Virtual Event, Ireland Ren. 2020. VN Network: Embedding Newly Emerging Entities with Vir-tual Neighbors

Conference Paper · February 2022

CITATIONS
2

READS
19

5 authors, including:

He Yongquan
Chinese Academy of Sciences
5 PUBLICATIONS **3** CITATIONS
SEE PROFILE

Zihan Wang
Shandong University
12 PUBLICATIONS **104** CITATIONS
SEE PROFILE

VN Network: Embedding Newly Emerging Entities with Virtual Neighbors

Yongquan He
heyongquan@iie.ac.cn
School of Cyber Security, University of Chinese Academy of Sciences
Institute of Information Engineering, Chinese Academy of Sciences

Zihan Wang
zihanwang.sdu@gmail.com
School of Computer Science and Technology, Shandong University

Peng Zhang*
pengzhang@iie.ac.cn
Institute of Information Engineering, Chinese Academy of Sciences

Zhaopeng Tu
tuzhaopeng@gmail.com
Tencent AI Lab

Zhaochun Ren
zhaochun.ren@sdu.edu.cn
Shandong University

ABSTRACT

Embedding entities and relations into continuous vector spaces has attracted a surge of interest in recent years. Most embedding methods assume that all test entities are available during training, which makes it time-consuming to retrain embeddings for newly emerging entities. To address this issue, recent works apply the graph neural network on the existing neighbors of the unseen entities. In this paper, we propose a novel framework, namely Virtual Neighbor (VN) network, to address three key challenges. Firstly, to reduce the **neighbor sparsity problem**, we introduce the concept of the virtual neighbors inferred by rules. And we assign soft labels to these neighbors by solving a rule-constrained problem, rather than simply regarding them as unquestionably true. Secondly, many existing methods only use one-hop or two-hop neighbors for aggregation and ignore the distant information that may be helpful. Instead, we identify both logic and symmetric path rules to capture **complex patterns**. Finally, instead of one-time injection of rules, we employ an iterative learning scheme between the embedding method and virtual neighbor prediction to **capture the interactions within**. Experimental results on two knowledge graph completion tasks demonstrate that our VN network significantly outperforms state-of-the-art baselines. Furthermore, results on Subject/Object-R show that our proposed VN network is highly robust to the neighbor sparsity problem.

CCS CONCEPTS

• **Computing methodologies** → *Knowledge representation and reasoning*; *Natural language processing*.

*Corresponding author

CIKM '20, October 19–23, 2020, Virtual Event, Ireland

ACM ISBN 978-1-4503-6859-9/20/10…$15.00
https://doi.org/10.1145/3340531.3411865

KEYWORDS

knowledge graph embedding, unseen entities, virtual neighbors, rule-constrained problem

ACM Reference Format:
Yongquan He, Zihan Wang, Peng Zhang, Zhaopeng Tu, and Zhaochun Ren. 2020. VN Network: Embedding Newly Emerging Entities with Virtual Neighbors. In *Proceedings of the 29th ACM International Conference on Information and Knowledge Management (CIKM '20), October 19–23, 2020, Virtual Event, Ireland*. ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3340531.3411865

1 INTRODUCTION

Recently, knowledge graphs (KGs) such as Freebase [1] and Google's Knowledge Vault [4] have proven to be extremely useful resources for many natural language processing(NLP) relevant applications [17, 31]. A typical KG can be considered as a multi-relational graph, where nodes and various types of edges reflect entities and relations, respectively. Each edge is presented as a triple of the form (*head entity*, *relation*, *tail entity*), e.g., (*John*, *bornedIn*, *Athens*). Although effective in representing structured data, the symbolic nature of triples often makes KGs hard to manipulate, and most of the KGs are far from complete compared to existing facts in the real world. To tackle this issue, KG embedding methods have been proposed, aiming to embed entities (nodes) and relations (edges) into the continuous vector spaces. In that way, such kind of methods can encode the structure of the KGs and simplify the manipulation. And KG embeddings contain rich semantic information and can facilitate KG completion and inference [3, 6, 15, 25].

Despite the success of embedding methods, previous approaches still can not handle newly emerging entities as all the entities need to be seen in the training process. In that case, newly emerging entities are unknown to the original system. Without retraining on the whole KG, the original system is difficult to predict missing facts about new entities. As knowledge graphs evolve dynamically, new entities appear along with emerging events or new products [8, 19]. Therefore, an inductive learning framework is needed for avoiding a time-consuming retraining process. The key idea of an inductive learning framework is to generalize the original system to the newly observed subgraph, which requires the recognition of the neighborhood structural properties of the unseen nodes [9].

MEAN [8] applies the graph neural network (GNN) on both the observed and unseen entities, mapping the newly emerging entities into embeddings by aggregating their known neighbors. However, it neglects the different edge types (relations) and adopts the simple mean pooling aggregator. Based on that, logic attention based neighborhood aggregation network (LAN) [23] is proposed, which employs both logic rules and neural network attention mechanisms to capture the redundancy information and concentrate on the query relevant facts.

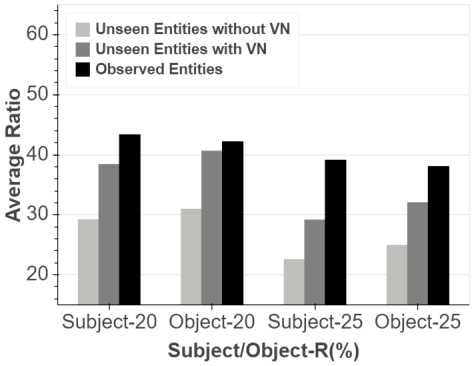


Figure 1: The average ratio of neighbors to predicted facts with and without virtual neighbors.

Although the above methods can solve the unseen entity problem to some extent, there are still several problems not fully addressed.

Neighbor Sparsity Problem: In fact, compared to observed entities, we generally do not have enough information to learn good representations for newly emerging entities. As shown in Figure 1, in the subject-{20, 25} and object-{20, 25} datasets [23] which obtained by selecting 20% or 25% triples from FB15K [2] test set, to sample the candidate unseen entities. And then use these candidate unseen entities to split the original training set, to ensure that these unseen entities are not observed during training process. We can see that the average ratio of the known neighbors to the predicted triples for the unseen entities is lower than the observed entities by at least 11 in the current graph, which makes the knowledge representation learning much more difficult when collecting neighborhood information. For example, in the subject-20, we only know average 29.19 triples of facts about the unseen entity, but for the observed entities is 43.32. Both methods above suffer this sparsity problem.

Complex Pattern: The previous methods mainly focus on the 1 or 2 hops of the connecting structures, or the 1 hop rules about the relation dependence, while there are many other long-distance dependencies that are helpful for the knowledge base completion. As Figure 2 describes, at test time, we receive new triples containing the entity "*Alec Guiness*", which is not observed in the original KG. From the first path (green) we can find the entity "*Tom*", which has similar semantic about actors' roles. We know the fact that "*Tom*" was in the science fiction "*The Infinite Worlds*", and such fact can help complete the missing edge "*starredIn*", which means an actor played a character who belongs to a science fiction, he must be in that science fiction. And the second path (blue) hints that they were born in different cities in the same country. We can analogy that "*Alec*" had lived in England by finding the pattern about "*Tom*", meaning that a person was born in a city of a country, he must have lived in that country. So we can obtain more information and capture long dependencies between entities through these long-distance paths to assist with short logic rules.

Interactions Between the Rule Inference and Embedding Learning: LAN regards the confidence level of the logic rules as the constant weights for the aggregator, while embeddings encoded with rich semantics can tune the results inferred by rules. As RUGE [6] and IterE [33] mentioned, rules can infer new facts more accurately with the refined embeddings, and newly inferred facts can also help to learn the embeddings of higher quality.

