Neurocomputing 491 (2022) 186–196

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

SLAN: Similarity-aware aggregation network for embedding out-of-knowledge-graph entities

Mingda Li [a,b], Zhengya Sun [a,b,*], Wensheng Zhang [a,b]

[a] University of Chinese Academy of Sciences (UCAS), Beijing 100049, China
[b] Research Center of Precision Sensing and Control, Institute of Automation, Chinese Academy of Sciences, Beijing 100190, China

ARTICLE INFO

Article history:
Received 29 October 2021
Revised 23 January 2022
Accepted 20 March 2022
Available online 23 March 2022

Keywords:
Knowledge graph embedding
Out-of-knowledge-graph entities
Knowledge graph completion
Similarity search

ABSTRACT

Knowledge graph embedding acts as a pivotal role in predicting the missing information in knowledge graphs (KGs). Due to the evolving nature of real-world KGs, one requires the ability to make predictions for newly emerging entities besides those already observed at training time. Current studies have made great efforts to develop a neighborhood aggregator and embed out-of-knowledge-graph (OOKG) entities inductively, with less focus on exploiting the similarity between the existing and newly emerging entities. Attaching importance to such similarity helps facilitate semantic transfer. In this work, we propose a similarity-aware aggregation network for embedding out-of-knowledge-graph entities. Motivated by the fact that similar entities are likely to occur in common graph context, we skillfully design a similarity-aware function, which measures the distance of each entity pair based on the contextual gap. Moreover, we aggregate the neighborhood surrounding the target entity and its similarity information by query-specific attention weights, which are optimized during the learning process. Extensive experiments on knowledge graph completion task show that our method achieves substantial improvements over baselines.

1. Introduction

Knowledge graphs (KGs) have recently become an effective tool for a wide range of applications such as machine translation [1], question answering [2] and recommendation [3] [4]. A typical KG may contain billions of facts in the form of triples, which is usually far from complete [5]. It becomes necessary to conduct knowledge graph completion for enhancing its utility for downstream applications [6]. In the real-world scenario, KGs often evolve fast with out-of-knowledge-graph (OOKG) entities added frequently. As reported in [7], around 200 OOKG entities emerge every day on the knowledge graph DBpedia [8]. In this sense, one should predict the missing triples not only about the existing entities, but also about the newly emerging ones.

Knowledge graph embedding has proven to be useful in handling the prediction task with the ever-increasing scale of KGs. The basic idea is to encode entities and relations in a low-dimensional space, which allows complex vector operations [6]. Most methods are designed in a transductive manner, where all test entities are assumed to be available in training time. Once new entities arrive, they need to retrain from scratch. This may induce substantially increased overhead due to frequently added entities [9]. To relieve this issue, some researchers resort to an inductive manner, where new entities are inferred according to the neighborhood properties in KGs. The first attempt [9] introduces the graph neural network (GNN) on the KG, which generates the embeddings of new entities by aggregating all their available neighbors. A related but different method [10] incorporates well-designed aggregation functions into transductive models so as to guide the inductive learning procedure towards reasonable solutions. A better method [11] distinguishes the importance of neighborhood, and improves the aggregator with attention mechanisms, where the attention weights are estimated by either logic rules or a neural network. A recent, state-of-the-art method [12] further simplifies the attention weight estimation and conceives a more efficient OOKG embedding framework.

As far as we know, while much attention has been given to effectively aggregate the neighborhood information, there are few attempts in the literature to make the aggregator more powerful by incorporating the similarity between the existing and newly emerging entities. The values in adding such similarity for predicting missing facts are as follows. On the one hand, compared to observed entities, newly emerging entities often have few connec-

* Corresponding author.
E-mail address: zhengya.sun@ia.ac.cn (Z. Sun).

https://doi.org/10.1016/j.neucom.2022.03.063
0925-2312/© 2022 Elsevier B.V. All rights reserved.

tions, without enough information to obtain sound representations. On the other hand, the graph context between two entities may possess similarity to some extent, which can be an important source of evidence for relating an existing entity to an unseen entity. Exploiting such similarity helps facilitate semantic transfer and benefit the prediction about unseen entities. As shown in Fig. 1, suppose A.Wiggins is an OOKG entity, and only appears in two facts, i.e., (A.Wiggins, profession, Athlete) and (A.Wiggins, son_of, M.Payne). According to the similarities between their graph context, A.Wiggins is semantically related to two existing entities, e.g., C.Parker and S.Curry, with different intensities. Here, by an intuitive comparison, we find that the existing entity S.Curry is more similar to A.Wiggins, due to the fact that they have an additional common edge (son_of). This implies that more semantic of S. Curry would be transferred to the newly emerging entity, which helps infer the missing fact (A.Wiggins, play_in, NBA).

Thus, in this paper, we propose a similarity-aware aggregation network SLAN for OOKG entity embeddings. This is motivated by the fact that similar entities are likely to occur in common graph context. Specifically, we conceive a similarity-aware function, which efficiently measures the distance of each entity pair by comparing their neighbor context and edge context. In this sense, given a target entity, we are allowed to retrieve an entity similarity list with different intensities. We then characterize the influence of the neighborhood surrounding each target entity and its similarity information by query-specific attention weights, which can be optimized during the learning process. When predicting the missing facts about the OOKG entities, we exploit the embeddings obtained by aggregating the neighborhood along with the similar counterparts in the learned aggregation network. Experimental studies indicate the superiority of our method compared to the state-of-the-art baselines on the task of knowledge graph completion.

Our contributions are three folds:

- We propose a novel aggregation network for embedding OOKG entities, which can jointly leverage the similarity and neighborhood information.
- We propose a simple yet powerful similarity measure, which can efficiently obtain the similarity between entities based on their neighbor context and edge context.
- The experimental results validate that our method can achieve substantial improvements over existing state-of-the-art methods, especially when there exist few neighbors surrounding the OOKG entities.

This paper is organized as follows. In Section 2, we survey and categorize the related studies. In Section 3, we present our similarity-aware aggregation network SLAN. In Section 4, we conduct experiments to test our method. Finally, we conclude our paper in Section 5.

2. Related work

In this section, we will introduce some representative works that are most related to our study.

2.1. Transductive embedding methods

Generally, transductive embedding methods apply different principles to design the score functions [13] [14], which can be roughly categorized into three groups: translational model, bilinear model and neural network model. TransE [15] is the basic translational model that translates the subject entity embedding $\mathbf{e}_s$ to the object entity embedding $\mathbf{e}_o$ by the relation embedding $\mathbf{r}$, formally as $\mathbf{e}_s + \mathbf{r} \approx \mathbf{e}_o$. There are many extensions of TransE [15] that project entities into different subspaces, such as TransR [16], TransH [17] and TransD [18]. The pioneering work of the bilinear model is RESCAL [19]. In this model, each entity is associated with a vector to capture its latent features, while each relation is represented as a matrix to model pairwise interactions between latent features. Extensions of RESCAL, such as DistMult [20], restrict the relational matrix to the diagonal matrix. Neural network models utilize complex neural structures to encode the knowledge graph. M-DCN [21] and IE_RCN [22] apply the convolutional neural network to generate expressive feature embeddings. M-DCN [23] proposes a heterogeneous graph neural network framework for learning knowledge graph embedding. However, in the face of OOKG entities, these methods usually need to repeat training from scratch, which is rather time-consuming.

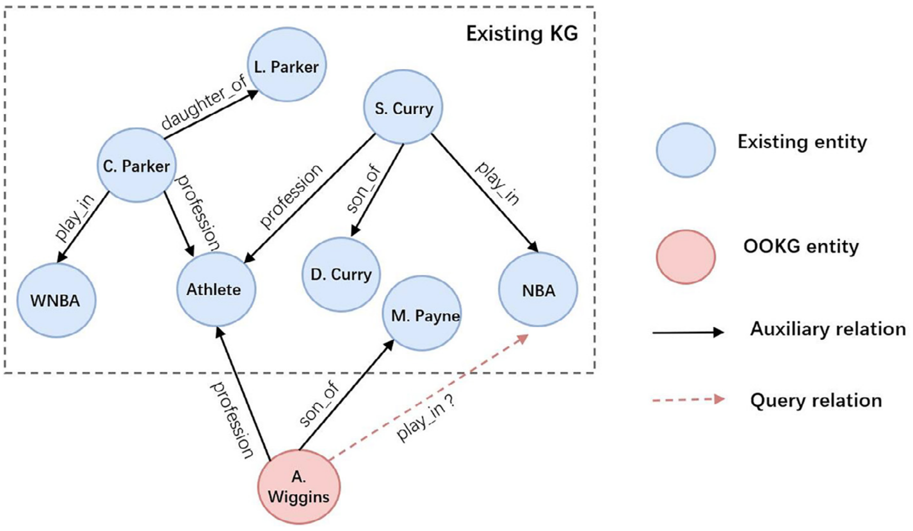


Fig. 1. An example for predicting the missing triple, where A.Wiggins is emerging as an OOKG entity with two auxiliary neighbors.

