Learning Graph Attention-Aware Knowledge Graph Embedding

Chen Li[a,b], Xutan Peng[c], Yuhang Niu[a,b], Shanghang Zhang[d], Hao Peng[a,e], Chuan Zhou[f], Lihong Wang[g], Jianxin Li[a,b]

[a]*Beijing Advanced Innovation Center for Big Data and Brain Computing, Beihang University, Beijing, China*
[b]*State Key Laboratory of Software Development Environment Lab, Beihang University, Beijing, China*
[c]*Department of Computer Science, The University of Sheffield, Sheffield, UK*
[d]*Department of Electrical and Computer Engineering, Carnegie Mellon University, Pittsburgh, CP, USA*
[e]*Academy of Mathematics and Systems Science, Chinese Academy of Sciences, Beijing, China*
[f]*School of Cyber Science and Technology, Beihang University, Beijing, China*
[g]*National Computer Network Emergency Response Technical Team/Coordination Center of China, CNCERTCC, Beijing, China*

Abstract

The knowledge graph, which utilizes graph structure to represent multi-relational data, has been widely used in the reasoning and prediction tasks, attracting considerable research efforts recently. However, most existing works still concentrate on learning knowledge graph embeddings straightforwardly and intuitively without subtly considering the context of knowledge. Specifically, recent models deal with each single triple independently or consider contexts indiscriminately, which is one-sided as each knowledge unit (i.e., triple) can be derived from its partial surrounding triples. In this paper, we propose a graph-attention-based model to encode entities, which formulates a knowledge graph as an irregular graph and explores a number of concrete and interpretable knowledge compositions by integrating the graph-structured information via multiple independent channels. To measure the correlation between entities from different angles (i.e., entity pair, relation, and structure), we respectively develop three attention metrics. By making use of our enhanced entity embeddings, we further introduce several improved factorization functions for updating relation embeddings and evaluating candidate triples. We conduct extensive experiments on downstream tasks including entity classification, entity typing, and link prediction to validate our methods. Empirical results validate the importance of our introduced attention metrics and demonstrate that our proposed method can improve the performance of factorization models on large-scale knowledge graphs.

Keywords: Knowledge Graph Embedding, Graph Attention Mechanism, Entity Typing, Link Prediction

1. Introduction

The knowledge graph can provide graph-structured representation for real-world knowledge, such as Freebase [4], DBPedia [1], and YAGO [52]. The multi-relational data stored in knowledge graph has been applied to numerous downstream machine learning tasks where optimization is subject to predefined rules and constraints, e.g., Information Retrieval [11], Question Answering [15, 69], and Emotion Classification [17, 37]. To facilitate the processing of these applications, some works attempt to model and standardize knowledge data in a more computable fashion. Among them, the knowledge graph embedding technique becomes a heated research topic which has witnessed rapid growth. Knowledge graph embedding aims to map entities to a low-dimensional continuous space while encodes their relations, thus inference on a knowledge graph can be simply solved with vector arithmetic.

Typically, a knowledge unit is stored as a standard triple, i.e., in the form of (*head entity*, `relation`, *tail entity*). While entities generally have several disparate *types* (which can be intuitively understood as multiple channels/aspects of an object), relations connected to those entities can also be assigned to the corresponding types [45, 34]. The relations with the same type are generally based on identical essential facts (e.g., all orange relations in Figure 1 are in the type of *US president*). Hence, they can be regarded as the basis for inferring the missing relations, while the relations with other types become useless. Besides, as illustrated in Figure 1, only partial knowledge in the context actually contributes to one specific inference process. For example, when modeling one of the orange relations in Figure 1, other orange relations and their connected entities are more valuable than the blue ones. However, most of the existing methods for embedding multi-relational data are either built upon an assumption with excessive independence (i.e., only considering the relation between the given entity pair) or integrating all context information indiscriminately, severely reducing the capacity of exploiting environmental information [14, 50].

Among these existing works, the translation models which are represented by TransE [5] and its variations [63, 30, 19, 13, 53, 72], treat each triple as a training sample and obtain corresponding knowledge representation by projecting entities and relations with an additional principle ($\mathbf{h}+\mathbf{r} \approx \mathbf{t}$, where $\mathbf{h}, \mathbf{r}, \mathbf{t}$ are the head, relation, and tail embeddings of a triple) under a variety of constraints. On the contrary, other lines of knowledge graph embedding studies [14, 48, 50] focus on constructing a graph whose vertices are entities and edges are relations, then combining the information of both graph structure and vertex to encode knowledge. One obvious limitation of these strate-

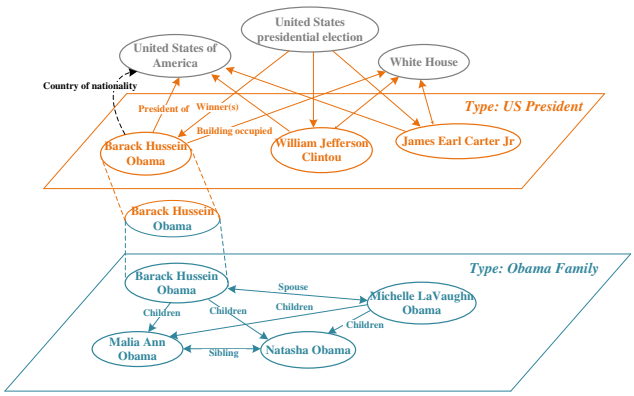

Figure 1: The entity *Barack Hussein Obama* has two types (*US President* (orange) and *Obama Family* (blue)). All of the orange entities have the same *type: US President*, and they also have the same relations with each gray entity (which is closely related to *type: US President*). This correlation can be understood as the intrinsic manifestation of such relations and corresponding type, while this phenomenon also appears in the entities with *type: Obama Family*. Obviously, such correlation (similar to collaborative filtering) will help screen corresponding information (i.e., different channels) for modeling specific relation.

gies is that they are not appropriate to process complex multi-relational data, because they either directly make inferential calculation without considering neighbor triples of training samples [5, 63, 30] or combining all environmental factors without any distinction [48, 14]. In particular, the performance of existing methods will see negative gains when there are abundant types of candidate entities or multiple relations between the single given entity pair (e.g., the relations between `Obama` and `United State` in Figure 1) [63, 30].

In this work, we introduce the graph attention mechanism to capture richer and more fine-grained information for knowledge graph embeddings. The backbone of our model is to acquire the hidden representation of the entities in various channels by selectively merging the information from their neighbor entities and utilizing a factorization function to determine the probability of candidate triples. This novel graph-based multi-head attention mechanism brings about several advantages: (1) by handling input information in multi-channels, the interpretablity of existing knowledge graph embeddings is substantially improved; (2) through iterating or stacking graph attention layers, multi-hop information acquisition can be easily achieved; (3) thanks to the parallelism and mobility of our algorithm, even generalization on arbitrary large-scale knowledge graph datasets is guaranteed. Moreover, we also propose three metrics to compute the attention coefficients based on diverse sources (i.e., entity pair, relation, and structure) and implementations to acquire high-quality entity embeddings. By utilizing specific entity embeddings injection strategies, we improve the factorization functions to train relation embeddings and score candidate triples. We conduct experiments to assess the performance of different attention in a wide range of tasks, including entity classification, entity typing, and link prediction. We also explain the working state of the graph attention in visual form. Comprehensive experimental results verify that our method is on par with or outperforms baseline methods and can be applied to large-scale knowledge graph with complex and agnostic structures.

The contribution of this paper is three-fold:

- We develop a novel graph attention mechanism to update entity embedding by adequately considering the structure and entity information.
- We propose three attention-based methods by considering different sources and calculative strategies, and conduct comparative studies on their ability to capture multi-channel information.
- We achieve new state-of-the-art performance on entity classification and entity typing, as well as improve the factorization model's effectiveness in link prediction.

The rest of this paper is organized as follows: Section 2 reviews traditional knowledge graph embedding methods and the graph attention mechanism. Section 3 details the composition of the whole framework, depicts each of the graph attention metrics, and conducts theoretical analysis. Section 4 describes the experiment designs and analyzes comprehensive results on typical downstream tasks. At length, Section 5 concludes this paper.

2. Related Work

We review works in the two fields that the subject matter of the current paper falls in, namely, knowledge graph embedding and graph attention mechanism.

2.1. Knowledge Graph Embedding

The knowledge graph embedding has attracted wide attention, leading to numerous related studies been proposed. According to the methods used to model triples, they can be divided into the following categories.

2.1.1. Translation

Bordes et al. [5] proposes TransE to learn the representation of multi-relational graph data based on their previous work [6], which treats the triple as a translation process in the semantic vector space. Later, to solve the problem of encoding complex relations, some works improve TransE by introducing assistant projection plane [63], independent relational space [30], or interpretation vectors [19] (which has achieved state-of-the-art performance in link prediction), respectively. Besides, by utilizing extra information (e.g., multi-hop paths [29], entity descriptions [64], and entity images [65]) as training constraints, some methods are no longer limited to regard the given triples as the only information source. Recent studies further consider circular correlation [38], torus [13] or complex space [56] to achieve more explainable knowledge graph embedding learning by exploiting more interpretable structural information. Similarly, Sun et al. [53] treats the relation as a rotation operation

and defines the triple with a new spatial relationship. However, the Trans-family methods only consider the single relations or multi-hop paths between given entity pair as training samples, while they do not effectively integrate the information from other triples surrounding the sample triple, which leads to the neglect of rich auxiliary information.

