ABM: Attention-based Message Passing Network for Knowledge Graph Completion

Weikai Xu, Lihui Liu, Hanghang Tong
Department of Computer Science
University of Illinois Urbana-Champaign
Urbana, United States
{weikaix2, lihuil2, htong}@illinois.edu

***Abstract*—Knowledge graph is ubiquitous and plays an important role in many real-world applications, including recommender systems, question answering, fact-checking, and so on. However, most of the knowledge graphs are incomplete which can hamper their practical usage. Fortunately, knowledge graph completion (KGC) can mitigate this problem by inferring missing edges in the knowledge graph according to the existing information. In this paper, we propose a novel KGC method named ABM (ATTENTION-BASED MESSAGE PASSING) which focuses on predicting the relation between any two entities in a knowledge graph. The proposed ABM consists of three integral parts, including (1) context embedding, (2) structure embedding, and (3) path embedding. In the context embedding, the proposed ABM generalizes the existing message passing neural network to update the node embedding and the edge embedding to assimilate the knowledge of nodes' neighbors, which captures the relative role information of the edge that we want to predict. In the structure embedding, the proposed method overcomes the shortcomings of the existing GNN method (i.e., most methods ignore the structural similarity between nodes.) by assigning different attention weights to different nodes while doing the aggregation. Path embedding generates paths between any two entities and treats these paths as sequences. Then, the sequence can be used as the input of the Transformer to update the embedding of the knowledge graph to gather the global role of the missing edges. By utilizing these three mutually complementary strategies, the proposed ABM is able to capture both the local and global information which in turn leads to a superb performance. Experiment results show that ABM outperforms baseline methods on a wide range of datasets.**

***Index Terms*—Attention, Knowledge Graph Completion**

I. Introduction

Knowledge graphs are ubiquitous data structures to explain the relationship between entities in the real world. A knowledge graph stores each fact as a triple (h, t, r) which means that the head h is related to the tail t through the relationship r. Many real-world knowledge graphs like WordNet, YAGO, Freebase, DBpedia have been widely used in various applications, such as recommender systems [31], dialog systems [32], question answering [33], and so on. However, most existing knowledge graphs are incomplete and noisy, which hamper their practical usage.

Knowledge graph completion aims to complete the input knowledge graph by predicting the missing link between two entities with the help of existing information in the knowledge graph, which has become an important research direction. An illustrative example of knowledge graph is shown in Figure 1.

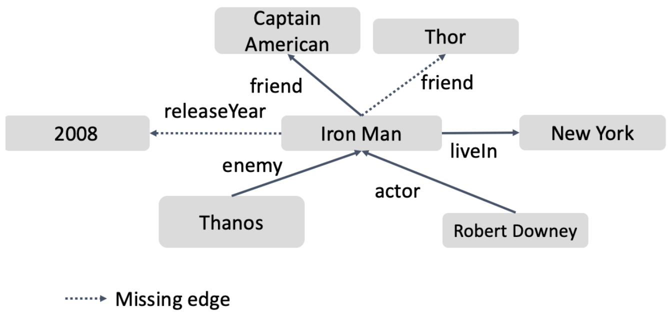

Fig. 1. An illustrative example of knowledge graph completion. Suppose we know that Iron Man and Captain America are friends, but this knowledge graph misses the relationship between Iron Man and Thor. Suppose we also know that Thanos and Iron Man are friends and Robert Downey is the actor of Iron Man. However, in this knowledge graph, we do not know that Iron Man is released in 2008, which is what we need to predict in the knowledge graph completion.

Generally speaking, existing KGC methods can be divided into three categories. The first type of method is based on translation models. The representative methods include TransE [1], TransD [2], TransH [3] and TransR [4]. They assume that in the valid triple, the head can be translated to the tail through the relationships. They define different score functions and minimize the loss function to learn the representation of entities and relationships. The second type of method is based on semantic matching models which include RESCAL [5], DistMult [6], HolE [7], ComplEx [8]. They use the score function based on semantic similarity to get the association between the embedding of entities and relationships. The last type of method is neural network-based methods, such as ConvE [9], SEAL [10]. These methods are based on network representation, which can extract the content information of the nodes and edges.

Despite the great accomplishments achieved recently, the traditional translation model, such as TransE, TransD, and TransH, do not consider the neighborhood information of given entities, while the neural network-based methods, such as ConvE, and SEAL, do not consider structure similarity between nodes, which could lose some important features. These shortcomings inevitably limit their performance since both the neighborhood information and the structure information should be important to the knowledge graph completion. Besides, most message passing based GNN methods assume that when doing aggregation for the node feature, the message passing from neighbors should get the same attention, which could not be suitable for the knowledge graph setting where both the relationship between nodes and the structure of the entire graph is important in the aggregation step.

To overcome the above limitations, we propose an attention-based method called ABM whose main idea is to combine structure embedding, context embedding, and path embedding to deal with the knowledge graph completion problem. More specifically, the proposed ABM is built upon three key components which are context embedding, structure embedding, and path embedding. In the context embedding, we embed the neighbor information by using the neighboring nodes' embedding and neighboring edges' embedding. We take the node embedding and relation embedding as the input of the multihead attention aggregation and then use the message passing mechanism to get the relational embedding between these two nodes. In the structure embedding, we combine the structure information with the context embedding. The nodes which bear higher similarity in the whole graph structure will get more attention when combining the context embedding. In the path embedding part, we generate several paths from the head to the tail and then use the Transformer [11] to obtain the relationship representation. In contrast to the previous models, the attention-based message passing neural network strengthens the message interactions between nodes and edges. We conduct extensive experiments and illustrate the effectiveness of our method.

The contributions of this paper are summarized as follows.

- We propose a novel KGC method named ABM for knowledge graph completion, which is storage efficient and explainable with existing embedding-based methods.
- ABM embraces three kinds of embedding: context embedding, structure embedding, and path embedding, all of which are critical to the relation prediction.
- ABM obtains superb performance than baseline methods on a variety of datasets.