2.2. Inductive embedding methods with neighborhood alone

To address the OOKG entity embedding problem, some methods are proposed to represent the newly emerging entities via their neighborhood information. MEAN [9] applies GNN along with a mean pooling function to compute the embeddings of OOKG entities. GEN [24] proposes a meta-learning framework to extrapolate the knowledge from seen to unseen entities. And oDistMult [10] introduces a simple yet efficient training algorithm to optimize its well-designed aggregation functions for embedding OOKG entities. To characterize the unordered and unequal natures of entities' neighbors, LAN [11] employs an attention mechanism and proposes an aggregator called logic attention network. InvTransE [12] further simplifies the attention mechanism and introduces correlation-based and degree-based weights to characterize the different contribution of neighbors. However, these attention mechanisms pay so much attention to the reversible relations that they can not deal with a low redundancy dataset. Besides, considering that newly emerging entities often have very few nearby neighbors, only neighborhood information is deficient for embedding OOKG entities.

2.3. Inductive embedding methods with additional information

Instead of purely relying on neighborhood, several methods exploit additional information to obtain the embeddings of OOKG entities. DKRL [25] explores bag-of-words and deep convolutional neural models to build representations for new entities according to text descriptions. IKRL [26] proposes a novel image embodied knowledge representation learning model, which leverages the visual information from entity images to represent the OOKG entity. Since external resources are not always available and hard to acquire, other methods try to extract rules from the existing knowledge graph, which can naturally generalize to the OOKG entities. GraIL [27] presents a graph neural network framework to represent logical rules in the knowledge graph, which can inductively predict the relations between unseen entities. VN network [28] constructs the logic and symmetric path rules to capture more virtual neighbors for embedding OOKG entities. However, to generalize to newly emerging entities, these methods need to select the suitable rules from a large amount of candidates, which is intractable when the OOKG entities emerge frequently. Compared with these methods, we introduce the contextual similarity information for enhancing OOKG entity embeddings, which is simple yet powerful.

2.4. Similarity measure methods

There are some works trying to measure similarities between nodes in the field of simple isomorphic graphs [29] [30]. The time requirement limits their capability of managing the knowledge graph with vast entities and multiple relations. A few researchers exploit the external sources such as domain concepts and content to measure the similarity of each entity pair in the knowledge graph [31]. However, such information suffers from incompleteness and inconsistency present in the data [32]. In contrast, we exploit the self-contained semantic information to measure the distance of each entity pair based on the contextual gap.

3. Method

In this section, we introduce a similarity-aware aggregation network SLAN, which allows to obtain an entity's embedding by fusing the similarity and neighborhood information as illustrated in Fig. 2. The three crucial components of SLAN, introduced in the next subsections, are: (i) how to measure the similarity between any two entities; (ii) how to aggregate the similar entities and the neighborhood for a target entity; (iii) how to optimize the embeddings that well match the aggregation network.

3.1. Background and notation

A knowledge graph $\mathcal{G}$ can be considered as a multi-relational graph, which usually contains many facts in the form of triples, namely $\mathcal{G} = \{(e_s, r, e_o) | e_s \in \mathcal{E}, r \in \mathcal{R}, e_o \in \mathcal{E}\}$, where $\mathcal{E}$ and $\mathcal{R}$ represent the set of entities and relations. In addition, for an entity e_i, we divide its neighborhood according to the relations' directions: outgoing neighborhood $\mathcal{N}_o(e_i) = \{(r, e) | (e_i, r, e) \in \mathcal{G}\}$ and incoming neighborhood $\mathcal{N}_i(e_i) = \{(r, e) | (e, r, e_i) \in \mathcal{G}\}$.

Knowledge graph embedding methods usually define a score function on each fact to measure its plausibility. In this respect, the score function assigns a higher value to a positive triple than that to a negative one. In this paper, we apply the widespread translational model TransE [15], whose score function can be described as follows,

$$f(e_s, r, e_o) = -|\mathbf{e}_s + \mathbf{r} - \mathbf{e}_o|, \tag{1}$$

where $\mathbf{e}_s, \mathbf{r}, \mathbf{e}_o \in \mathbb{R}^d$ correspond to the input embeddings of the subject entity, the relation and the object entity, respectively, and d is the embedding dimension. These embeddings are optimized by maximizing the total plausibility of observed facts. After that, the resulting embeddings are utilized to predict the missing entity in any given triple $(?, r, e_o)$ or $(e_s, r, ?)$, where r is the query relation. As the missing triples may contain newly emerging entities, one requires to incorporate an aggregation network to facilitate the inference of the OOKG entity embeddings at the test time.

3.2. Similarity search

This section introduces how to search for the similar counterparts in terms of any entity in the knowledge graph. It is well known that similar entities are likely to occur in common graph context. Here, we explore the graph context from two perspectives, i.e., neighbor context and edge context, which both belong to structural information surrounding the target entity and carry its certain semantic properties.

Neighbor Context. Given the target entity e_i, we treat its neighboring entities, along with the relations, as the neighbor context, namely $C_n(e_i) = \{(r, e) | (e_i, r, e) \in \mathcal{G}\} \vee \{(-r, e) | (e, r, e_i) \in \mathcal{G}\}$. It is the most common context and is widely used in the field of knowledge graph embedding[33] . In this work, we apply the Jaccard index [34] to measure the neighbor context similarity S_n between entity e_i and e_j as follows,

$$S_n(e_i, e_j) = \frac{|C_n(e_i) \wedge C_n(e_j)|}{|C_n(e_i) \vee C_n(e_j)|}, \tag{2}$$

where $|.|$ denotes the number of elements. Intuitively, the more similar neighbor contexts two entities have, the more similar these entities are. Due to the fact that knowledge graphs are usually sparse [35], there exist few common neighbor contexts between each entity pair. In this sense, only neighbor context is not enough. Thus, we additionally utilize the edge context to locate more potential similar entities.

Edge Context. Edge context refers to all kinds of relations relevant to the target entity e_i, namely $C_e(e_i) = \{r | (e_i, r, e) \in \mathcal{G}\} \vee \{-r | (e, r, e_i) \in \mathcal{G}\}$. Actually, the relations can provide us with valuable information about the types of the entities [36], which are largely deficient in the KGs. We make up for such information and measure the edge context similarity S_e between e_i and e_j as follows,

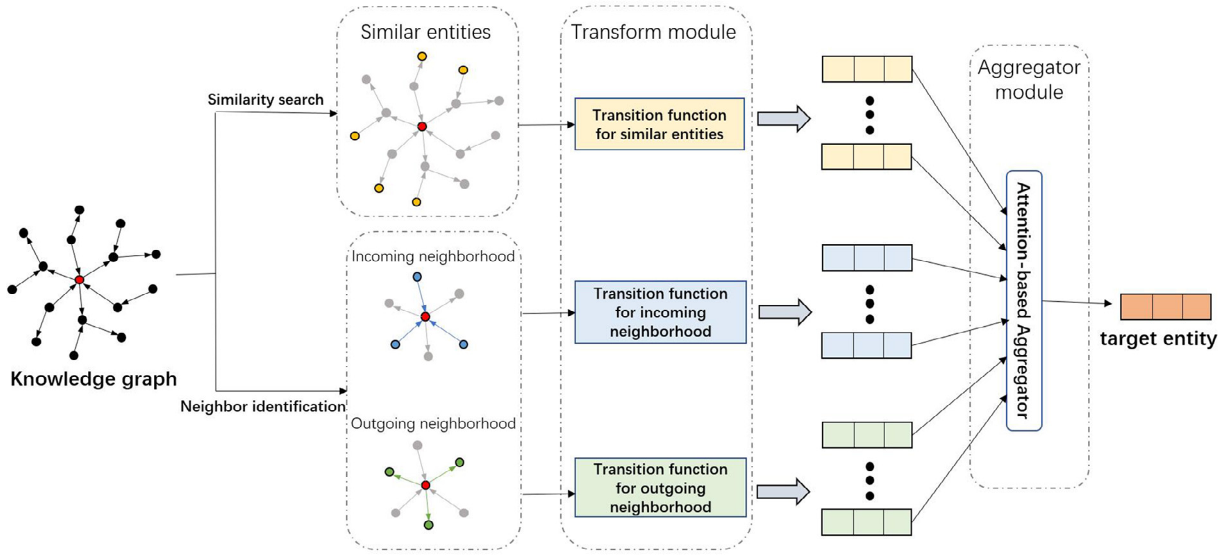


Fig. 2. The illustration of SLAN. Given a target entity (marked in red), we search the knowledge graph to obtain its similar entities (marked in yellow). Besides, we identify its incoming neighborhood (marked in blue) and outgoing neighborhood(marked in green). The similarity and the neighborhood information will be fused for embedding the target entity.

