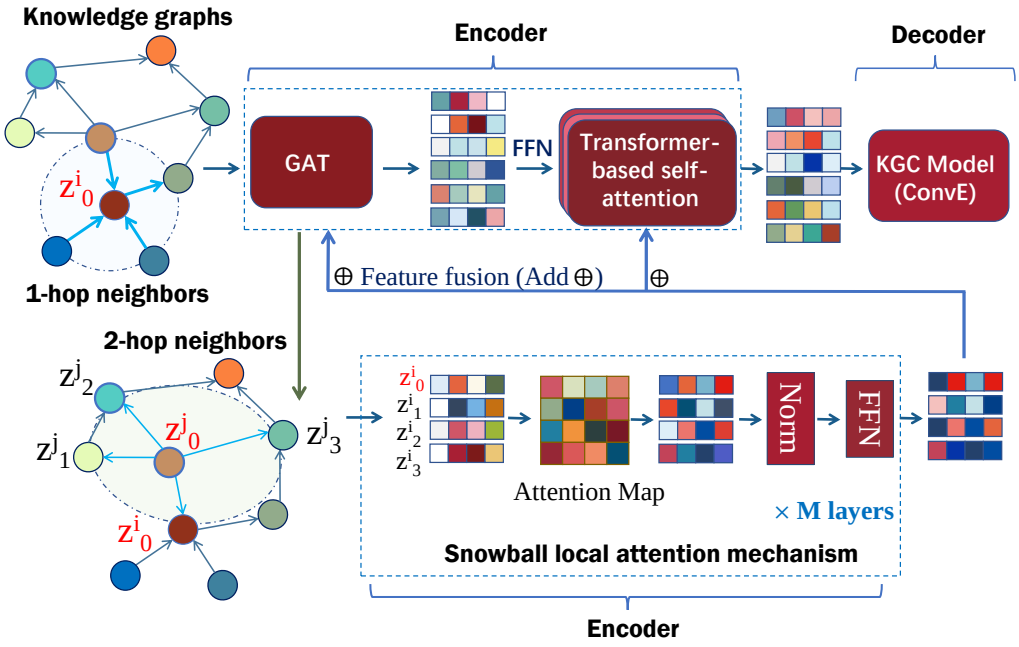


Figure 2: Architecture of MA-GNN. MA-GNN has two components: an encoder and a decoder. Specifically, the encoder includes three modules to capture local and global information: 1) local branch (GAT and Snowball local attention learn local graph structure information), 2) global branch (Transformer-based self-attention learns global, long-range relationships).

its one-hop neighbors $e_j \in N_i$, N_i is the one-hop neighbors of entity e_i.

$$\alpha_{i,j}^{l} = \frac{exp\left(\sigma\left(f^{l}\left[W_n^l z_i^l : W_n^l z_j^l\right]\right)\right)}{\sum_{k \in N_i} exp\left(\sigma\left(f^{l}\left[W_n^l z_i^l : W_n^l z_k^l\right]\right)\right)} \quad (1)$$

where z_i^l is the embedding of entity e_i in the l-th layer. For entity features, there is a learnable weight matrix called W_n^l. f^l refers to a fully connected neural network. Here, we use ReLU as the activation function, which is σ. By summing the weighted features of entity e_i's neighbors, we can aggregate local information for entity e_i:

$$s_i^l = \sum_{j \in N_i} \alpha_{i,j}^l z_i^l \quad (2)$$

The updated entity representation is then computed by:

$$z_i^{l+1} = z_i^l + \lambda W_w^l s_i^l \quad (3)$$

where λ determines how much information is passed between neighbors. W_w^l is the parameter that needs to be trained.

2.2 Transformer-based self-attention module

Transformer is presented as a unique encoder-decoder architecture constructed from several self-attention components (Vaswani et al., 2017). Let $Z \in R^{n \times d}$ be the input for each Transformer based self-attention layer, where n is the number of entities and d is the dimension of each entity. Then, a function $f_W : R^{n \times d} \rightarrow R^{n \times d}$ with the formula $f_W(Z) = z$ can be used as one Transformer-based self-attention layer:

$$A = \frac{1}{\sqrt{d}} Z W_Q \left(Z W_K\right)^{\top} \quad (4)$$

$$Z' = SoftMax\left(A\right)\left(Z W_V\right) \quad (5)$$

$$M = LayerNorm_1\left(Z' W_O + fZ\right) \quad (6)$$

$$F = \sigma\left(M W_1 + b_1\right) W_2 + b_2 \quad (7)$$

$$z = LayerNorm_2\left(M + F\right) \quad (8)$$

where LayerNorm() is the layer normalization function, Softmax() is the row-wise softmax function, and σ is the activation function (e.g., ReLU). In the layer, the following parameters are trainable: $W_Q, W_K, W_V, W_O, W_1, b_1, W_2$, and b_2. In more detail, W_Q, W_K, and W_V are broken down into H heads using the formulas $W_Q^{(h)}$, $W_K^{(h)}$, $W_V^{(h)}$, and

the matrices $Z^{(h)}$ from the attention heads are then concatenated to produce Z'.

$$A^{(h)} = \frac{1}{\sqrt{d}} Z W_Q^{(h)} \left(Z W_K^{(h)} \right)^\top \quad (9)$$

$$Z' = \|_{h=1}^{H} \left(SoftMax \left(A^{(h)} \right) Z W_V^{(h)} \right) \quad (10)$$

Once we obtain the final per-entity representation z_i^l encoded by GAT, where l is the total number of GAT layers, we pass it to the Transformer-based self-attention subnetwork of MA-GNN as shown in Figures 2 and 3. N and K refer to the number of stacked modules. In order to normalize the embedding, we first project the z_i^l into the Transformer-based self-attention dimension and normalize the embedding using a layer normalization:

$$p_i^l = LayerNorm \left(W_p z_i^l \right) \quad (11)$$

where $W_P \in R^{d_T \times d_G}$ is a learnable weight matrix, R^{d_T} is the self-attention dimension, and R^{d_G} is the dimension of the final GAT embeddings. Since the Transformer-based self-attention is permutation-invariant without a positional encoding (Wu et al., 2021), we utilize the random walk method (Perozzi et al., 2014) to obtain entity sequences.

$$\alpha_{i,j}^{l(h)} = \frac{1}{\sqrt{d_T}} p_i^{l-1} W_Q^{l(h)} \left(p_j^{l-1} W_K^{l(h)} \right)^\top \quad (12)$$

$$p_i' = \|_{h=1}^{H} \left(\text{SoftMax} \left(\alpha_{i,j}^{l(h)} \right) p_k^{l-1} W_V^{l(h)} \right) \quad (13)$$

where W_Q^l, W_K^l, and W_V^l are the learned query, key, and value matrices for a single attention head in layer l, respectively, and h is the number of attention heads.

