Learning to Compose Relational Embeddings in Knowledge Graphs

Wenye Chen[1], Huda Hakami[1,2], and Danushka Bollegala[1]

Department of Computer Science, University of Liverpool, United Kingdom
{W.Chen29@student, h.a.hakami,danushka}@liverpool.ac.uk

Abstract. Knowledge Graph Embedding methods learn lower-dimensional representations for entities and relations in knowledge graphs, which can be used to infer previously unknown relations between pairs of entities in the knowledge graph. This is particularly useful for expanding otherwise sparse knowledge graphs. Often the set of relations that exist between two entities are not independent, and it is possible to predict what other relations are likely to exist between two entities by composing the embeddings of the relations in which each entity participates. We introduce *relation composition* as the task of inferring embeddings for unseen relations by combining existing relations in a knowledge graph. Specifically, we propose a supervised method to compose relational embeddings for novel relations using pre-trained relation embeddings for existing relations. Our experimental results on a previously proposed benchmark dataset for relation composition and triple classification show that the proposed supervised relation composition method outperforms several unsupervised relation composition methods.

1 Introduction

Knowledge graphs (KGs) such as Freebase [1] organise the knowledge that we have about entities and the relations that exist between entities in the form of labelled graphs, where entities are denoted by the vertices and the relations are denoted by the edges that connect the corresponding entities. A KG can be represented using a set of relational tuples of the form (h, R, t), where the relation $R \in \mathcal{R}$ exists between the (head) entity $h \in \mathcal{E}$ and the (tail) entity $t \in \mathcal{E}$ such that the direction of the relation is from h to t. Here, $\mathcal{E}$ and $\mathcal{R}$ respectively denote the sets of entities and relations in the KG. For example, the relational tuple (Donald Trump, `president_of`, US) indicates that the `president_of` relation holds between Donald Trump and US. Despite the best efforts to create complete and large-scale KGs, most KGs remain incomplete and does not represent all the relations that exist between entities. In particular, new entities are constantly being generated and new relations are formed between new as well as existing entities. Therefore, it is unrealistic to assume that a real-world KG would be complete at any given time point.

Knowledge graph embedding (KGE) methods [17,28,14,18,24,25,4,2] learn representations (also known as *embeddings*) for the entities and relations in a

given KG. The learnt KGEs can be used for *link prediction*, which is the task of predicting whether a particular relation exists between two given entities in the KG. Specifically, given KGEs for entities and relations, in link prediction we predict the R that is most likely to exist between h and t according to some scoring formula. For example, in translational embeddings (TransE), h, t, R are all embedded into the same d-dimensional vector space respectively by $\boldsymbol{h} \in \mathbb{R}^d$, $\boldsymbol{t} \in \mathbb{R}^d$ and $\boldsymbol{R} \in \mathbb{R}^d$ and the tuple (h, R, t) is scored by $||\boldsymbol{h} + \boldsymbol{R} - \boldsymbol{t}||_2$. Once KGEs for all the entities and relations in a given KG are learnt, for two entities h' and t' that are not connected by a relation, TransE finds the relation $R' \in \mathcal{R}$ that minimises $||\boldsymbol{h'} + \boldsymbol{R'} - \boldsymbol{t'}||$. However, the relation types that can be predicted using KGEs are confined to $\mathcal{R}$, the set of relation types that *already exists* in the KG. In other words, we *cannot* predict novel relation types using the pre-trained KGEs alone.

On the other hand, the relations that exist in a KG are often closely related [22]. For example, given the embeddings for the relations `country_of_film` and `currency_of_country` relations, we can compose the embedding for a previously unseen relation such as `currency_of_film_budget` because entities are shared across many tuples such as (Movie, `country_of_film`, Country), (Country, `currency_of_country`, Currency), where Movie, Country, and Currency can be replaced respectively by valid instances such as The Italian Job, UK and GBP.