$$S_e(e_i,e_j) = \frac{|C_e(e_i) \wedge C_e(e_j)|}{|C_e(e_i) \vee C_e(e_j)|}. \tag{3}$$

To reinforce the advantages of neighbor context similarity and edge context similarity, we obtain the final similarity measure $S(e_i,e_j)$ between e_i and e_j as follows,

$$S(e_i,e_j) = \frac{S_n(e_i,e_j) + \alpha \cdot S_e(e_i,e_j)}{1+\alpha}, \tag{4}$$

where α weights the influence between $S_n(e_i,e_j)$ and $S_e(e_i,e_j)$. Here we can derive the value of α by computing their information quantity. Given two triples (e_i,r_1,e_1) and (e_j,r_2,e_2), let X denotes the event that e_i and e_j have the common neighbor context (formally as $r_1 = r_2, e_1 = e_2$), and Y denotes the event that they have the common edge context (formally as $r_1 = r_2$). According to the information theory, the quantity for X is $lg(n_e \cdot n_r)$, and that for Y is $lg(n_r)$, where n_e and n_r denote the number of entities and relations. Thus, we can obtain α by

$$\alpha = \frac{lg(n_r)}{lg(n_e n_r)}. \tag{5}$$

Search Complexity. As we need to compute the similarity value (refer to Eq. 4) for each entity-pair (e_i,e_j) in the knowledge graph, the time complexity is $O(n_e^2(d_1+d_2))$, where d_1 and d_2 are the average number of the elements from neighbor context and edge context for an entity. We notice that the entity pairs without common edge context, namely $|C_e(e_i) \wedge C_e(e_j)| = 0$, account for a large part of the total number of entity pairs in the knowledge graph. In this case, the similarity values between these entity pairs should be zero, namely $S(e_i,e_j) = 0$. To reduce the time requirement, we preferentially traverse all the relations between any entity pair (e_i,e_j) to compute the value of $|C_e(e_i) \wedge C_e(e_j)|$, which can be implemented efficiently due to the limited number of relations. Suppose there are on average $d_e \ll n_e$ entities that have common edge context with an entity, then the time complexity becomes $O(n_e d_e(d_1+d_2))$. To reduce the space requirement, we only store the top-k ($k \ll n_e$) similar entities for a given entity. Here we use the matrix $\mathbf{M}_{ind} \in \mathbb{R}^{n_e \times k}$ to store the indexes of the top-k similar entities in terms of each entity, and $\mathbf{M}_{val} \in \mathbb{R}^{n_e \times k}$ to store the corresponding similarity values. Then, the space complexity becomes $O(n_e k)$.

3.3. Aggregation network

In this section, we present the aggregation network, which comprises the transform module and the aggregator module, to aggregate the similar entities and the neighborhood for the target entity.

3.3.1. Transform module

The transform module aims to encode the information from similarity and neighborhood, respectively, which results in the candidate embeddings of the target entity e_i. Here, we apply different transition functions tailored for different information. In terms of the similarity information, we directly obtain the candidate embedding of e_i by the identity mapping function as follows,

$$T_s(e_j) = \mathbf{e}_j, \quad e_j \in \mathcal{N}_s(e_i), \tag{6}$$

where $\mathbf{e}_j$ denotes the input embedding of e_j, and $\mathcal{N}_s(e_i)$ denotes the top-k similar entity set of e_i, which obtained by retrieving the matrix $\mathbf{M}_{ind}$. In terms of the neighborhood information, we follow the translational assumption [12] that regards a relation as a geometric translation between entities, and obtain the candidate embedding of e_i by the translational functions as follows,

$$T_n(e_j,r_j) = \begin{cases} \mathbf{e}_j - \mathbf{r}_j, & (e_j,r_j) \in \mathcal{N}_o(e_i) \\ \mathbf{e}_j + \mathbf{r}_j, & (e_j,r_j) \in \mathcal{N}_i(e_i) \end{cases}. \tag{7}$$

where $\mathbf{e}_j$ and $\mathbf{r}_j$ denote the input embeddings of e_j and r_j. Thus, the candidate embedding set of the target entity e_i can be defined as $\mathcal{P}(e_i) = \{T_s(e_j)|e_j \in \mathcal{N}_s(e_i)\} \vee \{T_n(e_j,r_j)|(e_j,r_j) \in \mathcal{N}_i(e_i) \vee \mathcal{N}_o(e_i)\}$.

3.3.2. Aggregator module

The aggregator module reduces all the resulting candidate embeddings of e_i to the aggregation embedding by the weighted average. Note that candidate embeddings make different contribution to the representation of the taget entity depending on which relations will be queried. Thus, we propose the query-specific attention mechanisms to estimate the importance of these candidates. Formally, given the target entity e_i, the attention weight of the similarity candidate $T_s(e_j)$ is measured by

$$a^{sim}_{j|i,q} = S(e_i,e_j)\cdot|\mathbf{u}_s^{\mathrm{T}}\mathbf{w}_{r_q}|,\quad e_j\in\mathcal{N}_s(e_i),\ \mathbf{u}_s\in\mathbb{R}^d, \tag{8}$$

where $S(e_i,e_j)$ denotes the similarity value between e_i and e_j which is obtained by retrieving the matrix $\mathbf{M}_{val}$, and $\mathbf{w}_{r_q}\in\mathbb{R}^d$ is an additional embedding associated with the query relation r_q. Through trial and error tuning in our implementation, we notice that the influence of the neighborhood around the target entity varies mostly according to the involved relations and the query relation. In particular, the query relation often interacts with different relations to certain degree. For example, the query relation live_in has a stronger interaction with the relation born_in than with the relation son_of. Formally, given the target entity e_i, the attention weight of the neighborhood candidate $T_n(e_j,r_j)$ is measured by

$$a^{nei}_{j|i,q} = \begin{cases} |\mathbf{u}_{n_1}^{\mathrm{T}}(\mathbf{w}_{r_j}\circ\mathbf{w}_{r_q})|, & (e_j,r_j)\in\mathcal{N}_o(e_i),\ \mathbf{u}_{n_1}\in\mathbb{R}^d,\\ |\mathbf{u}_{n_2}^{\mathrm{T}}(\mathbf{w}_{r_j}\circ\mathbf{w}_{r_q})|, & (e_j,r_j)\in\mathcal{N}_i(e_i),\ \mathbf{u}_{n_2}\in\mathbb{R}^d,\end{cases} \tag{9}$$

where $\circ$ is Hadmard product, and $\mathbf{w}_{r_j}$ is an additional embedding associated with the neighboring relation r_j. Finally, to obtain the aggregation embedding of the target entity e_i, we design the aggregation function that aggregates the candidate embeddings via the corresponding attention weights as follows,

$$\mathbf{e}_i = \frac{1}{Z}\Big[\sum_{e_j\in\mathcal{N}_s(e_i)} a^{sim}_{j|i,q}\cdot T_s(e_j) + \sum_{(e_j,r_j)\in\mathcal{N}_o(e_i)\vee\mathcal{N}_i(e_i)} a^{nei}_{j|i,q}\cdot T_n(e_j,r_j)\Big], \tag{10}$$

where $Z=\sum_{e_j\in\mathcal{N}_s(e_i)}a^{sim}_{j|i,q}+\sum_{(e_j,r_j)\in\mathcal{N}_o(e_i)\vee\mathcal{N}_i(e_i)}a^{nei}_{j|i,q}$ is a normalization constant.