2.3 Snowball local attention module

In this subsection, we first construct one-hop neighborhood subgraphs that focus on the neighbors of the target entities (e.g., e_0^i, e_0^j, e_0^k), as shown in Figure 4, and then utilize the proposed snowball local attention mechanism to aggregate local graph structure information. The snowball local attention mechanism samples two-hop neighbors and one-hop neighbors of the target entity, and is capable of capturing entity similarity between two-hop neighborhood entities according to the attention scores between two-hop neighbors. Here, we only present

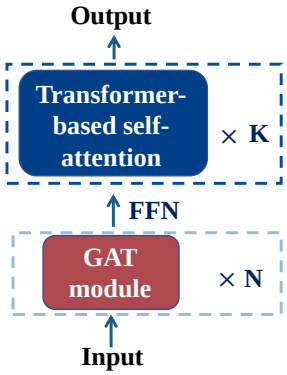


Figure 3: The combined framework for GNN and Transformer based attention modules.

one formulation for the snowball local attention layer:

$$\bar{z}_k^i = \sigma \left(\sum_{e_m^i, e_k^i \in \mathcal{N}_i} \alpha_{km}^i W z_k^i, k, m = 1, 2, 3 \ldots \right) \quad (14)$$

where $\bar{z}_k^i$ and z_k^i denotes the embeddings of the entity e_k^i. N_i refers to the two-hop neighbors of entity e^i. i in z_k^i refers to the i-th one-hop neighborhood subgraph or the target entity e_0^i (see Figures 2 and 4), $k = 0$ refers to the target entity, and k, m = 1, 2, 3... refers to the two-hop neighborhood entities. α_{km}^i is the semantic similarity between two-hop neighbors.

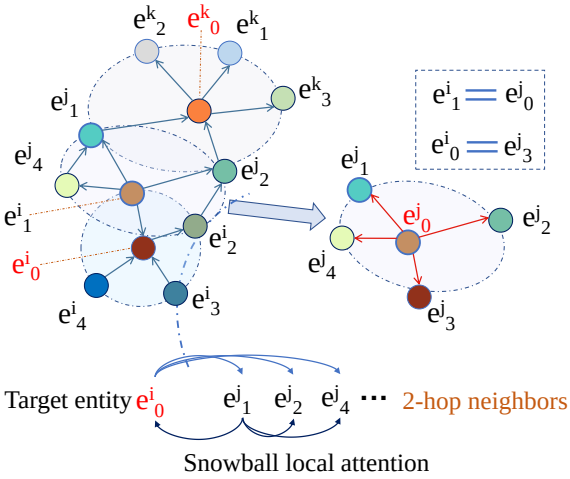


Figure 4: Architecture of Snowball local attention.

$$\alpha_{km}^i = \frac{exp \left(\left(z_m^i \right)^T \left(z_k^i \right) \right)}{\sum_{e_k^i, e_m^i, e_n^i \in \mathcal{N}_i} exp \left(\left(z_n^i \right)^T \left(z_k^i \right) \right)} \quad (15)$$

After passing through the snowball local attention layer, the two-hop neighborhood entities are normalized. The normalization output is then fed into a feedforward neural network, and the output vector z of the feedforward neural network is added with the output vectors of the GAT module and Transformer-based self-attention module. The "Add" is comparable to simplified feature fusion. The Snowball local attention module is stacked into M layers, as shown in Figure 2, with M = 2.

As illustrated in Figure 4, e_0^i is the target entity, and the one-hop neighboring subgraph belongs to it. Graph attention network module performs linear combination on embeddings of one-hop neighbors according to the attention scores and then aggregates these one-hop neighborhood entities on the target entity to learn its new entity representation.

However, GAT takes two stages of graph attention for entity e_0^i to aggregate entity e_1^j. e_1^j is a one-hop neighboring entity of e_0^j (e_1^i) and also is a two-hop neighborhood entity of the target entity e_0^i. Following this, e_0^k is a two-hop neighbor entity of e_1^i, snowball local attention mechanism looks like a snowball effect. To capture richer graph structure information and entity features, we use a snowball local attention mechanism that learns different semantic similarity information to generate entity features and then utilizes these features to fuse feature vectors resulting from the graph attention network module and Transformer-based self-attention architecture.

2.4 Knowledge graph completion module

We specifically choose ConvE (Dettmers et al., 2018) as the decoder. In our experiments, relational features are represented using the initialization feature representations. ConvE computes the knowledge triple scores based on the reshaped tensors after first reshaping the embeddings of triples (h, r, t) into 2D tensors. The prediction from (h, r, ?) to t or from (?, r, t) to h is then carried out by using the output embeddings. The ConvE score function is:

$$f(h,r,t) = \sigma\left(f(\sigma([\widetilde{h};\widetilde{r}] * \psi))W_Q\right)t \quad (16)$$

where $\widetilde{h}$ and $\widetilde{r}$ refer to 2D reshapings of h and r, $*$ stands for the convolution operator, and ψ represents a set of convolution kernels. The vectorization function is $f()$, and the weight matrix is W_Q. σ refers to the ReLU activation function. ConvE implies that triples with higher scores than those with lower scores are positive. The proposed MA-GNN model's loss function is defined as follows:

$$\mathcal{L} = \sum_{(h,r,t)\in\mathcal{T}_t} -\frac{1}{N}\sum_{i=1}^{N}\left(y_{(h,r,t_i)}\left(g\left(f\left(h,r,t_i\right)\right)\right)\right. \\ \left. + \left(1-y_{(h,r,t_i)}\right)\log\left(1-g\left(f\left(h,r,t_i\right)\right)\right)\right) \quad (17)$$

where $y(h,r,t_i)$ is the triple (h,r,t_i)'s label (1 or 0). The sigmoid function is represented by g, and N stands for the number of candidates for the tail entity.