In this paper, we propose a method for composing relation embeddings for novel (unseen in the KG) relation types by composing the embeddings for existing relation types. Our problem setting differs from that of KGE in two important ways. First, we do not learn relation embeddings from scratch for a given KG, but instead use pre-trained KGEs and learn a composition function to predict the embeddings for the relations that currently do not exist in the KG. In our experiments, we use the state-of-the-art matrix embeddings produced by the Relational Walk (RelWalk) method [2] as the relation embeddings. Second, the composition functions we learn are *universal* [19] in the sense that they are not parametrised by the entities or relations in the KG, thereby making the composition function independent from a particular KG. This is attractive because, theoretically the learnt composition function can be used to compose *any* relation type, not limited to the relations that exist in the KG used for training.

2 Background

2.1 Knowledge Graph Embedding Methods

Various methods have been proposed in the literature for representing entities and relations from a given KG. A popular choice for representing entities is to use vectors, whereas relations have been represented by vectors, matrices or tensors. For example, TransE [4], TransH [26], TransD [8], TransG [27], TransR [14], lppTransD [29], DistMult [28], HolE [17] and ComplEx [24] represent relations by vectors, whereas Structured Embeddings [4], TranSparse [9], STransE [16], RESCAL [18], RelWalk [2] use matrices and Neural Tensor Network (NTN) [21] uses three dimensional tensors. ComplEx [24] introduced complex embedding

vectors for KGEs to capture the asymmetry in semantic relations. [5] obtained state-of-the-art performance for KGE by imposing non-negativity and entailment constraints to ComplEx.

Given that the number of entities in a KG is significantly larger than that of its relations, from a space complexity point-of-view, it is desirable to represent entities using vectors. On the other hand, relations are often asymmetric and directional, which cannot be modelled using vector addition or multiplication. For example, the `is_father_of` relation requires the head entity to be the father of the tail entity whereas, the `is_son_of` relation requires the reverse. Matrices are a natural choice for representing relations because matrix multiplication in general is non-commutative and this property can be used to encode the directionality of a relation. On the other hand, it is desirable to embed both entities and relations in the same vector space for carrying out linear algebraic operations for the purpose of learning KGEs. This requires a $\mathcal{O}(d^2)$ space for storing relation embeddings relative to the $\mathcal{O}(d)$ required for the entity embeddings. This requirement can be infeasible for large KGs such as Freebase, which covers over 39 million topics[1]. Therefore, much prior work on KGE has opted to represent both entities as well as relations using vectors.

As already stated in the introduction, we emphasise that we *do not* propose a method for learning KGEs in this paper. Instead, given pre-trained entity and relation embeddings, our goal is to compose relation embeddings for the relations that currently do not exist in the KG. Our proposed method is agnostic to the algorithm used to learn the input KGEs. In this regard, it can be used to compose relation embeddings using KGEs produced by any KGE learning method. As a concrete example, we use the state-of-the-art relation embeddings produced by RelWalk [2] in our experiments. RelWalk represents relations using matrices and is further detailed in subsection 2.2.

2.2 Relational Walk

In this section, we briefly describe RelWalk, the method that produces the relation embeddings we use in this paper. For a detailed overview refer [2].

RelWalk assumes that the task of generating a relational triple (h, R, t) in a given KG to be a two-step process. First, given the current knowledge vector at time k, $\boldsymbol{c} = \boldsymbol{c}_k$ and the relation R, the probability of an entity h satisfying the first argument of R to be given by (1).

$$p(h \mid R, \boldsymbol{c}) = \frac{1}{Z_c} \exp\left(\boldsymbol{h}^\top \mathbf{R}_1 \boldsymbol{c}\right). \tag{1}$$

Here, $\mathbf{R}_1 \in \mathbb{R}^{d \times d}$ is a relation-specific orthogonal matrix that evaluates the appropriateness of h for the first argument of R and Z_c is a normalisation coefficient such that $\sum_{h \in \mathcal{E}} p(h \mid R, \boldsymbol{c}) = 1$. After generating h, the state of our random walker changes to $\boldsymbol{c}' = \boldsymbol{c}_{k+1}$, and the second argument of R is generated