3.4. Training objective

The training procedure for OOKG entity representation learning is proposed in Algorithm 1. After aggregating the candidate embeddings from the similarity and neighborhood information, we are ready to optimize the input embeddings and the output of the aggregator. To make the aggregation network approximate what is expected of the OOKG entity embeddings and make the input embeddings aware of the aggregation operation, we follow [10], and formulate the training objective as

$$L = L_1 + \lambda\cdot L_2 + \lambda\cdot L_3, \tag{11}$$

where the hyper-parameter $\lambda>0$ is used to trade off the weight of different losses. L_1, L_2, L_3 are three margin-based ranking losses defined as follows,

$$L_1 = \sum_{(e_s,r,e_o)\in\Delta_1}\sum_{(e_{s'},r',e_{o'})\in\Delta_2}[\gamma - f_1(e_s,r,e_o) + f_1(e_{s'},r',e_{o'})]_+, \tag{12}$$

$$L_2 = \sum_{(e_s,r,e_o)\in\Delta_1}\sum_{(e_{s'},r',e_{o'})\in\Delta_2}[\gamma - f_2(e_s,r,e_o) + f_2(e_{s'},r',e_{o'})]_+, \tag{13}$$

$$L_3 = \sum_{(e_s,r,e_o)\in\Delta_1}\sum_{(e_{s'},r',e_{o'})\in\Delta_2}[\gamma - f_3(e_s,r,e_o) + f_3(e_{s'},r',e_{o'})]_+, \tag{14}$$

where Δ_1 and Δ_2 denote the positive and negative triple set, $[x]_+ = max(0,x), \gamma>0$ denotes the margin value, and f_1, f_2, f_3 are the score functions for a triple (e_s,r,e_o), which are defined as follows,

$$f_1(e_s,r,e_o) = -|\mathbf{e}_s+\mathbf{r}-\mathbf{e}_o|, \tag{15}$$

$$f_2(e_s,r,e_o) = -|\mathbf{e}_s^A+\mathbf{r}-\mathbf{e}_o|, \tag{16}$$

$$f_3(e_s,r,e_o) = -|\mathbf{e}_s+\mathbf{r}-\mathbf{e}_o^A|. \tag{17}$$

where $\mathbf{e}_s^A$ and $\mathbf{e}_o^A$ denote the aggregation embeddings of e_s and e_o. When inferring the missing triples about OOKG entities at the test time, we can exploit the learned aggregation network to obtain their embeddings, and apply the score function f_2 or f_3 to measure the plausibility of the candidate triples in which the OOKG entities occurs as the subjects or the objects.

Note that the ideas we develop are general and can be applied to other translational models as well, such as TransR [16], TransH [17] and TransD [18].

Algorithm 1: Training Procedure of SLAN

```
Input:
  The positive triples and negative triples $\mathcal{T}=\{(e_s,r,e_o)\}$;
  The loss function $L$; The score functions $f_1,f_2,f_3$;
  The similarity matrices $\mathbf{M}_{ind}$ and $\mathbf{M}_{val}$; The iteration number $N$;
1: repeat
2:   for: each min-batch $\mathcal{T}_b$ do
3:     $scores, labels \leftarrow []$;
4:     for $(e_s,r,e_o), label$ in $\mathcal{T}_b$ do
5:       $\mathbf{e}_s,\mathbf{r},\mathbf{e}_o \leftarrow$ LookupEmbedding$(e_s,r,e_o)$;
6:       $N_i(e_s),N_o(e_s),N_i(e_o),N_o(e_o) \leftarrow$ LocateNeighborhood$(e_s,e_o)$;
7:       $N_s(e_s),N_s(e_o) \leftarrow$ RetrieveSimilarity$(M_i,e_s,e_o)$;
8:       $\mathcal{P}(e_s) \leftarrow$ Transform$(N_s(e_s),N_i(e_s),N_o(e_s))$;   Eqs. 6 and 7
9:       $\mathcal{P}(e_o) \leftarrow$ Transform$(N_s(e_o),N_i(e_o),N_o(e_o))$;   Eqs. 6 and 7
10:      $\mathbf{e}_s^A,\mathbf{e}_o^A \leftarrow$ Aggregate$(\mathcal{P}(e_s),\mathcal{P}(e_o))$;   Eq. 10
11:      $scores$.append([$f_1(e_s,r,e_o),f_2(e_s,r,e_o),f_3(e_s,r,e_o)$]);   Eqs. (15)–(17)
12:      $labels$.append($label$);
13:    end for
14:    UpdateParams$(L, scores, labels)$;   Eq. 11
15:  end for
16:  $n=n+1$;
17: until $n<N$
```

4. Experiment

In this section, we evaluate our method on the task of knowledge graph completion. Afterwards, we test its performance when there exist few neighbors surrounding the newly emerging entities. Finally, we carry out a case study to demonstrate the effectiveness of exploiting similarity information.

4.1. Dataset

In the experiment, the test set of datasets should contain OOKG entities that are unseen during training. The work [11] constructs required datasets FB15K-sub(10%) and FB15K-obj(10%) based on FB15K [15], which are the benchmark datasets for our experiment. Besides, since FB15K contains many redundant relations and is not recommended for further experiments in the knowledge graph [37], we select three low redundancy datasets FB15K237 [38], WN18RR [37] and YAGO3-10 [37] to construct more required datasets according to a similar protocol used in [11] as follows.

- Sampling OOKG entities: We select the entities that appear in the original test set, and randomly sample $R=\{10\%; 20\%\}$ of these entities to form the candidate unseen entities U'. For an entity $e \in U'$, if it does not have any neighbors in the original training set, such entity is filtered out, yielding the final unseen entity set U.

- Filtering and splitting training set: To ensure that OOKG entities would not appear in final training set, we split the original training set into two datasets: the new training set and auxiliary set. For a triple (e_s, r, e_o) in original training set, if $e_s \notin U \wedge e_o \notin U$, it is added to the new training set. Then, we select the entities that appear in the new training set, and form the final existing entity set E. In addition, if $e_s \in U \wedge e_o \in E$ or $e_s \in E \wedge e_o \in U$, it is added to the auxiliary set, which serves as the existing neighbor for OOKG entities at the test time.
- Forming the new validation set: For a triple (e_s, r, e_o) in the original validation set, if $e_s \in E \wedge e_o \in E$, it is selected to form the new validation set.
- Forming the new test set: For a triple (e_s, r, e_o) in the original test set, if $e_s \in U \wedge e_o \in E$ or $e_s \in E \wedge e_o \in U$, it is selected to form the new test set.

The statistics for the these eight datasets are listed in Table 1. The code and datasets along with their corresponding splits are available at https://github.com/lmdgit/OOKG.

4.2. Implementation

In this paper, given an entity, we randomly sample 16 incoming neighbors and 16 outgoing neighbors. Zero padding is used when the number of corresponding neighbors is less than 16. We conduct all the experiments on a GPU-enabled (NVIDIA GeForce RTX 3090) Linux machine. During training, SGD algorithm is used, and the number of batches is 128. We train our model until convergence but stop at most 2000 rounds. In addition, we fine-tune the hyper-parameters on the validation dataset. Here we use the function $f_1(e_s, r, e_o)$ to compute the score of the triple (e_s, r, e_o) in the validation set, since all entities are available. The ranges of the hyper-parameters for the grid search are set as follows: embedding dimension $d \in \{50, 100, 200\}$, SGD learning rate $lr \in \{5.0, 10.0\}$, margin value $\gamma \in \{1, 2, 4, 6, 8, 10, 12\}$, the dimension of similarity matrix $k \in \{4, 8, 16\}$, the weight of loss function $\lambda \in \{0.1, 0.2, 0.5\}$. Table 2 lists the optimal configurations of SLAN.

4.3. Knowledge graph completion

Knowledge graph completion aims to predict the missing triples about OOKG entities under the supervision of the existing knowledge graph, which is the most important benchmark task for knowledge graph embedding.

Evaluation protocol: For each test triple (e_s, r, e_o) where e_s or e_o is an OOKG entity, the other entity which exists in the KG is removed and replaced by each of the entities in E to create a set of corrupted triples. We use 'Filtered' setting protocol [17], which will not take any corrupted triples that appear in the existing KG into account. Then, the scores of those corrupted triples are computed. We rank the valid test triple and corrupted triples in descending order of their scores. The following evaluation metrics are employed: mean reciprocal rank(MRR) and the proportion of the valid test triples ranking in top n predictions (Hits@n). Higher MRR, higher Hits@n indicate better performance.