The rest of the paper is organized as follows. Section II introduces our preliminary and the definition of the problem. Section III introduces the detail of our method. Section IV compares our method with existing knowledge graph completion methods. Section V shows some related work of our paper. Section VI gives our conclusion of this paper.

II. Preliminary and Problem Definition

In this chapter, we formally define the knowledge graph completion and alternative message passing method. Table I summarizes the main symbols and notations used throughout the paper. Let $\mathcal{G} = \{\mathcal{V}, \mathcal{E}, \mathcal{R}\}$ be a knowledge graph, where $\mathcal{V}$ is the set of nodes/entities in the graph $\mathcal{G}$, $\mathcal{E}$ is the sets of edges in this graph $\mathcal{G}$, and $\mathcal{R}$ is the set of the relationships. We define a knowledge graph triple as (h, t, r), where h is the head entity, t is the tail entity and r is the relationship between them. Notice that for a pair of (h, t), it can have different relation types. For example, "Harry Potter" and "Hermione Granger" are both friends and classmates.

Symbol	Description
h, t	The head entity and tail entity
r	The relationship type
n_v^i	The hidden state of node v in the i^{th} iteration
$\widetilde{n}_v^i$	The output of the context aggregator for node v in i^{th} iteration
$\widehat{n}_v^i$	The output of the location aggregator for node v in i^{th} iteration
$s_{(h,t)}^i$	The edge embedding between the head and tail in the i^{th} iteration
r_j^i	The relation embedding for type j in the i^{th} iteration
$\mathcal{N}(v)$	The neighbor points of node v
$Q_i, K_i, V_i.$	The query, key, and value matrix of the attention for the task i
$W_i.$	MLP for the task i
α_{ij}	The attention weight between node i and node j

TABLE I
Notations and definitions

Traditional node-based message passing. Let us first briefly review the traditional message passing methods. For the traditional message passing, they typically use the node feature to do the message passing. In the i^{th} iteration during training, we get the node v's embedding which is n_v^i, then the embedding is updated by

$$\begin{aligned} m_v^i &= A(\{n_u^i | u \in \mathcal{N}(v)\}) \\ n_v^{i+1} &= F(n_v^i, m_v^i) \end{aligned}$$

where m_v^i is the message aggregated by the node v in the i^{th} iteration, $A(\cdot)$ is the aggregation function for the neighbors' message, and $F(\cdot)$ is the updated function after getting the aggregation result. After we get the node embedding of the entities, we can use these entities to predict the relation between them.

Using the previous framework, we can see that the traditional framework focuses on the embedding of nodes while ignoring the relationship and structure similarity between the nodes. However, these relationships and graph structure between the nodes are critical for the relationship prediction task. Thus, it is indispensable to take them into account.

Alternative message passing. In order to update the node embedding more accurately, we designed the alternative message passing method which use not only the node embedding but also the edge embedding to update the node embedding which should give each neighbor a different attention when aggregating. More specifically, the update function can be designed as follows:

$$\begin{aligned} m_v^i &= A(\{(n_u^i, s_{(u,v)}^i)n | u \in \mathcal{N}(v)\}) \\ n_v^{i+1} &= F(n_v^i, m_v^i) \end{aligned}$$

where $s_{(u,v)}^i$ should be the relationship embedding in the ith iteration. By using this alternative message passing function, different neighbors will get different attention by using the relationship between the node and their neighbors.

Problem Definition. Given a knowledge graph, we focus on predicting the relationship between two nodes which is defined as follows:

Problem Definition 1: Knowledge Graph Completion

Given: *the incomplete knowledge graph:* $\mathcal{G} = \{\mathcal{V}, \mathcal{E}, \mathcal{R}\}$, *the head entity* h, *the tail entity* t.

Output: *the relationship* r *between the head entity* h *and the tail entity* t.

III. Proposed Model

In this section, we first introduce the framework of our method in Subsection III-A. Then, we introduce the detail of ABM, including context embedding, structure embedding, and path embedding, in Subsection III-B-III-E.

A. Overview of the Framework

The proposed neural network framework of ABM is shown in Figure 2. It consists of three integral parts: context embedding, structure embedding, and path embedding. We will use these three types of embedding between two entities h and t to predict the relationship between them. First, we create the context embedding based on the message passing neural network with multihead attention, which assigns different weights to different neighbor relationships. Different from traditional message passing which only considers the neighbor nodes' embedding while ignoring the relationship between them, our attention message passing method can efficiently encode neighbor information. Second, for the structure embedding, we leverage the graph convolutional networks to encode the structure embedding of the entire graph. Then, we give different neighbor nodes different weights according to their structure similarity to the whole entire graph, which can encode the structure similarity between the node and its neighbors. Last but not the least, for the path embedding, we sample several paths from h to t and treat these paths as the input of a Transformer [11] to update the relationship embedding between h and t, which allows us to capture the global information between the head entity and tail entity.

Since the ABM generates the model by both node embedding and relationship embedding, it is able to gather the different types of information in the aggregation step, which in turn makes it easy for us to explain predictions. In the following subsections, we will use the illustrative example in Figure 3 to introduce each component in detail.

B. Context Embedding

Context embedding aims to update the node's embedding by using its neighbors' embedding. When encoding a node's context embedding, it is important to accurately capture its semantic information. For example, the relationships around node `Lebron James` are `Play for` and `Friend`, and the neighbors of node `Lebron James` are `Laker` and `Dwyane Wade`. We can infer that this node is a basketball player. If the relationships around node `Chris Paul` are `Play in` and `Friend`, and the node around node `NBA` and `Dwyane Wade`, then we can infer that this node should also be a basketball player. So, the relationship between `Lebron James` and `Chris Paul` is most likely `Teammate` or `Friend`. In other words, context information is very important when predicting the relationship.

However, traditional message passing algorithms simply aggregate neighbor information with add, mean, or max and assign the same weight to all neighbors which are likely to be sub-optimal.

To better aggregate the neighbor information, we use the attention mechanism to assign different weights to different neighbors according to their relationships.

