Convolutional Neural Knowledge Graph Learning

Feipeng Zhao\textsuperscript{1}, Martin Renqiang Min\textsuperscript{2}, Chen Shen\textsuperscript{3,1}, and Amit Chakraborty\textsuperscript{1}

\textsuperscript{1}Siemens Corporate Technology, 755 College Road East, NJ 08540, USA
\textsuperscript{2}NEC Laboratories America, 4 Independence Way, NJ 08540, USA
\textsuperscript{3}Temple University, 1925 N.12th Street, PA 19122, USA
feipeng.zhao, amit.chakraborty@siemens.com, renqiang@nec-labs.com, chen.shen@temple.edu

Abstract

Previous models for learning entity and relationship embeddings of knowledge graphs such as TransE, TransH, and TransR aim to explore new links based on learned representations. However, these models interpret relationships as simple translations on entity embeddings. In this paper, we try to learn more complex connections between entities and relationships. In particular, we use a Convolutional Neural Network (CNN) to learn entity and relationship representations in knowledge graphs. In our model, we treat entities and relationships as one-dimensional numerical sequences with the same length. After that, we combine each triplet of head, relationship, and tail together as a matrix with height 3. CNN is applied to the triplets to get confidence scores. Positive and manually corrupted negative triplets are used to train the embeddings and the CNN model simultaneously. Experimental results on public benchmark datasets show that the proposed model outperforms state-of-the-art models on exploring unseen relationships, which proves that CNN is effective to learn complex interactive patterns between entities and relationships.

Introduction

A knowledge graph (KG) stores real-world information as a directed multi-relational structured graph. In KG, graph nodes are entities, and directed edges correspond to different kinds of relationships. One piece of knowledge is represented as (head, relationship, tail) or \((h, \ell, t)\). For example, “Donald Trump is a Politician of USA” will be stored as (Donald Trump, isPoliticianOf, USA), where “Donald Trump” is the head entity, “isPoliticianOf” is the relationship and “USA” is the tail entity. In real world, there are different kinds of knowledge graphs such as WordNet (Miller 1995), Google Knowledge Graph and DBpedia (Lehmann et al. 2015). WordNet is a large lexical database of English, in which words are grouped into cognitive synonyms (synsets) and these synsets are interlinked with different relations. Google Knowledge Graph is a system that Google launched to understand facts about people, places and things and how they are connected. DBpedia (Lehmann et al. 2015) extracts information from Wikipedia as a structured knowledge base. Usually, knowledge graphs suffer from incompleteness. People try to exploit new triplets based on the existing incomplete graph: (1) given a head/tail and one kind of relationship \(\ell\), find the tail/head in the entity set; (2) given one head \(h\) and one tail \(t\), find the relationship \(\ell\) between these two entities. Various models are proposed to learn knowledge graph representations and solve these problems (Nickel et al. 2016). In (Nickel, Tresp, and Kriegel 2011), a 3-D binary tensor is constructed. Tensor Factorization method is applied on the tensor to learn entity and relationship embedding. This model is intuitive and simple, however the performance is not good. Similarly, in (Sutskever, Tenenbaum, and Salakhutdinov 2009), a bayesian clustered tensor factorization (BCTF) is applied on the 3-D binary tensor in order to get the balance between clustering and factorizations. In (Nickel, Rosasco, and Poggio 2016), holographic model is proposed to reduce the time complexity of tensor factorization, in which a novel circular correlation of vectors is proposed to represent pairs of entities, trying to keep the power of tensor product and make the model simple to learn.

Another group of models such as TransE (Bordes et al. 2013), TransH (Wang et al. 2014), TransR (Lin et al. 2015b) and TransA (Jia et al. 2016) learn low-dimensional representations for entities and relationships. They treat relationships as translations from heads to tails. The drawback of these models is that the translation structure assumption between entities and relationships is simple but in reality the connections between entities and relationships are more complex. Based on the translation models, people try to learn better representations to get more accurate predictions. In (Lin et al. 2015a), PTransE is proposed to improve TransE. This model considers “relation paths” instead of relationships as translations between entities. Similarly, in (Garca-Durr, Bordes, and Usunier 2015), the authors compose several relationships together as new relationships, After that new triplets are added to the training set to learn knowledge graph embedding. In (Xiao, Huang, and Zhu 2016), a manifold based embedding principle is proposed.