Baselines: MEAN [9], oDistMult [10], LAN [11] and InvTransE [12] are selected as baselines. As there exist several variants of oDistMult in the original paper [10], we apply the version oDistMult-ERAvg that achieves better performance. In addition, InvTransE applies two attention weights named corr and deg. And we find that the method InvTransE(corr) has a deficiency that it doesn't discuss how to design the corr attention when there is no correlation between the query and the target entity's neighboring relations. In this case, we modify InvTransE(corr) and employ the same mean pooling on the neighborhood as MEAN [9].

4.3.1. Comparison with state-of-the-art methods

Here we select the state-of-the-art inductive embedding methods as baselines. Since the corresponding original papers did not test these embedding methods on some of following datasets, we retrain MEAN, LAN and oDistMult based on the codes provided by the authors [10] [11] to enable a comprehensive comparison. As the code of InvTransE is not public, we re-implement it using the open source toolkit OpenKE [39], and list the hyper-parameter setting of these baselines in the appendix A. The evaluation results are reported in Tables 3–6. We could observe that:

Table 1
Statistics of datasets used in the experiments. Here α denotes the similarity weight in Eq. 4.

Dataset	#Rel	#Ent	#OOKG-Ent	#Train	#Valid	#Auxiliary	#Test	α
FB15K-sub(10%)	1323	12193	2102	108854	11339	249798	2811	0.42
FB15K-obj(10%)	1325	12275	1947	99783	10190	261341	2987	0.42
FB15K237(10%)	237	13443	1029	227863	14717	42178	3354	0.36
FB15K237(20%)	237	12378	2061	192144	12160	72726	6173	0.36
WN18RR(10%)	11	39785	505	82587	2891	3903	508	0.18
WN18RR(20%)	11	39154	993	79610	2786	6561	872	0.18
YAGO3-10(10%)	37	121381	789	963086	4479	114274	789	0.24
YAGO3-10(20%)	37	120554	1503	906539	4221	169271	1333	0.24

Table 2
Optimal configurations of SLAN on the datasets.

Dataset	d	lr	γ	K	λ
FB15K-sub(10%)	200	5.0	8.0	16	0.2
FB15K-obj(10%)	200	5.0	8.0	16	0.2
FB15K237(10%)	100	5.0	4.0	16	0.2
FB15K237(20%)	100	5.0	6.0	16	0.2
WN18RR(10%)	100	5.0	10.0	4	0.2
WN18RR(20%)	100	5.0	10.0	4	0.2
YAGO3-10(10%)	100	10.0	1.0	16	0.2
YAGO3-10(20%)	100	10.0	1.0	16	0.2

Table 3
Comparison with state-of-the-art embedding methods on FB15K-sub(10%) and FB15K-obj(10%).

	FB15K-sub(10%)				FB15K-obj(10%)			
	MRR	HITS@1	HITS@3	HITS@10	MRR	HITS@1	HITS@3	HITS@10
MEAN*	0.310	22.2	34.8	48.0	0.251	17.1	28.0	41.0
LAN*	0.394	30.2	44.6	**56.6**	0.314	22.7	35.7	**48.2**
oDistMult	0.222	15.9	23.9	34.7	0.197	12.8	22.4	32.8
InvTransE(corr)	0.320	24.4	35.1	46.5	0.240	17.0	26.5	37.6
InvTransE(deg)	0.287	21.0	31.5	43.4	0.208	13.8	23.4	34.1
SLAN	**0.417**	**34.2**	**45.6**	56.1	**0.323**	**25.6**	**35.9**	45.4

Results marked by * are taken from [11], and others are reproduced based on the public code [10] [39].

Table 4
Comparison with state-of-the-art embedding methods on FB15K237(10%) and FB15K237(20%).

	FB15K237(10%)				FB15K237(20%)			
	MRR	HITS@1	HITS@3	HITS@10	MRR	HITS@1	HITS@3	HITS@10
MEAN	0.233	15.6	25.5	37.7	0.238	15.9	25.7	39.5
LAN	0.262	18.9	28.4	41.0	0.265	18.7	28.9	42.2
oDistMult	0.220	15.5	23.9	34.7	0.232	16.6	25.0	36.3
InvTransE(corr)	0.260	18.1	28.9	40.9	0.269	18.9	29.9	43.0
InvTransE(deg)	0.268	18.8	29.5	43.0	0.276	19.4	30.5	44.2
SLAN	**0.303**	**22.1**	**33.1**	**47.1**	**0.307**	**22.0**	**33.9**	**48.1**

Table 5
Comparison with state-of-the-art embedding methods on WN18RR(10%) and WN18RR(20%).

	WN18RR(10%)				WN18RR(20%)			
	MRR	HITS@1	HITS@3	HITS@10	MRR	HITS@1	HITS@3	HITS@10
MEAN	0.097	4.7	11.0	19.9	0.104	4.8	12.7	20.5
LAN	0.119	5.3	14.1	25.2	0.118	7.0	12.6	21.5
oDistMult	0.280	21.2	30.3	41.1	0.264	19.6	29.7	38.5
InvTransE(corr)	0.100	6.1	11.8	17.6	0.087	5.3	10.0	15.0
InvTransE(deg)	0.148	9.6	17.1	24.4	0.139	9.0	15.9	22.6
SLAN	**0.297**	**23.3**	**34.0**	**41.7**	**0.286**	**22.8**	**31.5**	**40.3**

Table 6
Comparison with state-of-the-art embedding methods on YAGO3-10(10%) and YAGO3-10(20%).

	YAGO3-10(10%)				YAGO3-10(20%)			
	MRR	HITS@1	HITS@3	HITS@10	MRR	HITS@1	HITS@3	HITS@10
MEAN	0.051	3.1	5.2	10.2	0.042	1.7	3.8	9.1
LAN	0.104	6.1	10.9	19.7	0.097	5.1	10.1	19.5
oDistMult	0.106	6.2	11.3	19.8	0.104	5.6	11.2	20.7
InvTransE(corr)	0.095	5.6	9.9	17.6	0.103	6.1	10.7	18.1
InvTransE(deg)	0.086	5.2	8.3	16.5	0.090	4.9	9.9	16.2
SLAN	**0.136**	**8.6**	**14.6**	**23.7**	**0.143**	**9.6**	**14.7**	**23.3**

Table 7
The aggregation functions of SLAN-S, SLAN-T and SLAN-A.

Method	Aggegation function for the target entity e_i
SLAN-S	$\mathbf{e}_i = \frac{1}{Z}[\sum_{(e_j,r_j)\in N_o(e_i)\vee N_i(e_i)} a^{nei}_{j\|i,q} \cdot T_n(e_j,r_j)]$
SLAN-T	$\mathbf{e}_i = \frac{1}{Z}[\sum_{e_j\in N_s(e_i)} a^{sim}_{j\|i,q} \cdot \mathbf{e}_j + \sum_{(e_j,r_j)\in N_o(e_i)\vee N_i(e_i)} a^{nei}_{j\|i,q} \cdot \mathbf{e}_j]$
SLAN-A	$\mathbf{e}_i = \frac{1}{Z}[\sum_{e_j\in N_s(e_i)} T_s(e_j) + \sum_{(e_j,r_j)\in N_o(e_i)\vee N_i(e_i)} T_n(e_j,r_j)]$

- SLAN outperforms the baselines on almost of the metrics, which demonstrates the effectiveness of our method and the correctness of our intuition analysis.
- Due to the fact that there exist many reversible relations in FB15K-sub(10%) and FB15K-obj(10%), embedding methods which focus on this relatedness between relations will achieve better performance, such as LAN.

Table 8
The ablation study of SLAN on the knowledge graph completion task.