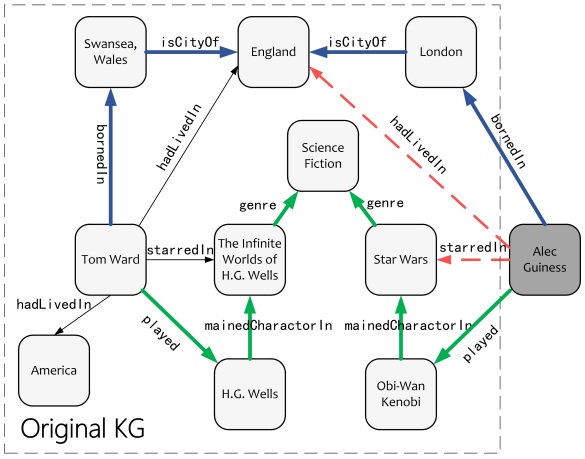


Figure 2: The example of the unseen entities problem about original KG.

In this work, we propose a novel inductive learning framework, VN network, to embed newly emerging entities. The VN network consists of three main components, including the virtual neighbor prediction, encoder, and decoder. To reduce the sparsity problem, we introduce the concept of the virtual neighbors and employ the rule-based virtual neighbor prediction algorithm. Meanwhile, to capture more information in the KGs, we adopt the logic rules and identify the long-distance symmetric path rules. In that case, the soft labels of triples inferred by rules are computed by solving a convex rule-constrained problem over the current KG embeddings. Then, the KG with the predicted virtual neighbors is inputted to the GNN-based encoder composed of several structure aware layers and one query aware layer. Finally, we output the embeddings from the encoder to the decoder for predicting the missing facts. At each iteration, to combine with rules in an iterative manner, we first refine the soft labels using the current embeddings and then obtain the optimal current KG embeddings by minimizing the global loss over both the hard labeled and soft labeled triples.

Our contributions are summarized as follows: 1)We propose a novel inductive learning framework to handle the unseen entities. 2)We develop a virtual neighbor prediction method to reduce the sparsity problem, identify the logic and symmetric path rules to capture more information and establish an iterative refinement

scheme over the soft labels and KG embeddings. 3)We conduct two types of the knowledge graph completion tasks on WordNet11, FB15K and YAGO37 to demonstrate the effectiveness of the VN network, and show about a significant improvement over the state-of-the-art LAN.

2 RELATED WORK

In recent years, we have witnessed increasing interests in embedding methods for KGs. Such methods have become one of the most important techniques on knowledge base completion (KBC), aiming to map the entities and relations into continuous vector space. Recent works can be mainly classified into two categories: some of them proposed more complicated scoring functions and deeper frameworks, such as TransE extensions [10, 26], RESCAL extensions [22, 32], and combining with deep neural network [18, 25]. Others tried to further incorporate other information available, including relation paths [14, 28, 35], typing information [34], visual information [29] and logic rules [3, 6, 33]. More detailed reviews can be found in [11] and [24].

Although proving the success in the KBC task, traditional KG embedding methods still fall short on the newly emerging entities issue. Previous embedding frameworks require all the entities should be seen during the training process. However, KG evolves frequently, and a large number of new entities emerge almost on a daily basis, especially between late 2015 and early 2016 [19]. Meanwhile, retraining on the whole KG to obtain embeddings of the new entities is extremely time-consuming. To address the newly emerging entities problem, several works utilize other types of information, such as text descriptions and image [19, 29], to predict new facts. These methods are still limited when text and image information are not sufficient or provided. Moreover, we also need knowledge which are usually graph-specific under domain expert guidance to use these information. Compared with above information, rules are easier to obtain, and there are many efficient rule mining methods for knowledge graph, such as AMIE+ [5] or RLvLR [16]. And rules are inherently inductive since they are independent of node identities, which can assist embedding learning as useful information for reasoning. The efficiency of learning embedding and rules in an iterative way has proved [6, 33]. But most general rule-based methods only focus on one-hop and two-hop relations. In this paper, through introducing the symmetric path rules, we can capture long-distance dependencies between entities.

There are some other works such as DKGE [27] aiming at solving the emerging facts about KGs, but the most relevant works to ours are LAN [8] and MEAN [23], which focus on the representation learning of the unseen entities by aggregating neighbors. MEAN [8] applies a simple mean pooling on the neighborhood structures of the emerging entities without distinguishing the edge types. To obtain a more effective neighborhood aggregator, LAN [23] employs two kinds of attention mechanisms, and the attention weights are estimated by either logic rules or a neural network. These methods above still face the neighbor-sparseness problem when few neighbors of unseen entities are known. They also ignore the meaningful complex patterns in the KGs which only focuses on one-hop neighbors and fail to recognize the interactions between rules and embeddings. In contrast, our method, VN network, proposes the concepts of the virtual neighbors to handle the sparseness problem, identify logic and symmetric path rules to capture more information, and optimize the KG embeddings and rule inference in an iterative manner.

3 METHOD

This section introduces the proposed Virtual Neighbor network (VN network). As Figure 3 shows, the model has three main components: the rule-based virtual neighbor prediction, a GNN-based encoder and an embedding-based decoder. In the following sections, we describe the definitions and notations used in this paper firstly. And then we give an overview of the model architecture and detail the three components.

3.1 Definitions

This section introduces the definitions and notations for the knowledge graph (KG) under the setting of unseen entities and rules used in this paper.

3.1.1 Knowledge Graph (KG). A knowledge graph can be considered as a multi-relational graph, consisting of a set of the **observed edges** (fact triples), i.e., $\mathcal{O} = \{x_o\}$, where $x_o = (e_i, r_k, e_j)$. Each edge (triple) consists of two nodes (entities) $e_i, e_j \in \mathcal{E}_o$, and the edge (relation) $r_k \in \mathcal{R}$, where $\mathcal{E}_o$ and $\mathcal{R}$ are the entity and relation sets respectively. And for an entity e, we define its neighborhood in $\mathcal{O}$ as $N_o(e)$, where $N_o(e) = \{(e^{'}, r)|(e^{'}, r, e) \in \mathcal{O} \vee (e, r, e^{'}) \in \mathcal{O}\}$.

In addition to the observed triples and entities, we collect the **newly emerging entities** and **auxiliary triples**. Newly emerging entities are unseen during the training process but appear in the test set. And auxiliary triples are newly added facts for the original KG, which connect the unseen entities with the original graph, to learn the embeddings for unseen entities during the aggregating process. For example, as Figure 2 shows, $(London, isCityOf, England) \in \mathcal{O}$ is an observed triple, where $London, England \in \mathcal{E}_O$ and $isCityOf \in \mathcal{R}$. $(Alec, bornedIn, London) \in \mathcal{AUX}$ is an auxiliary triple, composed of one unseen entity $Alec \in \mathcal{E}_u$, one observed entity $London \in \mathcal{E}_o$, and their relation $bornedIn \in \mathcal{R}$, where $\mathcal{AUX}$ is the auxiliary triple set, and $\mathcal{E}_u$ is the unseen entity set.

In the VN network, to address the sparseness problem of $\mathcal{AUX}$ as mentioned above, we introduce the concept of the **virtual neighbor**, which were obtained by using rules on $\mathcal{O}$ and $\mathcal{AUX}$ (detailed in §3.3). To be more specific, a triple (e_i, r_k, e_j) is not in $\mathcal{O}$ and $\mathcal{AUX}$, where $e_i \in \mathcal{E}_o$, $e_j \in \mathcal{E}_u$ and $r_k \in \mathcal{R}$, we define the head entity e_i as the **virtual neighbor** of the tail entity e_j under relation r_k. We denote the set containing such kind of triples as $\mathcal{VN} = \{x_{\mathcal{VN}}\}$. And as in Figure 2, $(Alec, bornedIn, London)$ is newly added fact containing unseen entity "$Alec$", so it belongs to $\mathcal{AUX}$. $(Alec, starredIn, StarWars)$ can be inferred by rules, and it is not in $\mathcal{O}$ and $\mathcal{AUX}$, so $(StarWars, starredIn)$ is the virtual neighbor of "$Alec$".