In knowledge graph, entity descriptions can be easily collected, people also try to use the entity descriptions to improve the model performance. In (Wang and Li 2016; Xiao et al. 2017), entity descriptions are incorporated into entity embedding. In (Xie et al. 2016), entity representations are learned from entity descriptions directly by using encod-
Neural Network models have also been proposed to learn KG embedding. Different from translation based models, the connections between entities and relationships are learned in neural networks. In (Socher et al. 2013), a neural tensor network (NTN) is proposed to learn the heads and tails over different relationships. In (Shi and Weninger 2017), ProjE is proposed, which uses combination operation and non-linear transformations applied to the triplet \((h, \ell, r)\) and calculates a score for the triplet.

TransE, TransH, TransR, and PTransE all consider relationships as simple \textit{translations} between entities and learn embedding based on this assumption, which makes entity and representations easy to learn. However, in reality, relationships between two entities are more complex. We try to learn distributed embedding to represent entities and relationships and learn a more complex correlations to connect entities and relationships.

Inspired by the success of neural networks based KG learning, in this paper we also try to use neural network to learn the entity and relationship embedding and their connections. Different from NTN or ProjE, in our model, we use Convolutional Neural Network (CNN) (Krizhevsky, Sutskever, and Hinton 2012) to learn sequential representations and high level non-linear connections between entities and relationships. In our CNN model, filters and convolution operations are used to exploit local features and high level features. Besides learning high level image features in computer vision, CNN is also used for learning word embedding in Natural Language Processing (NLP) (Dauphin et al. 2017). Because of the advantages of CNN in learning features, we try to apply CNN model to learn entity and relationship representations and their complex connections. In our model, entities and relationships are represented as low-dimensional sequential vectors. We treat each triple \((h, \ell, t)\) as an instance and combine head, relationship and tail sequential vectors together to create a matrix with height 3. CNN model is then used on this combination matrix to learn the entity and relationship representations and exploit the connection structure within \(h\), \(\ell\) and \(t\) simultaneously. A confidence score is learned as the output of the CNN model with a logistic unit. We use the existing triplets as positive samples and create negative samples by corrupting positive triplets to train CNN models. After the CNN model is learned, we can learn a score for each triplet in the test data. We predict new relationships in the knowledge graph based on the scores of the triplets.

Related Work

In this section, we will review some related methods for knowledge graph embedding learning, including tensor factorization based models, translation based models and neural network based models.

Tensor Factorization based Embedding Learning

In RESCAL (Nickel, Tresp, and Kriegel 2011), given the knowledge graph, for each relation \(\ell\), a binary matrix \(R_\ell\) is created for entity pairs. If \(i\)-th entity \(e_i\) and \(j\)-th entity \(e_j\) are connected with \(\ell\), \((e_i, \ell, e_j)\) hold and \(R_\ell(i, j) = 1\), otherwise \(R_\ell(i, j) = 0\). Finally a three-dimensional binary tensor \(R\) with size \(n \times n \times m\) is created for all the relationships, where \(n\) and \(m\) are the entity and relationship number respectively. After that, entity and relationships embedding are learned by using tensor factorization. A new tensor will be reconstructed based on the learned embedding.

Translation based Embedding Learning

In TransE (Bordes et al. 2013), entities and relationships are represented as low-dimensional embedding. If one triplet \((h, \ell, t)\) holds, the tail entity \(t\) can be represented as the head entity \(e\) with simple translation by using the relationship \(\ell\): ideally, \(h + \ell = t\). Positive triplets \((h, \ell, t)\) will have a small distance between \(h + \ell\) and \(t\) while negative triplets \((h', \ell, t')\) will have big distance between \(h'+\ell\) and \(t'\). Pairwise ranking method is applied to learn the entity and relationship embedding.

Similarly, TransH (Wang et al. 2014) also use relationship translation to connect head and tail entities. Different from TransE, in TransH, there is one hyperplane for every relationship. Given one relationship, head and tail will be projected to the hyperplane first as relationship-specified embedding, after that translation will be performed on this hyperplane.

In TransR, entities and relationships are represented in two different spaces. For every relationship, a mapping function will be learned to map the head and tail from entity space to relationship space. Similar to TransE and TransH, a translation will finally performed to connect the new head and tail on the relationship space.