3 Experiments

3.1 Experiment settings

3.1.1 Datasets

Following the standard train/test split, we evaluate our proposed method using the five benchmark datasets: FB15K-237 (Toutanova and Chen, 2015), WN18RR (Dettmers et al., 2018), FB15K (Bordes et al., 2011), WN18 (Bordes et al., 2011), and NELL-995 (Zhao et al., 2022). We further investigate the number of subgraphs in each dataset and the number of entities contained in them. WN18RR, WN18, and NELL-995 are much sparser than FB15K-237 and FB15K, which indicates that these datasets contain less structural information. Table 5 in the appendix shows statistics for all datasets.

3.1.2 Baselines

We compare MA-GNN with geometric methods including TransE (Bordes et al., 2013), RotatE (Sun et al., 2019), ATTH (Chami et al., 2020), TimE (Zhang et al., 2021), Rot-Pro (Song et al., 2021), BiQUE (Guo and Kok, 2021), HBE (Pan and Wang, 2021), DeepER (Zeb et al., 2022), RotatE-IAS (Yang et al., 2022), HousE (Li et al., 2022a), and GIE (Cao et al., 2022); Tensor decomposition methods including ComplEx (Trouillon et al., 2016), Procrustes (Peng et al., 2021), HypER (Balažević et al., 2019); Negative sampling (NS) methods including CAKE (Niu et al., 2022a), KGTuner (Zhang et al., 2022b); Deep learning and attention-based methods including ConvE (Dettmers et al., 2018), Interact (Vashishth et al., 2020a), HittER (Chen et al., 2021), KGA (Wang et al., 2022), PUDA (Tang et al., 2022), JointE (Zhou et al., 2022), EC2 (Niu et al., 2022b), and StructurE (Zhang et al., 2022a); and Graph neural network methods including R-GCN (Schlichtkrull et al.,

2018), KBGAT (Nathani et al., 2019), CompGCN (Vashishth et al., 2020b), SE-GNN (Li et al., 2022a), Rethinking (Zhang et al., 2022c), MRGAT (Li et al., 2022b), and EIGAT (Zhao et al., 2022).

3.2 Overall results

Table 1, 6, and 7 shows the link prediction performance on the test set on standard benchmarks. From the experimental results, we observe that our method significantly outperforms the benchmark methods, especially for sparse knowledge graphs, i.e., WN18RR and NELL-995, in which our method outperforms the second-best methods by **12.7%** and **15.1%** on the Hits@10, respectively.

Our approach ranks second on the H@10 metric on the FB15K-237, but it still performs best on the other metrics. On WN18, FB15K and NELL-995, MA-GNN also achieves competitive results compared to baseline models, with significant improvement on NELL-995 dataset. As shown in the table 1, other methods have competition on some measures, but our method has significant results on all metrics, which demonstrates the robust performance of the proposed approaches in capturing global-local structure information in KGs.

In Table 8, we find that the result of predicting the tail entity in both the validation set and the test set is significantly higher than the result of predicting the head entity, on FB15K-237 and WN18RR, which indicates that our method is more capable of capturing extra information by aggregating neighboring entities when it predicts the tail entity.

3.3 Ablation study

There are three primary modules in MA-GNN: GAT, snowball local attention module, and Transformer based self-attention module. On FB15K-237 and WN18RR, we evaluate the effect of different modules. “MA-GNN w/o A” is the model without using the snowball local attention module, “MA-GNN w/o T” is the model without using Transformer based self-attention module. Besides, “GAT w/ MLP” indicates that the snowball local attention module is replaced by a multilayer perceptron in the model, and the Transformer based self-attention module is removed. From Table 2, we can see that MA-GNN performs better than the model after removing different modules. Compared with “MA-GNN w/o A”, “MA-GNN w/o T ”, and “MA w/ MLP”, MA-GNN is more powerful because it can capture the rich global-local features by using multi-attention methods. On the WN18RR dataset, compared to MA-GNN, the Hits@10-value of “GAT w/ MLP” is **20.8%** lower, the Hits@10-value of “MA-GNN w/o A” is **11.8%** lower, and the Hits@10-value of “MA-GNN w/o T” is **7.5%** lower.

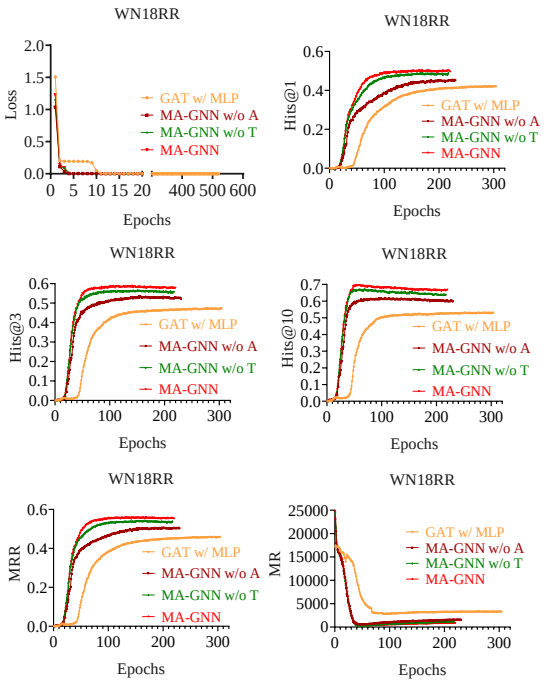


Figure 5: Influence of the different attention module on the WN18RR dataset in MRR, MR, Hits@1, Hits@3, Hits@10.

