Association for Information Systems
AIS Electronic Library (AISeL)

PACIS 2017 Proceedings

Pacific Asia Conference on Information Systems (PACIS)

Summer 2017

Learning Contextual Embeddings for Knowledge Graph Completion

Changsung Moon
North Carolina State University at Raleigh, cmoon2@ncsu.edu

Steve Harenberg
North Carolina State University Raleigh, sdharenb@ncsu.edu

John Slankas
Laboratory of Analytic Sciences Raleigh, jbslanka@ncsu.edu

Nagiza F. Samatova
North Carolina State University Raleigh, samatova@csc.ncsu.edu

Learning Contextual Embeddings for Knowledge Graph Completion

Completed Research Paper

Changsung Moon
North Carolina State University
Raleigh, NC 27695, USA
cmoon2@ncsu.edu

Steve Harenberg
North Carolina State University
Raleigh, NC 27695, USA
sdharenb@ncsu.edu

John Slankas
Laboratory of Analytic Sciences
Raleigh, NC 27695, USA
jbslanka@ncsu.edu

Nagiza F. Samatova*
North Carolina State University
Raleigh, NC 27695, USA
Oak Ridge National Laboratory
Oak Ridge, TN 37831, USA
samatova@csc.ncsu.edu

Abstract

Knowledge Graphs capture entities and their relationships. However, many knowledge graphs are afflicted by missing data. Recently, embedding methods have been used to alleviate this issue via knowledge graph completion. However, most existing methods only consider the relationship in triples, even though contextual relation types, consisting of the surrounding relation types of a triple, can substantially improve prediction accuracy. Therefore, we propose a contextual embedding method that learns the embeddings of entities and predicates while taking contextual relation types into account. The main benefits of our approach are: (1) improved scalability via a reduced number of epochs needed to achieve comparable or better results with the same memory complexity, (2) higher prediction accuracy (an average of 14%) compared to the related algorithms, and (3) high accuracy for both missing entity and predicate predictions. The source code and the YAGO43k dataset of this paper can be found from (https://github.ncsu.edu/cmoon2/kg).

Keywords: Knowledge Graph Completion, KG Embedding Method, Translation-based Model, Vector Embedding, Contextual Embedding

Introduction

Knowledge Graphs (KGs) have recently become mainstream in IT companies, such as Google, Facebook, Microsoft, and Yahoo, as a way of storing information about entities and enhancing search results based on the captured semantic information. KGs store information as SPO triples, with each triple consisting of two entities (subject and object) and the relationship (predicate), for example (*Star Wars*, *genre*, *Science Fiction*). Naturally, a KG can be modeled as a directed multigraph, with nodes representing entities and each edge labeled by the relation type. An example KG is shown in Figure 1.

Although KGs have been constructed in various ways, such as public collaborative efforts and curated approaches by closed groups, **KGs often suffer from numerous missing relationships**. For

* Corresponding author.

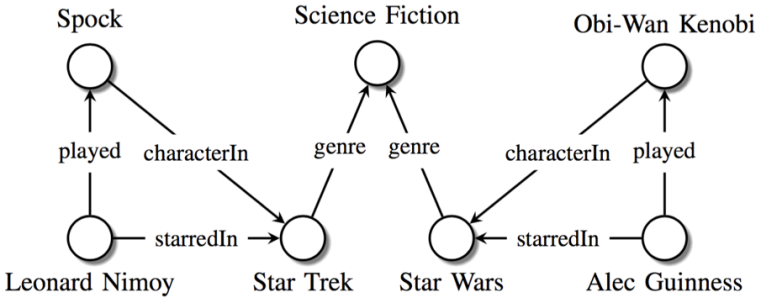


Figure 1. Sample Knowledge Graph. Figure adapted from (Nickel et al. 2016a).

example, every person has a place of birth; however, 71% of all people do not have place of birth data in the Freebase KG (West et al. 2014). Similarly, 75% of people have no known nationality in this dataset. Missing relationships such as these can disrupt semantic-based search (Guu et al. 2015). For example, missing nationality relationships would inhibit even a simple path query such as *"What languages are spoken by people living in Lisbon?"*. Therefore, it is critical to develop methods that address the task of *KG completion* - the prediction of missing entities and missing relation types.

Existing methods that address the task of KG completion can fall into one of two categories (Nickel et al. 2016a): (1) graph feature models that use features directly extracted from observed edges in a graph (Muggleton 1995; Galárraga et al. 2013) (e.g., rule mining, inductive logic programming, etc.) and (2) latent feature models that use features not directly observed in a graph (e.g., semantic embeddings, matrix factorization, tensor factorization, etc.). There have been a number of methods based on graph feature models, though these have scalability limitations and often do not work for large KGs. In contrast, latent feature models have been gaining considerable attention because they scale to large KGs by representing features of entities and relation types in a low-dimensional vector space. We call these features the "embeddings" because they are not directly observed in the data; rather, they are automatically inferred from the training data. In light of the improved scalability of latent feature approaches, we developed our method in this space via semantic embeddings.