3.1.2 Rule. A set of rules with different confidence levels are denoted as $\mathcal{F} = \{f_p, \lambda_p\}$, where f_p is the p-th logic rule defined over the given KG. For example, $(i, r_1, j) \Rightarrow (i, r_2, j) : i, j \in \mathcal{E}$ and $r_1, r_2 \in \mathcal{R}$, which indicates that any two entities connected by r_1 should also be connected by r_2. The left-hand side of the implication

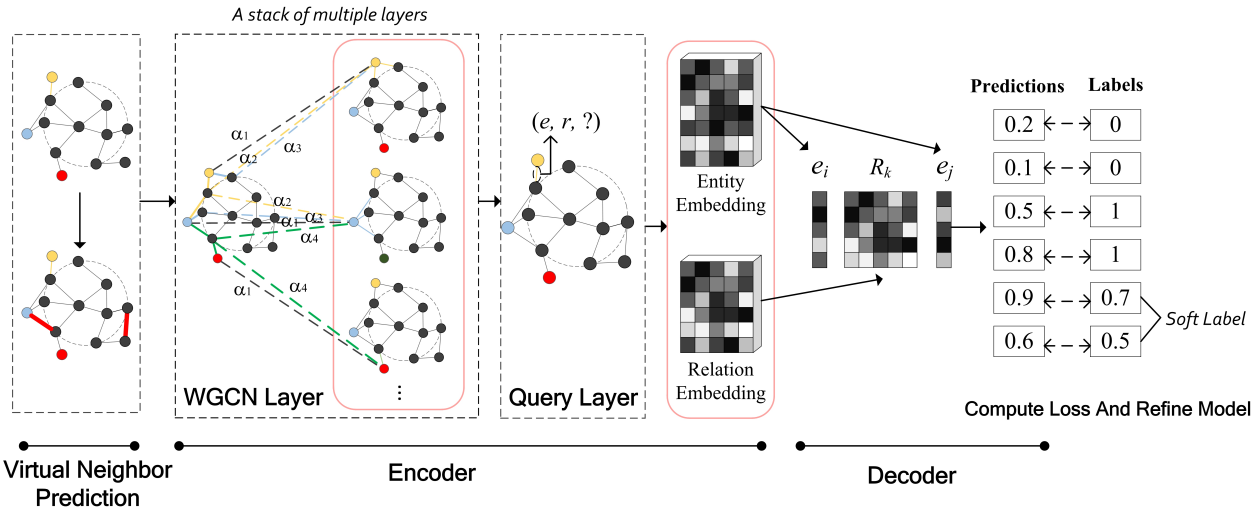


Figure 3: The framework of the VN network. VN network contains three components: the rule-based virtual neighbor prediction, the GNN-based encoder to capture structure information and embed entities, the decoder to calculate the probability of edges and then refine the model with soft labels.

$\Rightarrow$ is called the premise, and the right-hand side is conclusion. λ_p represents the confidence level of rule f_p, where $\lambda_p \in [0,1]$. The higher the confidence level of rule, the more likely it is.

3.1.3 Symmetric Path. Although logic rules can be mined through AMIE+, RLvLR or other useful rule mining methods, which can help us find the missing facts, but they still have limitations. As they mainly concentrate on one-hop or two-hop relations while there may be some long-distance dependencies helpful. Moreover, due to the sparsity problem mentioned above, we may lack neighbors to mine sufficient rules or obtain the grounding rules. Under such circumstance, **the symmetric paths** can help us find some entities that have similar contextual semantics, which contain two subpaths with the same relation order but the opposite direction. As shown in Figure 2, we start with the unseen entity "*Alec*", we can find two symmetric paths, namely, the blue one and the green one, and the end node at the other end of the path is "*Tom*" samely, but the two paths hint "*Alec*" is an actor and a resident of England respectively. Taking an example in meta-path learning [21], A-P-A (representing co-authorship relationship) and A-P-V-P-A (representing two papers published by two authors in the same venue) as meta-paths for a bibliographic graph, we can sample such node pairs and incorporate them in embeddings. But above method is usually graph-specific and highly rely on knowledge from domain experts, which requires empirical results from previous works on the same graph. Our proposed symmetric path do not have to learn such meta-path explicitly, and semantically similar entities can be found to help enrich the neighbors for unseen entities.

3.2 Model Architecture

This section briefly introduces the model architecture with three components: the rule-based virtual neighbor prediction, a GNN-based encoder, and an embedding-based decoder. In the VN network, as Figure 3 indicates, given a knowledge graph, we extract the logic and symmetric path rules to make the **virtual neighbor prediction**. We predict a soft label $s(x_{vn}) \in [0,1]$ to every unobserved triple containing virtual neighbors. To do so, we solve a convex optimization problem constrained by rules and calculate the soft labels using the current KG embeddings. As a result, the original KG becomes denser, which extremely facilitates embedding learning and predicting.

Then, the knowledge graph with virtual neighbors is inputted to the **GNN-based encoder**. The key idea of the GNN-based encoder is to collect information from the neighbors and project nodes (entities) to the continuous spaces. As mentioned in [30], modern GNNs follow the neighbor aggregation strategy, which is to update the representation of a node by aggregating representations of its neighbors repeatedly. For a multi-relational knowledge graph, we can formulate the l th layer of a GNN as:

$$\begin{aligned} a_i^{(l)} &= \text{AGGREGATE}^{(l)}(h_{r,j}^{(l-1)} : (i,r,j) \in \mathcal{O}), \\ h_i^{(l)} &= \text{COMBINE}^{(l)}(h_{r_0,i}^{(l-1)}, a_i^{(l)}), \end{aligned} \tag{1}$$

where a_i is the neighborhood aggregating information for the node i. $h_{r,j}^{(l)}$ denotes the message passing from the neighbor entity j under relation r at l th layer, $h_{r_0,i}^{(l)}$ denotes the self-connection message at l th layer, and r_0 means the self-connection relation. In this work, the encoder is composed of several **structure aware layers** and one **query aware layer**.

Since knowledge base completion task aims to predict new facts when given an incomplete knowledge graph, this task requires to determine how likely the unseen facts are true. To handle this task, the **decoder** should assign scores to the fact triples with the entity embedding e_i from the GNN-based encoder, where $e_i = h_i^L : i \in \mathcal{E}_o$. As the choice of the decoder is independent of the encoder, various methods can be used here. In this work, we mainly adopt DistMult

Table 1: An Example of KG Containing Two Types of Symmetric Path Starting from "*Bob*"

Node pair	sp_i	sp_j
(Bob, Zurich)	√	√
(Bob, Sam)	√	×
(Bob, Rome)	√	√
(Bob, Amy)	√	√
(Bob, Adam)	√	√

[32] as our decoder, and we also consider TransE [2] and ComplEX [22] decoders.

Finally, we optimize a global loss over observed facts and virtual neighbors to **refine embeddings**. In that way, we can obtain unseen entities' embeddings fitting the ground truths while satisfying the extracted rules. Note that, in the testing process, we only apply the rule-based virtual neighbor prediction and the GNN-based encoder on the auxiliary triple set $\mathcal{AUX}$, and then use the obtained embeddings to predict missing facts about newly emerging entities without refining the soft labels since the testing process is not iterative.

3.3 Virtual Neighbor Prediction

In this section, we describe how to predict the soft label $s(x_{vn})$ for each triple x_{vn} in $\mathcal{VN}$ inferred by the logic or symmetric path rules.

3.3.1 Logic and Symmetric Path Rules. For the logic rules obtained by the KG rule mining tool, we instantiate them with concrete entities to obtain the ground rules. For example, given a rule $(i, playsFor, j) \Rightarrow (i, isAffiliatedTo, j)$, we can instantiate this rule as $(Saidu, playsFor, Chiasso) \Rightarrow (Saidu, isAffiliatedTo, Chiasso)$. As there could be a huge number of groundings and our goal is to reduce the sparsity problem for unseen entities, we take as valid groundings only those where premise triples are observed in $\mathcal{O} \cup \mathcal{AUX}$ and conclusion triples that including the unseen entities in $\mathcal{E}_u$. Moreover, we only use the rule whose confidence is not less than our threshold, and the confidences of the rules are calculated directly by the AMIE+.