2.1.2. Multiplication

Tensor decomposition is a traditional solution to link prediction. To learn knowledge graph embedding, the so-called multiplication models are also applied on modeling triples in a knowledge graph. DistMult [67] defines the relation as a special matrix that associates the probability of a triple with the corresponding multiplication. ANALOGY [31] combines several traditional models by constructing an objective function in a differentiable fashion. SimplE [21] enhances the interpretablity of embeddings by weighting the background knowledge. TuckER [2] proposes a linear model based on Tucker decomposition of binary tensor representations of knowledge graph triples. ProcrustEs [42] utilises the closed-form solution to the Orthogonal Procrustes Problem and embeds knowledge graphs in an efficient fashion.

2.1.3. Neural Networks

Currently, a plethora of works have begun to focus on exploiting neural networks to learn knowledge graph embeddings. R-GCN [48] accumulates evidence in multiple inference steps through the message propagation network, which improves the performance of factorization models. To learn more contextual information, TranAt [43] introduces a specific graph attention mechanism for modeling triples. SACN [49] is composed of a weighted graph convolutional network encoder and a convolutional network (Conv-TransE) decoder to achieve end-to-end structure perception. KBGAT [35] proposes a complete set of inference models by using triples as the object of graph attention. ActiveLink [36] uses the underlying structure of the knowledge graph to expand uncertainty sampling and connect entities for improved sampling efficiency. COMPGCN [58] uses various entity relational combination operations in knowledge graph embedding technology and scales according to the number of relations. DPMPN [66] relies on the input sub-graph and self-expands through the streaming attention mechanism to explicitly model a sequence inference process.

2.1.4. Extra Information

In addition to the distinctions between architectures, a variety of extra information is introduced to constrain the learning of knowledge graph embeddings and enhance the model's ability to capture the hidden information. Among them, the multi-hop path between the entities is used to assist modeling triples, and consequently, a large number of additional positive samples are obtained [29, 28, 25]. Meanwhile, textual information [16], temporal information [23], and neighborhood information [61] are used to strengthen the semantic signals of triples, which makes the knowledge graph embeddings more robust. In addition, several strategies based on manual construction are also adopted to optimize the learning process, e.g., regularization [47], soft rules [44], and negative sampling [8].

2.2. Graph Neural Network

In the course of the growth of graph based methods for machine learning tasks, by combining the graph-structure with the vertex information, a large amount of related works learn to obtain the representations of each vertex as well as the entire graph [55].

2.2.1. Graph Convolutional Network

The first successful attempt in this direction is the Graph Convolutional Network (GCN) [26], which achieves representation learning based on a recurrent framework. Later, by utilizing spectral methods [7] or spatial-based methods [39], GCN introduces the convolution concept to irregular graphs. However, spectral-based methods are limited by scalability and universality, and cannot be flexibly applied to incremental graphs or directed graphs. The spatial-based model, on the other hand, is able to more reasonably aggregate the graph signals by defining different neighborhood information capture strategies.

2.2.2. Graph Autoencoder

Graph autoencoder has also be utilized for learning graph embeddings, whose purpose is to use the neural network structure to represent nodes as low-dimensional vectors. One popular implementation is to use a multilayer perceptron as an encoder for node embedding, while designs a decoder to reconstruct the node's statistical neighborhood information, such as Positive Pointwise Mutual Information (PPMI) or first-order and second-order approximations [40]. Among them, DNGR [10] and SDNE [62] learn the node embedding from topological structure, while ARGA [41], NetRA [70], and DRNE [57] are proposed to learn node embedding when both the topology information and the node feature are available. One challenge of the graph autoencoder is the sparseness of the adjacency matrix, which makes the number of positive entries of the decoder much smaller than that of negative entries. In order to mitigate this problem, DNGR reconstructs a denser matrix, namely the PPMI matrix, SDNE penalizes the zero entries of the adjacency matrix, GAE reweights the entries in the adjacency matrix, and NetRA linearizes the graph as a sequence.

2.2.3. Graph Generative Network

The goal of graph generation networks is to generate new graphs given a set of observed graphs. Many of these methods are applied to specific fields, and a number of generative methods have been proposed very recently. For instance, MolGAN [9] combines GCN and reinforcement learning goals to generate a graph with required attributes. Its generator tries to pose a pseudo graph and its feature matrix, while the goal of its discriminator is to distinguish between pseudo samples and empirical data. Meanwhile, MolGAN also introduces a reward network in parallel with the discriminator to encourage the generated graph to have certain attributes in line with

an external evaluator. DGMG [27] incorporates spatial-based GCN to obtain hidden representations of existing graphs. Specially, the decision-making process of generating nodes and edges is based on the representation of the entire graph. Apart from MolGAN and DGMG, there also exist methods which achieve graph generation by introducing different structures, e.g., GraphRNN [68] and NetGAN [3].

2.2.4. Graph Attention Network

As a particular instance of MoNet [33], Graph ATtention network (GAT) [60] presents a masked self-attention layer, which selectively adjusts the weights of neighbor vertex information and achieves state-of-the-art results in transductive and inductive learning. Specifically, by setting multiple attention heads, GAT learns multi-channel features which can satisfy varying requirements. This research strand is mainly inspired by the neural translation [59]: its efficient attention design allows the translation model to reach better effect without relying on any recurrent or convolution operations. Currently, ADSF utilizes the attention mechanism to structural information and obtains graph weights by calculating the degree of neighborhood overlap [71]. Beyond that, ELCO refines the graph topology based on the overlapping clustering to learn more interpretable graph embeddings [24]. As illustrated in Figure 1, relations in knowledge graph are clustered by entity type. Therefore, the extraction of entity features needs to be performed on multi-channels, which makes the multi-head attention mechanism a natural choice for modeling knowledge graph.

Prior to us, some authors have attempted to introduce graph attention into knowledge graph embedding learning. Nevertheless, their methods are limited because each triple is treated as a co-occurrence relationship while structural information is absent. Furthermore, their studies lack detailed analysis on the effects of attention modules. To the best of our knowledge, we are the first to analyze the graph attention of various sources in knowledge graph embedding learning.

3. Methodology

In this section, we first introduce various attention mechanisms to refine the learning of entity embeddings for the knowledge graph, then perform probabilistic analysis through learning relation embeddings. We detail the modules used to capture information in arbitrary graph structures and dive deeply to analyze its theoretical and practical benefits and limitations.

3.1. Problem Definition and Notations

A knowledge base is defined as a collection of triples $\mathcal{T} = \{(e_h, rel, e_t) \mid e_h, e_t \in E, rel \in REL\}$, where e_h, e_t respectively denote the head and tail entities, *rel* is the single relation type, E is the entity set and REL is the relation type set[1]. To integrate richer reference information, we collect triples to form a knowledge graph $\mathcal{G} = \{e, r \mid e \in E, r \in R\}$, where e, r are the vertex and the edge in $\mathcal{G}$. Given an entity e, its context $cont(e)$ is a set of other entities relevant to e : $\{e_i|(e, r, e_i) \in \mathcal{T}\}$. The task of knowledge graph embedding aims to represent each entity/vertex and each relation/edge by a vector $\vec{h}' \in \mathbb{R}^{d'}$ with real numbers, where d' is the dimension of $\vec{h}'$.

3.2. Graph Attention in Knowledge Graph

In general, an attention function [59] can be described as a query on a set of key-value pairs, where the query, keys, values, and output are all vectors. The output of attention function is computed as the weighted sum of all matched values, where the weights are determined through considering the matching degrees of query and corresponding keys.

In the graph attention model, based on the assumption commonly used to representation learning [32, 18, 60], i.e., the vertices with similar features have similar neighbors, the attention function is analogically defined and applied to learn the entity embedding in $\mathcal{G}$. Specifically, the input of the graph attention layer is a set of entity features: $h = \{\vec{h}_1, \vec{h}_2, ..., \vec{h}_{n_e}\}, \vec{h}_i \in \mathbb{R}^d$, where $n_e = |E|$, and d is the dimension of features of each entity. The graph attention layer produces a new set of entity embeddings as its outputs, $h' = \{\vec{h}'_1, \vec{h}'_2, ..., \vec{h}'_{n_e}\}, \vec{h}'_i \in \mathbb{R}^{d'}$, where d' is the dimension of $\vec{h}'$ and does not necessarily equal to the original d depending on the task requirements.

Because either the neighbor entities or the relations between entity pairs can be used as the measure of information acquisition, we divide the implementation of graph attention into two categories: *entity attention* and *relation attention*. In particular, to remove the constraints of triples on the relationship between entities, we also introduce *structural attention* to evaluate the correlation of neighbor distribution between vertices.

3.2.1. Entity Attention

As a vertex in graph $\mathcal{G}$, obtaining the entity embedding $\vec{h}'_i$ is the initial step for modeling multi-relational data. Based on the aforementioned assumption, $\vec{h}'_i$ can be learned by receiving partial information from its neighbor entities and itself. The learning scale of each information source is determined by the attention coefficient c_{ij} between entity pair (e_i, e_j). In order to obtain sufficient expressive power to transform the input entity features into high-level embeddings, we perform the calculation of c_{ij} by using *additive attention* as

$$c_{ij} = a(\mathrm{W}\vec{h}_i, \mathrm{W}\vec{h}_j), \tag{1}$$

where $\mathrm{W} \in \mathbb{R}^{d' \times d}$ is a learnable *weight matrix*. Additive attention is a *self-attention* mechanism $a : \mathbb{R}^{d'} \times \mathbb{R}^{d'} \to \mathbb{R}$ between entities, which indicates the degree of importance of e_i to e_j,

$$a(\mathrm{W}\vec{h}_i, \mathrm{W}\vec{h}_j) = \mathrm{ReLU}(\vec{a}^{\top}[\mathrm{W}\vec{h}_i \| \mathrm{W}\vec{h}_j]), \tag{2}$$

where $\vec{a} \in \mathbb{R}^{2d'}$ is the *weight vector*, $\cdot^{\top}$ denotes the transposition operation, $\mathrm{ReLU}(\cdot)$ is a rectified linear function and $\|$ is the concatenation operation. When calculating additive attention, the shared linear transformation W has been used to pre-process the input features of the entities, which is the key to ensure that entity features can be utilized to measure entity correlation [60].