	MRR	HITS@1	HITS@3	HITS@10	MRR	HITS@1	HITS@3	HITS@10
	FB15K237(10%)				FB15K237(20%)			
SLAN-S	0.289	20.2	32.3	46.1	0.290	20.7	32.4	46.2
SLAN-T	0.280	20.2	30.8	43.3	0.274	19.4	30.8	44.1
SLAN-A	0.283	20.9	30.8	43.1	0.274	19.7	30.3	43.7
SLAN	**0.303**	**22.1**	**33.1**	**47.1**	**0.307**	**22.0**	**33.9**	**48.1**
	WN18RR(10%)				WN18RR(20%)			
SLAN-S	0.291	22.8	33.8	40.3	0.282	22.2	31.0	39.9
SLAN-T	0.247	19.9	26.8	33.7	0.258	20.1	29.3	36.1
SLAN-A	0.091	6.1	9.9	15.0	0.086	6.1	8.7	13.2
SLAN	**0.297**	**23.3**	**34.0**	**41.7**	**0.286**	**22.8**	**31.5**	**40.3**
	YAGO3-10(10%)				YAGO3-10(20%)			
SLAN-S	0.116	6.3	12.5	21.5	0.112	6.4	11.7	20.5
SLAN-T	0.104	6.1	10.5	20.0	0.102	6.0	10.5	18.8
SLAN-A	0.125	7.9	13.2	21.0	0.124	7.2	12.9	22.7
SLAN	**0.136**	**8.6**	**14.6**	**23.7**	**0.143**	**9.6**	**14.7**	**23.3**

- Compared with LAN that also applies attention weights for neighborhood aggregation, SLAN achieves an improvement of 6.1% on HITS@10 on FB15K237(10%), 16.5% on WN18RR(10%) and 4.0% on YAGO-10(10%).

4.3.2. Ablation study

To test the effectiveness of each component of SLAN, we present three variants in this section. The difference among these variants is their aggregation functions (refer to Eq. 10) that output the embeddings of target entities, which are listed in Table 7. Specifically, the method SLAN-S removes the similarity information from SLAN. SLAN-T changes the transform module and obtains the target entity's neighborhood candidates by directly performing an identity mapping on the neighboring entities in Eq. 7. And SLAN-A changes the aggregator module and applies a simple mean pooling on the candidate embeddings to output the final aggregation embeddings. The evaluation results are listed in Table 8 and we could observe that:

- From the results of knowledge graph completion, we justify that each component of our method is effective and can improve the performance consistently.
- Compared with SLAN-S, SLAN can achieve a 2.0% improvement on MRR on YAGO3-10(10%) and 3.0% improvement on YAGO3-10(20%), which demonstrates that exploiting the similarity information can benefit the prediction about OOKG entities.
- Our attention mechanisms work better on the datasets WN18RR(10%) and WN18RR(20%), which can achieve around 20% improvement on MRR on both datasets. The reason may be that there exists strong interaction between the relations in WN18RR, which can be effectively captured by our attention mechanisms.

4.3.3. Generalization to other translational methods

Since the ideas in this paper are general and can be applied to other translational methods, such as TransR [16], TransH [17] and TransD [18], we use the score functions of these methods to modify the transition function in Eq. 7 according to [12], and design the loss functions in Eqs. (12)–(14) for testing the generalization performance of SLAN. From Table 9, we could observe that:

- Our method can empower the widespread translational methods with the ability to represent OOKG entities inductively. Among these methods, SLAN(TransE) can achieve stable and consistent performance.
- Compared with other translational models, SLAN(TransR) performs poorly on this task. The reason may be that TransR has more parameters to be learned in the training process, which limits its capability of generalizing to the OOKG entities.

4.3.4. Parameter analysis

Semantic search: When measuring the similarity for each entity pair, we use the weight α to characterize the influence of neighbor context and edge context in Eq. 4. In this section, we investigate the effect of SLAN with different values of α. Evaluation results are reported in Table 10 and we could observe that SLAN with the value computed by Eq. 5 works generally better than other constants, which demonstrates that the effectiveness of our proposal that derives α based on the information quantity, and we can improve the capability of SLAN by combining neighbor context and edge context properly.

Training procedure: In the training process, we propose the loss function (refer to Eq. 11) to optimize our proposal. To explore the effect of the weight λ in the training loss, the following experiments are conducted with $\lambda \in \{0.1, 0.2, 0.5\}$ on FB15K237(10%) and FB15K237(20%). From the results in Table 11, we could observe that the training weight λ has a impact on the performance of SLAN, and a proper scheduling of λ can result in more expressive embeddings for the OOKG entities.

4.3.5. Evaluation time analysis

Since new entities may emerge frequently, it's necessary for embedding methods to efficiently predict the missing triples about OOKG entities. Thus, we conduct this experiment to test the eval-

Table 9
The evaluation results with different translational scoring functions.

	FB15K237(10%)				YAGO-10(10%)			
	MRR	HITS@1	HITS@3	HITS@10	MRR	HITS@1	HITS@3	HITS@10
SLAN(TransH)	0.301	21.3	**33.5**	47.0	0.128	7.7	13.7	23.2
SLAN(TransR)	0.261	18.2	28.6	40.9	0.058	2.5	5.7	12.2
SLAN(TransD)	0.292	20.8	31.9	45.8	**0.164**	**10.9**	**18.3**	**27.2**
SLAN(TransE)	**0.303**	**22.1**	33.1	**47.1**	0.136	8.6	14.6	23.7

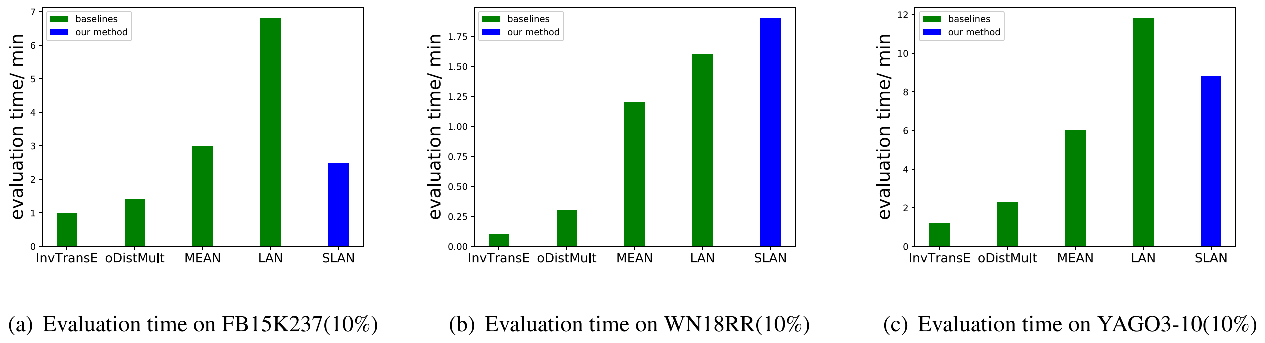
Table 10
Parameter analysis on similarity weight α.

	FB15K237(10%)		WN18RR(10%)		YAGO-10(10%)	
	MRR	HITS@1	MRR	HITS@1	MRR	HITS@1
SLAN($\alpha = 1$)	0.296	21.1	0.287	22.5	0.123	7.6
SLAN($\alpha = 0$)	0.298	21.4	0.293	23.0	0.126	7.9
SLAN($\alpha = \alpha^*$)	**0.303**	**22.1**	**0.297**	**23.3**	**0.136**	**8.6**

Note that α^* denote the value derived by Eq. 5.

Table 11
Parameter analysis on training loss weight λ.

	FB15K237(10%)				FB15K237(20%)			
	MRR	HITS@1	HITS@3	HITS@10	MRR	HITS@1	HITS@3	HITS@10
SLAN($\lambda = 0.1$)	0.291	20.9	32.0	45.8	0.301	21.2	33.3	47.8
SLAN($\lambda = 0.2$)	**0.303**	**22.1**	**33.1**	**47.1**	**0.307**	**22.0**	**33.9**	**48.1**
SLAN($\lambda = 0.5$)	0.299	21.9	32.3	46.3	0.303	21.4	33.8	47.9



(a) Evaluation time on FB15K237(10%) (b) Evaluation time on WN18RR(10%) (c) Evaluation time on YAGO3-10(10%)

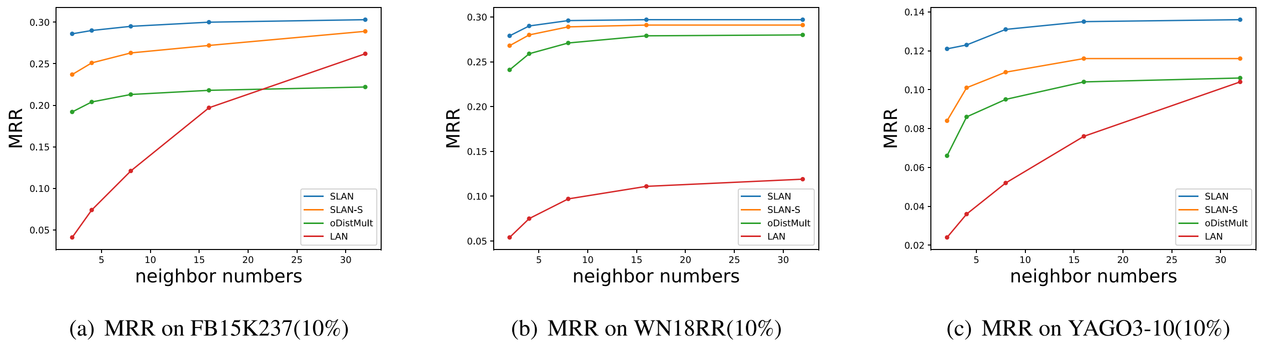
Fig. 3. Comparison of evaluation time between SLAN and the baselines.

uation time of above embedding methods. The results are reported in Fig. 3, and we could observe that:

- Compared with the baselines, SLAN can achieve the state-of-the-art performance without significantly increasing the evaluation time.
- The embedding method LAN generally requires longer evaluation time than other models, due to its complicated attention mechanism which is time-consuming.