Recently proposed KG embedding methods (Bordes et al. 2011; Nickel et al. 2011; Bordes et al. 2013; Dong et al. 2014; Wang et al. 2014b; Lin et al. 2015; Nickel et al. 2016b) can discover new knowledge via KG completion. However, most of them only capture the relationship in a triple, even though contextual relation types (i.e., surrounding relation types) of the triple can be used to significantly improve the accuracy of the discovery. For example, in our test datasets, over 97% of the predicates between S and O have already been used as a predicate of S to a different object or as a predicate of O from a different subject. Hence, the predictions using these methods are made without regard to valuable information contained in the KG. There are also recent KG embedding methods (Socher et al. 2013; Wang et al. 2014a; Zhong et al. 2015; Xie et al. 2016a; 2016b), which use rich information like text and entity type. However, usually, they are more complex to deal with the additional information, and the rich data might not be available for some KGs.

To address these issues, we introduce our method, Contextual Embeddings (ContE). The basic idea of our method is that ContE includes outgoing (from the subject) and incoming (to the object) relation types of a triple as *contextual relation types* or *contexts*, which are shown in Figure 2, into the embedding learning process. Thus, as a probable missing relation type between two entities, ContE considers predicates that have appeared in the outgoing relation types of the subject or incoming relation types of the object. In summary, our main contributions are as follows:

- **Our approach requires a reduced number of training epochs** to achieve comparable or better results without increasing the memory complexity. For example, for entity prediction on the FB15k dataset, ContE needs to run only 1,200 epochs to achieve the optimal result of the next best method, which requires 3,300 epochs. Our method can also scale to large KGs, taking linear time in the size of KG and storing only a low-dimensional vector for each entity and relation type.
- **We perform extensive experimental comparisons** of ContE against other state-of-the-art KG embedding methods on two real-world datasets for relation type prediction and entity prediction. In all experiments, ContE had a consistently higher accuracy (an average of 14% against the state-of-the-art methods).

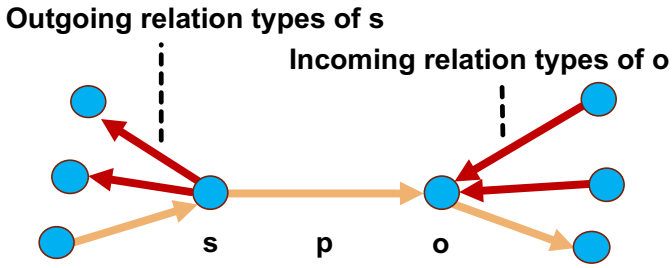


Figure 2. Contextual Relation Types for a Triple (*s, p, o*) in a KG

- **Our proposed method shows high accuracy for both entity and relation type predictions**, while some baseline methods show a high accuracy for one and a low accuracy for the other. To achieve better KG completion, it is critical to improve the accuracy performance for the both predictions.

Problem Definition

The problem of **KG completion** (i.e., relation type and entity predictions in knowledge graphs) can be formally defined as follows: Let $E = \{e_1, e_2, \cdots, e_n\}$ be a set of all entities, and let $R = \{r_1, r_2, \cdots, r_m\}$ be a set of all relation types. A triple can be denoted by (s, p, o) where $s, o \in E$ and $p \in R$, with s indicating the subject, p representing the predicate, and o being the object. The problem of relation type prediction is to determine the score or the probability $\psi(p = r_j \mid s, o)$, for all $r_j \in R$. The problem of entity prediction is to determine the score or the probability $\psi(s = e_i \mid p, o)$ or $\psi(o = e_i \mid s, p)$, for all $e_i \in E$.

For our analysis, the scores or the probabilities are ranked to compute the *Hits@N* prediction and the mean reciprocal rank (MRR). *Hits@N* prediction refers to the problem of determining the top-*N* most probable subjects, predicates, or objects for a given triple that has a missing entity or relation type. MRR is a statistical measure for evaluating a ranking process and is defined as:

$$MRR = \frac{1}{|Q|}\sum_{i=1}^{|Q|}\frac{1}{rank_i} \qquad (1)$$

where Q is a set of test triples, and $rank_i$ is the rank position of the true entity or relation type for the i-th triple.

Related Work

Translating Embeddings

Translation-based models have shown great improvements in KG completion. In TransE (Bordes et al. 2013), a predicate p is regarded as a translation from a subject s to an object o. In other words, when the triple (s, p, o) holds, TransE wants $\boldsymbol{e}_s + \boldsymbol{r}_p \approx \boldsymbol{e}_o$ where $\boldsymbol{e}_s, \boldsymbol{e}_o \in \mathbb{R}^k$ are the vector embeddings of the entities (s and o) and $\boldsymbol{r}_p \in \mathbb{R}^k$ is the vector embedding of the predicate (p). The score function is defined as follows:

$$\psi(s, p, o) = -d(\boldsymbol{e}_s + \boldsymbol{r}_p, \boldsymbol{e}_o) \qquad (2)$$

where $d()$ is a dissimilarity measure that is either the L_1 or the L_2-norm. When the triple (s, p, o) holds, the vector of $\boldsymbol{e}_s + \boldsymbol{r}_p$ should be close to $\boldsymbol{e}_o$. Otherwise, $\boldsymbol{e}_s + \boldsymbol{r}_p$ should be far away from $\boldsymbol{e}_o$. TransE is a simple and efficient model, but it has issues in dealing with relation type prediction and *1-to-N*, *N-to-1* and *N-to-N* triples for entity prediction.