We define that the node v's hidden embedding in the i^{th} iteration of aggregation as $n_v^i \in \mathbb{R}^d$ and the edge (u, v)'s hidden embedding in the i^{th} iteration as $s^i_{(u,v)} \in \mathbb{R}^d$ where d is the embedding dimension of the hidden embedding. Given a head entity h and a tail entity t, we use two different strategies to calculate the attention weights. The first one is a GAT-based method and the second one is based on multihead attention.

GAT-based Aggregation. In the GAT-based aggregation, we use the idea of GAT [30] and replace the neighbors' node feature with the neighbors' edge feature for the contextual correlation during aggregation. The contextual correlation between two connected nodes u, v can be defined as

$$f_{uv} = \frac{\exp(\text{LeakyReLU}(a^T s^i_{(u,v)}))}{\sum_{k \in \mathcal{N}(u)} \text{LeakyReLU}(a^T s^i_{(u,k)}))} \tag{1}$$

where $a \in \mathbb{R}^d$ is the parameter of the feed forward layer. After we get the contextual correlation f_{uv} between two connected nodes, the node h's embedding can be updated as follows

$$\widetilde{n}_h^i = \sum_{j \in \mathcal{N}(h)} f_{uj} W_f n_j^i \tag{2}$$

Note that this strategy is similar as the graph attention networks (GAT). It assigns each neighbor a different attention weight based on the relation types.

Multihead-attention Aggregation. As we have shown in Section II, we update the node embedding by using the neighbor nodes' embedding and neighbor edges' embedding. In this method, we use the multihead-attention when aggregating the neighbor information. For example, we wish to update the embedding of node h. We should use the node h's embedding as key, the node h's neighbor relation embedding as query, and the node h's neighbor nodes' embedding as value. The update method is

$$Q_c = n_h^i \cdot W_{Q_c} \tag{3}$$

$$V_c = \begin{bmatrix} n^i_{v_1} \\ \vdots \\ n^i_{v_k} \end{bmatrix} \cdot W_{V_c}, \quad v_j \in \mathcal{N}(h) \tag{4}$$

$$K_c = \begin{bmatrix} e^i_{(v_1,h)} \\ \vdots \\ e^i_{(v_k,h)} \end{bmatrix} \cdot W_{K_c}, \quad v_j \in \mathcal{N}(h) \tag{5}$$

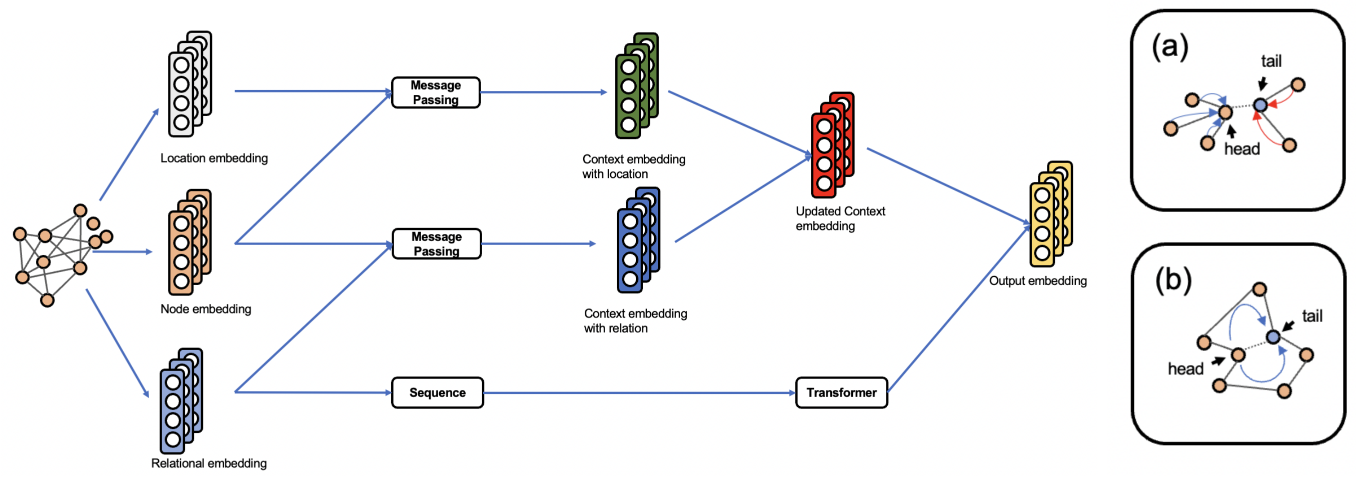

Fig. 2. An overview of the proposed framework of ABM. Right: (a) an illustration of Message Passing which aggregates the message of head and tail's neighbors and the edge connected with them, and (b) an illustration of using Transformer to take several paths as sequences to obtain the context embedding.

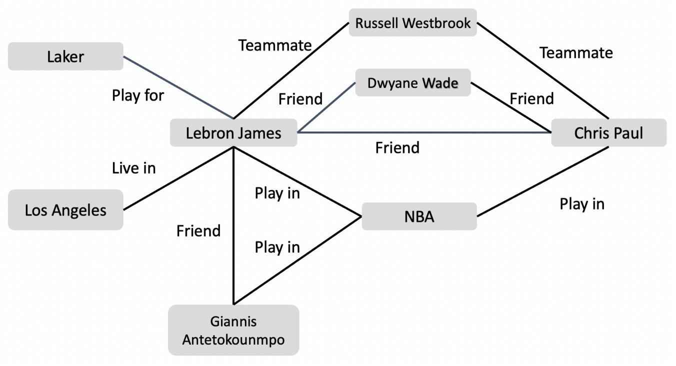

Fig. 3. An illustrative sample for the model explainability

where $W_{Q_c}, W_{K_c}, W_{V_c}$ are the projection matrices. Then, the attention of Q_c, K_c, V_c can be calculated as

$$\text{Attention}(Q_c, K_c, V_c) = \text{Softmax}(Q_c K_c^T/\sqrt{d})V_c \tag{6}$$

Furthermore, we focus on the multihead attention, where l attention layers are stacked together. The multihead attention is defined as

$$\widetilde{n}_h^i = \text{Cat}(head^1, \cdots, head^l)W_O \tag{7}$$

$$head^i = \text{Attention}(Q_c^i, K_c^i, V_c^i) \tag{8}$$

, where Cat should be the function that concatenate all the function together.