The fact that "GAT w/ MLP" performed poorly shows how powerful our snowball local attention and Transformer-based self-attention modules are. Additionally, as observed in Figure 5, it is evident that the values of "GAT w/ MLP" undergo the most significant changes, while "MA-GNN w/o A" and "MA-GNN w/o T" exhibit the least variation. This observation implies that the snowball local attention module possesses a greater capacity for encoding information. Due to our early stopping strategy, the convergence curves in Figure 5 are not all the same length. The integration of the GAT module into MA-GNN is crucial for two reasons. Firstly, the snowball local attention module aggregates the feature representations of two-hop neighbors, resulting in the exclusion of certain target entities during calculation. Secondly, the transformer-based self-attention module truncates subgraphs with a high number of multi-hop neighborhood entities during batch computation, further preventing the calculation of specific target entities.

Models	FB15K-237					WN18RR				
	MRR	MR	H@1	H@3	H@10	MRR	MR	H@1	H@3	H@10
Geometric methods										
TransE (2013)	0.330	173	0.231	0.369	0.528	0.223	3380	0.014	0.401	0.529
RotatE (2019)	0.338	177	0.241	0.375	0.533	0.476	3340	0.428	0.492	0.571
ATTH (2020)	0.348	-	0.252	0.384	0.540	0.486	-	0.443	0.499	0.573
TimE (2021)	0.346	171	0.250	0.382	0.537	0.477	2858	0.428	0.493	0.577
Rot-Pro (2021)	0.344	201	0.246	0.383	0.540	0.457	2815	0.397	0.482	0.577
BiQUE (2021)	0.365	-	0.270	0.401	0.555	0.504	-	0.459	0.519	0.588
HBE (2021)	0.336	-	0.239	0.372	0.534	0.488	-	0.448	0.502	0.570
RotatE-IAS (2022)	0.339	195	0.242	0.374	0.532	0.483	3862	0.467	0.502	0.570
HousE (2022)	0.361	153	0.266	0.399	0.551	0.511	1303	0.465	0.528	0.602
GIE (2022)	0.362	-	0.271	0.401	0.552	0.419	-	0.452	0.505	0.575
Tensor decomposition methods										
ComplEx (2016)	0.323	165	0.229	0.353	0.513	0.468	5542	0.427	0.485	0.554
Procrustes (2021)	0.345	-	0.249	0.379	0.541	0.474	-	0.421	0.502	0.569
Negative sampling (NS) methods										
CAKE (2022)	0.321	170	0.226	0.355	0.515	-	-	-	-	-
KGTuner (2022)	0.352	-	0.263	0.387	0.530	0.484	3380	0.440	0.506	0.562
Deep learning and attention-based methods										
ConvE (2018)	0.325	244	0.237	0.356	0.501	0.430	4187	0.400	0.440	0.520
HittER (2021)	0.373	-	0.279	0.409	0.558	0.503	-	0.462	0.516	0.584
KGA (2022)	0.357	-	0.265	-	0.540	-	-	-	-	-
PUDA (2022)	0.369	-	0.268	0.408	**0.578**	0.481	-	0.436	0.498	0.582
JointE (2022)	0.356	177	0.262	0.393	0.543	0.471	4655	0.438	0.483	0.537
StructurE (2022)	0.351	160	0.252	0.390	0.546	0.479	2865	0.425	0.500	0.585
Graph neural network methods										
CompGCN (2020)	0.355	197	0.264	0.390	0.535	0.479	3533	0.443	0.494	0.546
Rethinking (2022)	0.355	249	0.264	0.389	0.535	0.472	3434	0.437	0.485	0.544
SE-GNN (2022)	0.365	157	0.271	0.399	0.549	0.484	3211	0.446	0.509	0.572
MRGAT (2022)	0.358	-	0.266	0.386	0.542	0.481	-	0.443	0.501	0.568
MA-GNN	**0.379**	**145**	**0.282**	**0.415**	0.569	**0.565**	**886**	**0.507**	**0.592**	**0.679**

Table 1: Link prediction results on FB15K-237 and WN18RR. The best results are in bold. Results for TransE, ConvE, RotatE, and ComplEx are from (Li et al., 2022a). Other results are from the published paper.

Models	FB15K-237					WN18RR				
	MRR	MR	H@1	H@3	H@10	MRR	MR	H@1	H@3	H@10
GAT w/ MLP	0.345	214	0.252	0.380	0.532	0.465	3493	0.426	0.481	0.538
MA-GNN w/o A	0.365	159	0.272	0.400	0.550	0.491	1179	0.437	0.512	0.599
MA-GNN w/o T	0.371	**142**	0.275	0.409	0.561	0.509	888	0.446	0.540	0.628
MA-GNN	**0.379**	145	**0.282**	**0.415**	**0.569**	**0.565**	**886**	**0.507**	**0.592**	**0.679**

Table 2: Comparison of the results for different variants of MA-GNN.

4 Related work

4.1 GNN-based models

To date, the most existing GNN-based methods have been proposed to deal with the multi-relational edges in KGs (Nathani et al., 2019; Vashishth et al., 2020b; Yu et al., 2021; Zhang et al., 2022c; Li et al., 2022a). These approaches design different message-passing mechanisms to capture the graph structure and attributes of entities. CompGCN (Vashishth et al., 2020b) describes a compositional operator across each edge connecting the neighborhood entities of the target entity. SE-GNN (Li et al., 2022a) proposes a novel semantic evidence-aware graph neural network-based KGE model to assist KGE extrapolation. Rethinking (Zhang et al., 2022c) aims to explore the real effect of GCNs in KGC, and proposes the LTE-KGE framework, which combines linearly transformed entity embeddings with existing KGE models. MRGAT designed a heterogeneous GNN-based framework for KGs that directly applies GNN to the multi-relational graph (Li et al., 2022b). KBGAT utilizes GAT to incorporate both entity and relational features in any entity's neighborhood (Nathani et al., 2019).