[1] REL contains relations both in canonical direction (e.g. $president\_of$) and in inverse direction (e.g. $president\_of^{-1}$)

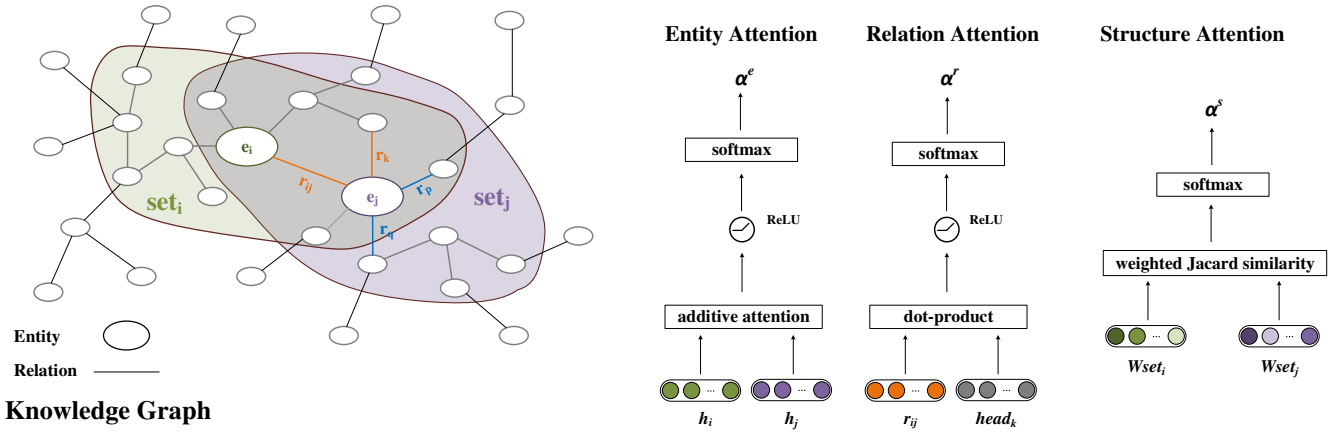

Figure 2: The calculation methods of attention from different sources (entity pair or relation).

Among them, $\vec{a}$ and W are the parameters included in the additive attention, and the multi-head attention contains a matching number of different initialization parameter groups.

We only select the first-order neighbors and itself as the information sources (multi-hop features can be obtained by stacking such graph attention layers). As shown in Figure 2, by using the softmax function, we normalized c_{ij} to make it easier to compare the entity attention coefficients α^e among different entity pairs

$$\alpha^e_{ij} = \text{softmax}(c_{ij}) = \frac{\exp(c_{ij})}{\sum_{l=0}^{n_i} \exp(c_{il})}, \tag{3}$$

where n_i is the number of first-order neighbors of e_i, and $\exp(\cdot)$ is the exponential function based on the natural constant e.

3.2.2. Relation Attention

As another important cohesive component, the relation type reflects the real-world connections between entities in the knowledge graph more clearly and directly. Even if the entity embeddings linked by the relation are updated or the same relation appears in different triples, the relation does not change semantically and always describes the semantics between entities stably (without ambiguity). Hence, it is possible to determine the weights of information captured from the neighbor entities by purely relying on the relation (not considering the entities). Meanwhile, since the relations in knowledge graph have their own types, the entity connected by similar relation types should be considered uniformly. For example, the orange and blue relations in Figure 2, with similar types (i.e., US president and Obama family in Figure 1), can be viewed as two different channels of information and should be dealt with separately.

Specifically, to collect the entity features provided by the neighbor entities connected to the relations with the same type, we put aside the information of entity pairs and calculate the attention coefficient using the relations only (as illustrated in Figure 2). In this way, the effects of entity feature initialization on the model performance and the convergence rate are substantially reduced. The relation information provided by the knowledge graph is also fully utilized. We additionally adopt a multi-head attention framework to achieve multi-channel feature learning similar to the *entity attention*, and the attention objects $head \in \mathbb{R}^d$ can be determined by the correlation of the relations in the knowledge graph, the task requirements, or the random initialization. The attention coefficient can be directly obtained by the *dot-product attention*:

$$c_{ij} = \text{ReLU}(r_{ij} \cdot head), \tag{4}$$

where r_{ij} is the relation representation of the entity pair (e_i, e_j), $head \in \mathbb{R}^d$ is the trainable attention parameter of various attention heads. In particular, since we intorduce the multi-head attention mechanism, there will be K different initialized attention heads learned to update entity representation in parallel. The weights of different neighbor entities can be obtained by normalizing the relation attention coefficients α^r_{ij} as follows:

$$\alpha^r_{ij} = \text{softmax}(c_{ij}) = \frac{\exp(c_{ij})}{\sum_{l=0}^{n_i} \exp(c_{il})}. \tag{5}$$

Among the relations connected with e_i, we add a self-loop relation that retains its own entity features, with the weight set by default. Similar to the final output embeddings of the *entity attention*, the output $\vec{h}'_i$ of *relation attention* also takes the form as shown in the right part of Figure 2, and *head* is trained by using entity classification.

3.2.3. Structure Attention

Triple is the smallest knowledge unit in the knowledge graph, and the aforesaid *entity attention* and *relational attention* are both measures of correlation within the triple. In fact, the neighbors of an entity can also be regarded as an implicit representation of its own information. Thus the correlation metric between two entities should consider the impact of their neighbors. Therefore, we propose to compute the correlation between two entity neighbors based on the *structural fingerprint*. First, we generate a structural fingerprint of each target entity, i.e., to determine the importance based on the structural relationship between the entity in the neighborhood and the target entity, and to adaptively divide its neighborhood according to the target entity. Then, when evaluating the correlation between

two entities, we calculate the results by analyzing the relationship between the structural fingerprints of the two entities.

Intuitively, the importance of entities in the neighborhood will decay as the distance from the target entity increases, but this attenuation will change slightly due to the connectivity/density of the internal structure of the neighborhood. The common Gaussian attenuation and non-parametric attenuation only rely on the distance between entities to achieve weight distribution, essentially ignoring the structural difference. To adjust the weights of neighbor entities according to the local graph structure (connected shape or density), we use the strategy of Random Walk with Restart (RWR) [54] to calculate the weights and to complete the corresponding adaptive neighborhood division. RWR explores the global topology of the network by simulating the iterative movement of particles between adjacent nodes. It quantifies the proximity between nodes in the network and is widely used in information retrieval and other fields.

To obtain the structural fingerprint of entity e_i, we consider its neighbor entity set E_i and the corresponding adjacency matrix A_i. The particle starts from the target entity e_i and randomly walks to its neighbor entity in E_i with a probability proportional to the relation weight. In each step, it also has a certain probability to return to the target entity. The iterative process can be written as

$$w_i^{(t+1)} = c \cdot \widetilde{A}_i w_i^t + (1-c) \cdot v_i, \tag{6}$$

where $\widetilde{A}_i$ is the transition probability matrix by normalizing columns of A_i, $c \in [0, 1]$ is a trade-off parameter between random walk and restart, and v_i is a vector of all zeros except the entry corresponding to the target entity e_i. The converged solution can be written in a closed form as

$$w_i = (I - c \cdot \widetilde{A}_i)^{-1} v_i, \tag{7}$$

where w_i quantifies the proximity of the target entity to all entities in its neighbors. Meanwhile, the entire structural fingerprint intuitively reflects the local structural details of the knowledge graph, effectively distinguishing the importance of entities. c controls the attenuation rate of weights in the neighborhood. When $c = 0$, except for the target entity, w_i is 0; when $c = 1$, w_i is a stable distribution of standard random walks on this graph. In practice, c will be optimized by downstream tasks, and the structural nature of sub-graphs will be mined in light of task requirements.

When analyzing the structural correlation between two entities, we can calculate the correlation between two entity structure fingerprints using the Jaccard similarity,

$$c_{ij} = \frac{\sum_{p \in (E_i \cup E_j)} \min(w_{ip}, w_{jp})}{\sum_{p \in (E_i \cup E_j)} \max(w_{ip}, w_{jp})}, \tag{8}$$

where w_{ip} and w_{jp} are the weights of an entity p in different entity neighbor set E_i and E_j. The structural weights α_{ij}^s of neighbor entities can be obtained by normalizing the following function:

$$\alpha_{ij}^s = \text{softmax}(c_{ij}) = \frac{\exp(c_{ij})}{\sum_{l=0}^{n_i} \exp(c_{il})}. \tag{9}$$

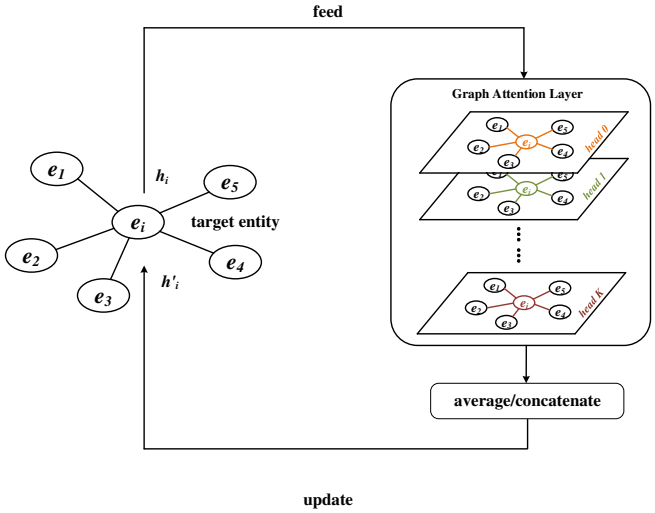

Figure 3: Updating the entity embeddings based on multi-head graph attention..