Neural Network based Knowledge Graph Learning

Besides tensor factorization and translation models, algorithms that use neural networks to learn knowledge graphs are also proposed. People try to use the neural networks to learn more complex translations to connect head \(h\), relation \(\ell\) and tail \(t\). In (Socher et al. 2013), a neural tensor network structure (NTN) is proposed to reason over relationships between one pair of head and tail. Given one triplet \((h, \ell, t)\), the neural tensor network uses bilinear tensor operation to compute a confidence score for it. Embedding and NTN structure are learned simultaneously.

In (Shi and Weninger 2017), combination operator is applied on triplet \((h, \ell, r)\) first to connect the entities and relationships, non-linear mapping and logistic regression is used to learn a score for the triplet.

In Table[1] we give a summary of the score function of different models.

Proposed Approach

In this section, we present a novel CNN based model to learn entity and relationship representations. Similar to TransE, we assume entities \(e \in E\) and relationships \(\ell \in L\) can be represented as sequential vectors in low dimensional embedding space: \(e, \ell \in \mathbb{R}^k\) where \(k\) is the embedding dimension. In our model, given one triplet \((h, \ell, t)\), if this relationship holds, we will assign positive score 1 to this triplet.
After convolution operation, we will use subsampling. In our model, we first use multiple $3 \times 3$ kernels to do convolution operation over the combined matrix $m$. Since the height of $m$ is 3, we will use kernels with the same height $g$ as the input matrix. As a result, convolution operation will only go through over the row of matrix $m$ as showed in Fig 2. Different from CNN kernels on images that go through rows and columns on image matrix. In our model, we will explore locally connected structure over the head, relationship and tail together. Suppose the kernel number(kernel channel) is $c$, for the matrix $m$, we will get $c$ feature maps with size $1 \times (k - g + 1)$. RELU activation function $RELU(x) = max(0, x)$ will be applied to get non-negative feature maps.

### Subsampling

After convolution operation, we will use max-pooling over the feature maps to get subsamples. In our model, we set the size of max pooling filter as $(1 \times 2)$ and stride as 2. As a result we get smaller feature maps with length as $((k - g + 1) - 1)/2 + 1$ which is equal to $(k - g)/2 + 1$.

### Fully Connection

In fully connection step, we flatten the subsampling feature maps into one feature vector $f_{flat}$ with size $c \times ((k - g)/2 + 1)$. After we get the feature vector, we will use a linear mapping method to map the feature into new fully connected feature $f_{fc1}$ as in Equation 2,

$$f_{fc1} = f_{flat}W_{flat} + b_{flat}$$

We use max pooling and dropout on the $f_{fc1}$ to get new feature $f_{fc2}$.

### Logistic Regression

Fully connected feature $f_{fc2}$ after max pooling and drop out will be used as the final high level feature. In our model, a positive triplet has score 1 while a negative triplet has score 0. It is proper to use logistic regression to calculate scores with range $(0,1)$ for every triplet. The final score function on $f_{fc2}$ is in Equation 3 where $W_{fc2}$ and $b_{fc2}$ are the linear mapping weight and bias need to be learned.

$$score(h, l, t) = \text{sigmoid}(f_{fc2}W_{fc2} + b_{fc2})$$ (3)

### Loss Function

We use $cnn(h, l, t)$ to represent the output score of proposed CNN model in short. Similar to TransE, we treat learning CNN model as a pairwise ranking problem that one positive triplet should have higher score than the negative triplets constructed according to Equation 1. A marginal ranking loss function can be used to learn the model. The loss function is showed in Equation 4.

$$\sum_{(h, l, t) \in S} \sum_{(h', l', t') \in S'(h, l, t)} [\gamma + cnn(h', l', t') - cnn(h, l, t)]_+$$

where $[\cdot]_+ = max(0, \cdot)$ and $\gamma$ is hyper-parameter of the ranking loss. In our algorithm we set the default value of $\gamma$ as 1.

### Learning CNN Model

We need to learn two sets of parameters: (1) the entity and relationship embedding in $E$ and $L$. (2) the CNN parameters set $\Phi_{CNN}$ including parameters of $c$ convolutional kernels with size $3 \times 3$, fully-connected mapping parameters $W_{flat}$ and $b_{flat}$ and logistic regression parameters $W_{fc2}$ and $b_{fc2}$. To learn parameters and optimize the loss function in Equation 4, we use mini-batch stochastic gradient descent method. The training batch samples is generated as follows: suppose we set batch size as $b$, we first randomly choose $b$ positive triplets from positive training set $S$, for every positive triplet $(h, l, t)$ we generate one negative triplet $(h', l', t')$ by using Equation 1. We need to point out here that when constructing negative samples, we can corrupt one positive triplet by randomly replacing its head or tail. However, since the training triplets in the knowledge graph is not complete, some constructed "negative" triplets may hold. As a result, these false negative triplets will be noise when training. 