To address these issues, TransH (Wang et al. 2014b) introduces a mechanism of projecting to relation-specific hyperplanes to enable an entity to have different embeddings in different triples. TransR (Lin et al. 2015) models entities and relation types in distinct entity and relation type spaces, and performs translation in relation type spaces. The score function of TransR is as follows:

$$\psi(s, p, o) = -d(\boldsymbol{e}_s^T\boldsymbol{M}_p + \boldsymbol{r}_p^T, \boldsymbol{e}_o^T\boldsymbol{M}_p) \qquad (3)$$

where $d()$ is a dissimilarity measure given by the L_1-norm, and $\boldsymbol{M}_p \in \mathbb{R}^{k\times k}$ is a projection matrix that

Method	Memory Complexity
Rescal	$\mathcal{O}(n_e k + n_r k^2)$
TransE	$\mathcal{O}(n_e k + n_r k)$
TransH	$\mathcal{O}(n_e k + 2n_r k)$
TransR	$\mathcal{O}(n_e k + n_r k + n_r k^2)$
TKRL	$\mathcal{O}(n_e k + n_r k + n_m k^2)$
HolE	$\mathcal{O}(n_e k + n_r k)$
ContE	$\mathcal{O}(n_e k + n_r k)$

Table 1. Memory Complexity of Embedding Models (n_e: \# of entities, n_r: \# of relation types, n_m: \# of projection matrices of all sub-types, and k: \# of dimensions)

projects entities from entity space to relation type space.

TransH and TransR improve the performance of TransE on *1-to-N*, *N-to-1* and *N-to-N* as well as *1-to-1* triples. However, they lose the simplicity and efficiency of TransE, leading to the higher memory complexities as shown in Table 1. Besides, they only concentrate on interactions within a triple. TKRL (Xie et al. 2016b) uses hierarchical entity type information as additional supplements to learn embeddings. It enables an entity to have multiple representations in different types. However, it is also complicated, with a higher memory complexity than TransE and requires additional data (i.e., well-defined hierarchical entity types), which might not be available for some KGs.

Other Models

Rescal (Nickel et al. 2011) is a collective matrix factorization method that captures the interactions between two entities via pairwise interactions of embedding vectors. The score of a triple (*s, p, o*) is modeled as:

$$\psi(s,p,o) = \boldsymbol{e}_s^T \boldsymbol{W}_p \boldsymbol{e}_o = \sum_{a=1}^{k}\sum_{b=1}^{k} \boldsymbol{w}_{abp} \boldsymbol{e}_{sa} \boldsymbol{e}_{ob} \qquad (4)$$

where $\boldsymbol{W}_p \in \mathbb{R}^{k \times k}$ is a weight matrix and $\boldsymbol{w}_{abp}$ indicates how much the latent features a and b interact with the relation type p. In our experiments, Rescal shows quite good results for relation type prediction, but it requires tuning many more parameters than other memory efficient models.

HolE (Nickel et al. 2016b) employs a circular correlation of entity embeddings to create compositional representations. The probability of a triple is modeled as follows:

$$\psi(s,p,o) = \sigma(\boldsymbol{r}_p^T(\boldsymbol{e}_s * \boldsymbol{e}_o)) \qquad (5)$$

where $\sigma(z) = (1 + exp(-z))^{-1}$, and $*$ indicates the circular correlation:

$$(\boldsymbol{x} * \boldsymbol{y})[j] = \sum_{i=0}^{k-1} \boldsymbol{x}[i]\, \boldsymbol{y}[j+i] \; mod \; k \qquad (6)$$

with k indicating the number of dimensions of a vector and j indicating the index of the output vector starting with 0. HolE can capture richer interactions between entities, but it has issues in capturing interactions between relation types and has a relatively slow training time.

Method

In this section, we propose a novel model for knowledge graph completion. Given a training triple (*s, p, o*), our model learns vector embeddings of outgoing relation types of *s* and incoming relation types of *o*

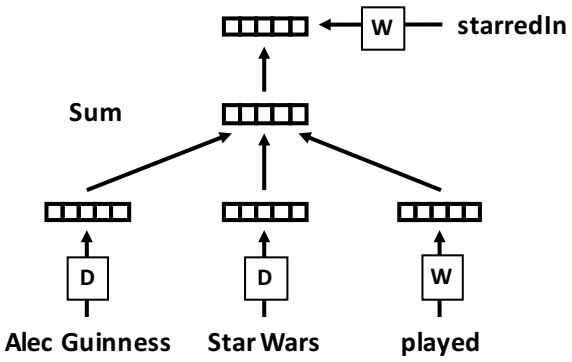


Figure 3. A framework for learning vectors of entities and relation types

as well as *s*, *p* and *o*. We label the outgoing relation types of *s* as $C_{out(s)}$ and the incoming relation types of *o* as $C_{in(o)}$. The set of both these relationships are referred to as the *contextual relation types* or *contexts* and denoted by $C_{(s,o)} = C_{out(s)} \cup C_{in(o)}$. ContE interprets contexts as translating operations between entities (subject and object) and the predicates between them. If a predicate has often appeared with a certain set of contexts, the predicate tends to appear again between two entities having a similar set of contexts.