3.2.4. Learning Entity Embedding

Once the attention coefficients are calculated, the final output feature $\vec{h}_i'$ of the entity e_i can be obtained through a nonlinear combination of the neighbor entities as follows,

$$\vec{h}_i' = \sigma(\sum_{j=0}^{n_i} \alpha_{ij} \mathrm{W} \vec{h}_j), \tag{10}$$

where $\sigma(x) = \frac{1}{1+\exp(-x)}$. In particular, to balance the three different sources of attention, we combine them into a final attention as

$$\alpha_{ij} = \frac{\lambda(\alpha_{ij}^e)\alpha_{ij}^e + \mu(\alpha_{ij}^r)\alpha_{ij}^r + \eta(\alpha_{ij}^s)\alpha_{ij}^s}{\lambda(\alpha_{ij}^e) + \mu(\alpha_{ij}^r) + \eta(\alpha_{ij}^s)}, \tag{11}$$

where λ, μ, and η are manually configured weights. In fact, experiments show that the adjustment of hyper-parameters does not lead to a significant performance boost, so we use the average pooling to obtain the final weight instead:

$$\alpha_{ij} = (\text{normalize}(\alpha_{ij}^e) + \text{normalize}(\alpha_{ij}^r) + \text{normalize}(\alpha_{ij}^s))/3 \tag{12}$$

where $\text{normalize}(\cdot) = \frac{\alpha_{ij} - min(\alpha)}{max(\alpha) - min(\alpha)}$ is the global normalized function.

To achieve multi-channel feature learning, we extend the graph attention mechanism by using a multi-head attention framework. Specifically, as illustrated in the right part of Figure 3, corresponding to K feature learning channels, K independent attention mechanisms are executed to complete feature learning of e_i, and are concatenated together to form the following output embedding

$$\vec{h}_i' = \|_{k=1}^{K} \sigma(\sum_{j=0}^{n_i} \alpha_{ij}^k \mathrm{W}^k \vec{h}_j), \tag{13}$$

where α_{ij}^k is the normalized attention coefficient via the k-th head attention mechanism a^k, and W^k is the corresponding transformation weight matrix.

The multi-head attention framework can be pre-set according to the requirements of downstream tasks, i.e., pre-training the transformation weight matrix W and the weight vector $\vec{a}$ using domain features, or random initialization. Then, we can feed partial features in some specific attention head according to the task requirements, or a nonlinear comprehensive feature representation as

$$\vec{h}'_i = \sigma(\frac{1}{K}\sum_{k=1}^{k}\sum_{j=0}^{n_i}\alpha_{ij}^k \mathrm{W}^k\vec{h}_j). \tag{14}$$

In order to obtain entity embedding that fully considers the structure and node information, we use the cross-entropy loss provided by the entity classification task to train the attention parameters [48, 56].

3.3. Modeling Multi-Relational Data

In the knowledge graph, multi-relational data is the structured representation of real-world knowledge, and its basic unit is a triple $(\vec{h}'_h, \vec{r}, \vec{h}'_t)$, where $\vec{h}'_h$ and $\vec{h}'_t$ are the vectorial representation of head entity and tail entity. The relation is an important element that connects two given entities, and a reasonable representation of relation can greatly help the downstream tasks. In this paper, the choice of relation representations reflects in the form of the scoring function for the candidate triples. Based on the updated entity embedding $\vec{h}'$, we need to select a commonly used function in the literature, which accumulates numerical evidence of knowledge reasoning by training in a real triple set. Then, we can utilize it to evaluate the probability of the candidate knowledge.

Since this task can also be described as predicting a missing element (either an entity or a relation) based on two given elements to form a new triple, we choose DistMult [67], ComplEx [56], and ConvE [12] to implement our method, which are known to perform well on standard link prediction benchmarks.

In DistMult [67], Yang et al. consider the basic bi-linear scoring function, which treats relation as a diagonal matrix $R_{\vec{r}} \in \mathbb{R}^{d'\times d'}$, and then the candidate triple $(\vec{h}'_h, \vec{r}, \vec{h}'_t)$ is scored as

$$f(\vec{h}'_h, \vec{r}, \vec{h}'_t) = \sigma(\vec{h}'^{\top}_h R_{\vec{r}}\vec{h}'_t). \tag{15}$$

In ComplEx [56], Trouillon et al. propose a method based on the representation of complex valued embeddings to better model the asymmetric relations, the specific score function is as following:

$$\phi(r, h_h, h_t; \Theta) = \mathrm{Re}(\sum_{p=1}^{P} w_{ek}h_{hk}h_{tk}), \tag{16}$$

where Θ denotes the parameters of ComplEx, $\mathrm{Re}(\cdot)$ is the real vector component, $w \in \mathbb{C}^P$ is a complex vector, and P is the dimension of the vector.

In ConvE [12], Minervini et al. reshape the input embeddings of the entity pair into a matrix, and extract features for it using the convolution kernel. The score function of ConvE is as following:

$$\psi(h_h, h_t) = f(\mathrm{vec}(f([\bar{h}_h; \bar{r}] * \omega))W)h_t, \tag{17}$$

where $f(\cdot)$ a non-linear function, $*$ denotes the convolution operator, $\bar{h}_h, \bar{r}$ denote a 2D reshaping of h_h and r, W denotes the trainable parameter matrix, and $\mathrm{vec}(\cdot)$ is the vectorization function.

Similar to the previous tasks, we train the model with negative sampling. For each triple corresponding to the positive knowledge, we randomly select n_s entities or relations to replace the corresponding elements in a real triple $t_r \in \mathcal{T}$ to generate the corresponding negative sample set $\mathcal{T}_{ns}$. We optimize the cross entropy loss to encourage our models to score higher for the real triples than the negative samples:

$$\mathcal{L} = -\frac{1}{n_s+1}\sum_{i=0,t_i\in\mathcal{T}_{ns}\bigcup t_r}^{n_s+1}(\hat{y}_i\log y_i + (1-\hat{y}_i)\log(1-y_i)), \tag{18}$$

where $\hat{y}_i \in \{0, 1\}$, $y_i = f(t_i)$ is the output probability of the candidate triple.

In particular, since we set attention heads in the *relation attention* based on the clustering phenomenon of these relations, we feed a entity embedding in a specific single attention heads to Eq. 15, Eq. 16, and Eq. 17 according to the cluster of the relation in the candidate triple. We will compare the performance of different entity embedding input strategies in the following experiments.

4. Experiments

In order to test the performance of our models, we verify them on three standard tasks: entity classification, entity typing, and link prediction.

4.1. Entity Classification

Here, since we learn the entity embedding of each vertex in the graph, we consider the task of classifying entities in a given knowledge graph [48, 67]. In fact, a mature model for learning knowledge graph embeddings needs to discriminate the categories for a given entity.

For the entity classification task, we design a two layer GAT. The first layer consists of $K_1 = 8$ attention heads computes $d' = d$ features, followed by an exponential linear unit nonlinearity. The second layer is utilized to classify the entities: a single attention head that computes c features (where c is the number of classes), followed by a softmax activation. The attention heads of *relation attention* are initialized based on unsupervised clustering results of relations (the number of categories corresponds to the number of attention heads). Our model is trained to minimize the cross-entropy loss on training entities with Adam [22] for 200 epochs using an initial learning rate of 0.01. Furthermore, we use L_2 regularization with $\rho = 0.001$ in our models to avoid over-fitting in small datasets. All of the hyper-parameters are tuned on a validation set, which is set aside 20% of the training set.

4.1.1. Datasets

In order to verify the effectiveness of our methods, we perform entity classification separately on the datasets from different fields. First, we choose two semantic web datasets in

Table 1: Statistics of three datasets for entity classification.

	AIFB (Research)	MUTAG (Molecular)	FB13 (Freebase)
#Entities	8285	23644	75043
#Relations	45	23	13
#Edges	29043	74227	0.31M
#Labeled	176	340	620
#Classes	4	2	6

Table 2: Entity classification results in accuracy (averaged 10 runs). ± is the standard deviation of the 10 runs.

Model	AIFB	MUTAG	FB13
RDF2Vec*	88.88	**76.20**	-
R-GCN*	95.83	73.23	-
TransE	84.82±0.39	63.37±0.47	79.44±0.45
GAKE	85.98±0.37	65.03±0.39	81.36±0.32
Ours(EA)	93.63±0.28	71.57±0.30	90.03±0.25
Ours(RA)	95.75±0.34	73.11±0.35	93.84±0.34
Ours(SA)	94.21±0.33	71.90±0.33	91.83±0.35
Ours(Mixed)	**96.14±0.31**	74.71±0.33	**94.27±0.29**

Resource Description Framework (RDF) format: AIFB, MUTAG [48, 46]. Second, we acquire a labeled dataset[2] based on the characteristics of the FB13 [51]. The exact statistics of the datasets can be found in Table 1.

4.1.2. Baselines

We select several targeted baseline methods for comparison. RDF2Vec [46] extracts the walking path in the labeled graph and models it to generate the entity embedding using the skip-gram. R-GCN [48] uses a GCN with a differentiable message-passing framework to learn the feature representation of entities in knowledge graph[3]. TransE [5] is a typical triple translation model whose source of entity representation is the positional relation in the semantic space. GAKE [14] is a method for obtaining vertex features based on multiple contexts and attention mechanisms in the graph structure.