To make the training step more stable, we add the residual connection between any two message passing layers, which can reduce the vanishing of gradient. This is achieved by

$$\widetilde{n}_h^i = \text{Layernorm}\left(n_h^i + \text{Layernorm}(\widetilde{n}_h^{i+1})\right) \tag{9}$$

, where Layernorm is to normalize the data after the hidden layer.

C. Structure embedding

When learning the node embedding or relation embedding on knowledge graphs, traditional GNN methods adopt message passing mechanisms to aggregate neighborhood information. Despite GNN's powerful representation learning ability, it fails to encode some important structural information, *e.g.* node degree, common neighborhoods of several nodes, and so on. However, this information is important for knowledge graph completion. For example, if we want to calculate the embedding for `Chris Paul`, `Lebron James` should get much more attention than other nodes, because both `Chris Paul` and `Lebron James` play in NBA. And they have many common friends (common neighborhoods) such as `Dwyane Wade`, `Russell Westbrook`, and so on. This means `Chris Paul` and `Lebron James` have high similarity. However, `Giannis Antetokounmpo` should get less attention, because the number of common neighbors between them are small. After precisely getting the node embedding, it will help us to more accurately get the relationship between two nodes.

Inspired by this observation, we propose structure embedding to help the context embedding encode more structure information. Note that context embedding is used to encode the neighborhood information of a given node, while structure embedding is used to indicate whether there is an edge between two nodes according to some structure information, e.g., shortest path, degree, and common neighbors.

Our proposed method takes into additional consideration on the structure by the structure embedding. According to [28], both the heuristic method and graph neural network based method could represent the graph structure. In particular, we first get the structure embedding of the knowledge graph based on the adjacency matrix and then we use the structure embedding to give each neighbor different attention. When aggregating the neighbors' information, we put more attention on the neighbors which have similar structure as the node and less attention on the neighbors which have low structure similarity with the node. Next, we present to use two methods to deal with this problem. The first one is the heuristic method and the second one is GNN-based method. The heuristic method use less computational time and the GNN-based method could more accurately gather the similarity between nodes.

Heuristic Method.: Heuristic methods are capable of providing some structural information for link prediction, such as shortest path, degree, and common neighbors. Although most GNN methods can get superior performance on many datasets, some heuristic methods also show strong empirical performance on the link prediction task. Especially, according to [31], overlap-based methods are usually efficient in the link prediction task. Typical neighborhood overlap methods are Common Neighbors, Resource Allocation, and Adamic Adar. We use different scores by these three methods to get the aggregation weight α_{ij}.

The common neighbors method measures the scores between u, v by counting the number of common neighbors between node u and v and the aggregation weight is

$$\alpha_{u,v} = |\mathcal{N}(u) \cap \mathcal{N}(v)| = \sum_{k \in \mathcal{N}(u) \cap \mathcal{N}(v)} 1 \tag{10}$$

The common neighbors method considers that all of the neighbors have the same weight. However, we should give different neighbors different weights since every neighbor should have different importance in the graph. Resource Allocation and Adamic-Adar fixed this problem. Resource allocation measures the scores between u, v by counting the inverse of the degree.

$$\alpha_{u,v} = \sum_{k \in \mathcal{N}(u) \cap \mathcal{N}(v)} \frac{1}{d_k} \tag{11}$$

where d_k is the degree of node k.

And Adamic-Adar decreases the penalty for higher degree by using the logarithm of the degree of the common neighbors between u, v

$$\alpha_{u,v} = \sum_{k \in \mathcal{N}(u) \cap \mathcal{N}(v)} \frac{1}{\log d_k} \tag{12}$$

After we getting the aggregation weight, we could use this aggregation weight to update the node embedding.

GNN-based Method. : Besides heuristic methods, we can also extract the structure embedding by using the graph neural network. In this paper, we propose a new structure to extract the structure embedding of the whole graph. The structure of GNN-based method is shown in Figure 4. First, we propose the structure feature generator to learn the structural features of each node using the adjacency matrix $A \in \mathbb{R}^{N \times N}$ of the graph. In the structure feature generator, we use four layers of graph convolutional network and concatenate them together. We assume that the output of i^{th} GCN layer is H_i and H_0 is an identity matrix. The output H_i is

$$H_i = \text{GCN}(H_{i-1}) \tag{13}$$

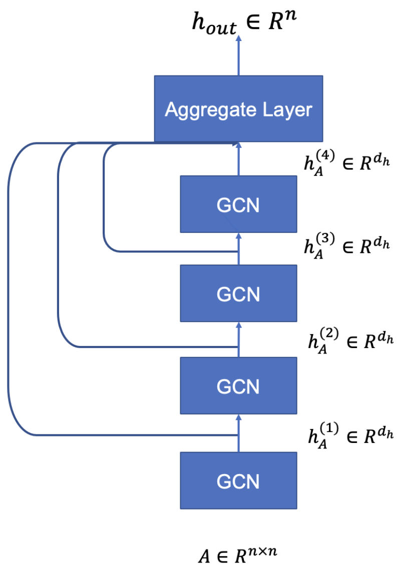

Fig. 4. The structure of the GNN-based method for structure embedding.

Then we aggregate all the H_i together to get the output

$$L = \sum_{i=1}^{4} H_i \tag{14}$$

where L is the structure embedding of the knowledge graph.

After we get the structure embedding, we use the structure embedding to compute the different attention of the neighbors. If we just use the attention on the relationship, they will overlook the relationship while ignoring the structure information. To overcome this issue, the attention is based on the structure embedding when doing the aggregation. Let L_i be the structure embedding of the node i. The structural attention parameter between the header h and its neighbor i is

$$\alpha_{hi} = \frac{\exp\left(L_h^T L_i\right)}{\sum_{k \in \mathcal{N}_h} \exp(L_h^T L_i)} \tag{15}$$

Given α_{ij}, we can get different attention with different structural information. Then we can aggregate the node's neighbor by using α. The output of the location aggregator from the header h could be defined as follows:

$$\widehat{n}_h^{i+1} = W^i n_h^i + \sum_{j \in \mathcal{N}_h} \alpha_{hj} W^i n_j^i \tag{16}$$

where W^i is a $\mathbb{R}^{d \times d}$ parameter matrix for feature mapping.