To get the symmetric path rule (SP rule) and its grounding, we do the following operations. Firstly, starting from one unseen entity e_i, we search all the symmetric paths $SP(e_i)$ by random walks. And if we find that the path is not symmetric, we terminate the search of this path early. Then we consider the pairwise combinations of symmetric paths, and the confidence level of them is calculated by the head coverage value [5], which mainly focus on the co-occurrence frequency of the premise and conclusion. As shown in Table 1, the confidence of the sp rule $sp_i \Rightarrow sp_j$ about "*Bob*" is 0.8, and we limit the number of sp_i to no less than 5.

For each symmetric path in the form of sp_i, we search from the other end of it. And if we can obtain the symmetric path in the form of sp_j by connecting the last edges with the unseen entity e_i, we get the grounding rule of $sp_i \Rightarrow sp_j$. In Figure 4, we add the edge "*starredIn*" to obtain the sp_j path which share the same ends of sp_i. To model the dependency between the two end entities and for brevity, we express the grounding rule of it as $(x_{first1}) \wedge (x_{last1}) \wedge (x_{last2}) \Rightarrow (x_{first2})$, where x_{first1} and x_{last1} are the first triple and last triple in sp_i, x_{last2} is the triple connect the other end of sp_j, and x_{first2} is the triple inferred as Figure 4 in sp_j. In this way, we can focus on specific entities and capture more information from long-distance nodes, to assist with logic rules to enrich the neighbors for unseen entities. We will demonstrate the effectiveness of SP rules through the experimental results.

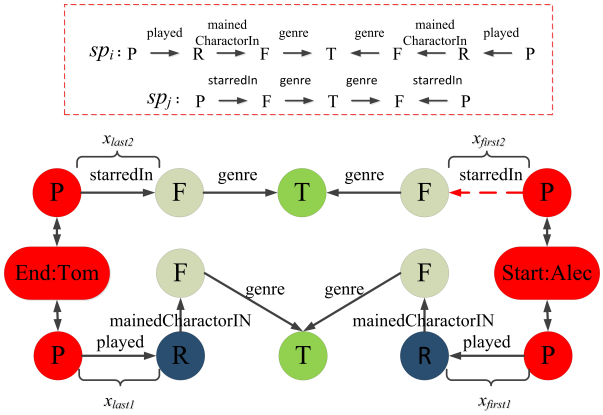


Figure 4: The example of the the symmetric path rule. P is person. R is role. F is film. And T is type. Note that for symmetric paths, we only care about the type and direction of the edges, and the letters of the nodes in the graph are just to illustrate the underlying meaning of them.

3.3.2 Soft Label Prediction. To model rules, we adopt t-norm fuzzy logics [7]. Following [6], the logical conjunction (∧), disjunction(∨), and negation(¬) are defined as follow:

$$\begin{aligned} \pi(f_1 \wedge f_2) &= \pi(f_1) \cdot \pi(f_2), \\ \pi(f_1 \vee f_2) &= \pi(f_1) + \pi(f_2) - \pi(f_1) \cdot \pi(f_2), \\ \pi(\neg f_1) &= 1 - \pi(f_1), \end{aligned} \tag{2}$$

Here, f_1 and f_2 are two logical expressions, which can either be single triples or be constructed by combining triples with logical connectives. $\pi(\cdot)$ assigns a truth value to each expression, indicating to what degree the logical expression is true. For triples, $\pi(\cdot)$ is the score function (DistMult) used in the decoder. For a conjunction of several triples, the truth value can be calculated recursively through Eq. 2. For example, for the rule $f_i \Rightarrow f_j$, the truth value can be computed as follows:

$$\pi(f_i \Rightarrow f_j) = \pi(\neg f_i \vee f_j) = \pi(f_i) \cdot \pi(f_j) - \pi(f_i) + 1. \tag{3}$$

To this end, our goal is to find a **soft label** $s(x_{vn}) \in S$ for each triple $x_{vn} \in \mathcal{VN}$, using the current KG embeddings Θ and the ground rules $\mathcal{F}_{rule}$. The optimal soft label $s(x_{vn})$ should keep close to truth value $\pi(x_{vn})$, while constrained by the ground rules. To do so, we introduce the conditional truth value $\pi(f_{rule}|S)$ for the ground rule $f_{rule} \in \mathcal{F}_{rule}$. For example, for the ground rule f_{rule} : $(e_i, r_s, e_u) \Rightarrow (e_j, r_k, e_u)$, where premise (e_i, r_s, e_u) is an known triple with unseen entity e_u, and conclusion is an unknown triple

$(e_j, r_k, e_u) \in \mathcal{VN}$ with unseen entity e_u. Then, we can calculate $\pi(f_{rule}|S)$ as:

$$\pi(f_{rule}|S) = I(e_i, r_s, e_u) \cdot s(e_j, r_k, e_u) - I(e_i, r_s, e_u) + 1, \quad (4)$$

where $I(e_i, r_s, e_u)$ is a truth value defined by the score function with the current embeddings, and $s(e_j, r_k, e_u)$ (can also be written as $s(x_{vn})$) is a soft label to be predicted for the triple containing the virtual neighbor. To get the optimal soft labels, we introduce the slack variables ξ_f for the rule f, and establish the following optimization problem:

$$\begin{aligned} &\min_{S,\xi} \frac{1}{2} \sum_{x_{vn} \in \mathcal{VN}} (s(x_{vn}) - I(x_{vn}))^2 + C \sum_{f \in F_{rule}} \xi_f, \\ &s.t. \lambda_f (1 - \pi(f|S)) \leq \xi_f, \xi_f \geq 0, 0 \leq s(x_{vn}) \leq 1, \end{aligned} \quad (5)$$

This kind of the optimization problem is convex [6]. Therefore, we can obtain the closed form solution:

$$s(x_{vn}) = \left[I(x_{vn}) + C \sum_{f \in F_{rule}} \lambda_f \nabla_{s(x_{vn})} \pi(f|S) \right]_0^1 \quad (6)$$

where C is the constant penalty parameter, and λ_f is the confidence value for the rule f, determined by the extracted algorithm. $[\cdot]_0^1 = \min(\max(x, 0), 1)$ is a truncation function. Through using currently learned embeddings and rules to predict soft labels for virtual triples, we consider the influence of embeddings on the rule-inference results rather than regarding them as necessarily true.

3.4 Encoder and Decoder

This section details the encoder and decoder in our framework.

3.4.1 Structure and Query Aware Layers. As mentioned above, our encoder consists of several structure aware layers and one query aware layer. In the first place, we adopt multiple GNN layers to map the connectivity structures of the KG into continuous spaces. To be specific, we use the weighted graph convolutional network (WGCN) [18] in the local aggregation process. Each structure aware layer assigns the different attention weights for each relation type, and the output embedding of the l th layer for the entity i can be formulated as follow:

$$\begin{aligned} a_i^{(l)} &= W^{(l)} \Big(\sum_{(i,r,j) \in O} \alpha_r^{(l)} h_j^{(l-1)} \Big), \\ h_i^{(l)} &= \tanh(a_i^{(l)} + h_i^{(l-1)} W^{(l)}), \end{aligned} \quad (7)$$

where α_r is the attention weight for relation r connected with the head entity i and tail entity j. $h_i^{(l)} \in R^{d^{(l)}}$ is the embedding of entity i at the l th layer. $W^{(l)} \in R^{d^{(l-1)} \times d^{(l)}}$ is the connection matrix for the l th layer, transforming $h_i^{(l-1)}$ to $h_i^{(l)}$. We randomly initialize the input entity embedding $h_i^{(0)}$, and stack multiple structure aware layers before the query aware layer.