In this work, we report the results of independent Entity Attention (EA), Relational Attention (RA), Structural Attention (SA), and their mixture (Mixed) according to Eq. 12.

4.1.3. Result Analysis

All the experimental results in accuracy are shown in Table 2. Our Mixed attention model achieves the state-of-the-art results on AIFB and FB13, however, there is a certain gap in the effect on MUTAG. To further understand the divergences of different datasets, we find out the reasons for two aspects: dataset composition and hyper-parameter selection. First, AIFB and MUTAG are not knowledge graphs. They are the RDF formatted semantic graph of AIFB staff composition and molecular composition. As a research organization, the relations in AIFB have the same correlation phenomena as described in Figure 1 (e.g., `employs` and `affiliation`), so it also adapt to our model. However, MUTAG is a molecular graph, and its edges are mainly atomic bonds, which leads to stronger independence of its context composition. Second, as illustrated in Figure 4, the number of channels (i.e. attention heads) we choose to learn features is smaller[4] than the number of clusters of relation types, which causes some features to be lost or overlapped.

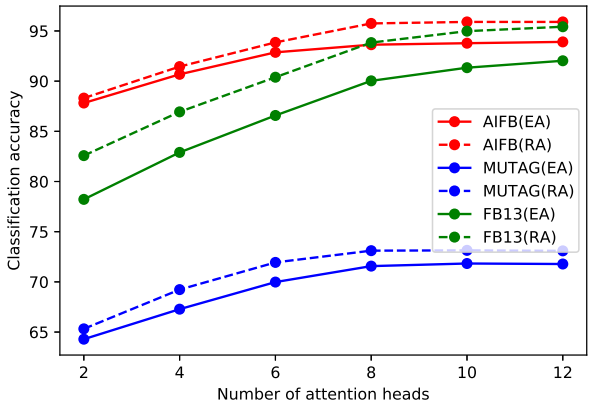

Figure 4: The relationship between the classification performance and the number of attention heads (i.e., K) in EA and RA.

Besides, compared with the results of models based on EA, RA, SA, and Mixed attention in Table 2, the EA based model is more stable, while the RA and SA based ones are more fluctuating, which is related to the choice of the relation clustering method and RWR. In addition, because the Mixed model integrates the above three attention mechanisms, its model stability is in the middle of the three. In general, Table 2 indicates that compositing neighbor features in multi-channels and equipping appropriate attention calculation methods provide a good supplement for learning entity embeddings.

4.2. Entity Typing

Entity typing is an important task in entity discovery and differs according to the granularity [20]. In heterogeneous graphs such as knowledge graphs, entity typing can be regarded as a multi-classification task. Specifically, it can be viewed as a kind of completion operation on entity attributes.

For entity typing tasks, we also designed a two-layer GAT. We set the number of neurons in the two hidden layers to be consistent with the input settings, and then implemented multi-classification through using binary classifiers corresponding to the number of categories. When training, the remaining parameters follow the settings in Section 4.1.

[2]We label the entities in knowledge graph dataset according to their related relations, e.g., we classify the head entities in (*mother_teresa*, `place_of_birth`, *skopje*) into *person* and the tail entity into *location*.

[3]Experimental results with $*$ are reused from the work of [48], and the other results with $**$ are reused from work of [56].

[4]To be fair, we select the same number of attention heads in three datasets. Obviously, the more relation clusters, the more number of attention heads required. Hence, if we want to achieve a better effect, we need to adjust the number of attention heads according to the number of relations.

Table 3: Statistics of two datasets for entity typing.

	FB15k-237_4000	WN18RR_4000
#Entities	4457	3846
#Relations	110	10
#Triples	27232	6439
#Features	100	100
#Classes	25	4
#Training	3565	3076
#Verification	446	385
#Testing	446	385

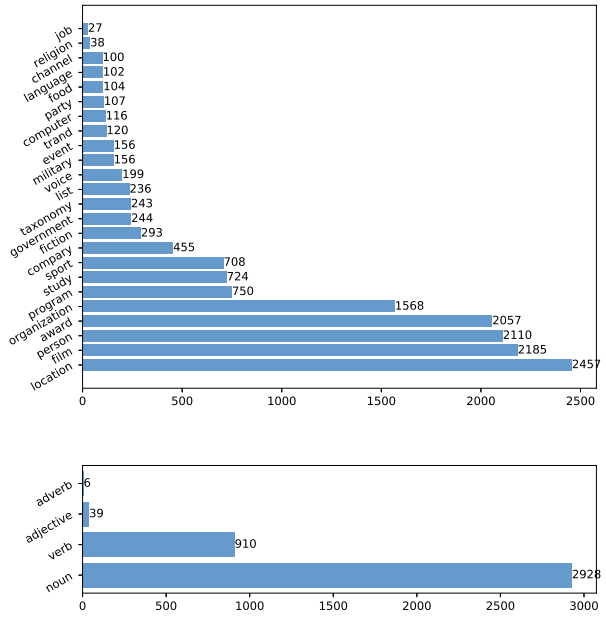

Figure 5: The distribution of different types in the FB15k-237_4000 and WN18RR_4000.

4.2.1. Datasets

To test the performance of the entity embeddings learned by the different graph attention mechanisms in the entity typing, we screened in standard FB15k-237 and WN18RR to form new independent datasets. Specifically, we first select the dense sub-graphs in the original datasets and ensure their connectivity. Next, by searching the attributes of the corresponding entities in the complete FB and WN datasets, each entity in the knowledge graph is given several attribute tags. (FB is the high-frequency words in the entity description text, and WN is the part-of-speech output of the related interface (NLTK.corpus.wordnet)). The training set, validation set, and testing set of the data set are divided into 8:1:1, where the training set ensures that each category has samples and is roughly balanced. The exact statistics of the datasets can be found in Table 3 and Figure 5.

4.2.2. Baselines

We still choose RDF2Vec [46] and R-GCN [48] as the baseline methods, and their multi-classifier setting is consistent with our method in this paper. In order to compare the different performances of different graph attention settings, we also set up four attention configurations, i.e., EA, RA, SA, and Mixed.

Table 4: Entity typing results in accuracy.

Model	FB15k-237_4000	WN18RR_4000
RDF2VEC(NB)	1.21±0.45	77.79±0.54
RDF2VEC(SVM)	17.11±0.44	75.48±0.49
R-GCN	43.08±0.63	91.05±0.60
Ours(EA)	46.32±0.42	89.92±0.52
Ours(RA)	51.35±0.34	90.04±0.44
Ours(SA)	52.72±0.38	91.83±0.60
Ours(Mixed)	**52.57±0.42**	**92.04±0.31**

4.2.3. Result Analysis

The experimental results are shown in Table 4, which indicates that neural-network-based models are significantly better than the traditional methods in terms of multi-class classification accuracy. Since the graph-attention-based method can better achieve multi-channel information capturing, it can more reasonably combine neighbor information than R-GCN, the classification performance is further improved.

Comparing different graph attention methods, we also find differences in the performance of various attention mechanisms. Specifically, for *relation attention*, the performance in FB15k-237_4000 far exceeds the *entity attention*, but it does not perform well on the WN18RR_4000: the result is slightly behind that of *entity attention*. We argue that this is due to the characteristics of the datasets, i.e., the FB15k-237_4000 has more relations, and the number and type of edges between entities are much more than the WN18RR_4000. Therefore, for *entity attention*, the weight between entities in *relation attention* can be more diversified, and it will not lack class separability like that in the WN18RR_4000. For *structure attention*, it has a significant improvement on the FB15k-237_4000 compared to the *entity attention*, and the other dataset basically maintains the same accuracy rate. The main reason is that WN18RR_4000 is more sparsely connected than FB15k-237_4000, thus the connection within the subgraph is not as tight as the latter, which may cause the noise between the remote entity and the central entity in the subgraph to be greater than The similarity of the two. In addition, since the entity of the sub dataset should be kept around 4000, when dividing the sub dataset, WN18RR_4000 selects the entity within 4 hops from the central entity. However, FB15k-237_4000 is 3 hops, for the edge part of the dataset not only did some of the neighboring entities be lost due to the segmentation of the dataset, but also the entity embeddings in the range of the 4-hop subgraph were collected, so the performance was not as good as expected.

4.3. Link Prediction

Link prediction is a standard task for verifying knowledge embedding[5]. We now validate our models using two sub-tasks

[5]In particular, because of the similarity between the triple classification task and the link prediction task, we did not show the results of the triple classifica-

Table 5: Statistics of three datasets for link prediction.

	FB15K (Freebase)	WN18RR (WordNet)	FB15k-237 (Freebase)
#Entities	14951	40943	14541
#Relations	1345	11	237
#Triples	0.48M	0.09M	0.27M
#Valid	50000	3034	17535
#Test	59071	3134	20466

of link prediction: entity prediction and relation prediction. Since the prediction methods of two sub-tasks are both ranking candidate triples in descending order using Eq. 15 Eq. 16 and Eq. 17, we describe the training details uniformly. We train our models by using the entity embedding obtained from *entity attention*, *relation attention*, *structure attention*, and mixed attention respectively. Besides, we further feed specific entity embedding to the downstream component, which is obtained from Average Nonlinearity (Eq. 14) (RA&AN) or Single Attention Head (RA&SAH) (corresponding to the relation in the candidate triple).

We utilize two layers of network structure with the number of attention heads and output feature dimensions as $K_1 = K_2 = 8, d' = d$, and the other hyperparameters in the model are obtained from the validation set. We feed n_s negative triples (constituted by replacing specific element in triples) and corresponding positive sample to Eq. 15, Eq. 16, and Eq. 17 to obtain the final scores. To minimize the cross-entropy loss described in Eq. 18, we update the model parameters using Adam optimizer with a learning rate of 0.01. Despite the training set is large enough, we still regularize our model through edge dropout with $p = 0.3$ and L_2 regularization with $\rho = 0.01$.