(Wang et al. 2014) proposed a new sampling method: in real knowledge graph, there are different kinds of relationships: one-to-many, many-to-one or many-to-many. When corrupting one triplet, different probabilities for replacing head or tail, relationship and tail together. Suppose the training triplets in the knowledge graph is not complete, some constructed “negative” triplets may hold. As a result, these false negative triplets will be noise when training.

### Table 1: Score functions on triplet $(h, l, t)$ of different knowledge graph models

| Model  | Score$(h, l, t)$ |
|--------|------------------|
| RESCAL | $hW_l$           |
| TransE | $||h + l - t||_p$ |
| TransH | $||h - W^T_lW_T + r - (t - W^T_lW_t)||_2$ |
| TransR | $||hM + t - tM||_2$ |
| NTN    | $u^T_f(h^T_W + W^T_fh + W_t(t + b_t))$ |
| ProjE  | $g(Wfc(h, l, t) + b)$ |
| CNN    | $cnn(h, l, t)$   |

In our CNN based Knowledge Graph model, both embedding and CNN based score function are unknown. We learn CNN model and entity and relationship representations simultaneously. In Fig 1 we show the framework of our model.

**Combination**

Given any triplet $(h, l, t)$, we first combine three vectors together as a matrix $m \in \mathbb{R}^{3 \times k}$. After that, CNN model is applied on this matrix and a score will be assigned to this triplet.

**Convolution Operation**

In the CNN model, we first use multiple $3 \times 3$ kernels to do convolution operation over the combined matrix $m$. Since the height of $m$ is 3, we will use kernels with the same height $g$ as the input matrix. As a result, convolution operation will only go through over the row of matrix $m$ as showed in Fig 2. Different from CNN kernels on images that go through rows and columns on image matrix. In our model, we will explore locally connected structure over the head, relationship and tail together. Suppose the kernel number(kernel channel) is $c$, for the matrix $m$, we will get $c$ feature maps with size $1 \times (k - g + 1)$. RELU activation function $RELU(x) = max(0, x)$ will be applied to get non-negative feature maps.

In fully connection step, we flatten the subsampling feature maps into one feature vector $f_{flat}$ with size $c \times ((k - g)/2 + 1)$. After we get the feature vector, we will use a linear mapping method to map the feature into new fully connected feature $f_{fc1}$ as in Equation 2.

$$f_{fc1} = f_{flat}W_{flat} + b_{flat}$$

We use max pooling and dropout on the $f_{fc1}$ to get new feature $f_{fc2}$.
Figure 1: CNN based Knowledge Graph Learning Framework

Figure 2: convolution kernels go through over the row of triplet matrix

tail entity are set in order to reduce the chance of generating false negative triplets. We will also use this method to create negative samples. After that, there are $b$ pairs of positive and negative triplets in the batch. We minimize the loss of these $b$ pairs of positive and negative triplets in the batch. We first initialize the embedding and the CNN model parameters random initial values. At each main iteration, multiple batches are created and used as training data, mini-batch stochastic gradient descent method is used to update all the parameters. The algorithm is stopped by using a fixed main iteration number. The details are in Algorithm 1.

Algorithm 1 Learning Knowledge Graph Embedding with CNN Model

Input: Training Set $S = (h, \ell, t)$, entity and relationship set $E$ and $L$, margin $\gamma$, embedding dimension $k$ 
initialize: $e, \ell \leftarrow \text{uniform}(-\frac{6}{\sqrt{k}}, \frac{6}{\sqrt{k}})$ for $e \in E, \ell \in L$, $\Phi_{CNN} \leftarrow \text{uniform}(-0.1, 0.1)$ for parameters in CNN.
loop
  for batch = 1 : batch_num
    1. $S_{batch} \leftarrow \text{sample}(S, b)$, construct negative triplets $S'_{batch}$.
    2. Calculate gradient $\nabla L_{batch}$ of Equation 4 w.r.t. $S_{batch}$ and $S'_{batch}$.
    3. Update embedding and $\Phi_{CNN}$ w.r.t. $\nabla L_{batch}$.
  end for
end loop

