Year: 2019

Interaction Embeddings for Prediction and Explanation in Knowledge Graphs

Zhang, Wen ; Paudel, Bibek ; Zhang, Wei ; Bernstein, Abraham ; Chen, Huajun

Interaction Embeddings for Prediction and Explanation in Knowledge Graphs

Wen Zhang
Zhejiang University
Alibaba-Zhejiang University Joint
Institute of Frontier Technologies
wenzhang2015@zju.edu.cn

Bibek Paudel
Department of Informatics
University of Zurich
Switzerland
paudel@ifi.uzh.ch

Wei Zhang
Alibaba Group
Alibaba-Zhejiang University Joint
Institute of Frontier Technologies
lantu.zw@alibaba-inc.com

Abraham Bernstein
Department of Informatics
University of Zurich
Switzerland
bernstein@ifi.uzh.ch

Huajun Chen[*]
Zhejiang University
Alibaba-Zhejiang University Joint
Institute of Frontier Technologies
huajunsir@zju.edu.cn

ABSTRACT

Knowledge graph embedding aims to learn distributed representations for entities and relations, and are proven to be effective in many applications. *Crossover interactions* — bi-directional effects between entities and relations — help select related information when predicting a new triple, but hasn't been formally discussed before. In this paper, we propose *CrossE*, a novel knowledge graph embedding which explicitly simulates crossover interactions. It not only learns one general embedding for each entity and relation as in most previous methods, but also generates multiple triple specific embeddings for both of them, named *interaction embeddings*. We evaluate the embeddings on typical link prediction task and find that CrossE achieves state-of-the-art results on complex and more challenging datasets. Furthermore, we evaluate the embeddings from a new perspective — giving explanations for predicted triples, which is important for real applications. In this work, explanations for a triple are regarded as reliable closed-paths between head and tail entity. Compared to other baselines, we show experimentally that CrossE is more capable of generating reliable explanations to support its predictions, benefiting from *interaction embeddings*.

ACM Reference Format:
Wen Zhang, Bibek Paudel, Wei Zhang, Abraham Bernstein, and Huajun Chen. 2019. Interaction Embeddings for Prediction and Explanation in Knowledge Graphs. In *The Twelfth ACM International Conference on Web Search and Data Mining (WSDM '19), February 11–15, 2019, Melbourne, VIC, Australia.* ACM, New York, NY, USA, 10 pages.

1 INTRODUCTION

Knowledge graphs (KGs) like Yago [31], WordNet [20], and Freebase [1] have numerous facts in the form of (*head entity*, *relation*, *tail entity*), or (h, r, t) in short. They are useful resources for many AI tasks such as web search [32] and question answering [42].

Knowledge graph embedding (KGE) learns distributed representations [10] for entities and relations, called *entity embeddings* and *relations embeddings*. The embeddings are meant to preserve the

---
[*]Corresponding author.

information in a KG, and are represented as low-dimensional dense vectors or matrices in continuous vector space.

Many KGEs have been proposed, such as tensor factorization based RESCAL [25], translation-based TransE [3], neural tensor network NTN [29] and linear mapping method DistMult [40]. They are proven to be effective in many applications like knowledge graph completion, question answering and relation extraction.

Despite their success in modeling KGs, none of them has formally discussed **crossover interactions**, which are the bi-directional effects between entities and relations. Crossover interactions are quite common and help select related information when predicting new triples. Proper information selection is necessary during prediction because there are various information about each entity and relation in KGs. Crossover interactions include *interactions from relations to entities* and *interactions from entities to relations*.
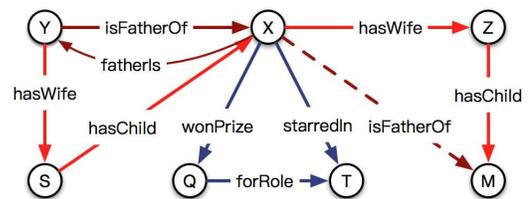

Figure 1: A hypothetical knowledge graph. Nodes and edges represent entities and relations. Solid lines represent existing triples and dashed lines represent triples to be predicted.

We explain the notion of crossover interactions with a running example. Given the subgraph in Figure 1, consider the task of predicting (*X, isFatherOf, ?*). There are six triples about entity *X*, but only four of them — (*X, hasWife, Z*), (*X, fatherIs, Y*), (*Y, isFatherOf, X*), (*S, hasChild, X*) — are related to this prediction, because they describe family relationships and are helpful to infer father-child relationship. The other two triples describing the career of *X* do not provide valuable information for this task. In this way, the relation *isFatherOf* affects the information of entities to be chosen for inference. We refer this as *interaction from relations to entities*.

In Figure 1, there are also two inference paths about the relation *isFatherOf*, but only one of them — ($X \xrightarrow{hasWife} Z \xrightarrow{hasChild} M$)

— is relevant for predicting (*X, isFatherOf, ?*), because information about (*X, hasWife, ?*) is available but (*?, fatherIs, X*) is not. In this way, the information of entities also affect the inference path of relations to use for inference. We refer this as *interactions from entities to relations*.

Considering **crossover interactions** in KGE, the embeddings for both entities and relations in specific triples should capture their interactions, and be different when involving different triples. However, most previous methods like TransE [3] learn a general embedding, which is assumed to preserve all information for each entity and relation. They ignore the *interactions* for both entities and relations. Some methods like TransH [37] and TransG [38] learn either multiple entity or relation embeddings but not both at the same time. They ignore that the crossover interactions are bi-directional and affect both entities and relations at the same time.

In this paper, we propose **CrossE**, a novel KGE which explicitly simulates *crossover interactions*. It not only learns one general embedding for each entity and relation, but also generates multiple triple specific embeddings, called *interaction embeddings*, for both of them. The *interaction embeddings* are generated via a relation specific interaction matrix. Given a (h, r, t), there are mainly four steps in CrossE: 1) generate *interaction embeddings* $\mathbf{h}_I$ for head entity h; 2) generate *interaction embeddings* $\mathbf{r}_I$ for relation r; 3) combine interaction embeddings $\mathbf{h}_I$ and $\mathbf{r}_I$ together; 4) compare the similarity of combined embedding with tail entity embedding $\mathbf{t}$.

We evaluate the embeddings on canonical link prediction task. The experiment results show that CrossE achieves state-of-the-art results on complex and more challenging datasets and exhibits the effectiveness of modeling crossover interactions in KGE.

Furthermore, we also propose an additional evaluation scheme for KGEs from the perspective of explaining their predictions. Link prediction task only evaluates the accuracy of KGEs at predicting missing triples, without explaining the predictions. But in real applications, it is valuable to give explanations for predictions, as they will likely improve the reliability of predicted results. To the best of our knowledge this is the first work to address both **link prediction and its explanation** of KGE.

The process of generating explanations for one triple (h, r, t) is modeled as searching reliable paths from h to t and similar structures to support the path explanations. We evaluate the quality of explanations based on *Recall* and *Average Support*. *Recall* reflects the coverage of triples that a KGE can generate explanations for and *Average Support* reflects the reliability of the explanations. Our evaluation on explanations show that CrossE, benefiting from *interaction embeddings*, is more capable of giving reliable explanations than other methods, including TransE [3] and ANALOGY [19].