4.3.1. Datasets

We evaluate our methods on the typical large-scale knowledge graphs: FreeBase and WordNet. FB15K [5] is a real dense subgraph captured from Freebase, which contains the triples consisting of two entities and a relation. FB15K-237 is a subset of FB15K, which removes the inverse triples for filtering. WN18RR [5] is a set of linguistic triples obtained from WordNet without inverse triples, which is a lexical English dictionary consists of linguistic relation between words, e.g. `hypernym`, `hyponym`, and `meronym`. Based on the mentioned datasets, we build a fair and dense labeled KB graph for our method and comparative approaches. Table 5 details the statistics information of the three datasets.

4.3.2. Baselines

In this work, because of the need of comparing and discussing the performance of entity embedding based on the graph attention mechanism, we selected the three basic factorization models as baselines, i.e., DistMult [67], ComplEx [56], and ConvE [12]. Correspondingly, we list the three independent graph attention mechanisms (i.e., EA, RA, and SA) and mixed attention (Mixed) as comparison items.

4.3.3. Entity Prediction

In the sub-task of entity prediction, we utilize two common measures as our evaluation metrics: the Mean of Reciprocal Rank (MRR) and Hits at n (Hits@n). As [5] mentioned, some triples in the test set have more than one correct missing entity prediction results after hiding the entity to be predicted, so the evaluation metrics will result in underestimation of some methods. Hence, we filter out all these triples before ranking candidate triples, and report both raw and filtered MRR, and filtered Hits. In order to show the performance more completely, we also make statistics on the Hits at $n = 1, 3$, and 10 respectively.

The evaluation results of entity prediction are shown in Table 6. For our models, we still use four attention implementation methods to obtain KB embedding: *entity attention*, *relation attention*, *structure attention*, and mixed attention. By observing the data in Table 6, our proposed graph attention based approaches outperform the original factorization methods. It shows that the entity embeddings learned based on the graph attention mechanism can steadily improve the ability of the factorization models to model triples. The main reason for these improvements is the graph attention mechanism can better capture the features more effectively for the downstream tasks from the neighbors. However, because it is difficult to reasonably label the categories of the entities (i.e., words) in WordNet, the improvement in WN18RR is not so obvious. We believe that this is due to the selection of the score function and the use of attention, and we will explore other combinations for achieving a better effect.

By observing the different sources and calculation methods of attention mechanism and their performances in Table 6, we find that the method based on *relational attention* performs better than the other independent graph attention methods in more cases. On the one hand, the relation type is a normalized representation of correlation, which is a more specific and stable expression than the entity pair. On the other hand, because we have a clearer and more reasonable definition and initialization of attention heads using clustering operation, and the dot product is easier to operate for computing similarity/correlation, the *relational attention* based method achieves better result in the subsequent feature multi-channel learning. In addition, we found that the model based on SAH performed better than the one based on AN, which verifies our assumption that relation clustering contributes to knowledge modeling. In addition, in most cases, the mixed attention model achieves the best results, which shows that different graph attention sources have the possibility of complementary information, and can steadily improve the performance of the model.

4.3.4. Relation Prediction

The sub-task of relation prediction is akin to the entity prediction, so we continue using the evaluation metrics of the previous sub-task. Since this task focuses on handling relations, we exploit FB15K-237, which has enough relations, to evaluate our methods and baselines. The experimental results are

tion task due to the space limitation.

Table 6: Results of entity prediction on the FB15K-237 and WN18RR datasets.

	FB15K-237					WN18RR				
	MRR		Hits@			MRR		Hits@		
	Raw	Filtered	1	3	10	Raw	Filtered	1	3	10
DistMult**	0.159	0.281	0.199	0.301	0.446	0.248	0.444	0.412	0.470	0.504
DistMult(EA)	0.161	0.293	0.206	0.315	0.453	0.251	0.452	0.417	0.478	0.521
DistMult(RA&AN)	0.171	0.301	0.227	0.321	0.461	0.257	0.454	0.421	0.486	0.527
DistMult(RA&SAH)	0.178	0.307	0.234	0.327	0.465	0.266	0.465	0.442	0.502	0.539
DistMult(SA)	0.167	0.298	0.228	0.311	0.450	0.259	0.456	0.429	0.480	0.529
DistMult(Mixed)	0.174	0.312	0.230	0.326	0.471	0.261	0.464	0.436	0.493	0.543
ConvE**	0.183	0.312	0.225	0.341	0.497	0.263	0.456	0.419	0.470	0.531
ConvE(EA)	0.198	0.334	0.237	0.354	0.509	0.268	0.465	0.425	0.478	0.540
ConvE(RA&AN)	0.207	0.339	0.243	0.360	0.515	0.272	0.471	0.432	0.489	0.548
ConvE(RA&SAH)	0.214	**0.346**	**0.251**	**0.367**	**0.521**	**0.280**	**0.480**	**0.441**	**0.498**	0.558
ConvE(SA)	0.203	0.333	0.240	0.356	0.511	0.277	0.476	0.436	0.492	0.552
ConvE(Mixed)	**0.220**	0.342	0.249	0.364	0.517	0.279	**0.480**	0.439	**0.498**	**0.563**
ComplEx**	0.142	0.278	0.194	0.297	0.449	0.264	0.449	0.409	0.469	0.530
ComplEx(EA)	0.146	0.281	0.198	0.303	0.456	0.267	0.451	0.413	0.472	0.534
ComplEx(RA&AN)	0.148	0.284	0.202	0.307	0.459	0.269	0.454	0.415	0.476	0.537
ComplEx(RA&SAH)	0.153	0.291	0.207	0.314	0.465	0.273	0.459	0.418	0.481	0.544
ComplEx(SA)	0.149	0.286	0.205	0.310	0.458	0.268	0.453	0.414	0.471	0.536
ComplEx(Mixed)	0.150	0.288	0.206	0.312	0.466	0.274	0.461	0.418	0.483	0.546

Table 7: Results of relation prediction on the FB15K-237 dataset.

	MRR		Hits@		
	Raw	Filtered	1	3	10
DistMult	0.302	0.516	0.361	0.584	0.845
DistMult(EA)	0.328	0.541	0.389	0.617	0.853
DistMult(RA&AN)	0.331	0.550	0.393	0.624	0.858
DistMult(RA&SAH)	0.335	0.556	0.396	0.626	0.862
DistMult(SA)	0.336	0.558	0.397	0.624	0.861
DistMult(Mixed)	0.338	0.556	0.396	0.627	0.864
ConvE	0.611	0.827	0.735	0.887	0.965
ConvE(EA)	0.614	0.831	0.740	0.891	0.966
ConvE(RA&AN)	0.619	0.837	0.744	0.892	0.967
ConvE(RA&SAH)	0.623	0.844	0.745	**0.897**	0.971
ConvE(SA)	0.618	0.836	0.741	0.893	0.967
ConvE(Mixed)	**0.624**	**0.847**	**0.746**	0.896	**0.973**
ComplEx	0.527	0.702	0.504	0.711	0.896
ComplEx(EA)	0.533	0.708	0.509	0.718	0.899
ComplEx(RA&AN)	0.536	0.710	0.521	0.725	0.903
ComplEx(RA&SAH)	0.544	0.717	0.529	0.728	0.909
ComplEx(SA)	0.545	0.715	0.531	0.732	0.907
ComplEx(Mixed)	0.547	0.721	0.534	0.737	0.913

shown in Table 7. Our proposed graph attention based methods can improve the original factorization models, which indicates that our methods are equally applicable to the task of relational element prediction in triples.

Comparing the various graph attention methods we proposed, we found that the entity embeddings obtained by *relation attention* and corresponding relation embeddings perform better. Besides, we found that using mixed attention can also achieve better performance in the relational prediction. We also found that when using the same number of attention heads, the fewer the relation types be in the dataset, the better the model's effect is, which is consistent with the conclusions obtained in the experiment for entity classification.

4.4. Parameter Sensitivity Analysis

In order to analyze the possible influence of the three hyper-parameter settings in Eq. 11 on the performance of the Mixed attention model, we conduct parameter sensitivity analysis for the three experiments mentioned above. Specifically, we try to observe the changing trend of experimental results by setting different combinations of hyper-parameters in the experiments of entity classification, entity annotation, and link prediction. The experimental results are shown in Figure 6. We can find that the performance of the Mixed attention model will not be significantly affected by adjusting the remaining parameter on the premise of fixing two parameters in three groups of experiments. In particular, in the entity classification task, reducing the corresponding weight of RA (i.e., μ) will bring relatively large performance fluctuation, which is consistent with the phenomenon that RA performance is higher than the remaining model in the original experiment. Similar phenomena also appear in the task of entity link prediction.

4.5. Visualization of Relation Embedding

Since we experiment with entity embedding, i.e. entity classification, we only report the visualization results of the relation embedding here. We select some of the high-frequency relations in FB15k-237, and use Principal Component Analysis to reduce the dimension of relation embeddings to 2 based on RA. We observe that the relation embedding nicely reflects the clustering structures among these relations either in the same attention head or the same prefixion (e.g. /people/person/languages and /people/person/nationality).