4.2 Attention-based models

Recently, attention-based methods, which measure the semantic similarity of neighbors, have become increasingly popular for knowledge graphs. The recent approaches (Zhang et al., 2020a; Rong et al., 2020; Dwivedi and Bresson, 2020) propose GNN-based frameworks that adopt Transformer-based attention to aggregate neighbor information. (Zhang et al., 2020a) and (Rong et al., 2020) investigate the issue of learning long-distance relationships without over-smoothing by focusing on graph structure rather than one-hop neighbors. In addition to offering different views of attention over the nodes and edges, we attempt to describe link prediction in knowledge graphs. The first effort to forecast missing entities in KGs using attention-based approaches was presented in (Nathani et al., 2019). Additionally, attention techniques are used to incorporate additional information into the learned entity representation (Wang et al., 2021; Zhang et al., 2022d). (Nathani et al., 2019) introduces a triple-level attention model that incorporates the combined information of both entities and relationships in the neighbors of a target entity. (Zhang et al., 2020b) offered a two-level hierarchical attention mechanism, decoupling the triple-level attention of (Nathani et al., 2019) into relationship-level attention and entity-level attention. MRGAT encodes each relation-path-based neighbors feature through the entity-level attention(Li et al., 2022b). Most recently, (Chen et al., 2021; Bi et al., 2022) propose the Transformer-based model to handle the heterogeneity of relations. For a variety of multimodal knowledge graph completion tasks, MKGformer (Chen et al., 2022) utilizes a hybrid transformer architecture with a unified input-output. In comparison, our method focuses on extracting all-sided semantic features by leveraging self-attention to learn long-distance global information on multi-hop paths and snowball local attention to capture local information.

5 Conclusion

In this paper, we propose MA-GNN, a simple and efficient framework for learning global-local structure information based on multi-attention. We incorporate a Transformer-based self-attention module into a standard GAT module to encode local graph structure information and learn long-range relationships. We design a snowball local attention mechanism to enrich the entity embeddings based on the similarities between two-hop neighborhood entities. On the five commonly known benchmark datasets, empirical experiments show that our proposed model achieves competitive performance compared with state-of-the-art performance.

6 Limitations

The paper has only focused on graphs with multi-type relations (knowledge graphs). When MA-GNN shows improvement over baselines, someone may doubt if MA-GNN will do well on single-type relation graphs. The limitations of the representational power of the MA-GNN model should be discussed more deeply.

7 Ethics Statement

MA-GNN focuses on improving the performance of link prediction and has no particular ethical consideration.

Acknowledgements

We appreciate the reviewers and the program committee for their helpful comments and suggestions. The work is supported by the CAAI-Huawei Mindspore Fund.

References

Marjan Albooyeh, Rishab Goel, and Seyed Mehran Kazemi. 2020. Out-of-sample representation learning for knowledge graphs. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 2657–2666.

Ivana Balažević, Carl Allen, and Timothy M Hospedales. 2019. Hypernetwork knowledge graph embeddings. In *International Conference on Artificial Neural Networks*, pages 553–565. Springer.

Zhen Bi, Siyuan Cheng, Ningyu Zhang, Xiaozhuan Liang, Feiyu Xiong, and Huajun Chen. 2022. Relphormer: Relational graph transformer for knowledge graph representation. *arXiv preprint arXiv:2205.10852*.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. *Advances in neural information processing systems*, 26.

Antoine Bordes, Jason Weston, Ronan Collobert, and Yoshua Bengio. 2011. Learning structured embeddings of knowledge bases. In *Twenty-fifth AAAI conference on artificial intelligence*.

Zongsheng Cao, Qianqian Xu, Zhiyong Yang, Xiaochun Cao, and Qingming Huang. 2022. Geometry interaction knowledge graph embeddings. In *AAAI Conference on Artificial Intelligence*.

Ines Chami, Adva Wolf, Da-Cheng Juan, Frederic Sala, Sujith Ravi, and Christopher Ré. 2020. Low-dimensional hyperbolic knowledge graph embeddings. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6901–6914, Online. Association for Computational Linguistics.

Deli Chen, Yankai Lin, Wei Li, Peng Li, Jie Zhou, and Xu Sun. 2020. Measuring and relieving the over-smoothing problem for graph neural networks from the topological view. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 3438–3445.

Sanxing Chen, Xiaodong Liu, Jianfeng Gao, Jian Jiao, Ruofei Zhang, and Yangfeng Ji. 2021. Hitter: Hierarchical transformers for knowledge graph embeddings. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 10395–10407.

Xiang Chen, Ningyu Zhang, Lei Li, Shumin Deng, Chuanqi Tan, Changliang Xu, Fei Huang, Luo Si, and Huajun Chen. 2022. Hybrid transformer with multi-level fusion for multimodal knowledge graph completion. SIGIR '22, page 904–915. Association for Computing Machinery.

Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. 2016. Convolutional neural networks on graphs with fast localized spectral filtering. *Advances in neural information processing systems*, 29.

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

Vijay Prakash Dwivedi and Xavier Bresson. 2020. A generalization of transformer networks to graphs. *arXiv preprint arXiv:2012.09699*.

Jia Guo and Stanley Kok. 2021. Bique: Biquaternionic embeddings of knowledge graphs. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 8338–8351.

Takuo Hamaguchi, Hidekazu Oiwa, Masashi Shimbo, and Yuji Matsumoto. 2017. Knowledge transfer for out-of-knowledge-base entities: A graph neural network approach. International Joint Conferences on Artificial Intelligence.

Magdalena Kaiser, Rishiraj Saha Roy, and Gerhard Weikum. 2021. Reinforcement learning from reformulations in conversational question answering over knowledge graphs. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 459–469.

Thomas N Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*.

Qimai Li, Zhichao Han, and Xiao-Ming Wu. 2018. Deeper insights into graph convolutional networks for semi-supervised learning. In *Thirty-Second AAAI conference on artificial intelligence*.