To reflect these observations in our model, we include $C_{(s,o)}$ in the embedding learning process for a triple as follows:

$$\boldsymbol{e}_s + \boldsymbol{e}_o + \boldsymbol{r}_c \approx \boldsymbol{r}_p \quad for\ \forall c \in C_{(s,o)} \tag{7}$$

where *c* is one of contextual relation types (as shown in Figure 2), and $\boldsymbol{e}_s, \boldsymbol{e}_o, \boldsymbol{r}_c\ and\ \boldsymbol{r}_p \in \mathbb{R}^k$ are vector embeddings of the entities (*s* and *o*) and the relation types (*c* and *p*), respectively. When the triple (*s, p, o*) holds, $\boldsymbol{r}_p$ should be close to $\boldsymbol{e}_s + \boldsymbol{e}_o + \boldsymbol{r}_c$, otherwise $\boldsymbol{e}_s + \boldsymbol{e}_o + \boldsymbol{r}_c$ should be far away from $\boldsymbol{r}_p$. To calculate the dissimilarity $d(\boldsymbol{x}, \boldsymbol{y})$ between two vectors $\boldsymbol{x}$ and $\boldsymbol{y}$, we use the L_1-norm as follows:

$$\boldsymbol{t} = \boldsymbol{e}_s + \boldsymbol{e}_o + \boldsymbol{r}_c \tag{8}$$

$$\eta_{spoc} = d(\boldsymbol{t}, \boldsymbol{r}_p) = \sum_{i=1}^{k} | \boldsymbol{t}[i] - \boldsymbol{r}_p[i] | \tag{9}$$

We can then use the negative of the distance, $-d(\boldsymbol{t}, \boldsymbol{r}_p)$, to measure how much $\boldsymbol{e}_s + \boldsymbol{e}_o + \boldsymbol{r}_c$ and $\boldsymbol{r}_p$ are similar. This similarity function is used for training embeddings and making predictions in our model.

Figure 3 shows our model with an example triple (*Alec Guinness*, *starredIn*, *Star Wars*) and one of contextual relation types, *played*. Every entity is mapped to a unique vector, which is a row in the matrix *D*. Every relation type is also mapped to a unique vector, which is represented by a row in the matrix *W*. The vectors of the entities (i.e., $\boldsymbol{e}_{\text{Alec Guinness}}$ and $\boldsymbol{e}_{\text{Star Wars}}$) and the context (i.e., $\boldsymbol{r}_{\text{played}}$) are combined as $\boldsymbol{e}_{\text{Alec Guinness}} + \boldsymbol{e}_{\text{Star Wars}} + \boldsymbol{r}_{\text{played}}$, and we compute the similarity between the combined vector and the vector of the predicate (i.e., $\boldsymbol{r}_{\text{starredIn}}$). Our model trains these vectors so that this similarity value will be higher. The vectors of the entities and the predicate are shared across all contexts in the training procedure for the triple, and the contexts are used as translating operations between the entities and the predicate. By doing this, we can capture interactions between relation types as well as interactions within triples.

The main advantages of our framework are as follows: (1) the structure is very simple, (2) it requires a low memory complexity that is linear in the dimensionality *k* of entity and relation type vectors (see Table 1), and (3) a new entity or relation type can be easily added and trained by adding a new row into the matrix *D* or *W* respectively.

Negative Sampling

Negative sampling, which was introduced by (Mikolov et al. 2013), makes a model differentiate positive samples from negative samples by means of logistic regression. It has been successfully applied to various areas such as Natural Language Processing, KG embedding, etc. For this reason, we also use negative samples, in addition to using the existing triples. Our negative sampling is defined as follows:

$$S'_{(s,p,o)} = \{(s', p, o) \mid s' \in E\} \cup \{(s, p, o') \mid o' \in E\} \cup \{(s, p', o) \mid p' \in R\} \quad (10)$$

That is, for each triple, we extract the set of negative samples with either the subject or object replaced by a random entity or the predicate by a random relation type. These negative samples are triples that have never appeared in the training set. For example, given a training triple (*s*, *p*, *o*) = (*Stephen_Hawking*, *hasWonPrize*, *Wolf_Prize*), its negative samples could be (*New_York*, *hasWonPrize*, *Wolf_Prize*), (*Stephen_Hawking*, *hasWonPrize*, *Finland*) and (*Stephen_Hawking*, *isLocatedIn*, *Wolf_Prize*). The following loss function (Equation 11) is used to distinguish each training triple from its negative samples by making the rank of the training triple higher than all of the triples in the negative samples.

Margin-Based Ranking Loss Function

To learn vector embeddings, we minimize the following margin-based ranking loss function over the training set:

$$\begin{aligned} \mathcal{L} = \sum_{(s,p,o)\in S} \Bigg(& \sum_{(s',p,o)\in S'_{(s,p,o)}} \sum_{c\in C_{in(o)}} \max\left(0, \gamma + \eta_{spoc} - \eta_{s'poc}\right) \\ & + \sum_{(s,p,o')\in S'_{(s,p,o)}} \sum_{c\in C_{out(s)}} max(0, \gamma + \eta_{spoc} - \eta_{spo'c}) \\ & + \sum_{(s,p',o)\in S'_{(s,p,o)}} \sum_{c\in C_{(s,o)}} max(0, \gamma + \eta_{spoc} - \eta_{sp'oc}) \quad \Bigg) \end{aligned} \quad (11)$$

where S is the set of positive triples, $S'_{(s,p,o)}$ is the set of negative samples of the triple (*s, p, o*), C is the set of contextual relation types of the triple, and $\gamma > 0$ is the margin, and η_{spoc} is the dissimilarity between $\boldsymbol{e}_s + \boldsymbol{e}_o + \boldsymbol{r}_c$ and $\boldsymbol{r}_p$ (see Equation 9).