4.4. Sparsity analysis

To further illustrate our improvements, the following experiments are conducted to verify the capabilities of our method of dealing with the sparsity problem, when there exist few neighbors surrounding the newly emerging entities. Take the dataset FB15K237(10%) for an example, for each OOKG entity, the average ratio of existing neighbors in auxiliary set is around 40. Thus, to test our method's performance on the sparse scenario, we remove the triples in the auxiliary set and ensure that each OOKG entity has no more than N (N = 2,4,8,16,32) neighbors. Then, the remaining auxiliary set along with the test set are used to evaluate our method. Experimental results of SLAN and baselines are plotted in Fig. 4. We can observe that the sparsity problem will lead to drastically performance reduction for the baselines that only rely on neighborhood information. And the performance of our method decreases slightly as the neighbor N reduces. The experiments jus-



(a) MRR on FB15K237(10%) (b) MRR on WN18RR(10%) (c) MRR on YAGO3-10(10%)

Fig. 4. Comparison on the OOKG entities with different neighbors.

Table 12
The sample cases. The left column contains the OOKG subject entity and the query relation. The middle column contains the information of the subject's similar entities ranked in a descending order according to the similarity value. The right column contains the ranked prediction from SLAN and SLAN-S. The correct predictions are marked in bold.

Subject and Query	The information of similar entities	Predicted Object from SLAN and SLAN-S
Michael McKean query: profession	profession(Christopher Guest)->Actor profession(Harry Shearer)->Screenwriter profession(Fred Willard)->Actor profession(Eugene Levy)->Screenwriter	SLAN: **Actor**, Screenwriter, Film_Producer, Musician, Comedian SLAN-S: Composer, Film_Director, Musician Television_Director, Film_Producer
Up query: genre	genre(WALL-E)->Comedy genre(Ratatouille)->Comedy genre(War Horse)->Drama genre(Rango)->Action Film	SLAN: **Comedy**, Science_Fiction, Drama, Action_Film, Musical SLAN-S: Science_Fiction, **Comedy**, Musical, Action_Film, Drama
City of London query: country	country(Tower Hamlets)->United Kingdom country(Newham)->United Kingdom country(Enfield)->United Kingdom country(Aachen)->United Kingdom	SLAN: **United_Kingdom**, England, Greater_London, West_Midlands, Warwickshire SLAN-S: Greater_London, England, **United_Kingdom**, Oxfordshire, Enfield

Table 13
Hyperparameters used to train the embedding methods in our paper.

Method	FB15K237(10%)	FB15K237(20%)	WN18RR(10%)	WN18RR(20%)	YAGO3-10(10%)	YAGO3-10(20%)
MEAN	$d = 100; \gamma = 5.0; Ep = 2000$	$d = 100; \gamma = 6.0; Ep = 2000$	$d = 100; \gamma = 20.0; Ep = 2000$	$d = 100; \gamma = 25.0; Ep = 2000$	$d = 100; \gamma = 2.0; Ep = 2000$	$d = 100; \gamma = 1.0; Ep = 2000$
LAN	$d = 100; \gamma = 3.0; Ep = 2000$	$d = 100; \gamma = 4.0; Ep = 2000$	$d = 100; \gamma = 10.0; Ep = 1000$	$d = 100; \gamma = 10.0; Ep = 1000$	$d = 100; \gamma = 1.0; Ep = 1000$	$d = 100; \gamma = 1.0; Ep = 1000$
oDistMult	$d = 200; lr = 0.001; reg = 0.0; Ep = 1000$	$d = 200; lr = 0.001; reg = 0.0; Ep = 1000$	$d = 200; lr = 0.01; reg = 0.001; Ep = 1000$	$d = 200; lr = 0.01; reg = 0.01; Ep = 1000$	$d = 200; lr = 1.0; reg = 0.001; Ep = 2000$	$d = 200; lr = 1.0; reg = 0.01; Ep = 2000$
InvTransE	$d = 100; \gamma = 4.0; Ep = 2000$	$d = 100; \gamma = 4.0; Ep = 2000$	$d = 100; \gamma = 10.0; Ep = 1000$	$d = 100; \gamma = 10.0; Ep = 1000$	$d = 100; \gamma = 1.0; Ep = 1000$	$d = 100; \gamma = 1.0; Ep = 1000$

d: embedding dimension; lr: learning rate; γ: margin; reg: regularization rate; Ep: training epochs.

tify the potential value of similarity information when dealing with the sparsity problem.

4.5. Case study

In this section, we would like to show how similar entities contribute to inferring the missing triple about OOKG entities. We sample some cases from the test set of FB15K237(10%) and evaluate the performance of SLAN and SLAN-S. From Table 12, we have the following observations. First, exploiting similarity information can facilitate semantic transfer and benefit the prediction about the unseen entity. In the first case, Guest and Willard are the similar entities of the subject McKean, and the semantic that they are Actors can be utilized by SLAN to infer the missing fact (Michael McKean, profession, Actor). Second, our similarity measure can effectively obtain the similar entities for a target entity. In the second case, the subject Up and its top-2 similar entities are all animated comedy. And in the third case, the subject and its top-3 similar entities are all boroughs in London. This case study analysis can demonstrate the effectiveness of exploiting similarity information to enhance the OOKG entity embeddings.

5. Conclusion

In this paper, we propose a novel similarity-aware aggregation network for effectively embedding OOKG entities. We first measure the similarity between entities based on their neighbor context and edge context. Additionally, we design query-specific attention weights to aggregate the similarity and neighborhood information. Experimental results show that our method achieves consistent improvements compared with the state-of-the-art baselines.

Currently, we assume that all the neighbors surrounding the OOKG entity are concurrent, which ignores the temporal information. In fact, due to the evolving nature of the knowledge graphs, the occurrence time of these neighbors may be different. And the temporally adjacent neighbors will have a major impact on the target entity's representation. In the future, we plan to study how to leverage this temporal information for enhancing OOKG entity embeddings.

CRediT authorship contribution statement

Mingda Li: Conceptualization, Methodology, Software, Validation, Writing - original draft, Visualization, Resources, Data curation. **Zhengya Sun:** Formal analysis, Investigation, Writing - review & editing, Supervision, Funding acquisition. **Wensheng Zhang:** Project administration, Funding acquisition.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

The authors are thankful for the financial support from the National Key Research and Development Program of China (No. 2018AAA0102100) and the National Natural Science Foundation of China (Nos. 61876183, 61961160707, 61976212)

Appendix A. Hyperparameters

We report here the hyperparameter setting used for each embedding methods in our experiments.
Table 13