Table 2: Statistics of datasets

| Dataset | #entity | #relation | #train  | #valid | #test  |
|---------|---------|-----------|---------|--------|--------|
| FB15k   | 14951   | 1345      | 483142  | 50000  | 59071  |
| WN18    | 40943   | 18        | 141442  | 5000   | 50000  |

Experiments

Dataset

We conduct the experiments on two public dataset which are widely used in knowledge graph learning models: FB15k and WN18 [Bordes et al. 2013]. FB15k is created based on Google Knowledge Graph Freebase dataset. This dataset contains various entities such as people, places, events and so on, it also contains thousands of relationships. WN18 is generated from WordNet [Miller 1995]. In Table 2, we show the statistical details including entity and relationship numbers, triplet size in training, validation and testing set.

Evaluation Protocol We use two evaluate metrics by following [Bordes et al. 2011]. For each test triplet, we corrupt the head by using other entities in the entity set $E$ in turn and calculate the scores for the test triplet and all the corrupted triplets. After that we rank these triplets with their scores by descending order. Finally we get the ranking of correct entity. If the ranking of the correct entity is smaller or equal to 10, $\text{Hit}@10$ for the test triplet is equal to 1, or it will be 0. For all the triplets in the testing data, we repeat the same procedure and get the Mean Rank scores and mean value $\text{Hit}@10$. We will also replace tails of the triplets and calculate the Mean Rank and $\text{Hit}@10$. We report the average scores on head prediction and tail prediction as final evaluation results.

When constructing corrupted triplets, some of them may hold in training or validation set. we will remove from the list first and then use the filtered triplets to get the two evaluation results.
### Experimental Setup

In our experiments, we implement the algorithm in Python with Tensorflow (www.tensorflow.org). We use Adam (Kingma and Ba 2014) to optimize the model and learn the parameters. In our model, we can set the width of convolutional kernels with different size, for simplicity we fix the kernel size as 3 × 3. When using pairwise ranking loss to learn CNN, we fix the margin value γ as 1. The learning rate in our model is fixed as 1e − 3 for FB15k and 1e − 4 for WN18. Batch size is 1000 for both dataset. Epoch number is set at 1000 for FB15k and 2000 for WN18. Other hyper-parameters in our model are: dimension of entity and relationship embedding k, number of convolutional kernel c, fully connected hidden dimension d_{fc2} and CNN dropout rate p. We select k from {50, 100, 200}, c from {8, 16, 32}, d_{fc2} from {128, 256, 512} and p from {0, 0.5}. We use the Hits@10 to select parameters on the validation set for both datasets. For FB15k the selected parameters are: k = 200, c = 16, d_{fc2} = 256, p = 0.5. For WN18 the selected parameters are k = 200, c = 16, d_{fc2} = 256, p = 0.

### Link Prediction Results

We compare our proposed CNN model with several Knowledge Graph learning models including translation based embedding learning models: TransE, TransH, TransR, PTransE and one neural network based model ProjE.

In Table 3 we show the experimental results on FB15k and WN18 with two evaluation metrics Mean Rank and Hits@10. The experimental results of other models are cited from the original papers. Missing values on WN18 means that the original papers did not show the results on this dataset.

From the table we can see that on FB15k, CNN can achieve 94.5 on Hits@10 which is much better than other methods. In fact our CNN approach is the only method that can achieve more than 90 in all the models. The MeanRank value is 68 that is worse than ProjE and PTransE. PTransE model has MeanRank value 34 which is hard to beat. On WN18, Our model has the best result on evaluation Hits@10, it can achieve to 92 while our model can get to 89.4. On MeanRank evaluation, our model is significantly better than others: for all the comparison method, their MeanRank values are higher than 200, our CNN model can achieve to a low value 17. From the experimental results we can conclude that CNN model can learn good entity and relationship embedding. CNN can have good Hits@10 performance on FB15k and extremely good MeanRank performance on WN18.

### Triplet Classification

We also test our model on triplet classification: given one pair of head h and a tail t, we try to find the correct relationship ℓ from relationship set L. Similar to Link Prediction evaluation protocol, for each test triplet, we corrupt the relationship by using other relationships in the relationship set L in turn and calculate the scores. After that we rank these triplets in descending order. We also use MeanRank to evaluate the triplet classification results. Instead of using Hits@10 we use Hits@1: we only check if the first relationship in the sorted list is the correct one. We also compare our CNN model with other methods. The results are in Table 4.