[1] `https://developers.google.com/freebase/guide/basic_concepts`

with probability given by (2).

$$p(t \mid R, \boldsymbol{c}') = \frac{1}{Z_{c'}} \exp\left(\boldsymbol{t}^\top \mathbf{R}_2 \boldsymbol{c}'\right). \tag{2}$$

Here, $\mathbf{R}_2 \in \mathbb{R}^{d \times d}$ is a relation-specific orthogonal matrix that evaluates the appropriateness of t as the second argument of R and $Z_{c'}$ is a normalisation coefficient such that $\sum_{t \in \mathcal{E}} p(t \mid R, \boldsymbol{c}') = 1$. In RelWalk, we consider ($\mathbf{R}_1$ and $\mathbf{R}_2$) to collectively represent the embedding of R. Bollegala et al. [2] proved Lemma 1 for a slow random walk over the KG.

Lemma 1 (Concentration Lemma). *If the entity embedding vectors satisfy the Bayesian prior $\boldsymbol{v} = s\hat{\boldsymbol{v}}$, where $\hat{\boldsymbol{v}}$ is from the spherical Gaussian distribution, and s is a scalar random variable, which is always bounded by a constant κ, then the entire ensemble of entity embeddings satisfies that*

$$\Pr_{c \sim \mathcal{C}}[(1 - \epsilon_z)Z \leq Z_c \leq (1 + \epsilon_z)Z] \geq 1 - \delta, \tag{3}$$

for $\epsilon_z = O(1/\sqrt{n})$, and $\delta = \exp(-\Omega(\log^2 n))$, where $n \geq d$ is the number of words and Z_c is the partition function for c given by $\sum_{h \in \mathcal{E}} \exp\left(\boldsymbol{h}^\top \mathbf{R}_1 \boldsymbol{c}\right)$.

Under the conditions required to satisfy Lemma 1, Bollegala et al. [2] proved the following theorem.

Theorem 1. *Suppose that the entity embeddings satisfy* (1)*. Then, we have*

$$\log p(h, t \mid R) = \frac{\left|\left|{\mathbf{R}_1}^\top \boldsymbol{h} + {\mathbf{R}_2}^\top \boldsymbol{t}\right|\right|_2^2}{2d} - 2\log Z \pm \epsilon. \tag{4}$$

for $\epsilon = O(1/\sqrt{n}) + \widetilde{O}(1/d)$, where

$$Z = Z_c = Z_{c'}. \tag{5}$$

Theorem 1 states that the log-likelihood of observing R between h and t is related to the squared ℓ_2 norm of ${\mathbf{R}_1}^\top \boldsymbol{h} + {\mathbf{R}_2}^\top \boldsymbol{t}$. This provides an objective function for learning KGEs. Specifically, we can randomly initialise entity and relation embeddings and iterate such that above ℓ_2 norm is approximately equal to the empirical probabilities $p(h, t \mid R)$, estimated from a KG.

2.3 Inference in Knowledge Graphs

Lao et al. [12] used the Path Ranking Algorithm (PRA) [11] for predicting relations between two entities in a KG. The number of paths connecting two entities in a large KG can be large and it is difficult to systematically enumerate them all. Instead, PRA performs random walks and selects paths that cover most of the entities in a given training set. Next, the likelihood of a path is computed as the product of the transition probabilities when moving from one vertex to another. Neelakantan et al. [15] used PRA to find paths connecting entity pairs and ran a recurrent neural net (RNN) to combine the vector embeddings of relations to compose an embedding for the relation between two entities. They

learnt separate RNNs for each relation type, but also proposed a zero-shot [13] version, where they used pre-trained KGEs and learn a single composition function. However, the performance of their zero-shot model was significantly worse than the relation-specific model.