Besides the common structure information in the KG, given the query relation (relation of the inputted triple), an ideal aggregator is able to focus on the relevant facts in the neighborhood [23]. To exploit the query-relevant information, we construct a query aware layer based on the neural network mechanism [23]. Specifically, given a query relation $q \in \mathcal{R}$, the importance of the neighbor j to entity i is calculated as follow:

$$\alpha_{j|i,q}^{\text{NN}} = \text{softmax}(\beta_{j|i,q}) = \frac{\exp(\beta_{j|i,q})}{\sum_{(i,r,j') \in O} \exp(\beta_{j'|i,q})}, \quad (8)$$

where the unnormalized attention weight $\beta_{j|i,q}$ can be computed by the following neural network:

$$\beta_{j|i,q} = \text{LeakyReLU}(u \cdot [W_e h_i; W_q z_q; W_e h_j]), \quad (9)$$

where $u \in R^{3d}$, W_e and $W_q \in R^{d \times d}$ are the global attention parameters, while $z_q \in R^d$ is a relation-specific attention parameter. Then, given the query relation q, the whole query aware layer can be defined as:

$$h_i^O = \sum_{(i,r,j) \in O} \alpha_{j|i,q}^{\text{NN}} \cdot h_j^I, \quad (10)$$

where $h_j^I = h_j^{(L)}$ is the embedding of the entity j from the last structure aware layer. $h_i^O = e_i$ is the output embedding of the entity i to the decoder. When obtaining embeddings for unseen entities, we apply the encoder on auxiliary and virtual triples by simply replacing all the observed triples $(i, r, j) \in O$ in Eq. 7, 8 and 10 with the triples $(i', r', j') \in \mathcal{AUX} \cup \mathcal{VN}$.

3.4.2 DistMult Decoder. Then we use DistMult as our decoder to assign scores for the fact triples with the entity embedding h_i^O output from the GNN-based encoder:

$$I(e_i, r_k, e_j) = e_i^T R_k e_j, \quad (11)$$

where $I(\cdot)$ is the score function. $e_i, e_j \in \Theta$ are the entity embeddings, and R_k is a diagonal matrix for the relation $r_k \in \mathcal{R}$.

Algorithm 1 Procedure of VN Network

Require:
The positive triples and negative triples $\mathcal{L} = \{(x_l, y_l)\}$;
The triples inferred by rules that containing virtual neighbors $\mathcal{VN} = \{x_{\mathcal{VN}}\}$;
The trained encoder and decoder on the original KG;
The iteration number N;

```
repeat
  for each mini-batch L_b, VN_b do
    Compute the soft labels for VN_b using Eq. 6;
    Input L_b, VN_b into encoder;
    Get the scores for L_b and VN_b from decoder;
    Minimize the global loss using Eq. 12;
    n ← n + 1
  end for
until n < N
```

Ensure: Θ^N

3.5 Training Objective

Algorithm 1 summarizes the learning procedure of our method. We randomly corrupt the head or tail entity of a positive triple to form negative triples. We are given a hard labeled set $\mathcal{L} = \{(x_l, y_l)\}$, each positive or negative triple in it has a hard label $y_l \in \{0, 1\}$. We also collect the set of virtual triples with soft labels, i.e., $\mathcal{VN} = \{x_{\mathcal{VN}}\}$. Our goal is to learn the optimal KG embeddings Θ^N with both

Table 2: The Statistics of Three Datasets

Dataset	Entities	Relations	Training	Validation	Test
WordNet11	38,696	11	112,581	5,218	21,088
FB15k	14,951	1,345	483,142	50,000	59,071
YAGO37	123,189	37	989,132	50,000	50,000

hard labeled and soft labeled triples. To do so, we establish a loss function over $\mathcal{L}$ and $\mathcal{VN}$ as follow:

$$\min_{\Theta} \frac{1}{|\mathcal{L}|} \sum_{\mathcal{L}} l(I(x_l), y_l) + \frac{1}{|\mathcal{VN}|} \sum_{\mathcal{VN}} l(I(x_{vn}), s(x_{vn})), \quad (12)$$

where we adopt the cross entropy $l(x, y) = -y \log x - (1 - y) \log(1 - x)$. $I(\cdot)$ is the score function defined in Eq.11. We use the ADAM algorithm [12] to minimize the global loss function. In this way, the resultant embeddings of unseen entities fit the newly emerging facts while constrained by rules.

4 EXPERIMENTS

In this section, we evaluate our proposed framework, VN network, in two KBC tasks: triple classification and link prediction. We mainly compare our model with three baselines, LSTM and LAN in [23], and MEAN in [8]. And we mainly evaluated our method from the following perspectives, 1)whether our model can learn better embeddings for unseen entities than the above methods with the neighbor sparsity problem, 2)whether each component of our framework is useful for the learning of embeddings.

Since the link prediction is a common task for knowledge graph completion which considers the ranks of all entities, we do further analysis on the link prediction task.

4.1 Datasets

We evaluate VN network on three datasets: WordNet11 [20], FB15k [2] and YAGO37 [6]. WordNet11 is a subset of WordNet, which is a database of lexical relations between words. FB15K is a subset of the multi-relational knowledge base Freebase. And YAGO37 is extracted from the core facts of YAGO3 where entities appearing less than 10 times are discarded. Table 2 summarizes the detail statistics of the above datasets.

For the triple classification task, we directly use the datasets released in [8] based on WordNet11, including Subject-{1000, 3000, 5000}, Object-{1000, 3000, 5000} and Both-{1000, 3000, 5000}. For the link prediction task, the published data sets of FB15K differ from what is written in [23], and there is no public dataset available on YAGO37 under the unseen entities setting. So we use the datasets on FB15K that they publish, and construct the datasets on YAGO37 by following the similar protocol mentioned in [23], including Subject-{5, 10, 15, 20, 25} and Object-{5, 10, 15, 20, 25}. The basic idea of the dataset construction is to randomly sample triples from test set, and select entities from these triples as candidate unseen entities. Then using these entities to split the training set. The specific process is as follows:

Sampling unseen entities. Firstly, $R = 5\%, 10\%, 15\%, 20\%, 25\%$ triples are randomly sampled from the original FB15K(YAGO37) test set as candidate test set. As for WordNet11, $N = \{1000, 3000, 5000\}$ testing triples are sampled. And Subject is the strategy that used to construct the candidate unseen entities sets $\mathcal{E}_u$, where only entities appearing as the head entities in the candidate test set are added to $\mathcal{E}_u$. The same goes for Both and Tail, where tail entities or both the head and tail entities are added to $\mathcal{E}_u$. Then entities in $\mathcal{E}_u$ that do not have any neighbors in the original training set are filtered. For a triple (e_i, r_k, e_j) in the candidate test set, if $e_i \notin \mathcal{E}_u \wedge e_j \notin \mathcal{E}_u$, it is removed from the candidate test set. The new test sets $\mathcal{T}$ are obtained after filtering.

Filtering and splitting data sets. The second step is to ensure that unseen entities would not appear in the final training set or validation set. The original training set are split into two data sets, the new training set $\mathcal{O}$ and auxiliary set $\mathcal{AUX}$. For a triple (e_i, r_k, e_j) in original training set, if $e_i \in \mathcal{E}_o \wedge e_j \in \mathcal{E}_o$, it is added to the new training set $\mathcal{O}$. If $e_i \in \mathcal{E}_o \wedge e_j \in \mathcal{E}_u$ or $e_i \in \mathcal{E}_u \wedge e_j \in \mathcal{E}_o$, it is added to the auxiliary set $\mathcal{AUX}$, which serves as existing neighbors for unseen entities in the aggregating process. For the new validation set, we keep only triples that have no unseen entities.