Table 4: Triplet Classification on FB15k.

| FB15k | Mean Rank | Hits@10(%) |
|-------|-----------|------------|
| TransE | 2.8 | 84.3 |
| PTransE | 1.4 | 94.0 |
| ProjE | 1.2 | 97.7 |
| CNN | 1.8 | 97.3 |

### Convolutional Kernels Analysis

We use convolutional kernels on knowledge graph triplets to learn complex connections between entities and relationships. In this section, we analysis the effect of CNN structure. We use a simpler MLP model directly without convolutional kernels and learn embedding: first of all, we connect the k dimensional h, ℓ and t together as a 3k dimension vector, after that we use a hidden layer with tanh activation function to get a new vector. Finally, logistic regression is applied on the hidden layer nodes to get a score. The learning algorithm is similar to our proposed model. We use the same way to get negative samples and also use mini-batch gradient descent method to learn the regression model. For both datasets, the embedding dimensions are selected from {50, 100, 200} and hidden dimensions are selected from {128, 256, 512}. For FB15k, embedding dimension is set as 200, hidden dimension is set as 128. For WN18, embedding dimension is In Table 5 we compare the results of CNN model and logistic regression model.
### Representation Distributions

Translation based embedding learning and CNN based model treat entity and relationship connections in different ways. After we learn the embedding, we compare the representation distributions of the entities of CNN model with TransE model. We learn the FB15k representations with TransE by using the settings reported in [Bordes et al. 2013] and get 50 dimensional vectors for each entity. We also learn representations with our proposed model with ranking loss and get 200 dimensional vectors for each entity. After that we use t-SNE method [van der Maaten and Hinton 2008] to reduce the representations to 2 dimensional space. We use Google Knowledge Graph API to collect the types of entities. 2188 of the whole 14951 entities in FB15k no longer exist and we have 12763 labeled entities left. There are one or multiple types for each entity. The types include Person, Movie, City and so on. In our experiments, we select a subset including Movie, Organization, Place and Person. If one entity belongs to more than one type, we will assign it to the type with fewer entities in the dataset. The representations distributions of TransE and CNN model is showed in Figure 3.

From Figure 3 we can get the similarity and difference between TransE and CNN. We can see that in TransE and our neural network model CNN, entities with the same type will be clustered automatically. However the cluster performance of TransE is not as good as CNN. The entity distribution in CNN is more smooth than TransE. For example, in TransE model (Figure 3a), some entities of type Place are represented together into subgroup which means they are represented similarly. On the other hand, the subgroups of Place are far away from each other and there are “blank” area in the group. This will lead to the result that when entities of other types translate to Place, it is difficult to distinguish the positive entity from its neighbors in the same subgroup. In contrast, in our model, the cluster results are better, in addition we connect entities and relationships in higher level non-linear mapping and it can avoid such situation.

### Conclusion and Future Work

In this paper, we propose a neural network based knowledge graph learning model. We use Convolutional Neural Network (CNN) to learn the sequential entity and relationship representations. Complex correlations between entities and relationships are also learned with CNN structure. Experimental results show that CNN model is effective to learn representations and entity and relationship connections. In our model, we treat the learning procedure as a pairwise ranking problem: a positive triplet should have higher score than the corresponding negative triplet generated from it. In fact, we can also treat the embedding learning as a binary classification problem, the positive triplets will have label 1 while negative triplets will have label 0. In future we will try to use different binary classification loss functions to train the model.

### References

[Bordes et al. 2011] Bordes, A.; Weston, J.; Collobert, R.; and Bengio, Y. 2011. Learning structured embeddings of knowledge bases. In Proceedings of the Twenty-Fifth Conference on Artificial Intelligence.

[Bordes et al. 2013] Bordes, A.; Usunier, N.; Garcia-Duran, A.; Weston, J.; and Yakhnenko, O. 2013. Translating embeddings for modeling multi-relational data. In Advances in Neural Information Processing Systems 26.

[Dauphin et al. 2017] Dauphin, Y. N.; Fan, A.; Auli, M.; and Grangier, D. 2017. Language modeling with gated convolutional networks. In Proceedings of the 34th International Conference on Machine Learning.