Guu et al. [6] considered path queries in a knowledge graph connecting two entities and proposed a composition method that multiplies the relation embedding matrices corresponding to the relations along the connecting path. They considered relation composition under the TransE model [3], where relational embedding vectors are added, and under the bilinear-diagonal model [28], where relations are represented using diagonal matrices. Both these composition operators can be seen as *unsupervised* in the sense that there are no learnable parameters in the composition function. In our experiments, we use both matrix addition and multiplication as unsupervised baseline methods for comparisons. On the other hand, our proposed method is a supervised relation composition method and we consider relations represented by orthogonal matrices, which are not diagonal in general.

3 Relation Composition

Let us assume that the two relations R_A and R_B jointly imply a third relation R_C. In this paper, we use the notation $R_A \wedge R_B \Rightarrow R_C$ to express this fact. Moreover, let us assume that the relational embeddings produced by RelWalk for R_A and R_B to be respectively $(\mathbf{R}_1^A, \mathbf{R}_2^A)$ and $(\mathbf{R}_1^B, \mathbf{R}_2^B)$. For simplify the explanation, let us assume all relation embedding matrices are in $\mathbb{R}^{d \times d}$. We model the problem of composing a relation embedding $(\hat{\mathbf{R}}_1^C, \hat{\mathbf{R}}_2^C)$ for R_C as learning two joint compositional operators (ϕ_1, ϕ_2) such that:

$$\phi_1 : \mathbf{R}_1^A, \mathbf{R}_2^A, \mathbf{R}_1^B, \mathbf{R}_2^B \longrightarrow \hat{\mathbf{R}}_1^C \tag{6}$$

$$\phi_2 : \mathbf{R}_1^A, \mathbf{R}_2^A, \mathbf{R}_1^B, \mathbf{R}_2^B \longrightarrow \hat{\mathbf{R}}_2^C \tag{7}$$

3.1 Unsupervised Relation Composition

When the compositional operators ϕ_1, ϕ_2 do not have learnable parameters we call them *unsupervised*. In the case of matrix relation embeddings, we consider the following unsupervised operators.

Addition

$$\mathbf{R}_1^A + \mathbf{R}_1^B = \hat{\mathbf{R}}_1^C \tag{8}$$

$$\mathbf{R}_2^A + \mathbf{R}_2^B = \hat{\mathbf{R}}_2^C \tag{9}$$

Matrix Product

$$\mathbf{R}_1^A \mathbf{R}_1^B = \hat{\mathbf{R}}_1^C \tag{10}$$

$$\mathbf{R}_2^A \mathbf{R}_2^B = \hat{\mathbf{R}}_2^C \tag{11}$$

Hadamard Product

$$\mathbf{R}_1^A \odot \mathbf{R}_1^B = \hat{\mathbf{R}}_1^C \tag{12}$$

$$\mathbf{R}_2^A \odot \mathbf{R}_2^B = \hat{\mathbf{R}}_2^C \tag{13}$$

Here, $\odot$ denotes the Hadamard (elementwise) product of two matrices. Unlike the matrix product, both addition and Hadamard product are commutative.

3.2 Supervised Relation Composition

The unsupervised compositional operators described in subsection 3.1 are not guaranteed to correctly predict the embeddings because they cannot be tuned to the relations in a given KG. Moreover, each unsupervised operator considers either one of $\mathbf{R}_1$ or $\mathbf{R}_2$, and do not model their possible interactions. Therefore, we propose to learn two *supervised* relation composition operators with shared parameters. The parameter sharing enables the two operators to learn a consistent relation embedding.