For each positive triple, we generate three kinds of negative samples, and a different set of contextual relation types is used for each kind of negative sample. For the negative samples with s' (i.e., *(s', p, o)*), we use only incoming relation types of *o*, $C_{in(o)}$, as the contextual relation types, and do not include $C_{out(s')}$. The reason is that the outgoing relation types of s' should not be treated as negative contexts to the predicate and the object even though s' is used as the negative. A similar reasoning is applied to the negative samples having o' (i.e., *(s, p, o')*). Only outgoing relation types of *s*, $C_{out(s)}$, are used as the contextual relation types. For the negative samples having p' (i.e., *(s, p', o)*), we use the full set of contextual relation types $C_{(s,o)}$.

Stochastic gradient descent (SGD) with AdaGrad (Duchi et al. 2011) is used as our optimization approach to minimize this loss function. This optimization algorithm adapts the learning rate by performing larger updates for infrequent entities and relation types and smaller updates for frequent entities and relation types.

Prediction

For KG completion, we make three kinds of predictions (the latter two being entity predictions): (1) relation type, (2) subject, and (3) object predictions. For each prediction, we use a different set of contextual relation types.

Given two entities *s* and *o*, we predict the relation type for the predicate *p* by using the following score function:

$$\psi(p = r_j \mid s, o) = \frac{1}{\mid C_{(s,o)} \mid} \sum_{c\in C_{(s,o)}} -\eta_{spoc} \quad for\ \forall r_j \in R \quad (12)$$

This score function computes the average similarity over all contextual relation types for each predicate.

To predict the subject given *p* and *o*, we use the following score function:

$$\psi(s = e_i \mid p, o) = \frac{1}{\mid C_{in(o)} \mid} \sum_{c\in C_{in(o)}} -\eta_{spoc} \quad for\ \forall e_i \in E \quad (13)$$

Dataset	FB15k	YAGO43k
Relation types	1,345	37
Entities	14,951	42,975
Training triples	483,142	331,687
Validation triples	50,000	30,000
Test triples	59,071	30,000

Table 2. Statistics of Datasets

This score function is similar to the score function for relation type prediction, but we use only $C_{in(o)}$ to compute the score for each subject candidate e_i. Each entity is connected with a various number of relation types (e.g., maximum: 127 relation types and minimum: 1 relation type in the FB15k dataset). If we added outgoing relation types of e_i into the set of contexts, then a different number of contexts is used to calculate the score, causing a score normalization issue. As such, it would be possible that an entity could have the highest score with only a few training contexts even though the entity is not the answer. Other probable entities could have lower scores due to having more training contexts but a small number of negative contexts. To avoid this issue, we fix the contexts with the incoming relation types of *o*.

The score function for the object prediction is similar to the subject prediction and is defined as follows:

$$\psi(o = e_i \mid s, p) = \frac{1}{\mid C_{out(s)} \mid} \sum_{c \in C_{out(s)}} -\eta_{spoc} \quad for\ \forall e_i \in E \quad (14)$$

The only difference is that only the outgoing relation types of the subject *s* are used as the contexts for the same reason as stated above.

Empirical Evaluation

In this section, we first describe the datasets used for evaluation and the baseline methods. Then, we outline our parameter optimization and, finally, discuss our experimental results.

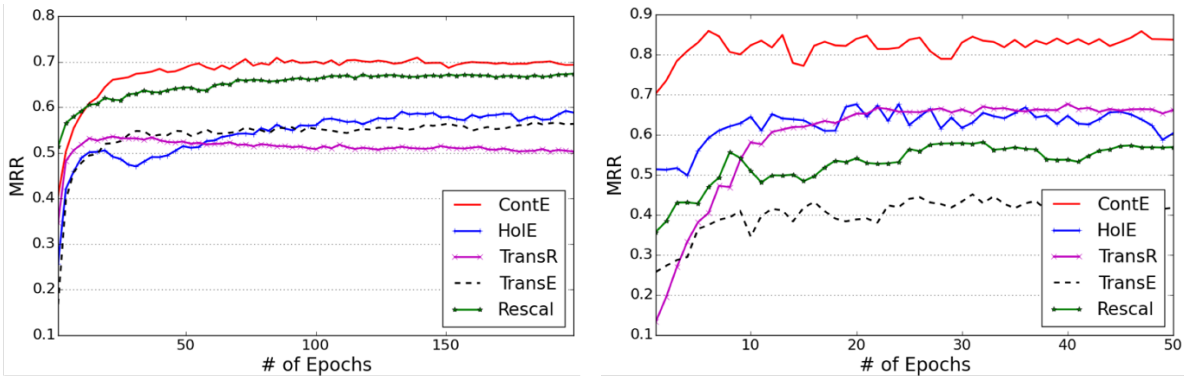
Datasets

ContE was evaluated on two real-world datasets that were extracted from Freebase and YAGO knowledge bases. The statistics of the datasets are given in Table 2. The main statistical differences being that FB15k has a much larger number of relation types, while YAGO43k has a much larger number of entities.

