Computing
https://doi.org/10.1007/s00607-020-00842-5

REGULAR PAPER

Enhancing knowledge graph embedding by composite neighbors for link prediction

Kai Wang[1] · **Yu Liu**[1] · **Xiujuan Xu**[1] · **Quan Z. Sheng**[2]

Received: 8 September 2019 / Accepted: 24 September 2020

Abstract

Knowledge graph embedding (KGE) aims to represent entities and relations in a low-dimensional continuous vector space. Recent KGE works focus on incorporating additional information, such as local neighbors and textual descriptions, to learn valuable representations. However, the non-uniformity and redundancy hinder the effectiveness of entity features from those information sources. In this paper, we propose a novel end-to-end framework, called composite neighborhood embedding (CoNE), utilizing composite neighbors to enhance the existing KGE methods. To ease past problems, the new composite neighbors are gathered from both entity descriptions and local neighbors. We design a novel Graph Memory Networks to extract entity features from composite neighbors, and fulfill the entity representation in the target KGE method. The experimental results show that CoNE effectively enhances three different KGE methods, TransE, ConvE, and RotatE, and achieves the state-of-the-art results on four real-world large datasets. Furthermore, our approach outperforms the recent text-enhanced models with fewer parameters and calculation. The source code of our work can be obtained from https://github.com/KyneWang/CoNE.

Keywords Knowledge graph embedding · Link prediction · Graph memory networks · Knowledge graphs

Mathematics Subject Classification 68T30

✉ Kai Wang
kai_wang@mail.dlut.edu.cn

Yu Liu
yuliu@dlut.edu.cn

Xiujuan Xu
xjxu@dlut.edu.cn

Quan Z. Sheng
michael.sheng@mq.edu.au

[1] School of Software, the Key Laboratory for Ubiquitous Network and Service Software of Liaoning Province, Dalian University of Technology, Dalian 116023, China

[2] Department of Computing, Macquarie University, Sydney, NSW 2109, Australia

Published online: 06 October 2020

1 Introduction

Knowledge graphs (KGs) have attracted widespread attention in the past few years with their enormous potential in artificial intelligence (AI) applications [8]. In most KGs, knowledge facts are stored as triples in the form of (head entity, relation, tail entity), e.g., ('Apple Inc.', 'Operating Systems Developed', 'Mac OS'). Despite that millions of facts have been extracted from the real world, the construction of large scale knowledge graphs is still confronted with incompleteness and sparseness [27].

In order to predict new facts and complete KGs automatically, knowledge graph embedding (KGE) has been proposed [21]. Different from normal graph embeddings, KGE methods focus on multi-relational graphs and learn the representation of entities and relations in a low-dimensional continuous vector space [24]. As one of the typical methods, TransE [3] treats every relation as a translation between head and tail entities. However, most of KGE methods solely learn from each triple in the KG, so they often suffer a decline in performance when processing entities with few triples [18]. To address the sparsity problem, some methods have been proposed to enhance KGE by utilizing textual descriptions or local neighbors [31,32].

Although the two kinds of additional information are valid, there are three problems when utilizing them in the engineering practice:

- *The information redundancy of entity descriptions* Textual descriptions often contain hundreds of words, in which unnecessary words dilute the entity features and hinder the expression of potential relationships [6]. Recent work intercepts the long text to a fixed length (usually the first 20 words) [31] but causes a loss of the entity semantic meaning.
- *The unbalanced distribution of local neighbors* Because of the power-law distribution of entities in real-world KGs, some commonly-used entities have hundreds of neighbors, which lead to heavy computational pressure to generate valuable representation [19]. Meanwhile, there is inevitably a long tail of sparse entities, which cannot gather enough information from the local neighbors [18].
- *Different features provided by the two parts* The features provided by local neighbors depend on the relation types in the KG. Differently, people edit the textual description to express the entity's meaning comprehensively, in which there may be other similar entities that have an indirect relation with the given entity.

To solve these problems, we propose 'Composite Neighbors', new additional information for KGE methods combining both local neighbors and entity descriptions. Figure 1 shows the composite neighbors related to one triple in Freebase [1], and the correlations with two past additional information. To reduce redundancy of entity descriptions, we first extract entities mentioned in a long text, called 'semantic neighbors'. Then, the composite neighbors of an entity contain a limited number of entities selecting from both local neighbors and semantic neighbors. For commonly-used entities, composite neighbors focus on the overlap of the two neighbor sets, rather than all hundreds of neighbors. For sparse entities, the two parts of neighbors complement each other, which can alleviate problems caused by missing data on one side.

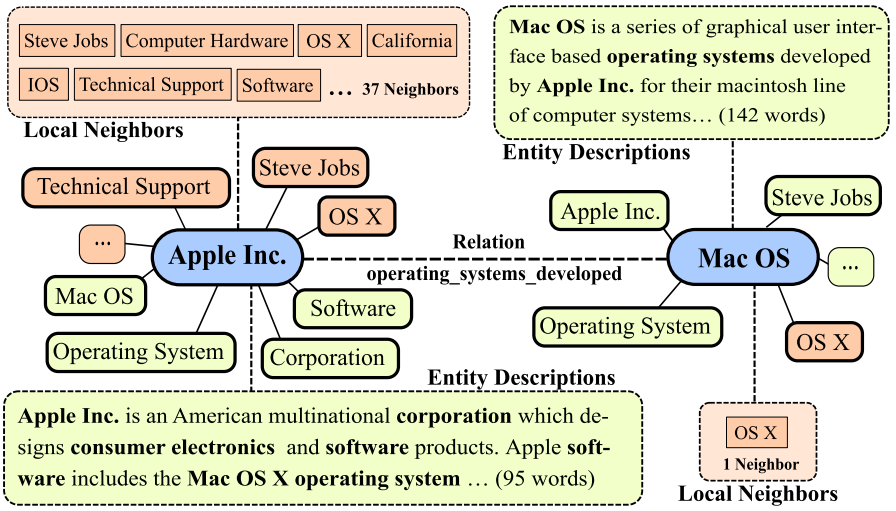

Fig. 1 The composite neighbors of two entities in a triple extracted from entity descriptions and local neighbors. Each entity has a hand-edited text to describe the entity meaning, and its local neighbors are the other entities having direct connections with it in existing KG triples

In this paper, we propose a novel encoder-decoder framework, named *Composite Neighborhood Embedding* (CoNE), which utilizes composite neighbors to enhance the existing KGE methods. To learn an entity's representation from its composite neighbors, a novel encoder model, Graph Memory Networks (GMN) is designed. Compared with general graph convolutional networks, the GMN encoder utilizes key-value memory units and multi-layered attention to avoid redundant calculations and improve the reasoning capability. After that, the entity representation enhanced by the GMN encoder is applied in a KGE model. In the experiements, we verify the performance of CoNE on three different kinds of KGE models, including TransE [3], ConvE [4], RotatE [23]. Experimental results on link prediction tasks show that our model effectively enhances the mainstream KGE methods with different score functions. The CoNE-TransE model outperforms existing models exploiting entity descriptions, while needs fewer parameters and calculation. The CoNE-RotatE model outperforms all the baseline methods on four popular datasets.