Different models can be used to express ϕ_1 and ϕ_2. In this paper, we use feed-forward neural nets, which are universal approximators [7] for this purpose. We first linearise the input $d \times d$ matrix relation embeddings to d^2-dimensional vector embeddings via a linearisation operator $\mathfrak{L}$. We then concatenate the four linearised relational embeddings $\mathfrak{L}(\mathbf{R}_1^A), \mathfrak{L}(\mathbf{R}_2^A), \mathfrak{L}(\mathbf{R}_1^B), \mathfrak{L}(\mathbf{R}_2^B)$ and feed it to the neural net. The weight and bias for the first layer are respectively $\mathbf{W} \in \mathbb{R}^{4d^2 \times m}$ and $\boldsymbol{b} \in \mathbb{R}^m$, where m is the number of neurones in the hidden layer. A nonlinear activation function, f is applied at the hidden layer. In our experiments, we used tanh as the activation function. The weight and bias for the output layer, respectively $\mathbf{U} \in \mathbb{R}^{m \times 2d^2}$ and $\boldsymbol{b}' \in \mathbb{R}^{2d^2}$, are chosen such that by appropriately splitting the output into two parts and applying the inverse mapping of the linearisation, we can predict $\hat{\mathbf{R}}_1^C$ and $\hat{\mathbf{R}}_2^C$. Denoting the concatenation by $\oplus$ and inverse linearisation by $\mathfrak{L}^{-1}$, we can write the predicted embeddings for r_C as follows:

$$\boldsymbol{x} = \mathfrak{L}(\mathbf{R}_1^A) \oplus \mathfrak{L}(\mathbf{R}_2^A) \oplus \mathfrak{L}(\mathbf{R}_1^B) \oplus \mathfrak{L}(\mathbf{R}_2^B) \tag{14}$$

$$\boldsymbol{h} = f(\mathbf{W}\boldsymbol{x} + \boldsymbol{b}) \tag{15}$$

$$\boldsymbol{y} = \mathbf{U}\boldsymbol{h} + \boldsymbol{b}' \tag{16}$$

$$\hat{\mathbf{R}}_1^C = \mathfrak{L}^{-1}\boldsymbol{y}_{:d^2} \tag{17}$$

$$\hat{\mathbf{R}}_2^C = \mathfrak{L}^{-1}\boldsymbol{y}_{d^2:} \tag{18}$$

Using a training set of relational tuples $\{(R_A, R_B, R_C)\}$, where $R_A \wedge R_B \Rightarrow R_C$ and their RelWalk embeddings, using Adam [10], we find the network parameters that minimise the squared Frobenius norm given in (19).

$$L(\mathbf{W}, \mathbf{U}, \boldsymbol{b}, \boldsymbol{b}') = \left|\left|\mathbf{R}_1^C - \hat{\mathbf{R}}_1^C\right|\right|_2^2 + \left|\left|\mathbf{R}_2^C - \hat{\mathbf{R}}_2^C\right|\right|_2^2 \tag{19}$$

4 Experiments

4.1 Datasets

We use the FB15k-237 dataset[2] created by Toutanova and Chen [23] for training KGEs using RelWalk. This dataset was created by removing the reverse relations between train and test portions in the original FB15k dataset and is considered as a more appropriate benchmark dataset for evaluating KGEs. FB15k-237 dataset contains 237 relation types for 14541 entities. To preserve the asymmetry property for relations, we consider that each relation $R^{<}$ in the relation set has its inverse $R^{>}$, so that for each triple $(h, R^{<}, t)$ in the KG we regard $(t, R^{>}, h)$ is also in the KG. Thus as a total we have 474 relation types to be learnt (we call this extended version as FB15k-474). The train, test and validation parts of this dataset contains respectively 544230, 40932 and 35070 tuples. Following the recommendations by the authors, RelWalk is trained using 100 minibatches for 1000 epochs until convergence. Negative sampling rate is set to 50 and we learn KGEs of dimensionalities $d = 20, 50$ and 100. We consider two tasks to evaluate relational composition operators namely, relation composition (subsection 4.2) and triple classification task (subsection 4.3). The matrix relational embeddings produced by RelWalk are used in the subsequent experiments described in the paper when learning supervised compositional operators.