After we get $n_{h,c}^i$ the updated context embedding and $n_{h,l}^i$ the updated structure embedding, we propose a method that combine the context embedding and the structure embedding. We define two learnable parameters k_c, k_s to determine the significance of the context embedding and the structure embedding, where $k_c \in \mathbb{R}, k_s \in \mathbb{R}$. We will normalize these two parameters to eliminate the possible gradient vanishing:

$$\hat{k}_c = \frac{\exp\left(k_c\right)}{\exp\left(k_c\right) + \exp\left(k_l\right)}, \hat{k}_l = \frac{\exp\left(k_l\right)}{\exp\left(k_c\right) + \exp\left(k_l\right)} \tag{17}$$

Then, the output of combining the context embedding and the structure embedding is

$$n_h^{i+1} = \hat{k}_c \widetilde{n}_h^i + \hat{k}_l \widehat{n}_h \tag{18}$$

And we will use the same method to update the embedding of the tail entity t. After we update the embedding of the head h and the tail t, we combine these two embeddings to get the relation embedding. The relation embedding between these two entities should be like

$$s_{(h,t)} = \sigma\left[W_r[n_h^{i+1}, n_t^{i+1}] + b_r\right] \tag{19}$$

where $\widetilde{s}_{(h,t)}$ is the relation embedding extract from two node embedding: context embedding and structural embedding. And from Eq. 19, we can get that the relation embedding $\widetilde{s}_{(h,t)}$ is not related to the relationship between them since during the training part we should ignore the relationship between head and tail.

This helps us determine the direct relationship embedding between the head and tail by using the information of their neighbors. Thus, this will collect the local information of the head entity and tail entity.

D. Path Embedding

Path embedding is another important way to deal with the relation prediction problem that should get which path would be important when dealing with this completion. For example, in Figure 3 the relationship between `Lebron James` and `Russell Westbrook` is `Teammate` and the relationship between `Lebron James` and `Lakers` is `play for`. Then the relationship between `Russell Westbrook` and `Lakers` is `play for`. We can use the first-order logic to explain this problem:

$$r(h, x_1) \wedge r(x_1, x_2) \wedge \cdots \wedge r(x_n, t) \Rightarrow r(h, t) \tag{20}$$

where $\bigwedge r$ is the conjunction of relations in a path and $r(h,t)$ is the predicted relation. Therefore, path embedding can also learn the logical rules of the knowledge graphs. We can predict the relationship between the head entity h and the tail entity t by using the path between them.

As shown in the previous paragraph, the path can be treated as a sequence and the Transformer is an efficient way to deal with the sequence. Now, we begin formally defining the sequence and show how we use the Transformer to update the relation embedding. Given a head entity h and a tail entity t in knowledge graph $\mathcal{G}$, we can generate several paths between them, and each path can be treated as a sequence. We take a length-L path as an example. It looks like: $h, v_1, v_2, \cdots, v_{L-1}, t$. Then, we transfer this path into a sequence. We add the edge embedding between nodes to form a sequence which could be $h, (h, v_1), v_1, (v_1, v_2), v_2, \cdots, v_{L-1}, (v_{L-1}, t), t$. Let us take the embedding sequence of the ith iteration as $S^i = \{s_1^i, s_2^i, \cdots, s_{2L+1}^i\}$, where $s_{2j-1} = n_{v_j}^i, s_{2j} = e_{(v_{j-1}, v_j)}^i$. And we also combine the context embedding into the path embedding. We add a sequence in the form of $n_h^i \rightarrow \widetilde{s}_{(h,t)} \rightarrow n_t^i$ which is a path of length 1. We update the context embedding in this part and it encodes the topological information of the knowledge graph.

To make use of the order of the sequence, we abstract some information from the position of each entity and edge in the sequence. We add the positional embedding into the node and edge embedding, which could be treated as the input embedding of the training method. Therefore, the positional embedding has the same dimension as the node and edge embedding. Formally, in the sequence $S^i = \{s_1^i, s_2^i, \cdots, s_{2L+1}^i\}$, we denote the position of s_j as j. Then the position embedding of the jth element s_j in the sequence is

$$\begin{aligned} e_j &= \text{Position-Embed}(j) & (21)\\ &= \left[\sin\left(\frac{j}{10000^{\frac{2k}{d}}}\right), \cos\left(\frac{j}{10000^{\frac{2k}{d}}}\right)\right]_{k=0}^{\lfloor \frac{d}{2} \rfloor} & (22)\end{aligned}$$

where k is the dimension.

After adding the positional embedding, we take the summation of entity/edge embedding and the positional embedding as the input of the Transformer, where $\hat{s}_j^i = s_j^i + p_j$. In the ith iteration, we use the Transformer to update s_j^i which can be formally defined as

$$s_j^{i+1} = \text{Transformer}(\hat{s}_j^i) = \text{Softmax}\left(\frac{Q_p K_p^T}{\sqrt{d}}\right) V \tag{23}$$

where

$$Q_p = \hat{s}_j^i \cdot W_{Q_p},\ K_p = \hat{s}_j^i \cdot W_{K_p},\ V_p = \hat{s}_j^i \cdot W_{V_p} \tag{24}$$

Since we have already put the updated relation embedding $\widetilde{s}_{(h,t)}$ into the Transformer as a sequence. Then by training the Transformer, we can get the relation embedding $\widehat{s}_{(h,t)}$ updated by the Transformer.

After getting the updated relation embedding $\widehat{s}_{(h,t)}$, this embedding not only contains the local information of the head and tail since it aggregates the information of their neighbors but also includes the global information of the head and tail in the whole graph since it has done the Transformer on several sequences and could view itself in the whole graph aspect.