The rest of our paper is organized as follows. We first review the related works in Sect. 2. In Sect. 3, we present the technical details of the CoNE model, including composite neighbors, the GMN encoder and the KGE decoder. Then we report our extensive experimental studies to validate the CoNE model in Sect. 4 and conduct specific analysis in Sect. 5. Finally, we offer some concluding remarks in Sect. 6.

2 Related work

In this section, we briefly overview the related works on knowledge graph embedding (KGE), and the approaches exploiting additional information for KGE.

2.1 Knowledge graph embedding

According to the model architecture, recent KGE methods can be broadly separated into three categories [10], including Translational Distance Models, Matrix Factorization Models, and Neural Networks Models.

Translational Distance Models compute the distance between two entities to measure the plausibility of a triple, represented by TransE model [3]. To solve flaws of TransE, some variants such as TransH [28], TransR [12] and TransD [9] are proposed. Meanwhile, Matrix Factorization Models formulate KGE models as three-way tensor decomposition. RESCAL [17] model utilizes three-way factorization over each relational slice of knowledge graph tensor. DistMult [33] and ComplEx [26] simplify it by introducing diagonal relation matrices and complex-valued matrices. The latest model, RotatE [23] is inspired by Euler's identity and can take relation as a rotation between entities. With the development of deep learning, Neural Networks Models have achieved remarkable performance in recent KGE studies, such as ConvE [4], ConvKB [15], and CapsE [16], they take entities and relations into deep neural networks and compute a semantic matching score. As a representative model, ConvE reshapes and concatenates the entity and relation embeddings, and utilizes a multi-layer convolutional network model for link prediction.

These three categories of KGE models have shown great performance on knowledge graph completion and link prediction tasks. The TransE model is highly efficient and commonly used, but it fails to encode symmetry relations. Earlier factorization models cannot infer the relation composition pattern, while RotatE fills this gap. Compared with those Linear or bilinear models, neural network models like ConvE usually achieve better performance but inevitably introduce additional parameters. In our work, we focus on proposing a generalized framework that integrates Composite Neighbors into KGE, which can enhance all three categories of models.

2.2 Incorporating additional information in KG

Additional information as a supplement of entity representation, can effectively alleviate the sparsity problem of KGE. Facing various information forms, different kinds of encoders based on deep neural networks have been utilized. In terms of entity descriptions, Xie et al. [31] jointly learn KG embedding by using Convolutional Neural Networks (CNNs) to process entity descriptions. Xu et al. [32] utilize Recurrent Neural Networks (RNNs) to integrate structural and textual information. In terms of local neighbors, Feng et al. [7] take the target entity and directed linked entities as neighbor context, combined with path context and edge context. Shi et al. [20] extract the outgoing edges of an entity, along with the entities they can reach, as neighbor information to rich entity representation.

To design an encoder for composite neighbors, some recent research work has inspired us. Graph Convolutional Networks (GCNs) [5], as the neighborhood message passing methods, have attracted much attention in the field of graph representation learning [30]. Utilizing neighborhood information for link prediction, R-GCNs [19] is proposed to deal with highly multi-relational data such as knowledge graphs. Deepak

et al. [14] utilize Graph Attention Networks to extract entity features from the n-hop neighborhood, but introduce a large number of parameters and heavy calculations. Meanwhile, Weston et al. [29] propose Memory Networks, which is designed for natural language processing tasks, reasoning on non-writable memory units to build a hierarchical memory representation. End-to-End Memory Networks (MemN2N) [22] improve Memory Networks to support end-to-end training, and operate via a multi-layered attention mechanism in which relevant memory pieces are adaptively selected based on the input query.

In our work, we design a novel Graph Memory Networks for composite neighbors encoding, which is inspired by some characteristics of MemN2N. Based on GCNs, we reduce its memory overhead by replacing the huge adjacency matrix with the memory units in MemN2N. Besides, the multi-layered attention mechanism is utilized for hierarchical reasoning in GMN encoder.

3 The CoNE model

3.1 Problem formulation and notations

We first formulate the link prediction task and introduce the notations used in this paper. Throughout this paper, we use bold uppercase characters to denote matrices and bold lowercase characters denote vectors. The notations used in this paper are illustrated in Table 1.

Table 1 Commonly used notations and terminologies

Items	Descriptions
$\mathcal{G}$	A knowledge graph
$\mathcal{T}$	The set of triples in a KG
$e \in E, r \in R$	The set of entities and relations in a KG
(e_h, r, e_t)	A triple of head entity, relation and tail entity
(e_q, r_q)	The input query of link prediction task
e_c	A candidate of the missing entity
$N(e)$	The composite neighbors of e
$(\mathbf{k}, \mathbf{v}) \in \mathbb{R}^{2*d}$	The key-value memory unit of a neighbor
$\mathbf{e}, \mathbf{r} \in \mathbb{R}^d$	The real-number embedding vector of e and r
$\mathcal{F}(\mathbf{h}, \mathbf{r}, \mathbf{t})$	Score Function of a KGE Model
$\mathcal{L}(\Theta)$	Loss function
$\|\| \cdot \|\|_{1/2}$	L1 or L2 normalization
$[h; r]$	Vectors/matrices concatenation
d	The dimension of the embedding vector
K	The maximum neighbor number of an entity
L	The layer number of multi-layered attention

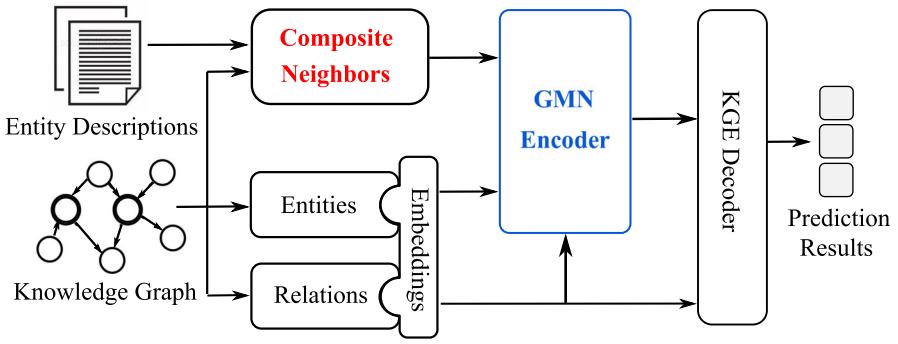

Fig. 2 The general architecture of the CoNE model

A knowledge graph (KG) is represented as $\mathcal{G} = (E, R, \mathcal{T})$ where E is the set of entities, and R is the set of relations. $\mathcal{T} = (e_h, r, e_t)$ denotes the set of factual triples in $\mathcal{G}$, where $r \in R$ is a relation, $e_h, e_t \in E$ are the head entity and tail entity of a triple. Given two items $(e_q, r_q)$of a triple, the link prediction task aims to predict the missing tail or head entity. The missing entity e_c is selected from the total entity set E. We define the composite neighborhood of an entity e as $N(e) = \{n_1, n_2, \ldots, n_K\}(n_i \in E)$, and its maximum length is denoted as K. Semantically related to e, each composite neighbor is extracted from existing triples T or entity descriptions.