To evaluate the ability of a relation composition operator, we use the dataset created by Takahashi et al. [22] from FB15-23k as follows. For a relation R, they define the *content set* $\mathcal{C}(R)$ as the set of (h, t) pairs such that (h, R, t) is a fact in the KG. Likewise, they define $\mathcal{C}(R_A \wedge R_B)$ as the set of (h, t) pairs such that $(h, R_A \rightarrow R_B, t)$ is a path in the KG. Next, $R_A \wedge R_B \Rightarrow R_C$ is considered as a compositional constraint if their content sets are similar; that is, if $|\mathcal{C}(R_A \wedge R_B) \cap \mathcal{C}(R_C)| \geq 50$ and the Jaccard similarity between $\mathcal{C}(R_A \wedge R_B)$ and $\mathcal{C}(R_C)$ is greater than 0.4. They obtained 154 compositional constraints of the form $R_A \wedge R_B \Rightarrow R_C$ after this filtering process. We name this dataset as the Relation Composition (**RC**) dataset in the remainder of the paper.

We perform 5-fold cross validation on the RC dataset to train a supervised relation composition operator using our proposed method described in subsection 3.2. Using a separate validation dataset, we set the initial learning rate for Adam to 5E-4 and minibatch size to 25. We apply dropout with rate 0.5 and ℓ_2 regularisation with coefficient 1E-10 to avoid overfitting during training. For $d = 20$ dimensional embeddings, we use a single hidden layer of 300 neurones, whereas for $d = 50$ and 100 we used two hidden layers, where each has 600 neurones. In all settings training converged after 25000 epochs, which took less than 5 minutes in a single GPU instance available in the Google Colab cloud-based ipython notebooks[3]. The source code implementation of the proposed method is available[4].

[2] `https://www.microsoft.com/en-us/download/details.aspx?id=52312`
[3] `https://colab.research.google.com/`
[4] `https://github.com/Bollegala/RelComp`

4.2 Relation Composition

Let us assume that the composition of the two relations R_A and R_B is the relation R_C. Moreover, let us denote the pre-trained RelWalk embeddings for a relation R_x to be $\mathbf{R}_1^x$ and $\mathbf{R}_2^x$, where $x \in \{A, B, C\}$. We will denote the composed embedding for R_C by $\hat{\mathbf{R}}_1^C$ and $\hat{\mathbf{R}}_2^C$.

Following Takahashi et al. [22], we rank the test relations R_L by its similarity to $\hat{R}_C$, the composed version of R_C using the distance function, $d(R_L, \hat{R}_C)$, given by (20).

$$d(R_L, \hat{R}_C) = \left|\left|\mathbf{R}_1^L - \hat{\mathbf{R}}_1^C\right|\right|_F + \left|\left|\mathbf{R}_2^L - \hat{\mathbf{R}}_2^C\right|\right|_F \tag{20}$$

If the R_C is ranked higher for $\hat{R}_C$, then it is considered better. We use Mean Rank (MR), Mean Reciprocal Rank (MRR) and Hits@10 to measure the accuracy of the composition.

Table 1 presents the average performance of relation compositions using 5-folds cross validation on **RC** compositional constraints. We consider the 474 relation types in FB15K-474 for this evaluation. Lower MR indicates better performance. As can be observed, the supervised relation composition achieves the best results for MR, MRR and Hits@10 with significant improvements over the unsupervised compositional operators. In fact, MR, MRR and Hits@10 results for the unsupervised operators are close to the random baseline.

Table 1. Performance in the relation composition task.

	d=20			d=50			d=100		
Method	MR	MRR	Hits@10	MR	MRR	Hits@10	MR	MRR	Hits@10
Supervised Relation Composition	**75**	**0.412**	**0.581**	**64**	**0.390**	**0.729**	**49**	**0.308**	**0.703**
Addition	238	0.010	0.012	250	0.008	0.019	247	0.007	0
Matrix Product	225	0.018	0.032	233	0.012	0.025	231	0.010	0.019
Hadamard Product	215	0.020	0.051	192	0.037	0.051	209	0.016	0.032