We employ AMIE+ [5] to extract the logic rules on all the datasets. We only keep the logic rules with the length not longer than 2 and the confidence not less than 0.8. Besides, through the use of random walks and drool rule engine tool [13], we extract symmetric paths with length 2, 4 and 6, and also keep the symmetric path rules with the confidence not less than 0.8. Then we enrich the neighbors for unseen entities with these rules. We omit the virtual neighbors that overlap with the existing neighbors, and take the one with the highest confidence when the virtual neighbors conflict with themselves. The detail statistics of WordNet11 can be found in [8]. And Table 3 summarize the detail statistics of FB15K and YAGO37. Since most of the tail entities of YAGO37 are locations or organizations which have many connected edges, so the subject-R and object-R are not balanced as FB15K. We can see that the average ratio of neighbors to predicted facts of unseen entities has increased by at least five after the supplement, which helps to solve the sparsity problem.

4.2 Triple Classification Task

This task is to classify every test triple as true or false. To tackle this task, we need to set a threshold δ_r for each relation r. The triple $x = (e_i, r, e_j)$ is predicted to be true when $I(x) \geq \delta_r$, otherwise the triple is false. We determine the optimal δ_r by maximizing classification accuracy on the validation set.

4.2.1 Experimental Setup. We randomly sample 64 negative triples for each triple in $\mathcal{O} \cup \mathcal{AUX}$. For all the datasets, we create 100 mini-batches on each dataset, and we conduct a grid search to find hyperparameters that maximize accuracy on the validation sets in at most 100 iterations. The embedding dimensions for the encoder and decoder are all set to 200. The learning rate is 0.002. The dropout rate during training is 0.3. The regularization penalty coefficient on KG embeddings is 0.001. The constant penalty C is 0.01. We employ three structure aware layers and one query aware layer in the encoder, and DistMult in the decoder. Our models are implemented by PyTorch and run on NVIDIA TITAN RTX Graphics Processing Units.

Table 3: Statistics of Processed FB15K and YAGO37 Dataset

	FB15K							YAGO37						
Dataset	$\mathcal{O}$	$\mathcal{AUX}$	$\mathcal{E}_u$	Valid	Test	$avg-\mathcal{R}_{before}$	$avg-\mathcal{R}_{after}$	$\mathcal{O}$	$\mathcal{AUX}$	$\mathcal{E}_u$	Valid	Test	$avg-\mathcal{R}_{before}$	$avg-\mathcal{R}_{after}$
Subject-5	188,238	235,746	1,465	19,454	1,834	85.54	125.89	929,037	59,453	2,408	49,972	2,458	23.79	35.04
Object-5	170,672	254,454	1,344	17,712	1,896	88.66	129.51	568,900	269,988	1,774	49,933	2,484	140.22	146.71
Subject-10	108,854	249,798	2,102	11,339	2,811	52.33	73.31	899,110	87,868	4,707	49,861	4,925	17.26	27.92
Object-10	99,783	261,341	1,947	10,190	2,987	53.63	74.58	441,916	382,082	3,188	49,815	4,926	92.58	99.61
Subject-15	71,407	228,484	2,358	7,310	3,250	37.14	50.36	848,034	137,255	6,846	49,752	7,299	17.43	22.49
Object-15	67,651	243,316	2,228	6,878	3,703	39.00	52.65	380,053	440,759	4,499	49,699	7,379	67.56	75.31
Subject-20	49,456	205,242	2,571	5,048	3,586	29.19	38.37	812,019	170,536	8,838	49,614	9,601	15.93	26.36
Object-20	46,982	222,200	2,388	4,843	4,132	30.92	40.59	332,204	488,332	5,884	49,556	9,745	55.25	63.71
Subject-25	37,986	179,656	2,704	3,908	3,889	22.53	29.12	781,209	200,073	10,897	49,526	12,048	15.76	26.18
Object-25	34,126	195,627	2,470	3,498	4,283	24.92	32.01	297,655	523,251	7,234	49,452	12,193	45.69	51.37

Table 4: Evaluation Accuracy on Triple Classfication(%)

	Subject			Object			Both		
Model	1000	3000	5000	1000	3000	5000	1000	3000	5000
MEAN	87.3	84.3	83.3	84.0	75.2	69.2	83.0	73.3	68.2
LSTM	87.0	83.5	81.8	82.9	71.4	63.1	78.5	71.6	65.8
LAN	88.8	85.2	84.2	84.7	78.8	74.3	83.3	76.9	70.6
VN network	**89.1**	**85.9**	**85.4**	**85.5**	**80.6**	**76.8**	**84.1**	**78.5**	**73.1**

4.2.2 Results. The detailed results are shown in Table 4. We can see that VN network achieves the best performance over all the datasets, and most of the improvements over other baselines are significant. The results demonstrate that embedding newly emerging entities with virtual neighbors indeedly improves the quality of KG embeddings. Note that the improvements on Subject-{1000,3000,5000} are not considerable as the other two groups. It may be because the Subject-*R* datasets are much easier to classify than the other two groups, the performances are more difficult to improve. Interestingly, with the number of unseen entities increase, the improvement is more obvious, which proves that our method is more effective in dealing with more unseen entities.

4.3 Link Prediction Task

Link prediction task aims at completing every test triple with subject or object missing. For example, in Subject-*R*(Object-*R*) data sets, we firstly hide the object(subject) of each testing triple to produce a missing part. Then we replace the missing part with all entities to construct candidate triples. We compute the scoring function defined in Eq. 11 for all candidate triples, and rank them in descending order. Finally, we evaluate whether the ground-truth entities are ranked ahead of other entities. We use three widely used metrics: *mean rank* (MR), *mean reciprocal rank* (MRR), *Hits at n* (Hits@n). And for this ranking process, we remove corrupted triples which is called "filtered" setting. We report filtered MRR and Hits at 1, 3 and 10.

4.3.1 Experimental Setup. In this task, we finetune the embedding dimension d in {100, 150, 200, 250, 300}, the dropout rate α in {0.1, 0.15, 0.2, 0.25, 0.3, 0.4}, the learning rate β in {0.001, 0.002, 0.005, 0.01, 0.1} with a grid search. The optimal configurations are d = 100, α = 0.25 and β = 0.002 for FB15K. And for YAGO37, the optimal configurations are d = 200, α = 0.2 and β = 0.002. Other hyperparameters are set the same as those in the triple classification task.

4.3.2 Results. As Table 5 shows, VN network outperforms all the baselines. Compared to the best performing baseline LAN, our model achieves an improvement of 13.5% in HITS@10 on FB15K, and an improvement of 21.4% on YAGO37 under the setting of Subject. And our model achieves an improvement of 14.5% in HITS@10 on FB15K, and an improvement oft 13.9% on YAGO37 under the setting of Object. The link prediction results demonstrate the superiority of our proposed framework again. To explore whether each component is useful in our VN network and generalizes to other configurations, we will do further analysis on the link prediction task in the following.

Ablation Study: To investigate the effectiveness of the components in our framework. We conduct the link prediction task on several variants of our method, as Table 6 describes. First, we only employ structure aware layers, which weighs the different types of relations differently. Then, without the soft label prediction process, we directly inject the hard rules which always hold with no exception. Next, to ensure the necessity of the soft label prediction, we adopt the soft logic rules setting. Besides, we add the symmetric path rules for the logic and SP rules setting, to ensure whether using long-distance dependencies between entities can be helpful. Finally, we evaluate our method with the query aware layer.

As Table 6 shows, even if we only use the structure aware layers, our method still outperforms the simple MEAN aggregator in most cases, which indicates that the structure aware layers are able to effectively encode the connectivity structure information.

Meanwhile, We can see two significant improvements, that is due to the use of hard labeled neighbors inferred by hard rules and iteratively soft labeled virtual neighbors. The first significant improvement strongly demonstrates the sparsity problem of unseen entities, and we can learn more expressive embeddings for the unseen entities by enriching the neighbors. The second significant improvement shows the importance of adding soft labeled virtual neighbors in an iterative manner. Because when we add virtual neighbors for unseen entities, we also introduce noise. Therefore, through considering the interactions between KG embeddings and rules, we can obtain embeddings that are more accurate and adapt to the rules.