3.2 Approach overview

For link prediction task, we propose a novel encoder-decoder framework, named *Composite Neighborhood Embedding* (CoNE), shown in Fig. 2. Previous KGE models represent entities and relations in a low-dimensional vector space and measure the plausibility of each candidate triple, while the CoNE aims to supplement the entity information through its composite neighbors to improve the prediction accuracy of those models. Following an encoder-decoder architecture, the CoNE framework consists of three major components as follows:

- *Composite Neighbors* are the additional information of CoNE. To gather composite neighbors for a given entity, we first generate two sets of neighbors from KG triples and entity descriptions respectively. Then we sample a limited number of composite neighbors from the two sets. This procedure will be detailed in Sect. 3.3.
- *Graph Memory Networks Encoder* is utilized to extract entity features from composite neighbors. We utilize key-value memory units to process composite neighbors, while the multi-layered attention mechanism in GMN can capture hidden correlations among neighbors based on a given relation. We will detail the GMN encoder in Sect. 3.4.
- *KGE Decoder* employs an existing KGE model to conduct the link prediction. To show the general applicability of CoNE, we apply three different models, including TransE, ConvE, and RotatE. The details of the KGE decoder and training process will be discussed in Sect. 3.5.

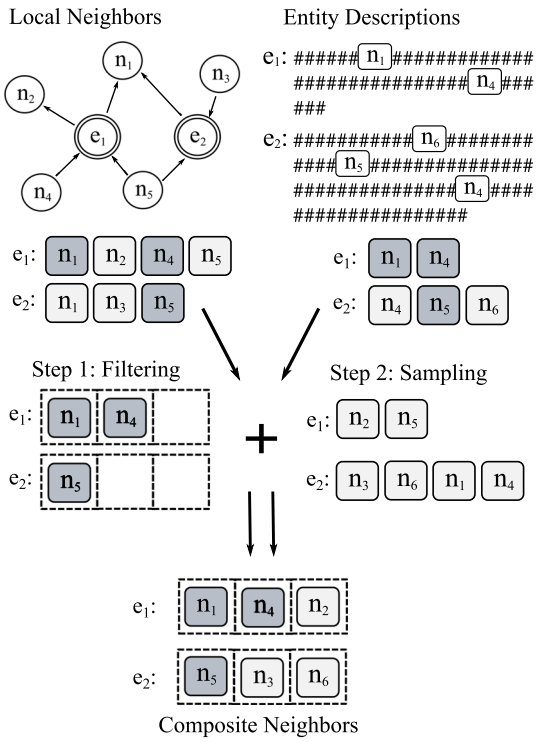

Fig. 3 An illustration on generating composite neighbors (K = 3)

The main process of CoNE is as follows: Given the input query (e_q, r_q), we first get the entity e_q's composite neighbors $N(e_q)$, and generate an enhanced entity representation in the GMN encoder. Finally, the new entity representation is used in the KGE decoder, to predict the missing entity of the query.

3.3 Composite neighbors

In this section, we introduce the composite neighbors, a new kind of additional information for KGE. By extracting representative entity neighbors from KG triples and entity descriptions, the composite neighbors can provide richer features for the entity representation, and overcome the problems of redundancy and unbalanced distribution. To generate composite neighbors for an entity e, a two-step process is proposed illustrated in Fig. 3.

First, two sets of neighbors are extracted from KG triples and entity descriptions respectively. In the neighbor set from KG triples, the local neighbors include the entities having at least one triple with e in the KG. As shown in Fig. 1, the local neighbors of 'Apple Inc.' include 'Steve Jobs', 'IOS' and so on. In the neighbor set from entity descriptions, we extract mentioned entities in the e's textual description by completely matching their names. Those semantic neighbors are defined as entities whose name appears in e's description, and entities whose description mentions the

name of e. For example, the 'Operating System' is mentioned in the textual description of 'Apple Inc.', so they are semantic neighbors for each other.

After that, at most K entities are sampled from two sets to make up the composite neighbors of e. As some entities own hundreds of neighbors, we first select the neighbors that appear in both sets simultaneously. For the entity 'Apple Inc.', its neighbors 'Software' and 'OS X' would be selected in this way. Then, the rest of the composite neighbors are filled by randomly sampling.

The aforementioned process is designed to overcome the problems in the entity descriptions and local neighbors. To reduce the redundant part, we only select a small number of entities from long textual descriptions. To address the unbalance problem, we limit the maximum number of composite neighbors, and then increase the neighbors of sparse entities by combining two neighbors sets. Note that, the composite neighbors can be applied easily in different KGs, as long as entity names and descriptions are available.

3.4 Graph memory networks encoder

Given an input query (e_q, r_q), the encoder aims to generate the e_q's representation by aggregating its own features $\mathbf{e_q}$ and composite neighbors' features. Various neural network models, such as CNNs and RNNs, have been used in entity description encoding [31,32]. Different from word sequences in those descriptions, the neighbor data is not chronological and has no spatial order. As the entity and its composite neighbors can be structured in an undirected graph, GCN-based encoder is relatively suitable to learn node representations from neighborhood information.

However, we argue that the traditional GCN model cannot efficiently encode composite neighbors in the KG. (1) As the size K of composite neighbors is much less than the entity amount, the adjacency matrix in GCN is huge and extremely sparse. The computation related to it leads to huge memory overhead. (2) Composite neighbors belong to 1-hop neighborhood, and cannot be utilized in multi-layer GCNs. However, a single-layer GCN model lacks enough fitting capability to process large-scale knowledge graphs. (3) Traditional GCN models usually learn representation in the graph inside, and have no operation to involve the input query. For a query (e_q, r_q), GCN models only process the e_q's neighborhood but the relation r_q is ignored.

To tackle the problems in the GCN model, we propose an improved GCN model for composite neighbors encoding, named Graph Memory Networks (GMN). The illustration of the GMN encoder is shown in Fig. 4. Inspired by Deep Memory Networks [22], we utilize external memory units and the multi-layered attention mechanism. The former replaces the adjacency matrix to record e_q's neighbors in a dense matrix, and then represent each neighbor as key-value vectors. The latter, multi-layered attention mechanism has multiple attention layers to learn representation with deep-level abstraction. Besides that, to solve the third problem, we improve the attention layer to assign different attention scores to neighbors under different relations.

Specifically, to generate a new representation of e_q under the specific relation r_q, we first represent the input items as low-dimensional vectors. we perform an embedding operation to represent e_q, r_q as the entity and relation embedding vec-