[Garca-Durn, Bordes, and Usunier 2015] Garca-Durn, A.; Bordes, A.; and Usunier, N. 2015. Composing relationships with translations. In Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing.

[Jia et al. 2016] Jia, Y.; Wang, Y.; Lin, H.; Jin, X.; and Cheng, X. 2016. Locally adaptive translation for knowledge graph embedding. In Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence.

[Kingma and Ba 2014] Kingma, D., and Ba, J. 2014. Adam: A method for stochastic optimizations. In Proceedings of the Third International Conference on Learning Representations.

[Krizhevsky, Sutskever, and Hinton 2012] Krizhevsky, A.; Sutskever, I.; and Hinton, G. E. 2012. Imagenet classification with deep convolutional neural networks. In Advances in Neural Information Processing Systems 25.

[Lehmann et al. 2015] Lehmann, J.; Isele, R.; Jakob, M.; Jentzsch, A.; Kontokostas, D.; Mendes, P. N.; Hellmann, S.; Morsey, M.; van Kleef, P.; Auer, S.; and Bizer, C. 2015. DBpedia - a large-scale, multilingual knowledge base extracted from wikipedia. Semantic Web Journal 6(2):167–195.

[Lin et al. 2015a] Lin, Y.; Liu, Z.; Luan, H.; Sun, M.; Rao, S.; and Liu, S. 2015a. Modeling relation paths for representation learning of knowledge bases. In Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing.

[Lin et al. 2015b] Lin, Y.; Liu, Z.; Sun, M.; Liu, Y.; and Zhu, X. 2015b. Learning entity and relation embeddings for knowledge graph completion. In Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence.

[Miller 1995] Miller, G. A. 1995. Wordnet: a lexical database for english. Commun. ACM 38(11):39–41.

[Nickel et al. 2016] 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):11–33.

[Nickel, Rosasco, and Poggio 2016] Nickel, M.; Rosasco, L.; and Poggio, T. A. 2016. Holographic embeddings of knowledge

### Table 5: CNN model vs MLP

| Model | Link Prediction | Triplet Classification |
|-------|----------------|------------------------|
| MR Hits@10 | MR Hits@1 |
| CNN | 68 | 95.4 | 1.8 | 97.3 |
| MLP | 674 | 82.9 | 51.3 | 87.7 |

| Model | Link Prediction | Triplet Classification |
|-------|----------------|------------------------|
| MR Hits@10 | MR Hits@1 |
| CNN | 41 | 89.4 | 1.2 | 90.8 |
| MLP | 4563 | 48.4 | 3.5 | 74.2 |
graphs. In Proceedings of the Thirtieth Conference on Artificial Intelligence.

[Nickel, Tresp, and Kriegel 2011] Nickel, M.; Tresp, V.; and Kriegel, H.-P. 2011. A three-way model for collective learning on multi-relational data. In International Conference on Machine Learning.

[Shi and Weninger 2017] Shi, B., and Weninger, T. 2017. Proje: Embedding projection for knowledge graph completion. In Association for the Advancement of Artificial Intelligence.

[Socher et al. 2013] Socher, R.; Chen, D.; Manning, C. D.; and Ng, A. 2013. Reasoning with neural tensor networks for knowledge base completion. In Advances in Neural Information Processing Systems.

[Sutskever, Tenenbaum, and Salakhutdinov 2009] Sutskever, I.; Tenenbaum, J. B.; and Salakhutdinov, R. R. 2009. Modelling relational data using bayesian clustered tensor factorization. In Advances in Neural Information Processing Systems.

[van der Maaten and Hinton 2008] van der Maaten, L., and Hinton, G. E. 2008. Visualizing high-dimensional data using t-sne. Journal of Machine Learning Research 9:2579–2605.

[Wang and Li 2016] Wang, Z., and Li, J. 2016. Text-enhanced representation learning for knowledge graph. In Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence.

[Wang et al. 2014] Wang, Z.; Zhang, J.; Feng, J.; and Chen, Z. 2014. Knowledge graph embedding by translating on hyperplanes. In Association for the Advancement of Artificial Intelligence.

[Xiao et al. 2017] Xiao, H.; Huang, M.; Meng, L.; and Zhu, X. 2017. SSP: semantic space projection for knowledge graph embedding with text descriptions. In Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence.

[Xiao, Huang, and Zhu 2016] Xiao, H.; Huang, M.; and Zhu, X. 2016. From one point to a manifold: Knowledge graph embed-