References

[1] A. Kazemi, A. Toral, A. Way, A. Monadjemi, M. Nematbakhsh, Syntax-and semantic-based reordering in hierarchical phrase-based statistical machine translation, Expert Syst. Appl. 84 (2017) 186–199.
[2] Y. Zhang, H. Dai, Z. Kozareva, A.J. Smola, L. Song, Variational reasoning for question answering with knowledge graph, in: Proceedings of Thirty-Second AAAI Conference on Artificial Intelligence, 2018.
[3] H. Wang, Z. Wang, S. Hu, X. Xu, S. Chen, Z. Tu, Duskg: A finegrained knowledge graph for effective personalized service recommendation, Future Gener. Comput. Syst. 100 (2019) 600–617.
[4] H. Liu, C. Zheng, D. Li, X. Shen, K. Lin, J. Wang, Z. Zhang, Z. Zhang, EDMF: Efficient Deep Matrix Factorization with Review Feature Learning for Industrial Recommender System, IEEE Trans. Industr. Inf. (2021).
[5] R. West, E. Gabrilovich, K. Murphy, S. Sun, R. Gupta, D. Lin, Knowledge base completion via search-based question answering, in: Proceedings of the 23rd international conference on world wide web, 2014, pp. 515–526.
[6] Q. Wang, Z. Mao, B. Wang, L. Guo, Knowledge graph embedding: A survey of approaches and applications, IEEE Trans. Knowl. Data Eng. 29 (12) (2017) 2724–2743.
[7] B. Shi and T. Weninger, Open-World Knowledge Graph Completion, in: Thirty-Second AAAI conference on artificial intelligence, 2018, pp. 1957–1964.
[8] J. Lehmann, R. Isele, M. Jakob, A. Jentzsch, D. Kontokostas, P.N. Mendes, S. Hellmann, M. Morsey, P.V. Kleef, S. Auer, DBpedia–a large-scale, multilingual knowledge base extracted from Wikipedia, Semantic Web (2015).
[9] T. Hamaguchi, H. Oiwa, M. Shimbo, Y. Matsumoto, Knowledge Transfer for Out-of-Knowledge-Base Entities: A Graph Neural Network Approach, IJCAI (2017) 1802–1808.
[10] M. Albooyeh, R. Goel, M. Kazemi, Out-of-Sample Representation Learning for Knowledge Graphs, in: Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, 2020, pp. 2657–2666.
[11] P. Wang, J. Han, C. Li, R. Pan, Logic Attention Based Neighborhood Aggregation for Inductive Knowledge Graph Embedding, AAAI (2019) 7152–7159.
[12] D. Dai, H. Zheng, F. Luo, B. Chang, Z. Sui, Inductively Representing Out-of-Knowledge-Graph Entities by Optimal Estimation Under Translational Assumptions, arXiv preprint arXiv:2009.12765 (2020).
[13] D. Q. Nguyen, A survey of embedding models of entities and relationships for knowledge graph completion, arXiv preprint, arXiv:1703.08098, 2017.
[14] S. Ji, S. Pan, E. Cambria, P. Marttinen, P.S. Yu, A survey on knowledge graphs: Representation, acquisition, and applications, IEEE Trans. Neural Networks Learn. Syst. (2021).
[15] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, O. Yakhnenko, Translating embeddings for modeling multi-relational data, in: Proceedings of Advances in neural information processing systems, 2013, pp. 2787–2795.
[16] Y. Lin, Z. Liu, M. Sun, Y. Liu, X. Zhu, Learning entity and relation embeddings for knowledge graph completion, in: Proceedings of the 29th AAAI conference on artificial intelligence, 2015, pp. 2181–2187.
[17] Z. Wang, J. Zhang, J. Feng, Z. Chen, Knowledge graph embedding by translating on hyperplanes, in: Proceedings of the 28th AAAI conference on artificial intelligence, 2014, pp. 1112–1119.
[18] G. Ji, S. He, L. Xu, K. Liu, J. Zhao, Knowledge graph embedding via dynamic mapping matrix, in: Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), 2015, pp. 687–696.
[19] M. Nickel, V. Tresp, H.-P. Kriegel, A three-way model for collective learning on multi-relational data, in: Proceedings of the 28th International Conference on Machine Learning, 2011, pp. 809–816.
[20] B. Yang, W. Yih, X. He, J. Gao, L. Deng, Embedding entities and relations for learning and inference in knowledge bases, in: Proceedings of International Conference on Learning Representations, 2015.
[21] Z. Zhang, Z. Li, H. Liu, N. Xiong, Multi-scale dynamic convolutional network for knowledge graph embedding, IEEE Trans. Knowl. Data Eng. (2020).
[22] Z. Li, H. Liu, Z. Zhang, T. Liu, J. Shu, Recalibration convolutional networks for learning interaction knowledge graph embedding, Neurocomputing 427 (2021) 118–130.
[23] Z. Li, H. Liu, Z. Zhang, T. Liu, N. Xiong, Learning knowledge graph embedding with heterogeneous relation attention networks, IEEE Trans. Neural Networks Learn. Syst. (2021).
[24] J. Baek, D.B. Lee, S.J. Hwang, Learning to Extrapolate Knowledge: Transductive Few-shot Out-of-Graph Link Prediction, in: Proceedings of the Thirty-fourth Conference on Neural Information Processing Systems, 2020.
[25] R. Xie, Z. Liu, J. Jia, H. Luan, Representation learning of knowledge graphs with entity descriptions, in: Proceedings of the AAAI Conference on Artificial Intelligence, 2016, pp. 2659–2665.
[26] R. Xie, Z. Liu, H. Luan, M. Sun, Image-embodied Knowledge Representation Learning, IJCAI (2017) 3140–3146.
[27] K. Teru, E. Denis, W. Hamilton, Inductive relation prediction by subgraph reasoning, in: Proceedings of International Conference on Machine Learning, 2020, pp. 9448–9457.
[28] Y. He, Z. Wang, P. Zhang, Z. Tu, Z. Ren, VN Network: Embedding Newly Emerging Entities with Virtual Neighbors, in: Proceedings of the 29th ACM International Conference on Information and Knowledge Management, 2020, pp. 505–514.
[29] G. Jeh, J. Widom, Simrank: a measure of structural-context similarity, in: Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining, 2002, pp. 538–543.
[30] D. Fogaras, B. Rácz, Scaling link-based similarity search, in: Proceedings of the 14th international conference on World Wide Web, 2005, pp. 641–650.
[31] I. Traverso, M.E. Vidal, B. Kämpgen, Y. Sure-Vetter, GADES a graph-based semantic similarity measure, in: Proceedings of the 12th International Conference on Semantic Systems, 2016, pp. 101–104.
[32] D. Krompaß, B. Stephan, T. Volker, Type-Constrained Representation Learning in Knowledge Graphs, in: Proceedings of the 14th international semantic web conference, 2015, pp. 640–655.
[33] J. Feng, M. Huang, Y. Yang, X. Zhu, GAKE: Graph aware knowledge embedding, in: Proceedings of the 26th International Conference on Computational Linguistics: Technical Papers, 2016, pp. 641–651.
[34] T.A. Sorensen, A method of establishing groups of equal amplitude in plant sociology based on similarity of species content and its application to analyses of the vegetation on Danish commons, Kongelige Danske Videnskabernes Selskab 5 (1–34) (1948) 4–7.
[35] D. Nathani, J. Chauhan, C. Sharma, M. Kaul, Learning attention-based embeddings for relation prediction in knowledge graphs, in: Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, 2019, pp. 4710–4723.
[36] H. Wang, H. Ren, J. Leskovec, Relational Message Passing for Knowledge Graph Completion, in: Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, 2021, pp. 1697–1707.
[37] T. Dettmers, P. Minervini, P. Stenetorp, S. Riedel, Convolutional 2d knowledge graph embeddings, in: Proceedings of the 32nd AAAI Conference on Artificial Intelligence, 2018, pp. 1811–1818.
[38] K. Toutanova, D. Chen, Observed versus latent features for knowledge base and text inference, in: Proceedings of the 3rd Workshop on Continuous Vector Space Models and their Compositionality, 2015, pp. 57–66.
[39] X. Han, S. Cao, X. Lv, Y. Lin, Z. Liu, M. Sun, J. Li, OpenKE: an open toolkit for knowledge embedding, in: Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, 2018, pp. 139–144.

Mingda Li is currently a Ph.D. student in institute of automation, Chinese Academy of Sciences. His research interests include Knowledge graph, natural language processing and approximate reasoning.

Zhengya Sun received the Ph.D. degree in computer science from Institute of Automation of Chinese Academy of Sciences, in 2011. She is currently an Associate Professor with the research center of precise perception and control, Institute of Automation of Chinese Academy of Sciences. Her current research interests include statistical relational learning, approximate reasoning, and data stream classification.

Wensheng Zhang received the Ph.D. degree in Pattern Recognition and Intelligent Systems from the Institute of Automation, Chinese Academy of Sciences (CAS), in 2000. He joined the Institute of Software, CAS, in 2001. He is a professor of machine learning and data mining and the director of Research and Development Department, Institute of Automation, CAS. His research interests include computer vision, pattern recognition and artificial intelligence.