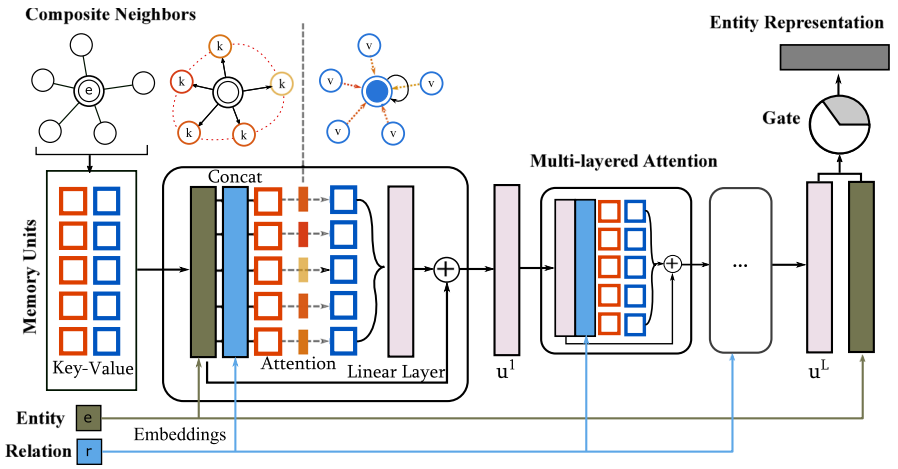

Fig. 4 An illustration of the GMN encoder in both graph view and model view

tors denoted as $\mathbf{e_q}, \mathbf{r_q} \in \mathbb{R}^d$. And the hyperparameter d is the embedding dimension. The composite neighbors of e_q are $N(e_q) = \{n_1, n_2, \ldots n_K\}$. Instead of using adjacency matrix like GCNs, those neighbors are represented as discrete memory units $\mathbf{M}(e_q) = \{(\mathbf{k_i}, \mathbf{v_i})\}(0 < i < K)$. Each memory unit is a key-value pair $(\mathbf{k_i}, \mathbf{v_i}) = (\mathbf{W_k n_i}, \mathbf{W_v n_i})$, in which the two item share one d-dimensional embedding vector $\mathbf{n_i}$. Then two projection matrices $\mathbf{W_k}, \mathbf{W_v} \in \mathbb{R}^{d \times d}$ transform $\mathbf{n_i}$ into different vector spaces.

Given the vectors $\mathbf{e_q}, \mathbf{r_q}$ and $\mathbf{M}(e_q)$, a L-layered attention mechanism is utilized to extract the potential features among composite neighbors. The input $\mathbf{u}^0$ of the first attention layer is $\mathbf{e_q}$, then the output of the layer $\mathbf{u}^1$ is input into the next layer. So that the entity vector can be iteratively updated by neighbor vectors.

In the l-th attention layer $(0 < l < L)$, the input vector $\mathbf{u}^{l-1}$ is first processed by a linear transformation followed by application of the LeakyRelu non-linearity, as shown in Equation 1.

$$\hat{\mathbf{u}} = LeakyReLU(\mathbf{W_{lin}}\mathbf{u}^{l-1} + b_{lin}), \tag{1}$$

where $\mathbf{W_{lin}} \in \mathbb{R}^{d \times d}$, $b_{lin} \in \mathbb{R}^d$. Then, the attention score for each neighbor n_i is defined as p_i, which is computed by the concatenation of the key vector of the neighbor k_i, the projected input $\hat{\mathbf{u}}$ and the relation vector $\mathbf{r_q}$:

$$g_i = LeakyReLU(\mathbf{W_{att}}[\mathbf{k_i}; \hat{\mathbf{u}}; \mathbf{r_q}] + b_{att}), \tag{2}$$

$$p_i = Softmax(g_i) = \frac{exp(g_i)}{\sum_{j=1}^{K} exp(g_j)}, \tag{3}$$

where $\mathbf{W_{att}} \in \mathbb{R}^{1 \times 2d}$, $b_{att} \in \mathbb{R}^1$. To get the relative attention scores $\mathbf{p}$, the Softmax function is applied over $\mathbf{g}$ of all neighbors. As the relation vectors are involved in

attention computing, neighbors can gather different attention scores under different relations.

With $\mathbf{p}$ as weights, the value vectors $\{\mathbf{v_i}\}$ of neighbors are summed as the neighbor representation. And the output of the l-th layer $\mathbf{u}^l$ is the addition of the neighbor representation and the updated entity vector $\hat{\mathbf{u}}$, as shown in Eq. 4. To keep the gradient stability in iterations, we constrain $\|\mathbf{u}^l\|_2 = 1$.

$$\mathbf{u}^l = \hat{\mathbf{u}} + \sum_{i=1}^{K} p_i \times \mathbf{v_i} \tag{4}$$

Finally, the output of the last L-th layer $\mathbf{u}^L$ is added with the initial entity vector $\mathbf{e_q}$ through a gating mechanism [32], which using trainable vectors as weights to adjust the proportions of two parts according to different entities. Differently, to reduce the space complexity, we use scalar weights to replace vectors. Each entity is assigned a trainable scalar weight $g_{e_q} \in [0, 1]$. As shown in Eq. 5, the final output vector $\hat{\mathbf{e_q}}$ is the enhanced entity representation, which utilized in the KGE decoder to perform link prediction.

$$\hat{\mathbf{e_q}} = g_{e_q} \odot \mathbf{e_q} + (1 - g_{e_q}) \odot \mathbf{u}^L \tag{5}$$

Compared with original end-to-end memory networks (MemN2N) [22], the GMN model has several differences. First, GMN is designed to process graph data instead of natural language, and the position encoding in MemN2N is deprecated. Besides, the key-value memory units in MemN2N utilize different parameter matrices which leads to high space complexity, while GMN fixes it by sharing the same parameters. In addition, the relation vector is involved in the attention function of GMN, which is ignored in MemN2N.

Take the triple ('Apple Inc.', 'operating systems developed', 'Max OS') in Fig. 1 as an example. The GMN encoder aims to learn a representation of 'Apple Inc.' under the relation 'operating systems developed'. To simplify the description, we only consider about two neighbors, 'Software' and 'OS X'. In a single attention layer, the relation vector is computed with each neighbor vector. As an operating system, 'OS X' is expected to get a higher attention score than 'Software'. Thus, more features about 'OS X' would be added into the entity representation. With the multi-layered iteration, the proportion difference between 'OS X' and'Software' would be more obvious. Finally, as the missing entity 'Mac OS' is tightly related to 'OS X', the new entity representation enhanced by the 'OS X' neighbor can fit with 'Mac OS' more easily.

3.5 KGE decoder and training objective

In our CoNE framework, the KGE decoder can employ different KGE models to perform the link prediction. To predict the missing entity, a score function $\mathcal{F}(e_q, r_q, e_c)$ is used to measure the potential triple. However, different from existing methods input entity and relation embedding vectors directly into the score function, the KGE decoder utilizes the encoder's output $\hat{\mathbf{e_q}}$ as the entity representation.

Table 2 The score functions and loss functions of three KGE models, where $\gamma > 0$ is the margin value, $\circ$ denotes the element-wise Hadmard product, $*$ denotes the convolution operator, $\bar{\mathbf{e}}$ denotes a 2D reshaping of $\mathbf{e}$, $vec(\cdot)$ denotes the 1D reshaping operator, $\mathbf{e^c}$ denotes a complex-number vector, $\sigma(\cdot)$ is the sigmoid function. N is the number of fact triples T, and $N_{T'}$ is the number of negative samples T'

Model	Score function	Loss function
TransE	$\mathcal{F}_T = -\lVert \mathbf{e_h} + \mathbf{r} - \mathbf{e_t} \rVert$	$\mathcal{L}_T = \sum_{t \in T} \sum_{t' \in T'} max(0, \gamma - \mathcal{F}(t) + \mathcal{F}(t'))$
ConvE	$\mathcal{F}_C = f(vec(f([\bar{\mathbf{e}}_\mathbf{h}; \bar{\mathbf{r}}] * \omega))\mathbf{W})\mathbf{e_t}$	$\mathcal{L}_C = -\frac{1}{N} \sum_{t \in T} (log(\mathcal{F}(t)) + \sum_{t' \in T'} log(1 - \mathcal{F}(t')))$
RotatE	$\mathcal{F}_R = -\lVert \mathbf{e^c_h} \circ \mathbf{r^c} - \mathbf{e^c_t} \rVert$	$\mathcal{L}_R = -\sum_{t \in T} (log\sigma(\gamma - \mathcal{F}(t)) + -\frac{1}{N_{T'}} \sum_{t' \in T'} log\sigma(\mathcal{F}(t') - \gamma))$

In the training procedure, given a set of fact triples $T \cup \mathcal{T}$, we use the Bernoulli sampling strategy described in [28] to generate the negative sampling set T'. The triple $t = (e_h, r, e_t) \in T$ denotes the positive sample, while the negative sample is denoted as $t' \in T'$. The training objective is to learn effective representations to divide the positive and negative samples.