Ren Li, Yanan Cao, Qiannan Zhu, Guanqun Bi, Fang Fang, Yi Liu, and Qian Li. 2022a. How does knowledge graph embedding extrapolate to unseen data: a semantic evidence view. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 5781–5791.

Zhifei Li, Yue Zhao, Yan Zhang, and Zhaoli Zhang. 2022b. Multi-relational graph attention networks for knowledge graph completion. *Knowledge-Based Systems*, 251:109262.

Deepak Nathani, Jatin Chauhan, Charu Sharma, and Manohar Kaul. 2019. Learning attention-based embeddings for relation prediction in knowledge graphs. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics.

Guanglin Niu, Bo Li, Yongfei Zhang, and Shiliang Pu. 2022a. Cake: A scalable commonsense-aware framework for multi-view knowledge graph completion. In *ACL*.

Guanglin Niu, Bo Li, Yongfei Zhang, Yongpan Sheng, Chuan Shi, Jingyang Li, and Shiliang Pu. 2022b. Joint semantics and data-driven path representation for knowledge graph reasoning. *Neurocomputing*, 483:249–261.

Zhe Pan and Peng Wang. 2021. Hyperbolic hierarchy-aware knowledge graph embedding for link prediction. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 2941–2948.

Xutan Peng, Guanyi Chen, Chenghua Lin, and Mark Stevenson. 2021. Highly efficient knowledge graph embedding learning with orthogonal procrustes analysis. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 701–710.

Yu Rong, Yatao Bian, Tingyang Xu, Weiyang Xie, Ying Wei, Wenbing Huang, and Junzhou Huang. 2020. Self-supervised graph transformer on large-scale molecular data. *Advances in Neural Information Processing Systems*, 33:12559–12571.

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.

Tengwei Song, Jie Luo, and Lei Huang. 2021. Rotpro: Modeling transitivity by projection in knowledge graph embedding. *Advances in Neural Information Processing Systems*, 34:24695–24706.

Zhiqing Sun, Zhi-Hong Deng, Jian-Yun Nie, and Jian Tang. 2019. Rotate: Knowledge graph embedding by relational rotation in complex space. In *International Conference on Learning Representations*.

Zhenwei Tang, Shichao Pei, Zhao Zhang, Yongchun Zhu, Fuzhen Zhuang, Robert Hoehndorf, and Xiangliang Zhang. 2022. Positive-unlabeled learning with adversarial data augmentation for knowledge graph completion. In *IJCAI*, pages 2248–2254.

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. PMLR.

Shikhar Vashishth, Soumya Sanyal, Vikram Nitin, Nilesh Agrawal, and Partha Talukdar. 2020a. Interacte: Improving convolution-based knowledge graph embeddings by increasing feature interactions. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 3009–3016.

Shikhar Vashishth, Soumya Sanyal, Vikram Nitin, and Partha Talukdar. 2020b. Composition-based multi-relational graph convolutional networks. In *International conference on learning representations*.

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

Jiang Wang, Filip Ilievski, Pedro Szekely, and Ke-Thia Yao. 2022. Augmenting knowledge graphs for better link prediction. In *IJCAI*, pages 2277–2283.

Xiang Wang, Tinglin Huang, Dingxian Wang, Yancheng Yuan, Zhenguang Liu, Xiangnan He, and Tat-Seng Chua. 2021. Learning intents behind interactions with knowledge graph for recommendation. In *Proceedings of the Web Conference 2021*, pages 878–887.

Zhanghao Wu, Paras Jain, Matthew Wright, Azalia Mirhoseini, Joseph E Gonzalez, and Ion Stoica. 2021. Representing long-range context for graph neural networks with global attention. *Advances in Neural Information Processing Systems*, 34:13266–13279.

Chenyan Xiong, Russell Power, and Jamie Callan. 2017. Explicit semantic ranking for academic search via knowledge graph embedding. In *Proceedings of the 26th international conference on world wide web*, pages 1271–1279.

Jinfa Yang, Xianghua Ying, Yongjie Shi, Xin Tong, Ruibin Wang, Taiyan Chen, and Bowei Xing. 2022. Knowledge graph embedding by adaptive limit scoring loss using dynamic weighting strategy. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 1153–1163.

Donghan Yu, Yiming Yang, Ruohong Zhang, and Yuexin Wu. 2021. Knowledge embedding based graph convolutional network. In *Proceedings of the Web Conference 2021*, pages 1619–1628.

Adnan Zeb, Summaya Saif, Junde Chen, and Defu Zhang. 2022. Learning knowledge graph embeddings by deep relational roto-reflection. *Knowledge-Based Systems*, 252:109451.

Jiawei Zhang, Haopeng Zhang, Congying Xia, and Li Sun. 2020a. Graph-bert: Only attention is needed for learning graph representations. *arXiv preprint arXiv:2001.05140*.

Qianjin Zhang, Ronggui Wang, Juan Yang, and Lixia Xue. 2021. Knowledge graph embedding by translating in time domain space for link prediction. *Knowledge-Based Systems*, 212:106564.

Qianjin Zhang, Ronggui Wang, Juan Yang, and Lixia Xue. 2022a. Structural context-based knowledge graph embedding for link prediction. *Neurocomputing*, 470:109–120.

Yongqi Zhang, Zhanke Zhou, Quanming Yao, and Yong Li. 2022b. Efficient hyper-parameter search for knowledge graph embedding. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2715–2735.

Zhanqiu Zhang, Jie Wang, Jieping Ye, and Feng Wu. 2022c. Rethinking graph convolutional networks in knowledge graph completion. In *Proceedings of the ACM Web Conference 2022*, pages 798–807.

Zhao Zhang, Fuzhen Zhuang, Hengshu Zhu, Zhiping Shi, Hui Xiong, and Qing He. 2020b. Relational graph neural network with hierarchical attention for knowledge graph completion. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 9612–9619.