In summary, **our contributions** in this paper are the following:

- We propose CrossE, a new KGE which models crossover interactions of entities and relations by learning an interaction matrix.
- We evaluate CrossE with various other KGEs on link prediction task with three benchmark datasets and show that CrossE achieves state-of-the-art results on complex and more challenging datasets with a modest parameter size.
- We propose a new evaluation scheme for embeddings, searching explanations for the predictions, and show that CrossE is able to generate more reliable explanations than other methods. The

evaluations suggest that interaction embeddings are better at capturing similarities between entities and relations in the different contexts of triples.

This paper is organized as follows. We review the literature in Section 2. We describe our model in Section 3 and explanation scheme in Section 4. We present the experimental results in Section 5 before concluding in Section 6.

2 RELATED WORK

Knowledge graph embedding (KGE) aims to embed a knowledge graph into continuous vector space and learns dense low dimensional representations for entities and relations. Various types of KGE methods have been proposed and majority of them learn the relationship between entities using training triples in a knowledge graph. Some methods also utilize extra information, such as random walks, external text corpus and hierarchical type information to improve the quality of embeddings. Our work focuses on learning from triples in a knowledge graph without extra information, so we mainly summarize those methods in this section. Considering the requirement of multiple representations for entities and relations from crossover interactions, prior studies are classified into two classes: (1) methods learning a general embedding for each entity and relation, and (2) methods learning multiple representations for either of them. In the end we also briefly summarize the methods that learn embeddings using extra information.

KGEs with general embeddings. Existing embedding methods with general embeddings all represent entities as low-dimensional vectors and relations as operations that combine the representations of head entity and tail entity. Most methods are proposed with different assumptions in vector space and model the knowledge graph from different perspectives. The first translation-based method TransE [3] regards a relation as translation from head entity to tail entity and assumes that the relation-specific translation of head entity should be close to the tail entity vector. It represents each relation as a single vector. RESCAL [25] regards the knowledge graph as a multi-hot tensor and learns the entity vector representation and relation matrix representation via collective tensor factorization. HOLE [24], which is a compositional vector space model, utilizes the interactions between different dimensions of embedding vectors and employs circular correlation to create compositional representations. RDF2Vec [26] uses graph walks and Weisfeiler-Lehman subtree RDF kernels to generate entity sequence and regards the entity sequence as sequence of words in natural language, then follows word2vec to generate embeddings for entities but not relations. NTN [29] represents each relation as a bilinear tensor operator followed by a linear matrix operator. ProjE [28] users a simpler but effective shared variable neural network method. DistMult [40] learns embeddings from a bilinear objective where each relation is represented as a linear mapping matrix from head entity to tail entity. It successfully captures the compositional semantics of relations. ComplEx [34] makes use of complex valued embeddings to handle both symmetric and antisymmetric relations because the Hermitian dot product of real values is commutative while for complex values it is not commutative. ANALOGY [19] is proposed from the analogical inference point of view and based on the linear mapping assumption. It adds normality and commutatively requirement constrains to the matrices for relations so as to

improve the capability of modeling analogical inference.It achieves the state-of-the-art results on link prediction task.

All these methods learn a general embedding for each entity and relation. They ignore the crossover interactions between entities and relations when inferring a new triple in different scenarios.

KGEs with multiple embeddings Some KGEs learn multiple embeddings for entities or relations under various considerations. Structured Embedding (SE) [4] assumes that the head entity and tail entity in the same triple should be close to each other in some subspace that depends on the relation. It represents each relation with two different matrices to transfer head entity and tail entity. ORC [43] focuses on the one-relation-circle and proposes to learn two different representations for each entity, one as a head entity and another as a tail entity. TransH [37] notices that TransE has trouble dealing with 1-N, N-1, and N-N relations. It learns a specific hyperplane for each relation and represents the relation as a vector close to its hyperplane. Entities are projected onto the relation hyperplane when involving a triple with this relation. TransR [18] considers that various relations focus on different aspects of entities. It represents the aspects by projecting entities from entity space to relation space and gets various relation specific embeddings for each entity. CTransR [18] is an extension of TransR that considers correlations under each relation type by clustering diverse head-tail entity pairs into groups and learning distinct relation vectors for each group. TransD [12] is a more fine-grained model which constructs a dynamic mapping matrix for each entity-relation pair considering the diversity of entities and relations simultaneously. TranSparse [13] is proposed to deal with the heterogeneity and imbalance of knowledge graph for relations and entities. It represents the transfer matrices with adaptive sparse matrices and the sparse degrees for the transfer matrices is determined by the number of entities linked by this relation.

These methods mostly consider the effects on entities from relations and learn multiple representations for entities. But they learn general embeddings for relations and ignore the interaction from entities to relations.

KGEs that utilize extra information Some KGEs learn embeddings utilizing not only training triples in a knowledge graph, but also extra information such as path rules from random walks. RTransE [6], PTransE [17] and CVSM [21] utilize the path rules as additional constrains to improves the embeddings. [35] considers three types of physical rules and one logical rule, and formulates the inference in knowledge graph as an integer linear programming problem. [15] and TKRL [39] propose that the hierarchical type information of entities is of great significance for representation learning in knowledge graphs. [15] considers entities types as hard constraints in latent variable models for KGs and TKRL considers hierarchical types as projection matrices for entities.

3 CrossE: MODEL DESCRIPTION

In this section we provide the details of our model CrossE. Our model simulates the crossover interactions between entities and relations by learning an *interaction matrix* to generate multiple specific interaction embeddings.

In our method, each entity and relation is represented by multiple embeddings: (a) *a general embedding*, which preserves high-level properties, and (b) multiple *interaction embeddings*, which preserve specific properties and information about crossover interactions. The interaction embeddings are obtained through Hadamard products among general embeddings and an *interaction matrix* $\mathbf{C}$

We denote a knowledge graph as $\mathcal{K} = \{\mathcal{E}, \mathcal{R}, \mathcal{T}\}$, where $\mathcal{E}, \mathcal{R}, \mathcal{T}$ represent the sets of entities, relations and triples respectively. The number of entities is n_e, the number of relations is n_r, and the dimension of embeddings is d. Bold letters denote embeddings. $\mathbf{E} \in \mathbb{R}^{n_e \times d}$ is the matrix of general entity embeddings with each row representing an entity embedding. Similarly, $\mathbf{R} \in \mathbb{R}^{n_r \times d}$ is the matrix of general relation embeddings. $\mathbf{C} \in \mathbb{R}^{n_r \times d}$ is the *interaction matrix* with each row related to a specific relation.

The basic idea of CrossE is illustrated in Figure 2. The general embeddings ($\mathbf{E}$ for entities and $\mathbf{R}$ for relations) and interaction matrix $\mathbf{C}$ are represented in the shaded boxes. The interaction embeddings $\mathbf{E}_c$ and $\mathbf{R}_c$ are the results of crossover interactions between entities and relations. Note that $\mathbf{E}$, $\mathbf{R}$, and $\mathbf{C}$ are parameters that need to be learned. The interaction embeddings $\mathbf{E}_c$ and $\mathbf{R}_c$ do not need to be learned as they are fully specified by the interaction operation on general embeddings.
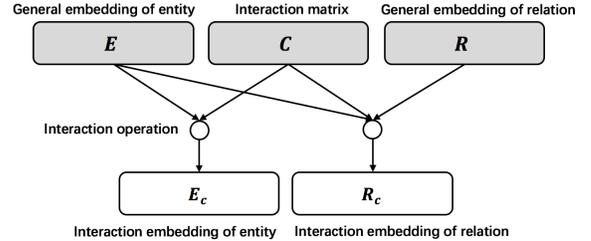

Figure 2: Overview of CrossE. Crossover interactions between general embeddings (E and R) and interaction matrix (C) resulting in interaction embeddings (unshaded boxes).