We employ three KGE models as the KGE decoder, including TransE, ConvE, and RotatE. We follow the score functions and loss functions described in original literature of those KGE models, as shown in Table 2. For TransE decoder, we minimize a hinge loss using stochastic gradient descent (SGD), and we apply the Adam [11] optimizer to minimize the loss functions of ConvE and RotatE.

4 Experiments

We describe experimental settings and report empirical results in this section. We evaluate our Composite Neighborhood Embedding (CoNE) model in the KG link prediction task, and verify the effectiveness of the Composite Neighbors and Graph Memory Networks Encoder. All experiments are performed on Intel Core i7-7700K CPU @ 4.20 GHz and NVIDIA GeForce GTX 1070 GPU, and implemented in Python using Pytorch.

4.1 Experimental setup

4.1.1 Datasets and hyperparameter settings

We introduce four commonly-used datasets used in our experiments. FB15k [3] is extracted from Freebase in which a large fraction of content describes knowledge facts about movies, actors, awards, and sports. WN18 [2] is a subset of the English lexical database, WordNet [13]. However, the drawback of these two datasets is that many test triples can be obtained simply by inverting triples in the training set. To solve this test leakage problem, FB15k237 [25] and WN18RR [4] are created by removing inverse relations respectively. Statistics of the four datasets are given in Table 3. '#Rel' and '#Ent' refer to the number of relations and entities in the dataset, and the other

Table 3 Statistics of datasets used in experiments

Dataset	#Rel	#Ent	#Train	#Valid	#Test
FB15k	1345	14,951	483,142	50,000	59,071
FB15k237	237	14,541	272,115	17,535	20,466
WN18	18	40,943	141,442	5000	5000
WN18RR	11	40,943	86,845	3034	3134

metrics refer to the number of triples in three subsets. The entity descriptions of those datasets are publicly available. We build the composite neighbors from the triples in the training set and the entity descriptions for each dataset.

We select the hyperparameters of our model via grid search according to the metrics on the validation set. We select embedding dimension d among {100, 200, 400, 600}, learning rate λ among {0.0001, 0.0005, 0.001}, the margin γ among {3, 6, 9, 12, 24}, the maximum number of neighbors K among {5, 10, 20} and the number of layers L for the GMN encoder L among {1, 3, 5}. We keep the original settings of employed KGE models as many as possible. To speed up the convergence and avoid overfitting, the entity embedding and relation embeddings are initialized by pre-trained results of KGE decoder. The neighbor embeddings and the rest parameters are initialized by randomly sampling from a uniform distribution in $[-0.01, 0.01]$.

4.1.2 Tasks and evaluation metrics

As a main task of knowledge graph completion, link prediction aims to predict the missing e_h or e_t in a triple (e_h, r, e_t). Different from other predicting tasks requiring the best answer, this task focuses on ranking a set of candidate entities in KG.

We adopt three evaluation metrics as follows: (1) Mean Rank (MR), the average rank of the test triples, (2) Mean Reciprocal Rank (MRR), the average inverse rank of the test triples, and (3) Hits@N, the proportion of correct entities ranked in top N (N = 1, 10). Lower MR, higher MRR, and Hits@N should be achieved by a good embedding model. By following the previous work [3], there are two evaluation settings, 'Raw' and 'Filter'. The former ranks in the whole entity set, while the latter removes the other candidate triples appearing in datasets before ranking. If no comments, the following results are in the 'Filter' setting by default.

4.2 Experimental results

4.2.1 Results of link prediction task

In the link prediction task, we compare CoNE with several recent KGE methods to validate our model's performance. The evaluation results on the four datasets are shown in Tables 4 and 5. 'CoNE-X' represents the results of CoNE applying different decoders, including TransE, ConvE, and RotatE.

Table 4 Link prediction results on FB15k and WN18

Methods	FB15K				WN18			
	MR↓	Hits@10↑		MRR↑	MR↓	Hits@10↑		MRR↑
		Raw	Filter			Raw	Filter	
TransE	70	43.4	61.8	30.7	312	76.3	91.1	44.8
DisMult	120	50.0	84.2	70.5	901	81.4	93.6	82.2
ComplEx	133	49.2	82.5	72.4	895	83.9	94.4	94.1
ANALOGY	121	50.5	84.3	72.2	788	**84.3**	94.7	94.2
ConvE	51	52.5	83.1	66.0	360	80.5	95.6	94.3
RotatE	57	50.2	84.6	70.7	432	81.0	95.3	93.8
CoNE-TransE	69	50.5	78.5	54.6	299	79.6	94.2	49.6
CoNE-ConvE	50	**53.2**	87.2	**73.7**	337	80.9	95.7	**94.6**
CoNE-RotatE	**32**	51.7	**88.6**	72.0	**252**	82.3	**96.8**	**94.6**

Best score is given in bold

Table 5 Link Prediction Results on FB15k237 and WN18RR

Methods	FB15K237				WN18RR			
	MR↓	MRR↑	Hits@10↑	Hits@1↑	MR↓	MRR↑	Hits@10↑	Hits@1↑
TransE	221	24.2	42.4	16.6	3793	18.0	45.7	1.3
DisMult	309	22.4	38.3	14.6	5110	43.0	49.0	39.0
ComplEx	289	22.7	39.1	14.5	5261	44.0	51.0	41.0
ANALOGY	309	22.7	38.7	14.8	8982	39.2	41.0	38.4
ConvE	244	31.6	50.1	23.7	4187	43.0	52.0	40.0
RotatE	177	33.8	53.3	24.1	3340	47.6	57.1	42.8
CoNE-TransE	211	30.6	48.5	21.7	3542	22.7	50.2	6.4
CoNE-ConvE	187	33.4	51.5	24.6	4165	45.5	52.7	42.3
CoNE-RotatE	**147**	**35.0**	**53.6**	**25.8**	**2776**	**49.4**	**58.2**	**45.1**

Best score is given in bold

From the experimental results, we can see some obvious improvements of these KGE methods after enhanced by our proposed CoNE model. As for the TransE-based model, the Hits@10 on FB15k-237 increases from 42.4 to 48.5%, while the Hits@10 on FB15k also rises from 61.8% to 78.5% in filter mode. Meanwhile, the MRR of the CoNE-ConvE model on FB15k increases from 66.0 to 73.7%, and the Hits@1 on WN18RR rises from 40.0 to 42.3%. Especially, the CoNE-RotatE model outperforms the state-of-the-art results of RotatE in FB15k-237 and WN18RR. As the three decoders belong to different categories of KGE methods, the results clearly demonstrate that our proposed method has the capability of enhancing different kinds of KGE models.