Zhenghao Zhang, Jianbin Huang, and Qinglin Tan. 2022d. Association rules enhanced knowledge graph attention network. *Knowledge-Based Systems*, 239:108038.

Yu Zhao, Huali Feng, Han Zhou, Yanruo Yang, Xingyan Chen, Ruobing Xie, Fuzhen Zhuang, and Qing Li. 2022. Eigat: Incorporating global information in local attention for knowledge representation learning. *Knowledge-Based Systems*, 237:107909.

Zhehui Zhou, Can Wang, Yan Feng, and Defang Chen. 2022. Jointe: Jointly utilizing 1d and 2d convolution for knowledge graph embedding. *Knowledge-Based Systems*, 240:108100.

A Appendix

A.1 Setup

In our model, Both the modules (GATs and snowball local attention module) have 2 layers with the ReLU activation function. Transformer-based self-attention is configured with eight heads and five layers. Additionally, we use the following measures to evaluate our models against baselines: (1) Mean Rank (MR, the mean of all expected rankings); (2) Mean Reciprocal Rank (MRR, the mean of all reciprocals of predicted ranks) (3) Hits@n, n=1, 3, 10 (H@1, H@3, H@10).

We perform a hyperparameter search on the dimensions, learning rate, optimizer, negative sample size, and batch size for our proposed model. For each dataset, we list the optimal hyperparameters (dimensionality, learning rate, optimizer, batch normalization, batch size, label smooth, entity dropout, convolution dropout, FC dropout, and Convolution kernel size) as follows: WN18RR:400, 0.0015, Adam, True, 256, 0.1, 0.2, 0.1, 0.4, 7; FB15K-237: 400, 0.00035, Adam, True, 1024, 0.1, 0.3, 0.3, 0.5, 7; FB15K: 400, 0.00035, Adam, True, 1024, 0.1, 0.3, 0.3, 0.5, 7; WN18: 400, 0.00035, Adam, True, 1024, 0.1, 0.3, 0.3, 0.5, 7; NELL995: 400, 0.00035, Adam, True, 1024, 0.1, 0.3, 0.3, 0.5, 7.

A.2 Number of parameters

We compare the number of parameters of the baselines and the MA-GNN on the FB15K-237 dataset in Table 3. MA-GNN is substantially more parameter-efficient than the ConvE while improving the test Hits@10 score from 0.497 to 0.569.

A.3 Dimension selection

To show the impact of entity embedding dimensions on performance, the performance of link prediction with the different number of embedding dimensions is shown in Table 4. It shows that MA-GNN's performance first rises and then falls, as the entity embedding dimension increases, and when the embedding dimension is 400, MA-GNN achieves optimal results on FB15K-237 and WN18RR datasets. This verifies that the entity embedding dimension has an impact on the link prediction task.

A.4 Impact of transformer-based self-attention

To study the effects of Transformer-based self-attention in MA-GNN, we random an instance and visualize its attention matrix to analyze the impact of transformer-based self-attention. From Figure 6, given the long-distance sequential paths, we observe that models with transformer-based self-attention have a vital impact on global weights and can capture the semantic similarity between long-range entities.

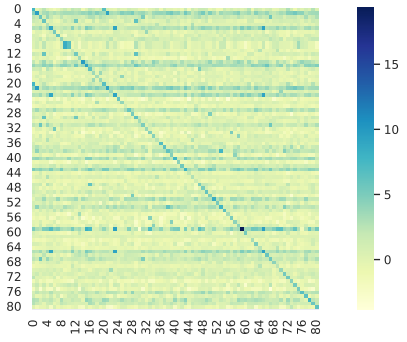


Figure 6: Attention map from Transformer-based self-attention.

Models	Parameters	MRR	Hits@1	Hits@10
ConvE (2018)	5.05M	0.312	0.225	0.497
HypER (2019)	**4.30M**	0.341	-	0.520
InteractE (2020)	10.70M	0.354	0.263	0.535
JointE (2022)	4.54M	0.356	0.262	0.537
MA-GNN	9.34M	**0.379**	**0.282**	**0.569**

Table 3: Parameter count. MA-GNN is substantially more parameter-efficient than the ConvE while improving the test Hits@10 score from 0.497 to 0.569.

Dimensions	FB15K-237					WIN18RR				
	MRR	MR	H@1	H@3	H@10	MRR	MR	H@1	H@3	H@10
200	0.338	151	0.250	0.370	0.512	0.504	1875	0.455	0.524	0.601
300	0.363	**138**	0.271	0.397	0.544	0.541	1040	0.488	0.564	0.646
400	**0.379**	145	**0.282**	**0.415**	**0.569**	**0.565**	**886**	**0.507**	**0.592**	**0.679**
500	0.372	149	0.277	0.408	0.560	0.556	1788	0.505	0.580	0.657
600	0.371	147	0.274	0.409	0.565	0.564	1487	0.510	0.587	0.667

Table 4: Impact of different embedding dimensions.

Datasets	Entities	Relations	Train triplets	Validation Triplets	Test triplets
FB15K-237	14541	237	272115	17535	20466
WN18RR	40943	11	86835	3034	3134
FB15K	14951	1345	483142	50000	59071
WN18	40943	18	141442	5000	5000
NELL-995	75492	200	149678	543	3992

Table 5: Statistics of datasets

Models	NELL-995			
	MRR	H@1	H@3	H@10
TransE (2013)	0.401	0.344	0.472	0.501
ComplEx (2016)	0.482	0.399	0.528	0.606
ConvE (2018)	0.491	0.403	0.531	0.613
R-GCN (2018)	0.12	0.082	0.126	0.188
KBGAT (2019)	0.530	0.447	0.564	0.695
EC2 (2022)	0.350	0.281	0.402	0.475
EIGAT (2022)	0.545	0.464	0.584	0.715
MA-GNN	**0.714**	**0.645**	**0.766**	**0.823**

Table 6: Link prediction results on NELL-995.