- **FB15k** (https://everest.hds.utc.fr/doku.php?id=en:transe) - Freebase is a huge knowledge graph of general facts. We used a subset of Freebase, called FB15k, which was introduced in (Bordes et al. 2013). It contains 14,951 entities, 1,345 relation types, 483,142 training, 50,000 validation, and 59,071 test triples. An example of the triple is (*/m/0677ng*, */music/artist/genre*, */m/03ckfl9*).
- **YAGO43k** - The core facts of YAGO3 (http://www.yago-knowledge.org) currently consist of 5,628,166 triples, 2,634,336 entities, and 37 relation types. For our experiments, we created a new dataset, called YAGO43k, by selecting triples that have entities appearing at least 20 times in the core facts of YAGO3, resulting in 391,687 triples, 42,975 entities, and 37 relation types. We split this dataset into 331,687 training, 30,000 validation, and 30,000 test triples. An example of the triple is (*156a81d*, *isLocatedIn*, *1lfplq*).

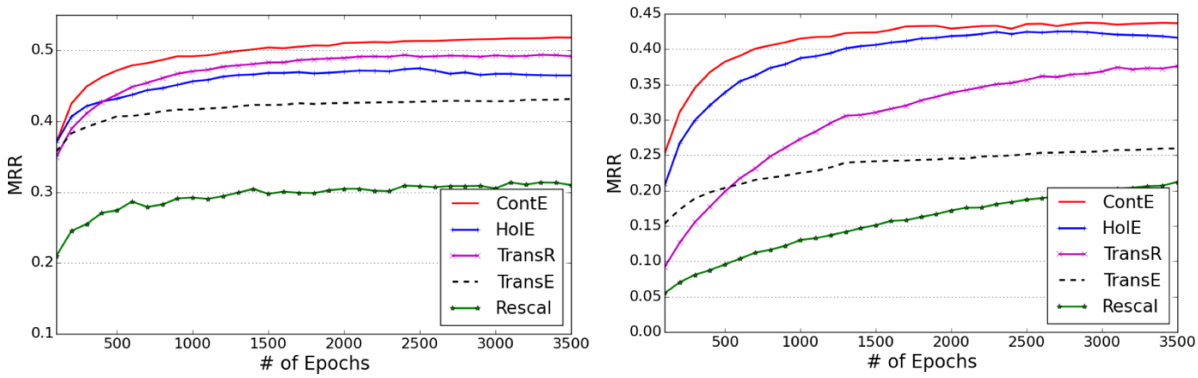
Baseline Methods and Implementation

The prediction accuracy of ContE was compared to four state-of-the-art KG embedding methods (Rescal, TransE, TransR and HolE) as well as a Null model. ContE was implemented in the *scikit-kge* (https://github.com/mnick/scikit-kge), which is a Python library computing KG embeddings. We also used the library to run Rescal, TransE and HolE. For TransR, we used the public code provided by (Lin et al. 2015).



(a) Relation Type Prediction for FB15k

(b) Relation Type Prediction for YAGO43k



(c) Entity Prediction for FB15k

(d) Entity Prediction for YAGO43k

Figure 4. Optimizing values of the number of epochs using validation datasets. MRR results for FB15k and YAGO43k datasets.

We applied SGD with AdaGrad for the optimization and a pairwise loss function to Rescal and TransE models to show better accuracy performances, which have been shown to be good for KG embedding methods in (Nickel et al. 2016b).

Our null model uses a simple heuristic. To predict a relation type, we randomly choose out of the N relation types in the contexts. For subject prediction, we randomly choose out of the N entities that have used the given predicate as the outgoing relation type. For object prediction, we run the same procedure by finding entities that have used the predicate as the incoming relation type. Although this null model is pretty simple, it shows good relation type prediction accuracy for *Hits@10*, as seen in Tables 3 and 4. In fact, this approach shows the best accuracy compared to the other model-based approaches for *Hits@10* on YAGO43k, which has a small number of relation types (see Table 2). These results support our observation that the contextual relation types of a triple can be considered as a missing relation type in the triple.

Parameter Selection

ContE and baseline methods depend mainly on four parameters. These parameters and their considered values for our experiments are the following:

- k - the number of dimensions; {20, 50, 100, 200}
- γ - margin; {0.2, 0.5, 1.0, 2.0, 4.0}
- λ - learning rate; {0.001, 0.01, 0.05, 0.1, 0.15, 0.2, 0.25}
- τ - training epochs; $\{n\}_1^{3500}$

To select these parameter values, our method and the baselines were run on the validation set of the two datasets, and the best combination of parameters, according to MRR on the both datasets, was selected. Ultimately, we obtained the following combinations (see Tables 3-6 for τ):

Method	FB15k					
	# of epochs	MRR		Hits @		
		Raw	Filt.	1	3	10
Null	-	-	-	5.29	15.76	52.39
Rescal	199	0.52	0.64	56.09	68.05	79.50
TransE	184	0.44	0.52	44.33	55.98	65.04
TransR	25	0.47	0.52	43.36	60.08	68.43
HolE	196	0.49	0.56	43.29	63.68	81.27
ContE	139	**0.56**	**0.67**	**57.13**	**73.47**	**86.37**