Compared with other KGE methods, CoNE-ConvE receives the best MRR and Hit@10 metrics on FB15k, while CoNE-RotatE achieves most of state-of-the-art MRR and Hits@N on 4 datasets. As shown in Table 5, the CoNE-TransE achieves 94.2% of

Table 6 Comparison results of CoNE and two text-enhanced methods on FB15k and WN18

Type	Method	FB15K		WN18	
		MR	Hits@10	MR	Hits@10
Baseline	TransE	70	61.8	312	91.1
Entity Descriptions	DKRL (CNN)	91	67.4	–	–
	Jointly (CBOW)	92	67.4	130	89.9
	Jointly (LSTM)	90	69.7	**95**	91.6
	Jointly (Att_LSTM)	73	75.5	123	90.9
Composite Neighbors	CoNE (CBOW)	80	72.7	351	92.5
	CoNE (GCN)	73	76.5	309	93.6
	CoNE (GMN)	**69**	**78.5**	299	**94.2**

Best score is given in bold

Hits@10 on WN18 outperforming 93.6% of DistMult, which is rarely that a TransE-based model defeats matrix factorization models in this dataset. Besides, the original ConvE on FB15k dataset is weaker than ComplEx and ANALOGY, while CoNE-ConvE outperforms both models on all metrics.

4.2.2 Comparison between composite neighbors and entity descriptions

In this section, we attempt to answer the following two questions:

Q1: As the KGE additional information, whether Composite Neighbors are better than entity descriptions?

Q2: What is the effect of using GMN encoder in CoNE compared to other encoders?

To answer the Q1, we select two comparison methods DKRL [31] and the Jointly model proposed by Xu et al. [32]. As both of them utilize entity descriptions to enhance the TransE model, we also use the TransE-based CoNE model in this experiment, called 'CoNE (GMN)'. To answer the Q2, CoNE (GMN) is compared with two CoNE variants with different encoders: (1) CoNE (CBOW), using CBOW encoder which generates representation by summing up all neighbor vectors; (2) CoNE (GCN), using the Graph Convolutional Networks (GCNs) as the encoder. Besides that, DKRL employs a CNN-based encoder with two 1D convolutional layers and two max-pooling layers, while the Jointly model utilizes three different encoders. We denote them as 'DKRL (CNN)', 'Jointly (CBOW)', 'Jointly (LSTM)' and 'Jointly (Att_LSTM)'.

The experimental results on FB15k and WN18 datasets are shown in Table 6. CoNE (GMN) obtains the best scores of Hits@10 on two datasets and the lowest MR score on FB15k, while Jointly models show relatively lower MR scores on WN18. Using the same simple encoder, the variant model CoNE (CBOW) achieves better performance than Jointly (CBOW) and Jointly(LSTM). It proves that composite neighbors as additional information are better than entity descriptions. Besides, the CoNE (GMN) outperforms than the other 2 variants on two datasets. It verifies that the GMN encoder has better capability for composite neighbors encoding, and proves the effectiveness of our improvements in GMN compared with GCN encoder.

Table 7 Comparison results of Composite neighbors with different length and the other two kinds of neighbors

Metric	FB15K237		WN18RR	
	MR	Hits@10	MR	Hits@10
RotatE (Baseline)	177	53.3	3340	57.1
CoNE (ComNei-3)	172	51.9	2791	57.1
CoNE (ComNei-5)	170	52.5	2718	57.9
CoNE (ComNei-10)	169	52.7	2702	**58.4**
CoNE (ComNei-20)	**147**	**53.5**	**2667**	58.2
CoNE (LocNei-20)	168	53.0	2707	57.4
CoNE (SemNei-20)	177	52.2	3126	55.1

Best score is given in bold

4.2.3 Comparison of different kinds of neighbors

In this section, we will answer the other two questions:

Q3: Does the sparse degree of Composite Neighbors influence the performance of link prediction?

Q4: How does Composite Neighbors perform on link prediction compared to two single-source neighbors?

We experiment CoNE-RotatE model with different neighbor information in two relatively sparse datasets, FB15k-237 and WN18RR. For Q3, we compare composite neighbors with different maximum length K, and the corresponding models are denoted as 'CoNE (ComNei-K)'($K \in [3, 5, 10, 20]$). The fewer neighbors of each entity lead to the sparser additional information. For Q4, CoNE (LocNei-20) and CoNE (SemNei-20) are employed, using local neighbors from KG triples and semantic neighbors from entity descriptions respectively. Neither these neighbor set is longer than 20.

As shown in Table 7, CoNE (ComNei-20) obtains the best Hits@10 on two datasets and the lowest MR score on FB15k-237, while its MR in WN18RR is sightly weaker than CoNE (ComNei-10). As the number of neighbors decreases, the Hit@10 value drops gradually, even weaker than the original RotatE model's. It indicates the negative effect of sparse additional information on link prediction. In spite of this, the MR metrics of CoNE (ComNei-3) are still better than RotatE on two datasets. Besides that, CoNE (ComNei-20) outperforms those using single-source neighbors significantly, which proves the necessity to integrate two neighbor sets. Using local neighbors is relatively better than semantic neighbors, the reason might be that the local neighbors are directly from KG triples.

4.2.4 Detailed results by relation categories

This experiment aims to explain why CoNE can improve the performance of existing KGE models. Considering the drawbacks of KGE methods in dealing with '1-to-N', 'N-to-1', and 'N-to-N' relations, we verify the Hits@10 metrics of the CoNE model on different relation categories and compare CoNE with the original KGE methods

Table 8 Detailed results by category of relation on FB15k

Tasks relation category	Prediction head (Hits@10)				Prediction tail (Hits@10)			
	1-to-1	1-to-N	N-to-1	N-to-N	1-to-1	1-to-N	N-to-1	N-to-N
TransE	70.4	68.0	39.3	51.7	71.7	38.2	64.8	53.5
ConvE	80.4	95.4	61.7	83.7	79.3	67.7	96.1	86.6
RotatE	93.0	97.3	57.1	84.4	93.8	80.3	97.6	88.5
CoNE-TransE	85.3	96.4	58.3	80.1	82.4	65.2	95.6	83.9
CoNE-ConvE	85.7	96.7	**67.7**	**87.5**	85.3	79.6	97.3	**90.3**
CoNE-RotatE	**94.3**	**97.4**	67.2	85.6	**95.2**	**85.8**	**97.8**	89.7

Best score is given in bold

on FB15k dataset. Mapping properties of relations follows the same rules in [3]. The detailed results on FB15k are reported in Table 8.

In all four relation categories, the CoNE model achieves a promising performance based on original KGE models. The CoNE-TransE model outperforms the baseline by an almost 50% improvement, especially when predicting the 1 side for 'N-to-1' relations, the Hits@10 metrics increase approximately from 65 to 95%. Meanwhile, the ConvE-based model also increases 5% in '1-to-1', 'N-to-1', and 'N-to-N' relations. The improvement of RotatE focuses on the 'N' side of 'N-to-1' relations, especially the head entity prediction from 57.1 to 67.2%.

Those results indicate the effectiveness of composite neighbors. As the representations of different entities are supplemented by different composite neighbors, the enhanced KGE model can better distinguish similar entities when processing multi-end relations.

5 Discussions

In this section, we analyze theoretically about how the proposed Composite Neighborhood Embedding (CoNE) framework eases two major problems, namely *unbalanced distribution* and *information redundancy*.