4.3 Triple Classification

To evaluate the effectiveness of the learnt operator for generating composed relation embeddings, we consider the triple classification task using the composed embeddings for R_C. Triple classification task is originally proposed by Socher et al. [20], and aims to predict whether a triple (h, R, t) is a valid triple or not given entity and relation embeddings and a scoring function that map the embeddings to a confidence score. Specifically, in this paper, we use the embeddings learnt by RelWalk for the entities and the relations in FB15k-474 and the joint probability $p(h, R, t)$ given by Theorem 1 to determine whether a relation R exists between

two given entities h and t. We need positive and negative triples for classification. The negative triples are generated by randomly corrupting entities of the positive examples. For example, for a test triple (h, R, t), we consider (h, R, t') as a negative example where t' is sampled from all entities that appear in the corresponding argument in the entire KG.

We perform 5 folds cross-validation on **RC** compositional constraints. Once the proposed supervised relation composition is learnt using a training set, we perform triple classification for those triples in FB15K-474 testing set that are connected by the relation types in the test compositional constraints of **RC**. We evaluate the performance using the accuracy which is the percentage of the correctly classified test triples. We use the validation set to find a threshold θ for each test relation such that if $p(h, R, t) > \theta$, the relation (h, R, t) holds, otherwise we consider it as a negative triple.

The performance of the supervised and unsupervised relational compositional operators for triple classification is shown in Table 2. Across the relational compositional operators and for different dimensionalities, the proposed supervised relational composition method achieves the best accuracy for this task. Despite increasing the dimensionality of relation embeddings from 20 to 100 leading to a complex model with a large number of parameters to be tuned using a small set of compositional constraints as in **RC**, the trained operator shows better performance in all cases.

Table 2. Triple classification accuracy for the different relational compositional operators.

Method	d=20	d=50	d=100
Supervised Relation Composition	**77.55**	**77.73**	**77.62**
Addition	68.9	70.44	69.45
Matrix Product	67.6	65.24	75.71
Hadamard Product	58.44	63.01	70.94

5 Conclusion

In this paper, we addressed the problem of composing pre-trained relation embeddings in KGs. Given a set of compositional constraints over relations in the form $R_A \wedge R_B \Rightarrow R_C$, our proposed method learns a supervised operator that maps the relation embeddings of two relation to a new relation embedding. The learnt operator can be used to infer relation embeddings for rare or unseen relation types. Evaluating the predicted relation embeddings for triple classification task indicates the effectiveness of the proposed relation composition method.

Acknowledgement

We would like to thank Ran Tian for sharing the relation composition benchmark dataset.