We now explain the score function and training objective of CrossE. Head entity h, tail entity t, and relation r correspond to high-dimensional 'one-hot' index vectors $\mathbf{x}_h$, $\mathbf{x}_t$ and $\mathbf{x}_r$. The learned general embeddings of h, r and t are written as:

$$\mathbf{h} = \mathbf{x}_h^\top \mathbf{E}, \quad \mathbf{t} = \mathbf{x}_t^\top \mathbf{E}, \quad \mathbf{r} = \mathbf{x}_r^\top \mathbf{R} \tag{1}$$

In CrossE, we define a score function for every triple such that valid triples receive high scores and invalid triples receive low scores. The score function has four parts and we describe them next.

(1) **Interaction Embedding for Entities.** To simulate the effect from relation to head entity, we define the interaction operation applied to head entity as:

$$\mathbf{h}_I = \mathbf{c}_r \circ \mathbf{h} \tag{2}$$

, where $\circ$ denotes *Hadamard product*, an element-wise operator that has been proved to be effective and widely used in previous methods, such as [24] and [34]. We call $\mathbf{h}_I$ the interaction embedding of h. Here, $\mathbf{c}_r \in \mathbb{R}^{1 \times d}$ is a relation specific variable which we get from the interaction matrix $\mathbf{C}$ as in (3). As $\mathbf{c}_r$ depends on relation r, there are as many interaction embeddings of h as there are relations.

$$\mathbf{c}_I = \mathbf{x}_I^\top \mathbf{C} \tag{3}$$

(2) **Interaction Embedding for Relations.** The second interaction operation applies to the relations so as to simulate the effects from head entities. This interaction operation is defined in (4). Similar to (2), this is the Hadamard product of $\mathbf{h}_I$ and $\mathbf{r}$ and we call $\mathbf{r}_I$

the interaction embedding of r. For every head entity, there is an interaction embedding of r.

$$\mathbf{r}_I = \mathbf{h}_I \circ \mathbf{r} \qquad (4)$$

(3) **Combination Operator** The third step is to get the combined representation and we formulate it in a nonlinear way:

$$\mathbf{q}_{hr} = tanh(\mathbf{h}_I + \mathbf{r}_I + \mathbf{b}) \qquad (5)$$

, where $\mathbf{b} \in \mathbb{R}^{1 \times d}$ is a global bias vector. $tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$, in which the output is bounded from -1 to 1. It is used to ensure the combined representation share the same distribution interval (both negative values and positive values) with the entity representation.

(4) **Similarity Operator** The fourth step is to calculate the similarity between combined representation $\mathbf{q}_{hr}$ and general tail entity representation $\mathbf{t}$:

$$f(h, r, t) = s_{(h,r,t)} = \sigma(\mathbf{q}_{hr}\mathbf{t}^\top) \qquad (6)$$

,where dot product is used to calculate the similarity and $\sigma(z) = \frac{1}{1+e^{-z}}$ is a nonlinear function to constrain the score $s_{(h,r,t)} \in [0, 1]$.

The overall score function is as follows:

$$f(h, r, t) = \sigma(tanh(\mathbf{c}_r \circ \mathbf{h} + \mathbf{c}_r \circ \mathbf{h} \circ \mathbf{r} + \mathbf{b})\mathbf{t}^\top) \qquad (7)$$

To evaluate the effectiveness of the crossover interactions, we devise a simplified CrossE called CrossE$_S$ by removing the interaction embeddings and using only the general embeddings in the score function:

$$f_S(h, r, t) = \sigma(tanh(\mathbf{h} + \mathbf{r} + \mathbf{b})\mathbf{t}^\top) \qquad (8)$$

Loss function We formalize a log likelihood loss function with negative sampling as the objective for training:

$$L(\mathcal{K}) = - \sum_{(h,r,t) \in \mathcal{K}} \sum_{x \in \mathcal{B}(h,r,t)} \Big[log(l(x)f(x))$$
$$+ log(1 - l(x))(1 - f(x)) \Big] + \lambda \sum \|\theta\|_2^2$$

Here, $\mathcal{B}(h, r, t)$ is the bag of positive examples with label 1 and negative examples with label 0 generated for (h, r, t). The label of example x is given by $l(x)$. For (h, r, t), positive examples are $(h, r, e) \in \mathcal{K}$ and negative examples are $(h, r, e) \notin \mathcal{K}$. The factor λ controls the L2 regularization of model parameters $\theta = \{\mathbf{E}, \mathbf{R}, \mathbf{C}, \mathbf{b}\}$. The goal of the training procedure is to minimize $L(\mathcal{K})$ and we train it iteratively using a gradient based approach.

Number of Parameters. The total number of parameters for CrossE is $(n_e + 2n_r + 1) \times d$, as there are $n_e + n_r$ general embeddings, n_r additional embeddings from the interaction matrix, and a bias term. Note that interaction embeddings are fully specified by these parameters. While predicting head entities, we model the task as the inverse of tail entity prediction, e.g., (t, r^{-1}, h). In such cases, we need $2n_r$ more embeddings for inverse relations. Since $n_r \ll n_e$ in most knowledge graphs, this does not add a lot of extra parameters.

The main benefits of CrossE compared to existing KGEs are as follows: (1) For an entity and a relation, the representations used during a specific triple inference are interaction embeddings (not general embeddings), which simulate the selection of different information for different triple predictions. (2) Multiple interaction embeddings for each entity and relation provide richer representations and generalization capability. We argue that they are capable of capturing different latent properties depending on the context of interactions. This is because each interaction embedding can

select different similar entities and relations when involving different triples. (3) It learns one general embedding for each entity and relation, and uses them to generate interaction embeddings. This results in much less extra parameters than when learning multiple independent embeddings for each entity and relation.

4 EXPLANATIONS FOR PREDICTIONS

In this section, we describe how we generate explanations for predicted triples. Explanations are valuable when KGEs are implemented in real applications as they help improve the reliability of and trust from people on the predicted results. The balance between predicting accuracy and explanation has already attracted research attention in other areas, such as recommender systems [9, 36].

We first introduce the motivation for our explanation scheme, followed by our embedding-based path-searching algorithm.

4.1 Background

Similar to inference chains and logical rules, meaningful paths from h to t can be regarded as explanations for a predicted triple (h, r, t). For example, in Figure 1, the fact that X is father of M can be inferred by $(X, hasWife, Z)$ and $(Z, haschild, M)$.

$$X \xrightarrow{hasWife} Z \xrightarrow{hasChild} M \Rightarrow X \xrightarrow{isFatherOf} M \qquad (9)$$

The right-hand side of the implication "$\Rightarrow$" is called *conclusion* and the left-hand side is *premise*. *Premise* is an explanation for the *conclusion*. In the above example, the path $X \xrightarrow{hasWife} Z \xrightarrow{hasChild} M$ is one explanation for the triple $(X, isFatherOf, M)$.