E. Objective Function

We treat the knowledge graph completion problem as a multiclass classification problem. We determine the relationship between the head entity and the tail entity as the highest probability of all the relation. Therefore, we minimize the cross-entropy loss over the training triples. The objective function could be like

$$\min \mathcal{L} = \sum_{(h,t,r)\in\mathcal{D}} J(\text{Softmax}(\widehat{ce}(h,t)), r) \tag{25}$$

where $\mathcal{D}$ is the training set and J is the cross-entropy loss.

IV. Experiments

A. Datasets

In this paper, we use three knowledge graph datasets to conduct the experiments: **FB15K**, **FB15K-237**, **NELL995**. The statistics of the three datasets are summarized in Table II. The basic description of the datasets is as follows:

- **FB15K** [12]: **FB15K** is from Freebase which includes facts in the real world, such as sports, films, and actors. Enitities in this dataset connect with each other in many to many relations.
- **FB15K-237** [13]: **FB15K-237** is the subset of the dataset **FB15K** by removing the inverse link in the **FB15K**.
- **NELL995** [14]: **NELL995** is the dataset which is generated from the 995^{th} iteration of the NELL system which is suitable for the multi-hop reasoning.

	#nodes	#relations	$E[d]$	$Var[d]$
FB15K	14,951	1,345	64.6	32,441.8
FB15K-237	14,541	237	37.4	12,336.0
NELL995	63,917	198	4.3	750.6

TABLE II
STATISTICS OF ALL DATASETS. $E[d]$ AND $Var[d]$ ARE THE MEAN AND VARIANCE OF THE NODE DEGREE.

B. Baselines Methods

To evaluate the performance of our method ABM, we compare it with several existing methods: **TransE** [1], **RotatE** [15], **ComplEx** [8], **DistMult** [6], **PathCon** [16]. The first two methods are based on the translation model. The next two methods are based on the semantic matching model. The last method is based on the graph neural network.

C. Evaluation Metrics

To evaluate the performance of our method, we use the standard MRR, hit@1, and hit@3 as evaluation metrics for the knowledge graph completion. MRR takes the reciprocal of the average score of the position of the true relationship between the head and the tail in the ranked list. Hit@n is the proportion of the true relationship in the top@n ranked lists.

D. Main Results

We first compare the performance of our method with the baseline methods. Then, we compare the alternative aggregation method for each embedding and determine which one is the most effective.

1) Effectiveness Comparison: The results of the evaluation metrics on the three datasets are reported in Table III. We can see that ABM outperforms all baselines in all three datasets. In this comparison, we only use the multihead-attention aggregation for the context embedding and the GNN-based method for the structure embedding. We separately study the effect of context embedding, structure embedding, and path embedding. Only context embedding, structure embedding and path embedding cannot beat the previous method while we use these three methods together.

Based on these results, we conclude that our proposed method ABM performs best with the least memory cost.

Datasets	Methods	MRR	hit@1	hit@3
FB15K	TransE	0.941	0.921	0.982
	RotatE	0.950	0.935	0.986
	ComplEx	0.879	0.824	0.952
	DistMult	0.661	0.439	0.868
	PathCon	0.954	0.942	0.995
	Context	0.939	0.918	0.994
	Location	0.935	0.915	0.993
	Path	0.924	0.909	0.991
	ABM	**0.960**	**0.953**	**0.995**
FB15K-237	TransE	0.932	0.924	0.984
	RotatE	0.946	0.929	0.980
	ComplEx	0.898	0.857	0.952
	DistMult	0.849	0.784	0.935
	PathCon	0.947	0.935	0.995
	Context	0.937	0.915	0.993
	Location	0.931	0.914	0.994
	Path	0.928	0.911	0.992
	ABM	**0.961**	**0.942**	**0.995**
NELL995	TransE	0.821	0.761	0.889
	RotatE	0.729	0.691	0.756
	ComplEx	0.703	0.625	0.765
	DistMult	0.634	0.524	0.720
	PathCon	0.862	0.795	0.941
	Context	0.815	0.741	0.875
	Location	0.835	0.766	0.899
	Path	0.801	0.731	0.864
	ABM	**0.887**	**0.819**	**0.942**

TABLE III
COMPARISON WITH THE BASELINES ON EVALUATION METRICS. THE UNDERLINE INDICATES THE SECOND BEST PERFORMANCE AND BOLD INDICATES THE BEST PERFORMANCE.

E. Model Variant

In this section, we try to use different variants to test the efficiency of the ABM when we modify some of the variants in the ABM. We alter several variants in three embedding methods and find which variant of our approach should achieve the best performance.

1) Context Aggregator: We study how different implementations of context aggregators affect the model preference. In Section III-B, we propose two other aggregation methods: GAT-based aggregation and multihead-based aggregator. In this section, we compare these two methods with aggregation without relation embedding (the mean aggregation). The result of different aggregation methods are shown in Fig. 5. The result shows that the mean aggregation performs worst on these three datasets. The GAT-based aggregation method performs worse than multihead-based aggregator. However, it is also interesting to notice that the GAT-based aggregation does not have many gaps between the multihead-based aggregator. The result shows that the relationship between two nodes must be an important factor when doing the aggregation.

2) Structure Aggregator: In this section, we try to discover whether the different types of structure representation should have different effects on prediction. As in Section III-C, we

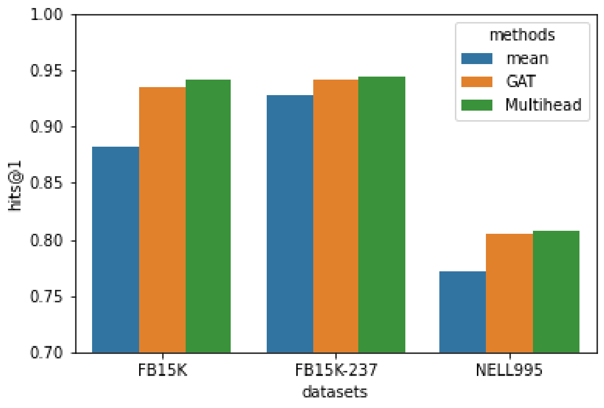

Fig. 5. The result of context embedding with different aggregator.

use four methods: Common neighbor, Resource Allocation, Adamic-Adar, and GNN-based structure embedding on three datasets. The result is shown in Fig 6. We can see that the Common neighbor is the worst method when doing the structure embedding and the GNN-based structure embedding performs best. The resource allocation and Adamic-Adar perform similarly. The result shows that we cannot treat every neighbor the same. The structure information between neighbors must be important when doing the aggregation.