Table 3. Relation Type Prediction Accuracy for FB15k

Method	YAGO43k					
	# of epochs	MRR		Hits @		
		Raw	Filt.	1	3	10
Null	-	-	-	10.11	29.70	**99.05**
Rescal	32	0.62	0.66	56.76	67.57	97.30
TransE	41	0.43	0.44	29.73	51.35	70.27
TransR	40	0.53	0.67	61.54	67.73	77.60
HolE	24	0.56	0.59	45.95	67.57	91.89
ContE	6	**0.78**	**0.81**	**72.97**	**86.49**	97.30

Table 4. Relation Type Prediction Accuracy for YAGO43k

- Rescal - $\{k = 150, \gamma = 0.2, \lambda = 0.1\}$
- TransE - $\{k = 50, \gamma = 2.0, \lambda = 0.1\}$
- TransR - $\{k = 100, \gamma = 1.0, \lambda = 0.001\}$
- HolE - $\{k = 200, \gamma = 0.2, \lambda = 0.1\}$
- ContE - $\{k = 200, \gamma = 2.0, \lambda = 0.1\}$

Figure 4 shows prediction accuracy as a function of the number of epochs on the validation sets. The number of epochs spans 1 and 200 for relation type prediction on FB15k (Figure 4 (a)) and between 1 and 50 on YAGO43k (Figure 4 (b)). Models tend to reach their optimal accuracy with the small number of epochs for relation type prediction because the number of relation types $|R|$ is relatively small compared to the number of entities $|E|$ (e.g., $|R|$=37 and $|E|$=42,975 on YAGO43k). Therefore, during each epoch, a relation type is exposed to updates many more times than an entity. So, the vectors of relation types tend to be well trained earlier than entities. In addition, we also have run the models at every 100 epoch between 100 and 3,500 epochs for entity prediction (Figure 4 (c) and (d)). The optimal number of epochs for each method is shown in Tables 3-6.

Experimental Results

For entity prediction, we used the following evaluation protocol that has been introduced in (Bordes et al. 2013) and carried out in succeeding publications like (Lin et al. 2015; Nickel et al. 2016b): For each triple (s, p, o) in the test set, s is replaced by s', and for $\forall s' \in E$ we compute the score of (s', p, o). Then we rank all the *corrupted* triples by the scores in decreasing order (i.e., the rank of a corrupted triple that has the highest score is 1). This procedure is repeated by replacing o for the object prediction. For the relation type prediction, we also repeated this procedure but, in this case, p was replaced with p' for $\forall p' \in R$.

It is possible that multiple corrupted triples of a triple exist in a dataset because a pair of two entities could have multiple relation types, and a pair of an entity and a relation type could be connected with multiple entities. In this case, only one corrupted triple is considered correct one, and the rest would be

Method	FB15k					
	# of epochs	MRR		Hits @		
		Raw	Filt.	1	3	10
Null	-	-	-	1.20	3.66	12.36
Rescal	3300	0.16	0.31	21.19	34.99	51.20
TransE	3500	0.24	0.43	30.59	49.80	66.72
TransR	3300	0.25	0.49	35.93	58.17	72.83
HolE	2500	0.20	0.47	35.78	54.94	66.64
ContE	3400	**0.26**	**0.52**	**40.29**	**58.45**	**74.53**

Table 5. Entity Prediction Accuracy for FB15k

Method	YAGO43k					
	# of epochs	MRR		Hits @		
		Raw	Filt.	1	3	10
Null	-	-	-	0.24	0.73	2.70
Rescal	3500	0.07	0.21	12.04	23.46	40.12
TransE	3500	**0.11**	0.26	14.43	29.84	49.85
TransR	3500	0.10	0.37	25.87	43.74	59.16
HolE	2800	0.10	0.42	33.59	47.57	57.07
ContE	2900	**0.11**	**0.44**	**33.97**	**50.75**	**62.38**

Table 6. Entity Prediction Accuracy for YAGO43k

counted as errors, even though they are all valid. This effect can lead to misleading results. To avoid this behavior, we remove all triples from the ranking if $(s', p, o) = true$ and $s \neq s'$ for a test triple (s, p, o) in the same way as in (Nickel et al. 2016b). That is, we remove any corrupted triples that appear in the training, validation, or test set aside from the triple being tested. We refer to the setting that does not remove the triples as *Raw*, while the setting that removes them is indicated as *Filtered (Filt.)*. To measure the quality of the ranking, we use MRR, and also report "*Hits @ 1, 3, 10*" that indicates the proportion of correct triples that appear in the top 1, 3 and 10.

Table 1 shows the memory complexity of ContE and baseline methods. ContE has the same or better complexity with other memory-efficient methods but shows higher accuracy than them, as shown in Tables 3-6. It can be seen that ContE outperforms all other methods for entity and relation type predictions over both datasets. For example, ContE predicts relation types with 72.97% accuracy, while Rescal, TransE, TransR and HolE show 56.76%, 29.73%, 61.54% and 45.95% accuracy for *Hits@1* on YAGO43k, respectively. ContE also shows consistently better results than all other methods. For example, Rescal shows good results for relation type prediction, but worse results for entity prediction compare to all the other baselines.