Searching paths between head entity and tail entity is the the first step for giving explanations for a triple. There are multiple works focusing on mining such paths as rules or features for prediction, e.g., AMIE+ [5] and PRA [16], in which the paths are searched and pruned based on random walks and statistical significance. An important aspect of efficient path search is the volume of search space. Selecting the candidate start entities and relations is the key point of reducing search space. Good embeddings can be useful for candidate selection with less effects on path searching results, because they are supposed to capture the similarity semantics of entities and relations. Thus giving explanations for predicted triples by searching for reliable paths based on embeddings not only improve the reliability of predicted results but also is a new perspective of evaluating embedding qualities.

In this paper, the explanation reliability for triple (h, r, t) is evaluated with the number of *similar structures* in knowledge graph, on which the inferences are mainly based, as noted by [23]. Two similar structures contain same relations but different specific entities. For example, Figure 3 is a subgraph with similar structures from Figure 1.
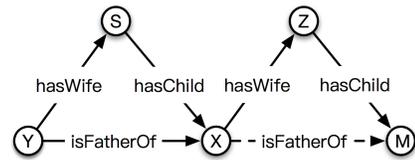

Figure 3: Similar structures for the example subgraph in Figure 1.

The left and right subgraph in Figure 3 are similar structures, as they both contain three entities e_1, e_2, e_3, a triple $e_1 \xrightarrow{isFatherOf} e_3$, and a path $e_1 \xrightarrow{hasWife} e_2 \xrightarrow{hasChild} e_3$. Thus the left subgraph is a *support* for $(X, isFatherOf, M)$ with path explanation $X \xrightarrow{hasWife} Z \xrightarrow{hasChild} M$ and vice versa. They support the reasonable existence and path explanation of each other. In general, the more similar structure *support* an explanation has, the more reliable it is.

4.2 Embedding-based explanation search

During the embedding-based explanation search, we first select candidate entities and relations embedding similarity to reduce the search space before generating explanations for a triple (h, r, t). The candidate selection is related with the quality of embeddings and will directly affect the final explanations. We assume the similarity of vector embeddings are related to Euclidean distance and matrix embeddings are related to Frobenuis norm. Then based in selected candidates, we do exhaustive search for explanations which involves (1) searching closed-paths from h to t as explanations and (2) searching similar structures for the explanations as *support*.

More specifically, there are four main steps:

• **Step 1: Search similar relations.** Find top k_r relations similar to r, denoted by the set $S_r = \{r_1, r_2, ..., r_t\}$, as possible first steps for path search. This step helps to prune unreasonable paths. For example, the path $h \xrightarrow{hasFriend} c \xrightarrow{likes} b$ doesn't indicate the relationship $h \xrightarrow{liveIn} b$ even if it may have a lot of support, because $hasFriend$ is not related with the inference of $liveIn$. To avoid such meaningless paths, the search is constrained to begin with relations similar to r, which are more likely to describe the same aspect of an entity.

• **Step 2: Search paths between h and t.** Search a set of path $\mathcal{P} = \{p | (h, p, t) \in \mathcal{K}\}$. For simplicity, we consider paths with length less than three as in [40]. There are six types of paths corresponding to six similar structures as shown in Table 1. The six possible paths are: $p_1 = \{h \xrightarrow{r_s} t\}$, $p_2 = \{h \xleftarrow{r_s} h\}$, $p_3 = \{h \xleftarrow{r_s} \circ \xrightarrow{r'} h\}$, $p_4 = \{h \xrightarrow{r_s} \circ \xleftarrow{r'} h\}$, $p_5 = \{h \xrightarrow{r_s} \circ \xrightarrow{r'} h\}$, $p_6 = \{h \xrightarrow{r_s} \circ \xrightarrow{r'} h\}$. Here, $\circ$ and r' denote any entity and relation in KG. To search for paths with length two, we apply a bi-direction search strategy. Take p_5 as an example, we first find two entity sets $\mathcal{E}_1 = \{e | (h, r_s, e) \in \mathcal{K}\}$ and $\mathcal{E}_2 = \{e | (e, r', t) \in \mathcal{K}, r' \in \mathcal{R}\}$. Then we get the path via intersection entities of $\mathcal{E}_1$ and $\mathcal{E}_2$, $p = \{(h \xrightarrow{r_s} \circ \xrightarrow{r'} t) | (e, r', t) \in \mathcal{K}, e \in \mathcal{E}_1 \cap \mathcal{E}_2\}$.

• **Step 3: Search similar entities.** Find top k_e similar entities for h, denoted by the set $S_h = \{h_1, h_2, ..., h_k\}$. and check for corresponding tail entity t_s of $(h_s, r, ?)$ in the KG, where $h_s \in S_h$. The tail checking results depend on the quality of selected similar entities, therefore the more capable a KGE is in capturing similarity between entities, the higher is the probability that (h_s, r, t_s) exist.

• **Step 4: Search similar structures as support.** Find support for path $p \in \mathcal{P}$ from step 2 based on the similar entities from step 3. If $(h_s, p, t_s) \in \mathcal{K}$, regard the path p as an explanation for (h, r, t) and $((h_s, p, r_s), (h_s, r, t_s))$ as a support for this explanation. We only consider the paths with at least one support in knowledge graph as explanations.

We summarize the process of embedding-based explanation search in Algorithm 1.

Algorithm 1 Search explanations for predicted triple (h, r, t)

Require: Knowledge graph $\mathcal{K}$, relation and entity embeddings **R** and **E**
Ensure: Explanations for (h, r, t) and their supports.
 1: Explanation set $E = \emptyset$, Support set $S = \emptyset$
 2: Select the set S_r of top k_r similar relations for r
 3: Search the corresponding path set $\mathcal{P} = \{p | (h, p, t) \in \mathcal{K}\}$ for each type: direct search for similar structure type 1 and type 2, and bidirectional search for type 3, type 4, type 5, and type 6.
 4: Select the set S_h of top k_e similar entities for h
 5: **for** $p \in \mathcal{P}$ **do**
 6:     **if** $(h_s, p, t_s) \in \mathcal{K}$ and $h_s \in S_h$ **then**
 7:         $E \leftarrow E \cup p$
 8:         $S \leftarrow S \cup ((h_s, p, t_s), (h_s, r, t_s))$
 9:     **end if**
10: **end for**

In order to make explanations and support more clear, in Table 1, we give real examples of path explanations and their supports based on the CrossE embeddings (Section 5.3 provides details about the implementation). For each types of path, we list a predicted triple with this kind of path explanation and one example support from the embedding-based explanation search results of CrossE. It is expected that the reliability of predicted triple increases when end-users are also provided with the explanation and its support in the form of similar path structures.

5 EXPERIMENTAL EVALUATION

We begin this section by describing datasets used in our experiments. We then explain a comparative study of different KGEs on two main tasks: (a) link prediction, and (b) generating explanations for predicted triples.

5.1 Datasets

We use three popular benchmark datasets: WN18 and FB15k introduced in [3], and FB15k-237 proposed by [33]. They are either subset of WordNet [20], a large lexical knowledge base about English, or Freebase [1], a huge knowledge graph describing general facts in the world. The details of these datasets are given in Table 2.

5.2 Evaluation I: Link Prediction

In this section, we evaluate the embeddings on canonical link prediction task, which contains two subtasks, one is tail entity prediction $(h, r, ?)$ and the other one is head entity prediction $(?, r, t)$.

5.2.1 Evaluation Metrics. The link prediction evaluation follows the same protocol as in previous works, in which all entities in the dataset are candidate predictions. During head entity prediction for (h, r, t), we replace the h with each entity and calculate their scores according to score function given by (7). Then we rank the scores in descending order and regard the rank of h as prediction result. The tail entity prediction is done similarly.