5.1 Comparison of neighbor distribution

We make quantitative analysis about different additional information, including entity descriptions, local neighbors from KG triples, semantic neighbors from descriptions, and composite neighbors we proposed. The related results are shown in Fig. 5.

Figure 5a shows the distributions of different neighbors. The number of neighbors per entity is sorted in descending order, and the maximum number K of composite neighbors is set to 20. We can see that both of local neighbors and semantic neighbors follow the power-law distribution approximately. There are only 2618 (35.89%) and 5219 (13.87%) entities having more than 20 neighbors in those two types respectively, while in the long tail more than 60% entities are sparse. By contrast, in the composite

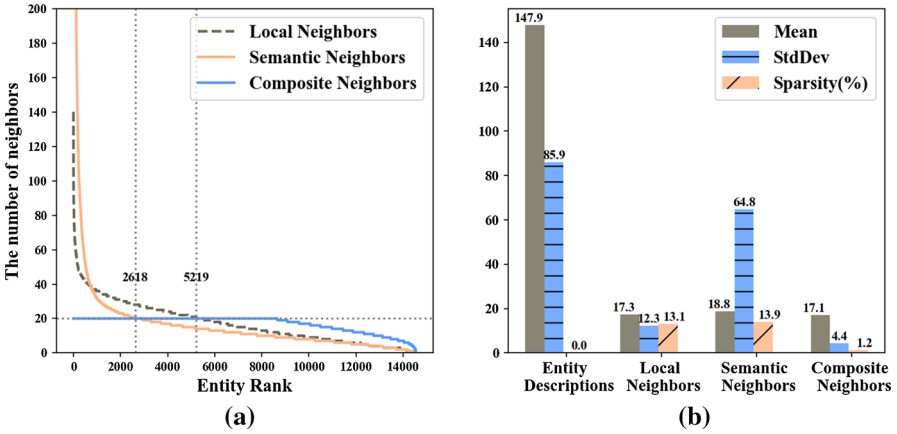

Fig. 5 The statistics of different additional information in FB15k-237. **a** The neighbor number distribution for all entities. The value on the horizontal axis indicated by the dotted line represents the total number of entities having at least 20 neighbors. **b** The histogram about multiple metrics, in which the sparsity degree is the proportion of entities that have at most 5 words or neighbors

neighbors we proposed, there are 8661 (57.92%) entities having 20 neighbors. Meanwhile, it is proved the necessity of setting the upper limit. Adding the two neighbor sets directly may lead to an explosion of neighbor number, which would hinder the effectiveness of attention mechanism.

In Fig. 5b, we count the number of neighbors, and the length of description for each entity in FB15k-237, and display four metrics including Mean, Standard Deviation, and Sparsity degree. According to the Mean metric, the length of entity descriptions is significantly more than the number of those neighbors. The standard deviation reflects the unbalanced distribution, as a higher value means more volatility. Compared with the other three types, composite neighbors has a relatively uniform distribution with the lowest standard deviation 4.4. In terms of Sparsity, both of semantic neighbors and local neighbors have more than 13% sparse entities. By contrast, composite neighbors have the least sparse entities, occupied by only 1.2%, it indicates that the sparsity problem is eased by combining two neighbor sets.

5.2 Redundancy and complexity analysis

It is verified that the additional information is beneficial to the KGE methods. However, information redundancy would lead to high computing pressure and performance degradation. Experimental results in the previous section have shown that replacing entity descriptions with composite neighbors can improve model performance. In this section, we focus on the influence on model complexity and memory overhead.

Table 9 compares the space complexity of the CoNE-TransE model and two text-enhanced models, where the parameter sizes of different components are given. The total parameters of a model contain trainable parameters in the model and the mapping matrix which recording additional information. The ‘Decoder’ and ‘Encoder’ part denote the parameters in two components respectively, while the ‘Extra Info’ part

Table 9 The space complexity comparison among CoNE and two text-enhanced methods, where the unit of total parameters is M(illion), d is the dimension of embedding vectors, K is the number of words or neighbors, $|E|, |R|, |W|$ are represented the total number of entities, relations and the vocabulary in descriptions respectively

Methods	Decoder	Extra Info	Encoder	Total parameters
DKRL (CNN)	$d\|E\| + d\|R\|$	$d\|W\| + K\|E\|$	$3d^2 + 2d$	9.3 M
Jointly (CBOW)	$d\|E\| + d\|R\|$	$d(\|W\| + \|E\|) + K\|E\|$	0	11.1 M
Jointly (LSTM)	$d\|E\| + d\|R\|$	$d(\|W\| + \|E\| + K\|E\|$	$4(2d^2 + d)$	11.2 M
Jointly (Att_LSTM)	$d\|E\| + d\|R\|$	$d(\|W\| + \|E\| + K\|E\|$	$5(2d^2 + d)$	11.2 M
CoNE (GCN)	$d\|E\| + d\|R\|$	$d\|E\| + \|E\|^2$	$d^2 + d$	226 M
CoNE (GMN)	$d\|E\| + d\|R\|$	$d\|E\| + K\|E\|$	$3d^2 + 3d$	3.4 M

contains the parameters related to additional information. We estimate the specific number of parameters when using the FB15k dataset, with $d = 100$, $K = 20$, $|E| = 14,951$, $|R| = 1345$ and $|W| = 76,758$. From Table 9 we can find that, overall, the total parameters of our CoNE model on FB15k dataset is roughly half of those of DKRL and Jointly models, while the variant of CoNE using GCN encoder need the most memory overhead. There is no obvious difference in the 'Decoder' and 'Encoder' parts, as all models employ TransE as the decoder, and the parameters in encoders are much less than embeddings.

In the comparison of the 'Extra Info' part, two text-enhanced models need more parameters for word embeddings, as the vocabulary of entity descriptions are much more than the entity number ($d|W| \gg d|E|$). As the general length of the entity description is longer than 20, the total parameters would be increased further. Those situations indicate the redundancy problem of text descriptions, while composite neighbors solve the problem by reducing the total number and limiting the maximum size of an entity's neighbors.

Then, comparing different encoders, it is clear that the Graph Memory Networks(GMN) encoder has fewer parameters than CNN and LSTM. Especially, the parameters of the GMN encoder is about one-tenth of that of Att_LSTM encoder. Although GCN encoder costs less than GMN, the adjacent matrix as its input leads to numerous memory overhead when facing large-scale knowledge graphs. This is one of the reasons we propose GMN to improve it.

Note that, although those extension models introduce more parameters into the KGE model, the training cost is acceptable benefited from pre-trained KGE embeddings. Besides that, as the strong fitting capability of the neural network encoder, the model can converge faster.

6 Conclusion

In this paper, we have proposed the Composite Neighborhood Embedding (CoNE) model for knowledge graph embedding. Instead of entity descriptions, we define composite neighbors as new additional information to overcome the distribution imbal-

ance problem. We then exploit a novel Graph Memory Networks (GMN) encoder to extract latent semantics from neighbors. Experimental results show that our model can enhance different kinds of KGE models and achieve the state-of-the-art results on the link prediction task. In comparison with text-enhanced methods, our model can get better performance and reduce the space complexity. The encouraging results are stimulating a number of further researches to extend the current work. In particular, we will explore the following research activities in the future:

- Our approach currently selects neighbors from entity descriptions by name matching , which inevitably results in certain omissions. We plan to develop a more effective mechanism for neighbor selection.
- The CoNE model utilizes Graph Memory Networks as the neighbor encoder, which is effective but still needs many additional parameters. We would like to try other forms of encoders to further reduce the memory overhead.
- Composite neighbors are designed to overcome the sparsity of KGs. We will further validate our CoNE model using more real KGs of specific domains such as finance and health.