We also compared the number of epochs needed for our method and second-ranked baselines to reach the optimal MRR results of the baseline methods or better than them. As shown in Figure 4, ContE needs to run only 1,200 epochs to reach TransR's the best MRR, while the second-ranked method has to run 3,300 epochs to get its best results for entity prediction on FB15k. For relation type prediction on YAGO43k, ContE can achieve HolE's best MRR by running only 1 epoch, while the baseline has to run 24 epochs. These results imply that ContE is more scalable than the other methods because it can build the better embeddings with a reduced number of epochs while having the same memory complexity as the memory-efficient methods.

Conclusion and Future Work

In this paper, we focused on making a better model for KG completion, with scalability and prediction performance as the most critical issues. We proposed ContE, a contextual embedding model for KGs. We have shown that by including contextual relation types, in addition to the entities and the predicate of a triple, higher quality vector embeddings can be achieved. It is intuitive, memory-efficient, and easily supports the addition of new entities and relation types. From our experiments, we show that our method is more scalable and has a higher prediction accuracy than all other baselines for both relation type and entity predictions.

We evaluated our method for static KGs, but this work could also be extended for dynamic graphs that have changes over time. For example, anomalies could be detected by capturing changes in the contextual edges of nodes in the embedded space. This area will be the focus of our future work.

Acknowledgements

This material is based upon work supported in whole or in part with funding from the Laboratory for Analytic Sciences (LAS). Any opinions, findings, conclusions, or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the LAS and/or any agency or entity of the United States Government.

References

Dong, X., Gabrilovich, E., Heitz, G., Horn, W., Lao, N., Murphy, K., Strohmann, T., Sun, S., and Zhang, W. 2014. "Knowledge Vault: A Web-Scale Approach to Probabilistic Knowledge Fusion," *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*: ACM, pp. 601-610.

Duchi, J., Hazan, E., and Singer, Y. 2011. "Adaptive Subgradient Methods for Online Learning and Stochastic Optimization," *Journal of Machine Learning Research* (12:Jul), pp. 2121-2159.

Bordes, A., Weston, J., Collobert, R., and Bengio, Y. 2011. "Learning Structured Embeddings of Knowledge Bases," *Conference on Artificial Intelligence*.

Bordes, A., Usunier, N., Garcia-Duran, A., Weston, J., and Yakhnenko, O. 2013. "Translating Embeddings for Modeling Multi-Relational Data," *Advances in Neural Information Processing Systems*, pp. 2787-2795.

Galárraga, L. A., Teflioudi, C., Hose, K., and Suchanek, F. 2013. "Amie: Association Rule Mining under Incomplete Evidence in Ontological Knowledge Bases," *Proceedings of the 22nd international conference on World Wide Web*: ACM, pp. 413-422.

Guu, K., Miller, J., and Liang, P. 2015. "Traversing Knowledge Graphs in Vector Space," *Proceedings of Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 318-327.

Lin, Y., Liu, Z., Sun, M., Liu, Y., and Zhu, X. 2015. "Learning Entity and Relation Embeddings for Knowledge Graph Completion," *AAAI Conference on Artificial Intelligence*, pp. 2181-2187.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. 2013. "Distributed Representations of Words and Phrases and Their Compositionality," *Advances in Neural Information Processing Systems*, pp. 3111-3119.

Muggleton, S. 1995. "Inverse Entailment and Progol," *New generation computing* (13:3-4), pp. 245-286.

Nickel, M., Tresp, V., and Kriegel, H. P. 2011. "A Three-Way Model for Collective Learning on Multi-Relational Data," *Proceedings of the 28th international conference on machine learning (ICML-11)*, pp. 809-816.

Nickel, M., Murphy, K., Tresp, V., and Gabrilovich, E. 2016. "A Review of Relational Machine Learning for Knowledge Graphs," *Proceedings of the IEEE* (104:1), pp. 11-33.

Nickel, M., Rosasco, L., and Poggio, T. 2016. "Holographic Embeddings of Knowledge Graphs," *AAAI Conference on Artificial Intelligence*, pp. 1955-1961.

Socher, R., Chen, D., Manning, C. D., and Ng, A. 2013. "Reasoning with Neural Tensor Networks for Knowledge Base Completion," *Advances in Neural Information Processing Systems*, pp. 926-934.

Wang, Z., Zhang, J., Feng, J., and Chen, Z. 2014. "Knowledge Graph and Text Jointly Embedding," *Proceedings of Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1591-1601.

Wang, Z., Zhang, J., Feng, J., and Chen, Z. 2014. "Knowledge Graph Embedding by Translating on Hyperplanes," *AAAI Conference on Artificial Intelligence*, pp. 1112-1119.

West, R., Gabrilovich, E., Murphy, K., Sun, S., Gupta, R., and Lin, D. 2014. "Knowledge Base Completion Via Search-Based Question Answering," *Proceedings of the 23rd international conference on World wide web*: ACM, pp. 515-526.

Xie, R., Liu, Z., Jia, J., Luan, H., and Sun, M. 2016. "Representation Learning of Knowledge Graphs with Entity Descriptions," *AAAI Conference on Artificial Intelligence*, pp. 2659-2665.

Xie, R., Liu, Z., and Sun, M. 2016. "Representation Learning of Knowledge Graphs with Hierarchical Types," *Proceedings of International Joint Conference on Artificial Intelligence (IJCAI-16)*, pp. 2965-2971.

Zhong, H., Zhang, J., Wang, Z., Wan, H., and Chen, Z. 2015. "Aligning Knowledge and Text Embeddings by Entity Descriptions," *Proceedings of Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 267-272.