Type			Head entity	Relation	Tail entity
Type 1		Predicted triple	Mel Gibson	award nominations	Best Director
		Explanation	Mel Gibson	awards won	Best Director
		Support	Vangelis	award nominations	Best Original Musical
			Vangelis	awards won	Best Original Musical
Type 2		Predicted triple	Aretha Franklin	influenced	Kings of Leon
		Explanation	Kings of Leon	influenced by	Aretha Franklin
		Support	Michael Jackson	influenced	Lady Gaga
			Lady Gaga	influenced by	Michael Jackson
Type 3		Predicted triple	Cayuga County	containedby	New York
		Explanation	Auburn	capital of	Cayuga County
			Auburn	containedby	New York
		Support	Onondaga County	containedby	New York
			Syracuse	capital of	Onondaga County
			Syracuse	containedby	New York
Type 4		Predicted triple	South Carolina	country	USA
		Explanation	Columbia	state	South Carolina
			United States of America	contains	Columbia
		Support	Mississippi	country	USA
			Jackson	state	Mississippi
			United States of America	contains	Jackson
Type 5		Predicted triple	World War I	entity involved	German Empire
		Explanation	World War I	commanders	Erich Ludendorff
			Erich Ludendorff	military commands	German Empire
		Support	Falklands War	entity involved	United Kingdom
			Falklands War	commanders	Margaret Thatcher
			Margaret Thatcher	military commands	United Kingdom
Type 6		Predicted triple	Northwestern University	major field of study	Computer Science
		Explanation	Northwestern University	specialization	Artificial intelligence
			Computer Science	specialization	Artificial intelligence
		Support	Stockholm University	major field of study	Philosophy
			Stockholm University	specialization	Political_philosoph
			Philosophy	specialization	Political_philosoph

Table 1: Six similar structures with real examples of explanations and support for predictions made by CrossE.

Dataset	$|\mathcal{E}|$	$|\mathcal{R}|$	|Train Set|	|Validation Set|	|Test Set|
WN18	40,943	18	141,442	5,000	5,000
FB15k	14,951	1,345	483,142	500,00	59,071
FB5k-237	14,541	237	272,115	17,535	20,466

Table 2: Datasets statistics.

Aggregating the head and tail entity prediction ranks of all test triples, there are three evaluation matrices: The *Mean Rank(MR)*, *Mean Reciprocal Rank(MRR)* and Hit@N, which is the proportion of ranking scores within *N* of all test triples. *MRR* is similar to *MR* but more immune to extremely bad case. Similar to most recent works, we evaluate CrossE on *MRR* and *Hit@1, Hit@3* and *Hit@10*.

We also apply *filter* and *raw* settings. In *filter* setting, we filter all candidate triples in train, test or validation datasets before ranking, as they are not negative samples. *Raw* is the setting without filtering.

5.2.2 Implementation Details. Parameter initialization are as follows: **E**, **R** and **C** are initialized from the uniform distribution $U[-\frac{6}{\sqrt{d}}, \frac{6}{\sqrt{d}}]$ as suggested in [7]. **b** is initialized to zero.

The positive samples for training triple (h, r, t) are generated by retrieving all $(h, r, e) \in \mathcal{K}_{train}$. Negative triples are generated randomly by sampling n entities such that $(h, r, e) \notin \mathcal{K}_{train}$.

The head entity prediction $(?, r, t)$ is transformed to $(t, r^{-1}, ?)$ in which r^{-1} is the inverse relation of r. We generate inverse triples (t, r^{-1}, h) for each (h, r, t) during training, as done in [17, 21].

We implement our model using TensorFlow with Adaptive Moment Estimation (Adam) [14] optimizer and dropout [30] with dropout rate 0.5 applied to similarity operator given in (6). The maximum training iteration is set to 500 for all datasets.

The configurations for the results of CrossE are as follows: number of negative examples $n = 50$, learning rate $r = 0.01$, dimension

	parameters	WN18	FB15k
Unstructured [2]	$n_e d$	38.2	6.3
RESCAL[25]	$n_e d + n_r d^2$	52.8	44.1
NTN [29]	$n_e d + n_r(sd^2 + 2sd + 2s)$	66.1	41.4
SE [4]	$n_e d + 2n_r d$	80.5	39.8
LFM [11]	$n_e d + n_r d^2$	81.6	33.1
TransH[37]	$n_e d + 2n_r d$	86.7	64.4
TransE[3]	$n_e d + n_r d$	89.2	47.1
TransR[18]	$n_e d + n_r d^2 + n_r d$	92.0	68.4
RTransE [6]	$n_e d + n_r d$	-	76.2
TransD[12]	$2n_e d + 2n_r d$	92.2	77.3
CTransR [18]	$n_e d + n_r d^2$	92.3	70.3
KG2E [8]	$2n_e d + 2n_r d$	93.2	74.0
STransE [22]	$n_e d + n_r d + 2n_r d^2$	93.4	79.7
DistMult [40]	$n_e d + n_r d^2$	93.6	78.3
TranSparse [13]	$n_e d + n_r d + (1 - \theta)n_r d^2$	93.9	79.9
PTransE-MUL [17]	$n_e d + n_r d$	-	77.7
PTransE-RNN [17]	$n_e d + n_r d + d^2$	-	82.2
PTransE-ADD [17]	$n_e d + n_r d$	-	84.6
ComplEx [34]	$2n_e d + 2n_e d$	94.7	84.0
ANALOGY [19]	$n_e d + n_r d$	94.7	85.4
HolE [24]	$n_e d + n_r d$	94.9	73.9
CrossE	$n_e d + 2n_r d + d$	**95.0**	<u>87.5</u>†
CrossE$_S$	$n_e d + n_r d + d$	87.3	72.7

Table 3: HIT@10(filter) results of 22 KGEs on WN18 and FB15k. "-" indicates missing results form original paper. Boldface scores are the best results among all methods. Underlined scores are the better ones between CrossE and CrossE$_S$. In *parameters* column, n_e and n_r denote number of entities and relations respectively, d is the embedding dimension, s is the number of hidden nodes of a neural network, θ is the sparse degree of matrix.

$d = 100$, regularizer parameter $\lambda = 10^{-4}$ and batch-size $B = 2048$ for WN18; $n = 50$, $r = 0.01$, $d = 300$, $\lambda = 10^{-6}$ and $B = 4000$ for FB15K; $n = 50$, $r = 0.01$, $d = 100$, $\lambda = 10^{-5}$ and $B = 4000$ for FB15k-237. CrossE$_S$ is trained with the same parameters.

5.2.3 Link Prediction Results. In Table 3, we show the results of CrossE and 21 baselines with their published results of *Hit@10* (*filter*) on WN18 and FB15k from the original papers [1]. This represents the most applied evaluation metric on the most commonly used datasets in prior works, as we want to compare with as many baselines as possible.

For fair comparison, models using external information such as text are not considered as baselines.

All CrossE results that are significantly different from the second-best results are marked with †. We used one-sample proportion test at the 5% p-value level for testing the statistical significances [2].

In Table 4, we compare CrossE with seven baseline methods whose *MRR, Hit@1* and *Hit@3* results are available. The performance score of TransE, DistMult and ComplEx are from [34], RESCAL and HOLE from [24], ANALOGY from [19] and R-GCN from [27].