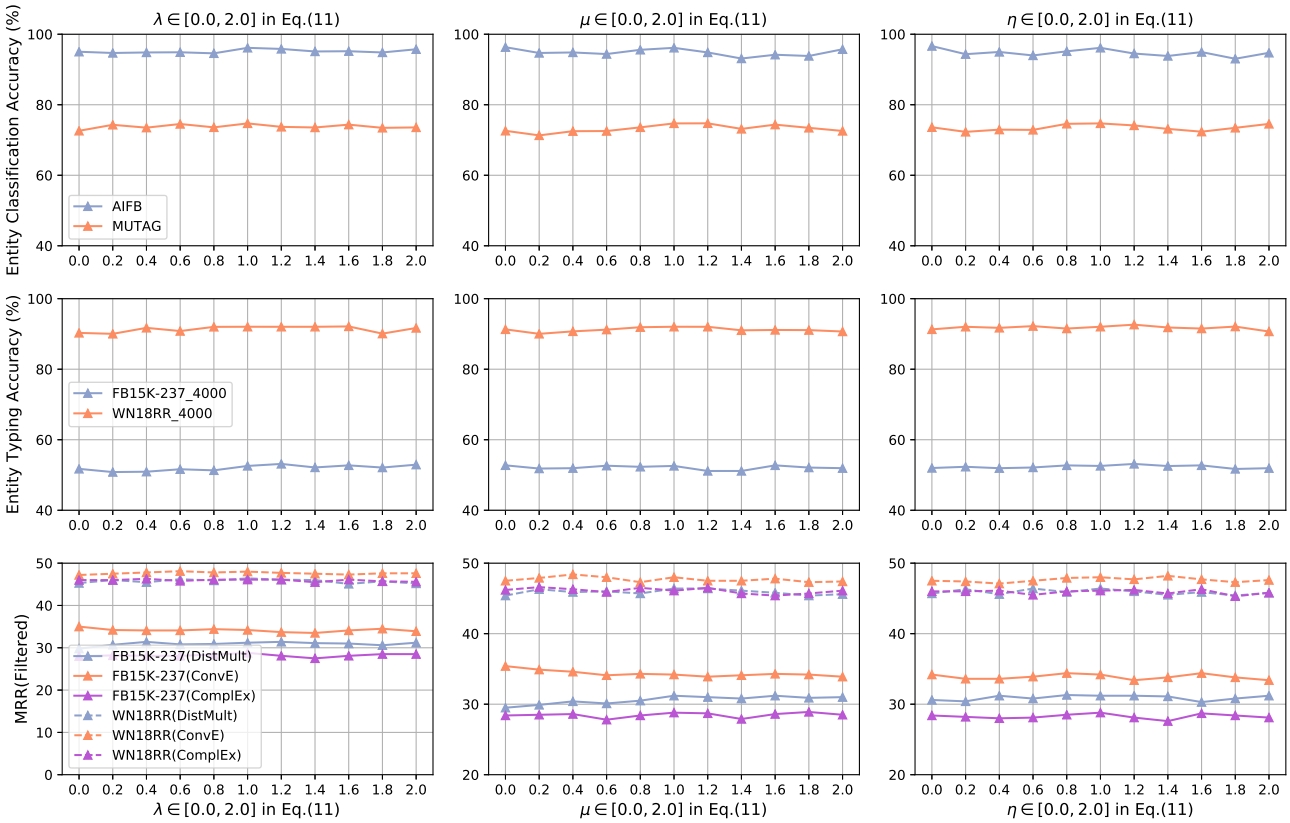

Figure 6: Experimental results of different hyper-parameter combinations in three experiments. The rows represent three experiments from top to bottom: entity classification, entity typing, and entity link prediction. The columns from left to right represent experiments with different hyper-parameter settings in Eq. 11: λ, μ, and η. Among them, the settings in each column are to adjust a certain hyper-parameter and fix the remaining two hyper-parameters. For example, in the first column, $\mu = 1.0, \eta = 1.0$, and $\lambda \in [0.0, 2.0]$.

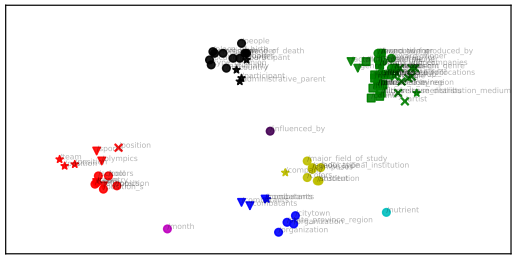

Figure 7: Visualization of relation embeddings. The colors denote different attention heads and the shapes represent different prefixes in each attention head.

5. Conclusion

In this paper, we introduce the graph multi-head attention mechanism into relational data modeling and demonstrate its effectiveness in knowledge embedding through extensive experiments. In particular, we use three different sources of attention (entity pair, relation, and structure) to achieve the learning of vertex features in the graph. Experimental results validate our assumptions about entity types and relations. In the future, we will explore the essential similarities between graph attention and graph convolution in feature learning.

Acknowledgements

The corresponding author is Jianxin Li. This work was supported by the NSFC program (No. U20B2053, 62002007 and 61772151), S&T Program of Hebei through grant 20310101D, and SKLSDE-2020ZX-12.

References

[1] Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary G. Ives. Dbpedia: A nucleus for a web of open data. In *The Semantic Web, 6th International Semantic Web Conference, 2nd Asian Semantic Web Conference, ISWC 2007 + ASWC 2007, Busan, Korea, November 11-15, 2007*, pages 722–735, 2007.

[2] Ivana Balazevic, Carl Allen, and Timothy M. Hospedales. Tucker: Tensor factorization for knowledge graph completion. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*, pages 5184–5193, 2019.

[3] Aleksandar Bojchevski, Oleksandr Shchur, Daniel Zügner, and Stephan Günnemann. Netgan: Generating graphs via random walks. In Jennifer G. Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 609–618. PMLR, 2018.

[4] Kurt D. Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2008, Vancouver, BC, Canada, June 10-12, 2008*, pages 1247–1250, 2008.

[5] Antoine Bordes, Nicolas Usunier, Alberto García-Durán, Jason Weston, and Oksana Yakhnenko. Translating embeddings for modeling multi-relational data. In *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States*, pages 2787–2795, 2013.

[6] Antoine Bordes, Jason Weston, Ronan Collobert, and Yoshua Bengio. Learning structured embeddings of knowledge bases. In *Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2011, San Francisco, California, USA, August 7-11, 2011*, 2011.

[7] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. Spectral networks and locally connected networks on graphs. 2014.

[8] Liwei Cai and William Yang Wang. KBGAN: adversarial learning for knowledge graph embeddings. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2018, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 1 (Long Papers)*, pages 1470–1480, 2018.

[9] Nicola De Cao and Thomas Kipf. Molgan: An implicit generative model for small molecular graphs. *CoRR*, abs/1805.11973, 2018.

[10] Shaosheng Cao, Wei Lu, and Qiongkai Xu. Deep neural networks for learning graph representations. In Dale Schuurmans and Michael P. Wellman, editors, *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA*, pages 1145–1152. AAAI Press, 2016.

[11] Jeffrey Dalton, Laura Dietz, and James Allan. Entity query feature expansion using knowledge base links. In *The 37th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '14, Gold Coast , QLD, Australia - July 06 - 11, 2014*, pages 365–374, 2014.

[12] Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. Convolutional 2d knowledge graph embeddings. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pages 1811–1818, 2018.

[13] Takuma Ebisu and Ryutaro Ichise. Toruse: Knowledge graph embedding on a lie group. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pages 1819–1826, 2018.

[14] Jun Feng, Minlie Huang, Yang Yang, and Xiaoyan Zhu. GAKE: graph aware knowledge embedding. In *COLING 2016, 26th International Conference on Computational Linguistics, Proceedings of the Conference: Technical Papers, December 11-16, 2016, Osaka, Japan*, pages 641–651, 2016.

[15] David A. Ferrucci, Eric W. Brown, Jennifer Chu-Carroll, James Fan, David Gondek, Aditya Kalyanpur, Adam Lally, J. William Murdock, Eric Nyberg, John M. Prager, Nico Schlaefer, and Christopher A. Welty. Building watson: An overview of the deepqa project. *AI Magazine*, 31(3):59–79, 2010.

[16] Cong Fu, Tong Chen, Meng Qu, Woojeong Jin, and Xiang Ren. Collaborative policy learning for open knowledge graph reasoning. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*, pages 2672–2681, 2019.

[17] Evgeniy Gabrilovich and Shaul Markovitch. Wikipedia-based semantic interpretation for natural language processing. *CoRR*, abs/1401.5697, 2014.

[18] Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*, pages 855–864, 2016.

[19] Guoliang Ji, Shizhu He, Liheng Xu, Kang Liu, and Jun Zhao. Knowledge graph embedding via dynamic mapping matrix. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 1: Long Papers*, pages 687–696, 2015.

[20] Shaoxiong Ji, Shirui Pan, Erik Cambria, Pekka Marttinen, and Philip S. Yu. A survey on knowledge graphs: Representation, acquisition and applications. *CoRR*, abs/2002.00388, 2020.

[21] Seyed Mehran Kazemi and David Poole. Simple embedding for link prediction in knowledge graphs. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, 3-8 December 2018, Montréal, Canada*, pages 4289–4300, 2018.

[22] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. 2015.

[23] Timothée Lacroix, Guillaume Obozinski, and Nicolas Usunier. Tensor decompositions for temporal knowledge base completion. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*, 2020.

[24] Chen Li, Xutan Peng, Hao Peng, Jianxin Li, Lihong Wang, and Philip S. Yu. Forming an electoral college for a graph: a heuristic semi-supervised learning framework. *CoRR*, abs/2006.06469, 2020.

[25] Chen Li, Xutan Peng, Shanghang Zhang, Hao Peng, Philip S. Yu, Min He, LinFeng Du, and Lihong Wang. Modeling relation paths for knowledge base completion via joint adversarial training. *Knowl. Based Syst.*, 201-202:105865, 2020.

[26] Yujia Li, Daniel Tarlow, Marc Brockschmidt, and Richard S. Zemel. Gated graph sequence neural networks. 2016.