Acknowledgements This research is supported by the National Natural Science Foundation in China (Grant: 61672128) and the Fundamental Research Fund for Central University (Grant: DUT20TD107).

References

1. Bollacker KD, Evans C, Paritosh P, Sturge T, Taylor J (2008) Freebase: a collaboratively created graph database for structuring human knowledge. In: Proceedings of the 2008 ACM SIGMOD international conference on management of data, pp 1247–1250
2. Bordes A, Glorot X, Weston J, Bengio Y (2014) A semantic matching energy function for learning with multi-relational data. Mach Learn 94:233–259
3. Bordes A, Usunier N, García-Durán A, Weston J, Yakhnenko O (2013) Translating embeddings for modeling multi-relational data. In: Proceedings of advances in neural information processing systems (NIPS 2013), pp 2787–2795
4. Dettmers T, Minervini P, Stenetorp P, Riedel S (2018) Convolutional 2D knowledge graph embeddings. In: Proceedings of the 32th AAAI conference on artificial intelligence, pp 1811–1818
5. Duvenaud DK, Maclaurin D, Aguilera-Iparraguirre J, Gómez-Bombarelli R, Hirzel T, Aspuru-Guzik A, Adams RP (2015) Convolutional networks on graphs for learning molecular fingerprints. In: Proceedings of advances in neural information processing systems (NIPS 2015), pp 2224–2232
6. Fan M, Zhou Q, Zheng TF, Grishman R (2017) Distributed representation learning for knowledge graphs with entity descriptions. Pattern Recognit Lett 93:31–37
7. Feng J, Huang M, Yang Y, Zhu X (2016) Gake: Graph aware knowledge embedding. In: Proceedings of the 32th international conference on computational linguistics (COLING 2016), pp 641–651
8. Guo D, Tang D, Duan N, Zhou M, Yin J (2018) Dialog-to-action: conversational question answering over a large-scale knowledge base. In: Proceedings of advances in neural information processing systems (NIPS 2018), pp 2942–2951
9. Ji G, He S, Xu L, Liu K, Zhao J (2015) 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, pp 687–696
10. Ji S, Pan S, Cambria E, Marttinen P, Yu PS (2020) A survey on knowledge graphs: Representation, acquisition and applications. CoRR arXiv:2002.00388
11. Kingma DP, Ba JL (2015) Adam: a method for stochastic optimization. In: Proceedings of international conference on learning representations (ICLR 2015)
12. Lin Y, Liu Z, Sun M, Liu Y, Zhu X (2015) Learning entity and relation embeddings for knowledge graph completion. In: Proceedings of the 29th AAAI conference on artificial intelligence, pp 2181–2187

13. Miller GA (1995) Wordnet: a lexical database for english. Commun ACM 38:39–41
14. Nathani D, Chauhan J, Sharma C, Kaul M (2019) Learning attention-based embeddings for relation prediction in knowledge graphs. In: Proceedings of the 57th conference of the association for computational linguistics, ACL 2019, pp 4710–4723
15. Nguyen DQ, Nguyen TD, Nguyen DQ, Phung DQ (2017) A novel embedding model for knowledge base completion based on convolutional neural network. In: Proceedings of the 2017 conference of the North American chapter of the Association for Computational Linguistics, vol 2, pp 327–333
16. Nguyen DQ, Vu T, Nguyen TD, Nguyen DQ, Phung DQ (2019) A capsule network-based embedding model for knowledge graph completion and search personalization. In: Proceedings of the 2019 conference of the North American chapter of the Association for Computational Linguistics, NAACL-HLT 2019, pp 2180–2189
17. Nickel M, Tresp V, Kriegel HP (2011) A three-way model for collective learning on multi-relational data. In: Proceedings of the 28th international conference on machine learning, pp 809–816
18. Pujara J, Augustine E, Getoor L (2017) Sparsity and noise: where knowledge graph embeddings fall short. In: Proceedings of the 2017 conference on empirical methods in natural language processing, pp 1751–1756
19. Schlichtkrull MS, Kipf TN, Bloem P, van den Berg R, Titov I, Titov I, Welling M, Welling M (2018) Modeling relational data with graph convolutional networks. In: Proceedings of extended semantic web conference, pp 593–607
20. Shi J, Gao H, Qi G, Zhou Z (2017) Knowledge graph embedding with triple context. In: Proceedings of the 2017 ACM on conference on information and knowledge management, pp 2299–2302
21. Socher R, Chen D, Manning CD, Ng AY (2013) Reasoning with neural tensor networks for knowledge base completion. In: Proceedings of advances in neural information processing systems (NIPS 2013), pp 926–934
22. Sukhbaatar S, Szlam A, Weston J, Fergus R (2015) End-to-end memory networks. In: Proceedings of advances in neural information processing systems (NIPS 2015), pp 2440–2448
23. Sun Z, Deng Z, Nie J, Tang J (2019) Rotate: Knowledge graph embedding by relational rotation in complex space. In: 7th international conference on learning representations, ICLR 2019
24. Sun Z, Hu W, Zhang Q, Qu Y (2018) Bootstrapping entity alignment with knowledge graph embedding. In: Proceedings of the 26th international joint conference on artificial intelligence, pp 4396–4402
25. Toutanova K, Chen D (2015) Observed versus latent features for knowledge base and text inference. In: Proceedings of the 3rd workshop on continuous vector space models and their compositionality
26. Trouillon T, Welbl J, Riedel S, Gaussier É, Bouchard G (2016) Complex embeddings for simple link prediction. In: Proceedings of the 33th international conference on machine learning, pp 2071–2080
27. Wang Q, Mao Z, Wang B, Guo L (2017) Knowledge graph embedding: a survey of approaches and applications. IEEE Trans Knowl Data Eng 29:2724–2743
28. Wang Z, Zhang J, Feng J, Chen Z (2014) Knowledge graph embedding by translating on hyperplanes. In: Proceedings of the 28th AAAI conference on artificial intelligence, pp 1112–1119
29. Weston J, Chopra S, Bordes A (2015) Memory networks. In: Proceedings of international conference on learning representations (ICLR 2015)
30. Wu Z, Pan S, Chen F, Long G, Zhang C, Yu PS (2019) A comprehensive survey on graph neural networks. CoRR arXiv:1901.00596
31. Xie R, Liu Z, Jia J, Luan H, Sun M (2016) Representation learning of knowledge graphs with entity descriptions. In: Proceedings of the 30th AAAI conference on artificial intelligence, pp 2659–2665
32. Xu J, Qiu X, Chen K, Huang X (2017) Knowledge graph representation with jointly structural and textual encoding. In: Proeedings of the 26th international joint conference on artificial intelligence, pp 1318–1324
33. Yang B, tau Yih W, He X, Gao J, Deng L (2015) Embedding entities and relations for learning and inference in knowledge bases. In: Proceedings of international conference on learning representations (ICLR 2015)