In Table 5, we separately show the link prediction results of CrossE on FB15k-237, as this is a recently proposed more challenge dataset and there are fewer baselines to compare than with WN18 and FB15k. We gather as many baselines as possible for FB15k-237 and list them in Table 5. The results of DistMult, Node+LinkFeat and Neural LP are from [41], and of R-GCN and R-GCN+ from the original paper [27]. For ComplEx and ANALOGY, we used the published code of [19][3] to search the best parameters among all possible combination of embedding dimension $d \in \{100, 200\}$ and regularizer weight $\lambda \in \{10^{-1}, 10^{-2}, 10^{-3}\}$ with six negative samples as used in the ANALOGY paper [19] [4]. The parameters used for ComplEx and ANALOGY are $d = 100, \lambda = 10^{-1}$.

	WN18			FB15k		
	MRR	Hit@		MRR	Hit@	
	filter/raw	1	3	filter/raw	1	3
RESCAL	0.890 / 0.603	84.2	90.4	35.4 /18.9	23.5	40.9
TransE	0.455 / 0.335	8.9	82.3	38.0 / 22.1	23.1	47.2
DistMult	0.822 / 0.532	72.8	91.4	65.4 / 24.2	54.6	73.3
HolE	0.938 / 0.616	93.0	**94.5**	52.4 / 23.2	40.2	61.3
ComplEx	0.941 / 0.587	93.6	**94.5**	69.2 / 24.2	59.9	75.9
ANALOGY	**0.942 / 0.657**	**93.9**	94.4	72.5 / 25.3	**64.6**	78.5
R-GCN	0.819 / 0.561	69.7	92.9	69.6/26.2	60.1	76.0
CrossE	<u>0.830</u> / <u>0.570</u>	<u>74.1</u>	<u>93.1</u>	**72.8 / 26.7**	<u>63.4</u>	**80.2**†
CrossE$_S$	0.469 / 0.396	21.7	70.6	46.4 / 25.4	28.4	61.9

Table 4: Link prediction results on WN18 and FB15k.

For WN18 (Table 3), CrossE achieves *Hit@10* results that are comparable to the best baselines. On the same dataset (Table 4), CrossE achieves better results than a majority of baselines on *MRR, Hit@1* and *Hit@3*. With only 18 relations, WN18 is a simpler dataset

---

[1]We follow the established practice in the KGE literature to compare link-prediction performance with published results on the same benchmarks.
[2]Similar to [19], we conducted the proportion tests on the Hit@k scores but not on MRR. Proportion tests cannot be applied to non-proportional scores such as MRR.
[3]https://github.com/quark0/ANALOGY
[4]Our parameter search includes the same range of values used in the original papers, and the best parameters obtained for FB15.

	FB15k-237				
	MRR (raw)	MRR (filter)	Hit@1 (filter)	Hit@3 (filter)	Hit@10 (filter)
DistMult	-	0.25	-	-	40.8
Node+LinkFeat	-	0.23	-	-	34.7
Neural LP*	-	0.24	-	-	36.2
R-GCN	0.158	0.248	15.3	25.8	41.4
R-GCN+	0.156	0.249	15.1	26.4	41.7
ComplEx	0.120	0.221	13.2	24.4	40.8
ANALOGY	0.118	0.219	13.1	24.0	40.5
CrossE	**0.177**	**0.299**	**21.1**†	**33.1**†	**47.4**†
CrossE$_S$	0.064	0.110	6.7	11.7	19.8

Table 5: Link prediction results on FB15k-237.

compared to FB15k. We can see in Table 4 that each method performs well on WN18 and much better than FB15k. For example, the scores of *Hit@3* on WN18 with all methods are above 90% while the best score on FB15k is just around 80%.

For FB15k (Table 3 and Table 4) we see that CrossE achieves state-of-the-art results on majority evaluation metrics, including *MRR, Hit@3*, and *Hit@10*. These results support that CrossE can better encode the diverse relations in knowledge graph, as FB15k is a complex linked dataset with 1, 345 relations and is more challenging to model than WN18. Compared to ANALOGY, which achieves best results on *Hit@1*, CrossE performs better on *Hit@3* and *Hit@10*, two metrics that we think are more fair to evaluate all relations, because there are four different relation types in knowledge graph, 1-to-1, 1-to-many, many-to-1 and many-to-many. Based on the Open World Assumption, the correct answer for link prediction on 1-to-many, many-to-1 and many-to-many relations might be more than 1 even under the *filter* setting. Thus we think *Hit@3, Hit@10* are better evaluation metrics for those datasets with diverse types of relations such as FB15k, in which there are 26.2% 1-to-1 relations, 22.7% 1-to-many, 28.3% and 22.8% many-to-many relations.

For FB15k-237 (Table 5), CrossE achieves state-of-the-art results and significant improvements compared to all baselines on all evaluation metrics. Compared to FB15k, FB15k-237 removes the redundant triples containing inverse relation information leakage that can be captured by simpler approaches to achieve good prediction result [33]. Without properly encoding the diverse semantics in knowledge graph, a method can't achieve good performance on FB15k-237. The significant performance improvement achieved by CrossE indicates that it is more capable of capturing and utilizing the complex semantics in knowledge graph during prediction.

Compared to the simpler model CrossE$_S$, CrossE performs much better on all three datasets(Table 3, Table 4, Table 5). As the only difference between them is whether there are *crossover interactions* between entities and relations or not, the huge performance difference shows the importance of modeling *crossover interactions*, which is common during inference on knowledge graphs with diverse topics.

Table 6 compares *Hit@10 (filter)* results on FB15k after mapping different relation types, 1-to-1, 1-to-many, many-to-1 and many-to-many represented as 1-1, 1-N, N-1 and N-N respectively, because we want to investigate which type of relations CrossE can encode better. The separate rules of relation types follows [3].
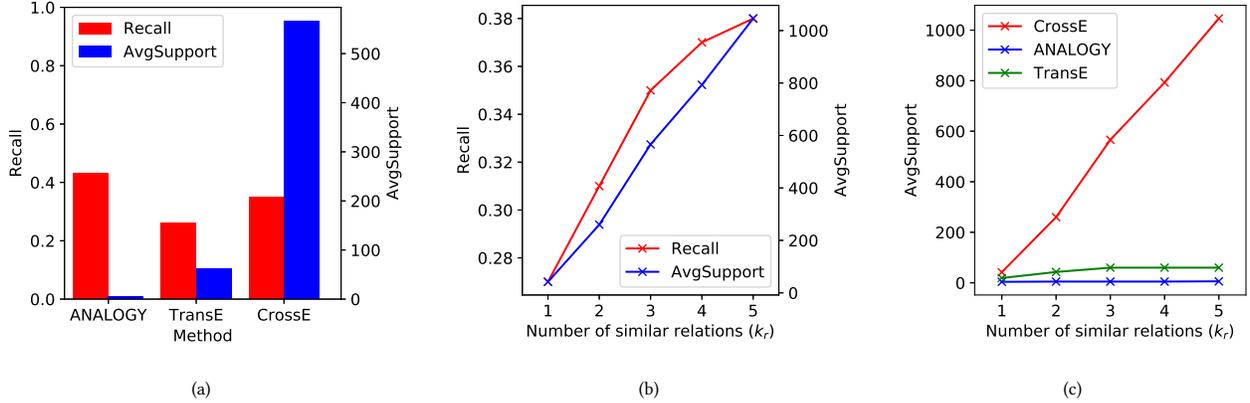
Figure 4: Evaluation results on generating explanations with different KGEs. 4(a) shows *Recall* and *AvgSupport* with $k_e = 10$, $k_r = 3$. 4(b) shows *Recall* and *AvgSupport* for CrossE with $k_r \in [1, 5]$. 4(c) shows *AvgSupport* with $k_e = 10$ and $k_r \in [1, 5]$.