References

1. Bollacker, K., Evans, C., Paritosh, P., Sturge, T., Taylor, J.: Freebase: a collaboratively created graph database for structuring human knowledge. In: Proc. of SIGMOD. pp. 1247 – 1250 (2008)
2. Bollegala, D., Hakami, H., Yoshida, Y., Kawarabayashi, K.i.: Relwalk – a latent variable model approach to knowledge graph embedding (2019), `https://openreview.net/forum?id=SkxbDsR9Ym`
3. Bordes, A., Usunier, N., Garcia-Durán, A., Weston, J., Yakhenko, O.: Translating embeddings for modeling multi-relational data. In: Proc. of NIPS (2013)
4. Bordes, A., Weston, J., Collobert, R., Bengio, Y.: Learning structured embeddings of knowledge bases. In: Proc. of AAAI (2011)
5. Ding, B., Wang, Q., Wang, B., Guo, L.: Improving knowledge graph embedding using simple constraints. In: Proc. of ACL. pp. 110–121 (2018)
6. Guu, K., Miller, J., Liang, P.: Traversing knowledge graphs in vector space. In: Proc. of EMNLP. pp. 318–327 (2015)
7. Hornik, K.: Multilayer feedforward networks are universal approximators. Neural Networks **2**, 359–366 (1989)
8. Ji, G., He, S., Xu, L., Liu, K., Zhao, J.: Knowledge graph embedding via dynamic mapping matrix. In: Proc. of ACL. pp. 687–696 (2015)
9. Ji, G., Liu, K., He, S., Zhao, J.: Knowledge graph completion with adaptive sparse transfer matrix. In: Proc. of AAAI. pp. 985–991 (2016)
10. Kingma, D.P., Ba, J.L.: Adam: A method for stochastic optimization. In: Proc. of ICLR (2015)
11. Lao, N., Cohen, W.W.: Relational retrieval using a combination of path-constrained random walks. Machine Learning **81**(1), 53–67 (Jul 2010). https://doi.org/10.1007/s10994-010-5205-8
12. Lao, N., Mitchell, T., Cohen, W.W.: Random walk inference and learning in a large scale knowledge base. In: Proc. of EMNLP. pp. 529–539 (2011)
13. Larochelle, H., Erhan, D., Bengio, Y.: Zero-data learning of new tasks. In: Proc. of AAAI. pp. 646–651 (2008)
14. Lin, Y., Liu, Z., Sun, M., Liu, Y., Zhu, X.: Learning entity and relation embeddings for knowledge graph completion. In: Proc. of AAAI. pp. 2181–2187 (2015)
15. Neelakantan, A., Roth, B., McCallum, A.: Compositional vector space models for knowledge base completion. In: Proc. of ACL. pp. 156–166 (2015)
16. Nguyen, D.Q., Sirts, K., Qu, L., Johnson, M.: Stranse: a novel embedding model of entities and relationships in knowledge bases. In: Proc. of NAACL-HLT. pp. 460–466 (2016)
17. Nickel, M., Rosasco, L., Poggio, T.: Holographic embeddings of knowledge graphs. In: Proc. of AAAI (2016)
18. Nickel, M., Tresp, V., Kriegel, H.P.: A three-way model for collective learning on multi-relational data. In: Proc. of ICML. pp. 809–816 (2011)
19. Riedel, S., Yao, L., McCallum, A., Marlin, B.M.: Relation extraction with matrix factorization and universal schemas. In: Proc. of NAACL. pp. 74–84 (2013)

20. Socher, R., Chen, D., Manning, C.D., Ng, A.: Reasoning with neural tensor networks for knowledge base completion. In: Advances in neural information processing systems. pp. 926–934 (2013)
21. Socher, R., Chen, D., Manning, C.D., Ng, A.Y.: Reasoning with neural tensor networks for knowledge base completion. In: Proc. of NIPS (2013)
22. Takahashi, R., Tian, R., Inui, K.: Interpretable and compositional relation learning by joint training with an autoencoder. In: Proc. of ACL. pp. 2148–2159 (2018)
23. Toutanova, K., Chen, D.: Observed versus latent features for knowledge base and text inference. In: Proc. of 3rd Workshop on Continuous Vector Space Models and their Compositionality. pp. 57–66 (2015)
24. Trouillon, T., Welbl, J., Riedel, S., Gaussier, É., Bouchard, G.: Complex embeddings for simple link prediction. In: Proc. of ICML (2016)
25. Wang, Q., Mao, Z., Wang, B., Guo, L.: Knowledge graph embedding: A survey of approaches and applications. IEEE Transactions on Knowledge and Data Engineering **29**(12), 2724–2743 (Dec 2017). https://doi.org/10.1109/TKDE.2017.2754499
26. Wang, Z., Zhang, J., Feng, J., Chen, Z.: Knowledge graph embedding by translating on hyperplanes. In: Proc. of AAAI. pp. 1112 – 1119 (2014)
27. Xiao, H., Huang, M., Zhu, X.: TransG : A generative model for knowledge graph embedding. In: Proc. of ACL. pp. 2316–2325 (2016)
28. Yang, B., Yih, W.t., He, X., Gao, J., Deng, L.: Embedding entities and relations for learning and inference in knowledge bases. In: ICLR (2015)
29. Yoon, H.G., Song, H.J., Park, S.B., Park, S.Y.: A translation-based knowledge graph embedding preserving logical property of relations. In: Proc. of NAACL. pp. 907–916 (2016)