[27] Yujia Li, Oriol Vinyals, Chris Dyer, Razvan Pascanu, and Peter W. Battaglia. Learning deep generative models of graphs. *CoRR*, abs/1803.03324, 2018.

[28] Xi Victoria Lin, Richard Socher, and Caiming Xiong. Multi-hop knowledge graph reasoning with reward shaping. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 3243–3253, 2018.

[29] Yankai Lin, Zhiyuan Liu, Huan-Bo Luan, Maosong Sun, Siwei Rao, and Song Liu. Modeling relation paths for representation learning of knowledge bases. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*, pages 705–714, 2015.

[30] Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. Learning entity and relation embeddings for knowledge graph completion. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, January 25-30, 2015, Austin, Texas, USA*, pages 2181–2187, 2015.

[31] Hanxiao Liu, Yuexin Wu, and Yiming Yang. Analogical inference for multi-relational embeddings. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, pages 2168–2178, 2017.

[32] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. 2013.

[33] Federico Monti, Davide Boscaini, Jonathan Masci, Emanuele Rodolà, Jan Svoboda, and Michael M. Bronstein. Geometric deep learning on graphs and manifolds using mixture model cnns. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 5425–5434, 2017.

[34] Changsung Moon, Paul Jones, and Nagiza F. Samatova. Learning entity type embeddings for knowledge graph completion. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management, CIKM 2017, Singapore, November 06 - 10, 2017*, pages 2215–2218, 2017.

[35] Deepak Nathani, Jatin Chauhan, Charu Sharma, and Manohar Kaul. Learning attention-based embeddings for relation prediction in knowl-

edge graphs. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4710–4723, Florence, Italy, July 2019. Association for Computational Linguistics.

[36] Deepak Nathani, Jatin Chauhan, Charu Sharma, and Manohar Kaul. Learning attention-based embeddings for relation prediction in knowledge graphs. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 4710–4723, 2019.

[37] Hien T. Nguyen, Phuc H. Duong, and Erik Cambria. Learning short-text semantic similarity with word embeddings and external knowledge sources. *Knowl. Based Syst.*, 182, 2019.

[38] Maximilian Nickel, Lorenzo Rosasco, and Tomaso A. Poggio. Holographic embeddings of knowledge graphs. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA*, pages 1955–1961, 2016.

[39] Mathias Niepert, Mohamed Ahmed, and Konstantin Kutzkov. Learning convolutional neural networks for graphs. In *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, pages 2014–2023, 2016.

[40] Shirui Pan, Ruiqi Hu, Guodong Long, Jing Jiang, Lina Yao, and Chengqi Zhang. Adversarially regularized graph autoencoder for graph embedding. In Jérôme Lang, editor, *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13-19, 2018, Stockholm, Sweden*, pages 2609–2615. ijcai.org, 2018.

[41] Shirui Pan, Ruiqi Hu, Guodong Long, Jing Jiang, Lina Yao, and Chengqi Zhang. Adversarially regularized graph autoencoder for graph embedding. In Jérôme Lang, editor, *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13-19, 2018, Stockholm, Sweden*, pages 2609–2615. ijcai.org, 2018.

[42] Xutan Peng, Guanyi Chen, Chenghua Lin, and Mark Stevenson. 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*, Online, June 2021. Association for Computational Linguistics.

[43] Wei Qian, Cong Fu, Yu Zhu, Deng Cai, and Xiaofei He. Translating embeddings for knowledge graph completion with relation attention mechanism. In *IJCAI*, pages 4286–4292, 2018.

[44] Hongyu Ren, Weihua Hu, and Jure Leskovec. Query2box: Reasoning over knowledge graphs in vector space using box embeddings. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*, 2020.

[45] Xiang Ren, Zeqiu Wu, Wenqi He, Meng Qu, Clare R. Voss, Heng Ji, Tarek F. Abdelzaher, and Jiawei Han. Cotype: Joint extraction of typed entities and relations with knowledge bases. In *Proceedings of the 26th International Conference on World Wide Web, WWW 2017, Perth, Australia, April 3-7, 2017*, pages 1015–1024, 2017.

[46] Petar Ristoski and Heiko Paulheim. Rdf2vec: RDF graph embeddings for data mining. In *The Semantic Web - ISWC 2016 - 15th International Semantic Web Conference, Kobe, Japan, October 17-21, 2016, Proceedings, Part I*, pages 498–514, 2016.

[47] Daniel Ruffinelli, Samuel Broscheit, and Rainer Gemulla. You CAN teach an old dog new tricks! on training knowledge graph embeddings. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*, 2020.

[48] Michael Sejr Schlichtkrull, Thomas N. Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. Modeling relational data with graph convolutional networks. In *The Semantic Web - 15th International Conference, ESWC 2018, Heraklion, Crete, Greece, June 3-7, 2018, Proceedings*, pages 593–607, 2018.

[49] Chao Shang, Yun Tang, Jing Huang, Jinbo Bi, Xiaodong He, and Bowen Zhou. End-to-end structure-aware convolutional networks for knowledge base completion. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*, pages 3060–3067, 2019.

[50] Jun Shi, Huan Gao, Guilin Qi, and Zhangquan Zhou. Knowledge graph embedding with triple context. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management, CIKM 2017, Singapore, November 06 - 10, 2017*, pages 2299–2302, 2017.

[51] Richard Socher, Danqi Chen, Christopher D. Manning, and Andrew Y. Ng. Reasoning with neural tensor networks for knowledge base completion. In *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States*, pages 926–934, 2013.

[52] Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. Yago: a core of semantic knowledge. In *Proceedings of the 16th International Conference on World Wide Web, WWW 2007, Banff, Alberta, Canada, May 8-12, 2007*, pages 697–706, 2007.

[53] Zhiqing Sun, Zhi-Hong Deng, Jian-Yun Nie, and Jian Tang. Rotate: Knowledge graph embedding by relational rotation in complex space. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*, 2019.

[54] Hanghang Tong, Christos Faloutsos, and Jia-Yu Pan. Fast random walk with restart and its applications. In *Proceedings of the 6th IEEE International Conference on Data Mining (ICDM 2006), 18-22 December 2006, Hong Kong, China*, pages 613–622, 2006.

[55] Ha Nguyen Tran and Erik Cambria. A survey of graph processing on graphics processing units. *J. Supercomput.*, 74(5):2086–2115, 2018.

[56] Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. Complex embeddings for simple link prediction. In *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, pages 2071–2080, 2016.

[57] Ke Tu, Peng Cui, Xiao Wang, Philip S. Yu, and Wenwu Zhu. Deep recursive network embedding with regular equivalence. In Yike Guo and Faisal Farooq, editors, *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2018, London, UK, August 19-23, 2018*, pages 2357–2366. ACM, 2018.

[58] Shikhar Vashishth, Soumya Sanyal, Vikram Nitin, and Partha P. Talukdar. Composition-based multi-relational graph convolutional networks. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*, 2020.

[59] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pages 5998–6008, 2017.

[60] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. 2018.

[61] Chun-Chih Wang and Pu-Jen Cheng. Translating representations of knowledge graphs with neighbors. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval, SIGIR 2018, Ann Arbor, MI, USA, July 08-12, 2018*, pages 917–920, 2018.

[62] Daixin Wang, Peng Cui, and Wenwu Zhu. Structural deep network embedding. In Balaji Krishnapuram, Mohak Shah, Alexander J. Smola, Charu C. Aggarwal, Dou Shen, and Rajeev Rastogi, editors, *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*, pages 1225–1234. ACM, 2016.

[63] Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. Knowledge graph embedding by translating on hyperplanes. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence, July 27 -31, 2014, Québec City, Québec, Canada*, pages 1112–1119, 2014.

[64] Ruobing Xie, Zhiyuan Liu, Jia Jia, Huanbo Luan, and Maosong Sun. Representation learning of knowledge graphs with entity descriptions. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA*, pages 2659–2665, 2016.

[65] Ruobing Xie, Zhiyuan Liu, Huanbo Luan, and Maosong Sun. Image-embodied knowledge representation learning. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, Melbourne, Australia, August 19-25, 2017*, pages 3140–3146, 2017.

[66] Xiaoran Xu, Wei Feng, Yunsheng Jiang, Xiaohui Xie, Zhiqing Sun, and Zhi-Hong Deng. Dynamically pruned message passing networks for large-scale knowledge graph reasoning. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*, 2020.

[67] Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. Embedding entities and relations for learning and inference in knowledge

bases. 2015.

[68] Jiaxuan You, Rex Ying, Xiang Ren, William L. Hamilton, and Jure Leskovec. Graphrnn: Generating realistic graphs with deep auto-regressive models. In Jennifer G. Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 5694–5703. PMLR, 2018.

[69] Tom Young, Erik Cambria, Iti Chaturvedi, Hao Zhou, Subham Biswas, and Minlie Huang. Augmenting end-to-end dialogue systems with commonsense knowledge. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pages 4970–4977, 2018.

[70] Wenchao Yu, Cheng Zheng, Wei Cheng, Charu C. Aggarwal, Dongjin Song, Bo Zong, Haifeng Chen, and Wei Wang. Learning deep network representations with adversarially regularized autoencoders. In Yike Guo and Faisal Farooq, editors, *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2018, London, UK, August 19-23, 2018*, pages 2663–2671. ACM, 2018.

[71] Kai Zhang, Yaokang Zhu, Jun Wang, and Jie Zhang. Adaptive structural fingerprints for graph attention networks. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*, 2020.

[72] Zhanqiu Zhang, Jianyu Cai, Yongdong Zhang, and Jie Wang. Learning hierarchy-aware knowledge graph embeddings for link prediction. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 3065–3072, 2020.