	1-1 (head/tail)	1-N (head/tail)	N-1 (head/tail)	N-N (head/tail)
Unstructured	34.5/34.3	2.5/4.2	6.1/1.9	6.6/6.6
SE	35.6/34.9	62.6/14.6	17.2/68.3	37.5/41.3
SME(linear)	35.1/32.7	53.7/14.9	19.0/61.6	40.3/43.3
SME(Bilinear)	30.9/28.2	69.6/13.1	19.9/76.0	38.6/41.8
TransE	43.7/43.7	65.7/19.7	18.2/66.7	47.2/50.0
TransH	66.8/65.5	87.6/39.8	28.7/83.3	64.5/67.2
TransD	86.1/85.4	95.5/50.6	39.8/**94.4**	78.5/81.2
TransR	78.8/79.2	89.2/37.4	34.1/90.4	69.2/72.1
CTransR	81.5/80.8	89.0/38.6	34.7/90.1	71.2/73.8
CrossE	**88.2**[†]/**87.7**[†]	**95.7**[†]/**75.1**[†]	**64.2**[†]/92.3	**88.1**[†]/**90.8**[†]
CrossE$_S$	78.6/81.6	85.1/54.2	45.3 /85.8	71.7/76.7

Table 6: Hit@10 on FB15k by mapping to different relation types.

From Table 6 we see that CrossE significantly outperforms all other embedding methods except in the tail prediction for N-1 relations. CrossE achieves significant improvement, with 11.7% average improvement, on the more difficult tasks with more multiple correct answers including the head prediction for N-1 relations, the tail prediction for 1-N relations and both head and tail prediction for N-N relations. As a result, CrossE performance is more balanced than other methods on different types of relations.

5.3 Evaluation II: Generating Explanations

5.3.1 Evaluation. Aggregating all the path explanations and their supports from embedding results, we evaluate the capability of a KGE to generate explanations from two perspectives: (1) the fraction of triples (out of all test triples) that it can give explanations for (*Recall*), (2) the average support among the triples for which it can find explanations (*AvgSupport*). We argue that higher the *AvgSupport* for a triple, the more reliable the explanation will be.

Generally, a KGE that can generate better explanations will achieve higher *Recall* and *AvgSupport* when selecting the same number of similar entities (k_e) and relations (k_r).

5.3.2 Experimental Details. For the explanation experiment, we use the FB15k dataset. We choose two different KGEs as baselines to compare with CrossE. One is the popular translation-based embedding method TransE and the other is the linear-mapping based

embedding method ANALOGY which achieves one of the best results in the link prediction task.

The embeddings of CrossE used for searching explanations are those used in the link prediction experiment. We re-implement TransE ourselves and the implementation of ANALOGY is from [19]. The embeddings used for similar entity and relation selection are the representations for head entities and relations when they involved specific triples, which are interaction embeddings in CrossE and general embeddings in TransE and ANALOGY.

5.3.3 Explanation Results. The results are summarized in Figure 4. In Figure 4(a), we see that when selecting ten similar entities and three similar relations, the *recall* of three methods varies from 0.26 to 0.43 and the *AvgSupport* varies from 5 to 566. ANALOGY achieves the best results on *Recall* while it can give only a few examples for each explanation, quantified by its *AvgSupport* performance. TransE achieves the lowest *recall* but has about ten times more *AvgSupport* than ANALOGY for each triple. CrossE achieves the second best result in *recall* and has about 100 times more *AvgSupport* than ANALOGY. From the perspective of giving reliable explanations for predicted triples, CrossE outperforms the baselines.

The explanation and similar structures search are based on the similar entity and relation selection results. CrossE generates multiple crossover embeddings for entities and relations. Based on the interaction embeddings, the similar items for each entity or relation are different when it involves different triples. This makes CrossE more efficient at selecting similar items which will results in better explanation results. Based on the general embeddings, the similar items selection results will always be the same regardless of the triple-specific context. In our opinion, this is why the *AvgSupport* of CrossE is much higher than of TransE and ANALOGY.

In Figure 4(b), we see that *recall* improves slightly when increasing the number of similar relations to select while the *AvgSupport* increase a lot. This behavior is the same when increasing the number of similar entities to select. Figure 4(c) shows that *AvgSupport* for CrossE increases much faster than with TransE and ANALOGY. This also demonstrates the effectiveness of CrossE at selecting similar entities and relations.

To figure out what types of paths and similar structures are easier to find for the compared models, in Figure 5 for each model we show the share of *AvgSupport* for the six types of similar structures.
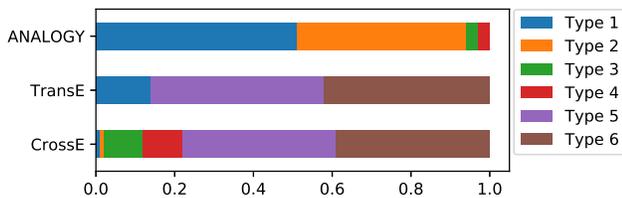
Figure 5: Share of *AvgSupport* for six similar structure types.

We can see that among all types of similar structures, type 5 is the one that both TransE and CrossE have the most *AvgSupport* for. From our point of view, type 5 is the most natural path in which the two relation along the path are in same direction with the relation between h and t. Thus it may be likely to be constructed when a knowledge graph is constructed. Although the shares of type 1 and 2 are high for ANALOGY, AvgSupport is very low (Figure 4(c)).

In summary, we can state that the modeling and vector-space assumptions of different KGEs effect the way they explain the predictions, including the types of path-explanations they can provide.

From these two evaluation tasks, we can conclude that the capability of KGE on link prediction and explanations are not directly related. A method that performs well on link prediction may not necessarily be good at giving explanations. The balance between prediction accuracy and giving explanations is important.

6 CONCLUSION

In this paper, we described a new knowledge graph embedding named CrossE. CrossE successfully captures the crossover interactions between entities and relations during new triple inference and achieves state-of-the-art results on link prediction task with complex linked datasets. We believe that improving the reliability of embedding method is as important as achieving high accuracy for prediction. This work is a first step for explaining the prediction results. There are still much work to do with explanations, such as how to enable KGEs to give explanation for all predicted triples. In our future work, we will focus on improving the capability of KGEs in predict missing triples and also giving more reliable explanations.