Table 5: The Link Prediction Results

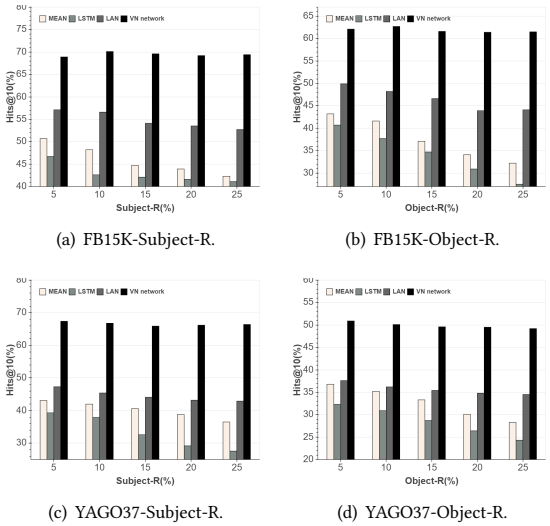
	FB15K										YAGO37									
	Subject-10					Object-10					Subject-10					Object-10				
Model	MR	MRR	Hits@10	Hits@3	Hits@1	MR	MRR	Hits@10	Hits@3	Hits@1	MR	MRR	Hits@10	Hits@3	Hits@1	MR	MRR	Hits@10	Hits@3	Hits@1
MEAN	293	31.0	48.0	34.8	22.2	353	25.1	41.0	28.0	17.1	2393	21.5	42.0	24.2	17.8	4763	17.8	35.2	17.5	12.1
LSTM	353	25.4	42.9	29.6	16.2	504	21.9	37.3	24.6	14.3	3148	19.4	37.9	20.3	15.9	5031	14.2	30.9	16.1	11.8
LAN	263	39.4	56.6	44.6	30.2	461	31.4	48.2	35.7	22.7	1929	24.7	45.4	26.2	19.4	4372	19.7	36.2	19.3	13.2
VN network	**175**	**46.3**	**70.1**	**52.6**	**34.5**	**212**	**42.3**	**62.7**	**44.6**	**28.2**	**1757**	**46.5**	**66.8**	**53.8**	**35.7**	**3145**	**27.4**	**50.1**	**36.4**	**19.5**

Table 6: Effectiveness of Each Component on Subject-10

Model	MR	MRR	Hits@10	Hits@3	Hits@1
Structure aware	291	34.2	56.7	40.3	24.8
Hard rules	237	40.6	64.2	46.5	30.3
Logic rules	218	43.9	66.2	50.6	32.3
Logic and SP rules	185	45.9	69.8	51.9	33.6
VN network	**175**	**46.3**	**70.1**	**52.6**	**34.5**

Table 7: Different Decoders on Subject-10

Encoder	Decoder	MRR	Hits@10	Hits@3	Hits@1
MEAN	ComplEx	28.6	44.7	32.2	20.4
MEAN	TransE	31.0	48.0	34.8	22.2
MEAN	DisMult	29.7	45.8	33.5	21.2
LAN	ComplEx	37.1	53.1	42.2	28.7
LAN	TransE	39.4	56.6	44.6	30.2
LAN	DisMult	37.8	53.4	43.2	29.3
VN	ComplEx	39.6	60.1	45.4	29.7
VN	TransE	43.5	66.7	49.4	31.6
VN	DisMult	46.3	70.1	52.6	34.5



(a) FB15K-Subject-R. (b) FB15K-Object-R.

(c) YAGO37-Subject-R. (d) YAGO37-Object-R.

Figure 5: Link prediction results on Subject/Object-R.

Although the use of symmetric path rules does not improve as the two components above, the improvement proves that the representational power of the original embeddings can be improved by introducing the long-distance dependencies between entities. Our proposed framework achieves the best result by the use of the query aware layer, and the main improvements are Hits@1 and Hits@3, because it can exploit the query-relevant information which helps to focus on more relevant facts in the neighborhood.

Impact of Other Decoders: To find out the influence of the different decoders, we consider three typical embedding methods here: DistMult, TransE and ComplEx. The LSTM is still inferior to MEAN as described in [23], so we also omit the results of LSTM. The results are reported in Table 7. We can see that our method outperforms MEAN consistently by a large margin on all the evaluation metrics. And as for the LAN, VN network performs better on most metrics. VN network with the ComplEx decoder results in the worst performance, due to the high parameter complexity. In contrast, VN network with the DistMult or TransE decoder can achieve the state-of-the-art results, while the DistMult leads to the best result. The experiment results show that the superiority of our model to the baselines can generalize to other scoring functions and learn more expressive embeddings for unseen entities.

Impact of the Percentage of Unseen Entities: In order to investigate the impact of the percentage of unseen entities, we conduct experiment under the setting of the Subject-*R* and Object-*R* on the FB15K and YAGO37 datasets. And it seems reasonable that with the ratio of the unseen entities over the training entities increases (namely the observed knowledge graph becomes sparser), the accuracy would decline.

As Figure 5 shows, our method and LAN are much better than MEAN and LSTM. And with the percentage of the unseen entities increasing and the KG becoming spaser, VN network still achieves better results than other baselines on all data sets. We observe that the increasing proportion of unseen entities certainly has a negative impact on all models because of the sparsity problem as mentioned above, especially MEAN and LSTM can not learn effective embeddings as LAN and VN network when the proportion of unseen entities is high. But the improvements of our model are relatively stable. And on the YAGO37 data sets, there only drops of less than 2% from 5 percent to 25 percent in our model, while LAN drops more obvious. We can conclude that when unseen entities appear, with the iterative guidance from virtual neighbors, our framework can reduce the sparsity problem and accurately predict the missing facts of the unseen entities to learn more expressive embeddings.

5 CONCLUSIONS

In this paper, we discuss the problems of the KG embedding task under the setting of unseen entities. And we propose a novel framework, VN network, to address the unseen entity problems. To handle the sparsity problem, we introduce the short logic rules and symmetric path rules to capture more information and enrich the neighbors for unseen entities. We also use three structure aware layers and one query aware layer, which can adapt the amount of information from neighbors used in local aggregation and concentrate on more relevant information, leading to more accurate embeddings of unseen entities. And through introducing the concept of the virtual neighbor and employ a rule-based prediction algorithm to assign soft labels using the current KG embeddings, we consider the interactions between the rule predictions and KG embedding learning rather than making a one-time injection of logic rules. Experimental results show that VN network achieves improvements over state-of-the-art baselines.

ACKNOWLEDGMENTS

This work was supported by the National Key R&D Program with No.2016QY03D0503,2016YFB081304, and Strategic Priority Research Program of Chinese Academy of Sciences, Grant No.XDC02040400.