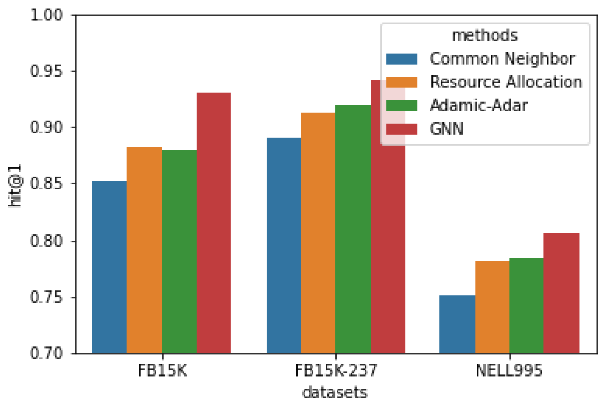

Fig. 6. The result of structure embedding with different aggregator.

Also, for the GNN-based method, we investigate the sensitivity of structure embedding by tuning the different number of GNN layers on the different dataset. In GCN, we will use several fully-connected layers to update the structure embedding. If we use k layers in this step, GCN should help us to aggregate the k-hop neighbors' information. Then we want to figure out how many layers we should use in GCN to get ABM to perform best. The result is shown in Fig 7. We can see that when the number of layers is small (e.g., smaller than 4), increasing the number of layers can significantly improve prediction accuracy. When the number of layers gets to 4, the result converges.

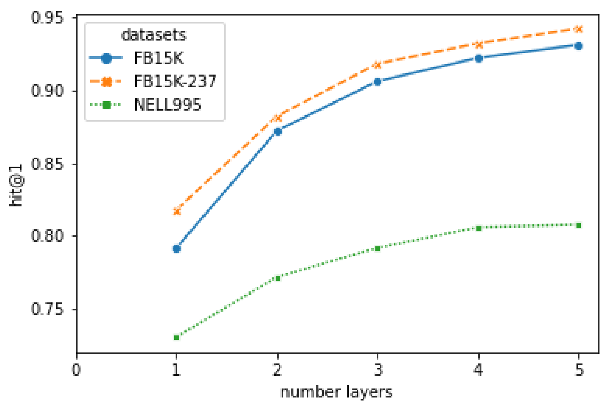

Fig. 7. The result of GNN-based structure embedding with different number of layers.

3) Path Embedding: During the path embedding, we variate the path length we collect to observe the prediction performance. The result is shown in Fig 8. We can see that the increase in the path length can significantly improve the outcome. However, the marginal benefit will appear with the increase of path length when the length of the path gets larger.

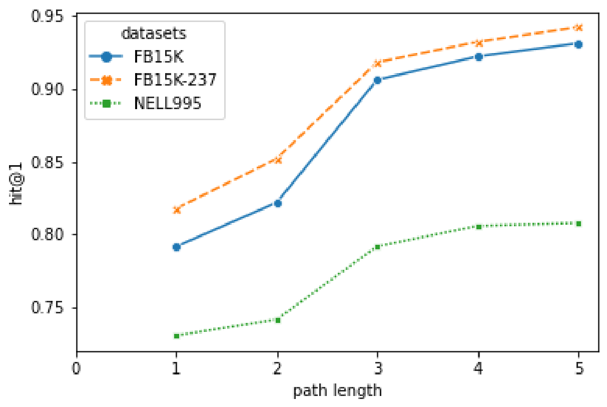

Fig. 8. The result of path embedding with different length of path.

V. Related work

A. Knowledge Graph Completion

Knowledge graph completion can provide information for some industry work, such as recommender systems and semantic analysis. Most existing methods dealing with knowledge graph completion should depend on the embedding. These methods assign each node with a unique node embedding and each edge with a unique edge embedding in the continuous embedding space. Using the existing datasets to train and get updated embedding. One of the mainstream methods is the translation model, they use the node embedding to represent the node and the relationship between them as the translation between nodes. After translating the head entities, they make the translated head entities close to the tail entities. TransE [1] used Euclidean distance as the distance metric to score the distance between translated head entities and tail entities. TransH [3] embeds knowledge graph into the hyperplane of a specific relationship to measure the distance. TransR [4] represents entities and relationships in separate entity and relationship spaces. There also exist some other translation-based methods [2], [15], [17]–[19] which achieve the state-of-art result to handle the knowledge graph completion problem. The other mainstream method should be the semantic-matching-based method. The semantic matching model uses semantic similarity to score the relationship between head entities and tail entities. RESCAL [5] treats each entity as a vector to capture its implied semantics and uses the relationship matrix to model the interactions between latent factors. QuatE [20] uses two rotating planes to model the relations to a hyper-complex space. HolE [7] employs cyclic correlation to represent the composition of the graph. However, neither of these methods captures the structure information of the graph which should be important to the graph.

B. Link Prediction

Link prediction is a common topic in graph learning. Both the heuristic method and the GNN method should be feasible for this problem. The heuristic method basically measures the score between the head entity and the tail entity by using the structural information *e.g.*, overlapped neighbors, shortest path, and degrees of the node. Overlapped neighbors [21] construct the graph structural information by using one-hop neighbors to compute the score of the graph. Adamic-Adar [22], resource allocation [23], PageRank [24] are proposed by using higher-order heuristic methods. However, all of these methods should design the score function of the structure manually. The embedding-based methods do not have the same problem since they compute the similarity scores by using the connection between nodes. Deepwalk [25] and node2vec [26] learn node embedding by using random walk and applying the Skip-Gram techniques on it. Also, link prediction should be based on the adjacency matrix which includes the connectivity information of the graph. VGAE [27] uses the auto-encoder to reconstruct the graph and learn the node embedding by using the existing structure of GNNs. SEAL [10] use the classification of enclosing subgraphs to predict the relationship between nodes.