ACKNOWLEDGMENTS

REFERENCES

[1] Kurt D. Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. *Proceedings of SIGMOD* (2008), 1247–1250.
[2] Antoine Bordes, Xavier Glorot, Jason Weston, and Yoshua Bengio. 2014. A semantic matching energy function for learning with multi-relational data - Application to word-sense disambiguation. *Machine Learning* 94, 2 (2014), 233–259.
[3] Antoine Bordes, Nicolas Usunier, Alberto García-Durán, Jason Weston, and Oksana Yakhnenko. 2013. Translating Embeddings for Modeling Multi-relational Data. *Proceedings of NIPS* (2013), 2787–2795.
[4] Antoine Bordes, Jason Weston, Ronan Collobert, and Yoshua Bengio. 2011. Learning Structured Embeddings of Knowledge Bases. *Proceedings of AAAI* (2011).
[5] Luis Galárraga, Christina Teflioudi, Katja Hose, and Fabian M. Suchanek. 2015. Fast rule mining in ontological knowledge bases with AMIE+. *VLDB J.* 24, 6 (2015), 707–730.
[6] Alberto García-Durán, Antoine Bordes, and Nicolas Usunier. 2015. Composing Relationships with Translations. *Proceedings of EMNLP* (2015), 286–290.
[7] Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. *Proceedings of AISTATS* (2010), 249–256.
[8] Shizhu He, Kang Liu, Guoliang Ji, and Jun Zhao. 2015. Learning to Represent Knowledge Graphs with Gaussian Embedding. *Proceedings of CIKM* (2015), 623–632.
[9] Reinhard Heckel, Michail Vlachos, Thomas P. Parnell, and Celestine Dünner. 2017. Scalable and Interpretable Product Recommendations via Overlapping Co-Clustering. *Procedding of ICDE* (2017), 1033–1044.
[10] G. E. Hinton, J. L. McClelland, and D. E. Rumelhart. 1986. Distributed Representations. *Parallel distributed processing: explorations in the microstructure of cognition* (1986).
[11] Rodolphe Jenatton, Nicolas Le Roux, Antoine Bordes, and Guillaume Obozinski. 2012. A latent factor model for highly multi-relational data. *Proceddings of NIPS* (2012), 3176–3184.
[12] Guoliang Ji, Shizhu He, Liheng Xu, Kang Liu, and Jun Zhao. 2015. Knowledge Graph Embedding via Dynamic Mapping Matrix. *Proceedings of ACL* (2015), 687–696.
[13] Guoliang Ji, Kang Liu, Shizhu He, and Jun Zhao. 2016. Knowledge Graph Completion with Adaptive Sparse Transfer Matrix. *Proceedings of AAAI* (2016), 985–991.
[14] Diederik P. Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization. *CoRR* abs/1412.6980 (2014).
[15] Denis Krompaß, Stephan Baier, and Volker Tresp. 2015. Type-Constrained Representation Learning in Knowledge Graphs. *Proceddings of ISWC* (2015), 640–655.
[16] Ni Lao, Tom M. Mitchell, and William W. Cohen. 2011. Random Walk Inference and Learning in A Large Scale Knowledge Base. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, EMNLP 2011, 27-31 July 2011, John McIntyre Conference Centre, Edinburgh, UK, A meeting of SIGDAT, a Special Interest Group of the ACL.* 529–539.
[17] Yankai Lin, Zhiyuan Liu, Huan-Bo Luan, Maosong Sun, Siwei Rao, and Song Liu. 2015. Modeling Relation Paths for Representation Learning of Knowledge Bases. *Proceedings of EMNLP* (2015), 705–714.
[18] Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. 2015. Learning Entity and Relation Embeddings for Knowledge Graph Completion. *Proceedings of AAAI* (2015), 2181–2187.
[19] Hanxiao Liu, Yuexin Wu, and Yiming Yang. 2017. Analogical Inference for Multi-relational Embeddings. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017.* 2168–2178.
[20] George A. Miller. 1995. WordNet: A Lexical Database for English. *Commun. ACM* 38, 11 (1995), 39–41.
[21] Arvind Neelakantan, Benjamin Roth, and Andrew McCallum. 2015. Compositional Vector Space Models for Knowledge Base Completion. *Proceedings of ACL* (2015), 156–166.
[22] Dat Quoc Nguyen, Kairit Sirts, Lizhen Qu, and Mark Johnson. 2016. STransE: a novel embedding model of entities and relationships in knowledge bases. In *NAACL HLT 2016, The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego California, USA, June 12-17, 2016.* 460–466.
[23] Maximilian Nickel, Kevin Murphy, Volker Tresp, and Evgeniy Gabrilovich. 2016. A Review of Relational Machine Learning for Knowledge Graphs. *Proc. IEEE* 104, 1 (2016), 11–33.
[24] Maximilian Nickel, Lorenzo Rosasco, and Tomaso A. Poggio. 2016. Holographic Embeddings of Knowledge Graphs. *Proceedings of AAAI* (2016), 1955–1961.
[25] Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. 2011. A Three-Way Model for Collective Learning on Multi-Relational Data. *Proceedings of ICML* (2011), 809–816.
[26] Petar Ristoski and Heiko Paulheim. 2016. RDF2Vec: RDF Graph Embeddings for Data Mining. *Proceedings of ISWC* (2016), 498–514.
[27] Michael Schlichtkrull, Thomas N. Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. 2018. Modeling Relational Data with Graph Convolutional Networks. *Proceddings of ESWC* (2018).
[28] Baoxu Shi and Tim Weninger. 2017. ProjE: Embedding Projection for Knowledge Graph Completion. *Proceedings of AAAI* (2017), 1236–1242.
[29] Richard Socher, Danqi Chen, Christopher D. Manning, and Andrew Y. Ng. 2013. Reasoning With Neural Tensor Networks for Knowledge Base Completion. *Proceddings of NIPS* (2013), 926–934.
[30] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research* 15(Jun) (2014), 1929–1958.
[31] Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. 2007. Yago: a core of semantic knowledge. *Proceedings of WWW* (2007), 697–706.
[32] Sean R. Szumlanski and Fernando Gomez. 2010. Automatically acquiring a semantic network of related concepts. (2010), 19–28.
[33] Kristina Toutanova and Danqi Chen. 2015. Observed versus latent features for knowledge base and text inference. In *n Proceedings of the 3rd Workshop on Continuous Vector Space Models and their Compositionality.* 57–66.

[34] Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. 2016. Complex Embeddings for Simple Link Prediction. *Proceedings ICML* (2016), 2071–2080.

[35] Quan Wang, Bin Wang, and Li Guo. 2015. Knowledge Base Completion Using Embeddings and Rules. *Proceedings of IJCAI* (2015), 1859–1866.

[36] Xiang Wang, Xiangnan He, Fuli Feng, Liqiang Nie, and Tat-Seng Chua. 2018. TEM: Tree-enhanced Embedding Model for Explainable Recommendation. In *Proceedings of WWW*. 1543–1552.

[37] Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014. Knowledge Graph Embedding by Translating on Hyperplanes. *Proceedings of AAAI* (2014), 1112–1119.

[38] Han Xiao, Minlie Huang, and Xiaoyan Zhu. 2016. TransG : A Generative Model for Knowledge Graph Embedding. *Proceedings of ACL* (2016).

[39] Ruobing Xie, Zhiyuan Liu, and Maosong Sun. 2016. Representation Learning of Knowledge Graphs with Hierarchical Types. *Proceedings of IJCAI* (2016), 2965–2971.

[40] Bishan Yang, Wen tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. 2015. Embedding Entities and Relations for Learning and Inference in Knowledge Bases. *Proceedings of ICLR* (2015).

[41] Fan Yang, Zhilin Yang, and William W. Cohen. 2017. Differentiable Learning of Logical Rules for Knowledge Base Reasoning. In *NIPS*. 2316–2325.

[42] Jun Yin, Xin Jiang, Zhengdong Lu, Lifeng Shang, Hang Li, and Xiaoming Li. 2016. Neural Generative Question Answering. In *Proceedings of IJCAI*.

[43] Wen Zhang. 2017. Knowledge Graph Embedding with Diversity of Structures. *Proceedings WWW Companion* (2017), 747–753.