REFERENCES

[1] Kurt D. Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *SIGMOD*. 1247–1250. https://doi.org/10.1145/1376616.1376746

[2] Antoine Bordes, Nicolas Usunier, Alberto García-Durán, Jason Weston, and Oksana Yakhnenko. 2013. Translating Embeddings for Modeling Multi-relational Data. 2787–2795. http://papers.nips.cc/paper/5071-translating-embeddings-for-modeling-multi-relational-data

[3] Boyang Ding, Quan Wang, Bin Wang, and Li Guo. 2018. Improving Knowledge Graph Embedding Using Simple Constraints. In *ACL*. 110–121. https://doi.org/10.18653/v1/P18-1011

[4] Xin Dong, Evgeniy Gabrilovich, Geremy Heitz, Wilko Horn, Ni Lao, Kevin Murphy, Thomas Strohmann, Shaohua Sun, and Wei Zhang. 2014. Knowledge vault: a web-scale approach to probabilistic knowledge fusion. In *SIGKDD*. 601–610. https://doi.org/10.1145/2623330.2623623

[5] Luis Galárraga, Christina Teflioudi, Katja Hose, and Fabian M. Suchanek. 2015. Fast rule mining in ontological knowledge bases with AMIE+. *VLDB J.* 24, 6 (2015), 707–730. https://doi.org/10.1007/s00778-015-0394-1

[6] Shu Guo, Quan Wang, Lihong Wang, Bin Wang, and Li Guo. 2018. Knowledge Graph Embedding With Iterative Guidance From Soft Rules. In *AAAI*. 4816–4823. https://www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/16369

[7] Petr Hájek. 1998. *Metamathematics of Fuzzy Logic*. Trends in Logic, Vol. 4. Kluwer. https://doi.org/10.1007/978-94-011-5300-3

[8] Takuo Hamaguchi, Hidekazu Oiwa, Masashi Shimbo, and Yuji Matsumoto. 2017. Knowledge Transfer for Out-of-Knowledge-Base Entities : A Graph Neural Network Approach. In *IJCAI*. 1802–1808. https://doi.org/10.24963/ijcai.2017/250

[9] William L. Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive Representation Learning on Large Graphs. In *NIPS*. 1024–1034. http://papers.nips.cc/paper/6703-inductive-representation-learning-on-large-graphs

[10] Guoliang Ji, Shizhu He, Liheng Xu, Kang Liu, and Jun Zhao. 2015. Knowledge Graph Embedding via Dynamic Mapping Matrix. In *ACL*. 687–696. https://www.aclweb.org/anthology/P15-1067/

[11] Shaoxiong Ji, Shirui Pan, Erik Cambria, Pekka Marttinen, and Philip S. Yu. 2020. A Survey on Knowledge Graphs: Representation, Acquisition and Applications. *CoRR* abs/2002.00388 (2020). arXiv:2002.00388 https://arxiv.org/abs/2002.00388

[12] Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *ICLR*, Yoshua Bengio and Yann LeCun (Eds.). http://arxiv.org/abs/1412.6980

[13] Dingcheng Li. 2012. Applying JBoss® Drools Business Rules Management System for Electronic Health Records Driven Phenotyping. http://knowledge.amia.org/amia-55142-a2012a-1.636547/t-006-1.640361/f-001-1.640362/a-245-1.640478/a-246-1.640475

[14] Arvind Neelakantan, Benjamin Roth, and Andrew McCallum. 2015. Compositional Vector Space Models for Knowledge Base Completion. In *ACL*. 156–166. https://www.aclweb.org/anthology/P15-1016/

[15] Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. 2011. A Three-Way Model for Collective Learning on Multi-Relational Data. In *Proceedings of the 28th International Conference on Machine Learning, ICML 2011, Bellevue, Washington, USA, June 28 - July 2, 2011*. 809–816. https://icml.cc/2011/papers/438_icmlpaper.pdf

[16] Pouya Ghiasnezhad Omran, Kewen Wang, and Zhe Wang. 2018. Scalable Rule Learning via Learning Representation. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13-19, 2018, Stockholm, Sweden*. ijcai.org, 2149–2155. https://doi.org/10.24963/ijcai.2018/297

[17] Evgenia Wasserman Pritsker, William W. Cohen, and Einat Minkov. 2015. Learning to Identify the Best Contexts for Knowledge-based WSD. In *EMNLP*. 1662–1667. https://www.aclweb.org/anthology/D15-1192/

[18] Chao Shang, Yun Tang, Jing Huang, Jinbo Bi, Xiaodong He, and Bowen Zhou. 2019. End-to-End Structure-Aware Convolutional Networks for Knowledge Base Completion. In *AAAI*. 3060–3067. https://doi.org/10.1609/aaai.v33i01.33013060

[19] Baoxu Shi and Tim Weninger. 2018. Open-World Knowledge Graph Completion. In *AAAI*. 1957–1964. https://www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/16055

[20] Richard Socher, Danqi Chen, Christopher D. Manning, and Andrew Y. Ng. 2013. Reasoning With Neural Tensor Networks for Knowledge Base Completion. 926–934.

[21] Yizhou Sun, Jiawei Han, Xifeng Yan, Philip S. Yu, and Tianyi Wu. 2011. PathSim: Meta Path-Based Top-K Similarity Search in Heterogeneous Information Networks. *Proc. VLDB Endow.* 4, 11 (2011), 992–1003. http://www.vldb.org/pvldb/vol4/p992-sun.pdf

[22] Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. 2016. Complex Embeddings for Simple Link Prediction. In *ICML*. 2071–2080. http://proceedings.mlr.press/v48/trouillon16.html

[23] PeiFeng Wang, Jialong Han, Chenliang Li, and Rong Pan. 2019. Logic Attention Based Neighborhood Aggregation for Inductive Knowledge Graph Embedding. In *AAAI*. 7152–7159. https://doi.org/10.1609/aaai.v33i01.33017152

[24] Quan Wang, Zhendong Mao, Bin Wang, and Li Guo. 2017. Knowledge Graph Embedding: A Survey of Approaches and Applications. *IEEE TKDE* 29, 12 (2017), 2724–2743. https://doi.org/10.1109/TKDE.2017.2754499

[25] Zihan Wang, Zhaochun Ren, Chunyu He, Peng Zhang, and Yue Hu. 2019. Robust Embedding with Multi-Level Structures for Link Prediction. In *IJCAI*. 5240–5246. https://doi.org/10.24963/ijcai.2019/728

[26] Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014. Knowledge Graph Embedding by Translating on Hyperplanes. In *AAAI*. 1112–1119. http://www.aaai.org/ocs/index.php/AAAI/AAAI14/paper/view/8531

[27] Tianxing Wu, Arijit Khan, Huan Gao, and Cheng Li. 2019. Efficiently Embedding Dynamic Knowledge Graphs. *CoRR* abs/1910.06708 (2019). arXiv:1910.06708 http://arxiv.org/abs/1910.06708

[28] Han Xiao, Minlie Huang, Lian Meng, and Xiaoyan Zhu. 2017. SSP: Semantic Space Projection for Knowledge Graph Embedding with Text Descriptions. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA*. 3104–3110. http://aaai.org/ocs/index.php/AAAI/AAAI17/paper/view/14306

[29] Ruobing Xie, Zhiyuan Liu, Huanbo Luan, and Maosong Sun. 2017. Image-embodied Knowledge Representation Learning. In *IJCAI*. 3140–3146. https://doi.org/10.24963/ijcai.2017/438

[30] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2019. How Powerful are Graph Neural Networks?. In *ICLR*. https://openreview.net/forum?id=ryGs6iA5Km

[31] Bishan Yang and Tom M. Mitchell. 2017. Leveraging Knowledge Bases in LSTMs for Improving Machine Reading. In *ACL*. 1436–1446. https://doi.org/10.18653/v1/P17-1132

[32] Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. 2015. Embedding Entities and Relations for Learning and Inference in Knowledge Bases. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*. http://arxiv.org/abs/1412.6575

[33] Wen Zhang, Bibek Paudel, Liang Wang, Jiaoyan Chen, Hai Zhu, Wei Zhang, Abraham Bernstein, and Huajun Chen. 2019. Iteratively Learning Embeddings and Rules for Knowledge Graph Reasoning. In *The World Wide Web Conference, WWW 2019, San Francisco, CA, USA, May 13-17, 2019*. 2366–2377. https://doi.org/10.1145/3308558.3313612

[34] Zhao Zhang, Fuzhen Zhuang, Meng Qu, Fen Lin, and Qing He. 2018. Knowledge Graph Embedding with Hierarchical Relation Structure. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*. 3198–3207. https://doi.org/10.18653/v1/d18-1358

[35] Yukun Zuo, Quan Fang, Shengsheng Qian, Xiaorui Zhang, and Changsheng Xu. 2018. Representation Learning of Knowledge Graphs with Entity Attributes and Multimedia Descriptions. In *Fourth IEEE International Conference on Multimedia Big Data, BigMM 2018, Xi'an, China, September 13-16, 2018*. 1–5. https://doi.org/10.1109/BigMM.2018.8499179