Models	FB15K				WN18			
	MRR	H@1	H@3	H@10	MRR	H@1	H@3	H@10
TransE (2013)	0.463	0.297	0.578	0.749	0.495	0.113	0.888	0.943
RotatE (2019)	0.797	0.746	0.830	0.884	0.949	0.944	0.952	0.959
ConvE (2018)	0.657	0.558	0.723	0.831	0.943	0.935	0.946	0.956
R-GCN (2018)	0.348	0.252	0.384	0.540	0.486	0.443	0.499	0.573
ComplEx (2016)	0.692	0.599	0.759	0.839	0.941	0.936	0.945	0.947
HypER (2019)	0.790	0.734	0.829	0.885	0.951	0.947	0.955	0.958
CompGCN (2020)	0.801	0.715	0.834	0.862	0.942	0.921	0.933	0.941
DeepER (2022)	0.759	0.692	0.804	0.875	**0.952**	**0.948**	0.955	0.958
EC2 (2022)	0.715	0.651	0.750	0.857	0.946	0.940	0.945	0.952
MRGAT (2022)	0.813	0.741	0.853	0.893	0.947	0.932	0.946	0.971
MA-GNN	**0.817**	**0.747**	**0.871**	**0.932**	0.905	0.851	**0.959**	**0.991**

Table 7: Link prediction results on FB15K and WN18.

Models	FB15K-237					WN18RR				
	MRR	MR	H@1	H@3	H@10	MRR	MR	H@1	H@3	H@10
Valid(Head)	0.289	192	0.193	0.316	0.476	0.525	891	0.468	0.548	0.639
Valid(Head)	0.487	100	0.384	0.525	0.672	0.595	615	0.541	0.583	0.704
Valid(Head)	0.285	183	0.189	0.310	0.473	0.519	1036	0.457	0.548	0.636
Valid(Head)	0.473	107	0.375	0.520	0.665	0.612	738	0.557	0.636	0.722

Table 8: Evaluation for predicting head-and-tail entities.

A.5 Effectiveness of snowball local attention

On the WN18RR dataset, we visualize the similarity weights of the two-hop neighbors of entity e_0. From the dark color in the red dashed box in Figure 7, we observe some degree of similarity between the two-hop neighbors. This result matches our motivation that captures richer graph structure information and entity features using two-hop neighbors and verifies the effectiveness of our proposed snowball local attention mechanism.

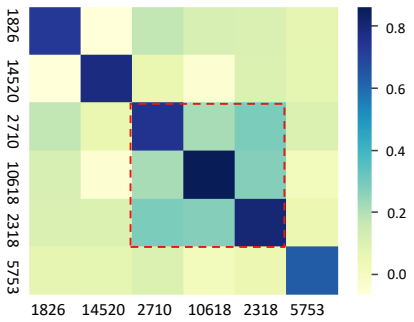


Figure 7: The attention values between two-hop neighbors.

ACL 2023 Responsible NLP Checklist

A For every submission:

☑ A1. Did you describe the limitations of your work?
6

☑ A2. Did you discuss any potential risks of your work?
6

☑ A3. Do the abstract and introduction summarize the paper's main claims?
1

☑ A4. Have you used AI writing assistants when working on this paper?
Grammarly was used to check for grammatical errors in this paper.

B ☒ Did you use or create scientific artifacts?

Left blank.

☐ B1. Did you cite the creators of artifacts you used?
No response.

☐ B2. Did you discuss the license or terms for use and / or distribution of any artifacts?
No response.

☐ B3. Did you discuss if your use of existing artifact(s) was consistent with their intended use, provided that it was specified? For the artifacts you create, do you specify intended use and whether that is compatible with the original access conditions (in particular, derivatives of data accessed for research purposes should not be used outside of research contexts)?
No response.

☐ B4. Did you discuss the steps taken to check whether the data that was collected / used contains any information that names or uniquely identifies individual people or offensive content, and the steps taken to protect / anonymize it?
No response.

☐ B5. Did you provide documentation of the artifacts, e.g., coverage of domains, languages, and linguistic phenomena, demographic groups represented, etc.?
No response.

☐ B6. Did you report relevant statistics like the number of examples, details of train / test / dev splits, etc. for the data that you used / created? Even for commonly-used benchmark datasets, include the number of examples in train / validation / test splits, as these provide necessary context for a reader to understand experimental results. For example, small differences in accuracy on large test sets may be significant, while on small test sets they may not be.
No response.

C ☑ Did you run computational experiments?

Appendix

☑ C1. Did you report the number of parameters in the models used, the total computational budget (e.g., GPU hours), and computing infrastructure used?
Appendix

The Responsible NLP Checklist used at ACL 2023 is adopted from NAACL 2022, with the addition of a question on AI writing assistance.

☑ C2. Did you discuss the experimental setup, including hyperparameter search and best-found hyperparameter values?
Appendix

☑ C3. Did you report descriptive statistics about your results (e.g., error bars around results, summary statistics from sets of experiments), and is it transparent whether you are reporting the max, mean, etc. or just a single run?
3

☑ C4. If you used existing packages (e.g., for preprocessing, for normalization, or for evaluation), did you report the implementation, model, and parameter settings used (e.g., NLTK, Spacy, ROUGE, etc.)?
3, Appendix

D ☒ Did you use human annotators (e.g., crowdworkers) or research with human participants?

Left blank.

☐ D1. Did you report the full text of instructions given to participants, including e.g., screenshots, disclaimers of any risks to participants or annotators, etc.?
No response.

☐ D2. Did you report information about how you recruited (e.g., crowdsourcing platform, students) and paid participants, and discuss if such payment is adequate given the participants' demographic (e.g., country of residence)?
No response.

☐ D3. Did you discuss whether and how consent was obtained from people whose data you're using/curating? For example, if you collected data via crowdsourcing, did your instructions to crowdworkers explain how the data would be used?
No response.

☐ D4. Was the data collection protocol approved (or determined exempt) by an ethics review board?
No response.

☐ D5. Did you report the basic demographic and geographic characteristics of the annotator population that is the source of the data?
No response.