VI. Conclusion

In this paper, we propose the multihead message passing neural network for the knowledge graph completion. The proposed ABM simultaneously gathers relationship information by context embedding, structure similarity information by structure embedding, and the topological information by path embedding. Our proposed method consistently outperforms baseline methods in several datasets.

Acknowledgement

This work is supported by NSF (1947135, and 2134079), the NSF Program on Fairness in AI in collaboration with Amazon (1939725), DARPA (HR001121C0165), NIFA (2020-67021-32799), and ARO (W911NF2110088). The content of the information in this document does not necessarily reflect the position or the policy of the Government or Amazon, and no official endorsement should be inferred. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation here on.

References

[1] Bordes, Antoine, et al. "Translating embeddings for modeling multi-relational data." Advances in neural information processing systems 26 (2013).

[2] Ji, Guoliang, 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), pp. 687-696. 2015.

[3] Wang, Zhen, Jianwen Zhang, Jianlin Feng, and Zheng Chen. "Knowledge graph embedding by translating on hyperplanes." In Proceedings of the AAAI Conference on Artificial Intelligence, vol. 28, no. 1. 2014.

[4] Lin, Yankai, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. "Learning entity and relation embeddings for knowledge graph completion." In Twenty-ninth AAAI conference on artificial intelligence. 2015.

[5] Nickel, Maximilian, Volker Tresp, and Hans-Peter Kriegel. "A three-way model for collective learning on multi-relational data." Icml. 2011.

[6] Yang, Bishan, 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).

[7] Nickel, Maximilian, Lorenzo Rosasco, and Tomaso Poggio. "Holographic embeddings of knowledge graphs." In Proceedings of the AAAI Conference on Artificial Intelligence, vol. 30, no. 1. 2016.

[8] Balland, Pierre-Alexandre, and David Rigby. "The geography of complex knowledge." Economic Geography 93, no. 1 (2017): 1-23.

[9] Shang, Chao, 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, vol. 33, no. 01, pp. 3060-3067. 2019.

[10] Zhang, Muhan, and Yixin Chen. "Link prediction based on graph neural networks." Advances in neural information processing systems 31 (2018).

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

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

[13] Toutanova, Kristina, 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, pp. 57-66. 2015.

[14] Xiong, Wenhan, Thien Hoang, and William Yang Wang. "Deeppath: A reinforcement learning method for knowledge graph reasoning." arXiv preprint arXiv:1707.06690 (2017).

[15] Sun, Zhiqing, 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).

[16] Wang, Hongwei, Hongyu Ren, and Jure Leskovec. "Relational message passing for knowledge graph completion." In Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining, pp. 1697-1707. 2021.

[17] Ji, Guoliang, Kang Liu, Shizhu He, and Jun Zhao. "Knowledge graph completion with adaptive sparse transfer matrix." In Thirtieth AAAI conference on artificial intelligence. 2016.

[18] Xiao, Han, Minlie Huang, Yu Hao, and Xiaoyan Zhu. "TransA: An adaptive approach for knowledge graph embedding." arXiv preprint arXiv:1509.05490 (2015).

[19] Xiao, Han, Minlie Huang, and Xiaoyan Zhu. “TransG: A Generative Model for Knowledge Graph Embedding.” In Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pp. 2316-2325. 2016.

[20] Zhang, Shuai, Yi Tay, Lina Yao, and Qi Liu. “Quaternion knowledge graph embeddings.” Advances in neural information processing systems 32 (2019).

[21] Barabási, Albert-László, and Réka Albert. “Emergence of scaling in random networks.” science 286, no. 5439 (1999): 509-512.

[22] Adamic, Lada A., and Eytan Adar. “Friends and neighbors on the web.” Social networks 25, no. 3 (2003): 211-230.

[23] Zhou, Tao, Linyuan Lü, and Yi-Cheng Zhang. “Predicting missing links via local information.” The European Physical Journal B 71, no. 4 (2009): 623-630.

[24] Page, Lawrence, Sergey Brin, Rajeev Motwani, and Terry Winograd. “The PageRank citation ranking: Bringing order to the web.” Stanford InfoLab, 1999.

[25] Perozzi, Bryan, Rami Al-Rfou, and Steven Skiena. “Deepwalk: Online learning of social representations.” In Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 701-710. 2014.

[26] Grover, Aditya, and Jure Leskovec. “node2vec: Scalable feature learning for networks.” In Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 855-864. 2016.

[27] Kipf, Thomas N., and Max Welling. “Variational graph auto-encoders.” arXiv preprint arXiv:1611.07308 (2016).

[28] Yun, Seongjun, Seoyoon Kim, Junhyun Lee, Jaewoo Kang, and Hyunwoo J. Kim. “Neo-GNNs: Neighborhood Overlap-aware Graph Neural Networks for Link Prediction.” Advances in Neural Information Processing Systems 34 (2021).

[29] He, Tiantian, Yew Soon Ong, and Lu Bai. “Learning conjoint attentions for graph neural nets.” Advances in Neural Information Processing Systems 34 (2021): 2641-2653.

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

[31] Wang, Hongwei, Fuzheng Zhang, Jialin Wang, Miao Zhao, Wenjie Li, Xing Xie, and Minyi Guo. “Ripplenet: Propagating user preferences on the knowledge graph for recommender systems.” In Proceedings of the 27th ACM international conference on information and knowledge management, pp. 417-426. 2018.

[32] Xu, Lin, Qixian Zhou, Ke Gong, Xiaodan Liang, Jianheng Tang, and Liang Lin. “End-to-end knowledge-routed relational dialogue system for automatic diagnosis.” In Proceedings of the AAAI Conference on Artificial Intelligence, vol. 33, no. 01, pp. 7346-7353. 2019.

[33] Huang, Xiao, Jingyuan Zhang, Dingcheng Li, and Ping Li. “Knowledge graph embedding based question answering.” In Proceedings of the twelfth ACM international conference on web search and data mining, pp. 105-113. 2019.