MultiE: Multi-Task Embedding for Knowledge Base Completion

Zhao Zhang[1,2], Fuzhen Zhuang[1,2], Zheng-Yu Niu[3], Deqing Wang[4*], Qing He[1,2]
[1]Key Lab of Intelligent Information Processing of Chinese Academy of Sciences (CAS),
Institute of Computing Technology, CAS, Beijing 100190, China
[2]University of Chinese Academy of Sciences, Beijing 100049, China
[3]Baidu Inc., Beijing, P.R.China
[4]School of Computer Science and Engineering, Beihang University, China
{zhangzhao2017,zhuangfuzhen,heqing}@ict.ac.cn,dqwang@buaa.edu.cn
*corresponding author

ABSTRACT

Completing knowledge bases (KBs) with missing facts is of great importance, since most existing KBs are far from complete. To this end, many knowledge base completion (KBC) methods have been proposed. However, most existing methods embed each relation into a vector separately, while ignoring the correlations among different relations. Actually, in large-scale KBs, there always exist some relations that are semantically related, and we believe this can help to facilitate the knowledge sharing when learning the embedding of related relations simultaneously. Along this line, we propose a novel KBC model by *Multi*-Task *E*mbedding, named MultiE. In this model, semantically related relations are first clustered into the same group, and then learning the embedding of each relation can leverage the knowledge among different relations. Moreover, we propose a three-layer network to predict the missing values of incomplete knowledge triples. Finally, experiments on three popular benchmarks FB15k, FB15k-237 and WN18 are conducted to demonstrate the effectiveness of MultiE against some state-of-the-art baseline competitors.

KEYWORDS

Multi-Task Learning; Knowledge Base Completion; Embedding

ACM Reference Format:

Zhao Zhang, Fuzhen Zhuang, Zheng-Yu Niu, Deqing Wang, Qing He. 2018. MultiE: Multi-Task Embedding for Knowledge Base Completion. In *The 27th ACM International Conference on Information and Knowledge Management (CIKM'18), October 22–26, 2018, Torino, Italy.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3269206.3269295

1 INTRODUCTION

Knowledge bases (KBs), such as WordNet [5] and Yago [9] are extremely useful resources for many AI-related applications. Despite their large sizes, modern KGs suffer from incompleteness. Therefore, how to complete KBs is still a crucial but challenging problem. The task of knowledge base completion (KBC) aims to fill the missing values into incomplete triples. More formally, the goal of KBC is to predict either the head entity given a query $(?, r, t)$ or the tail entity given $(h, r, ?)$.

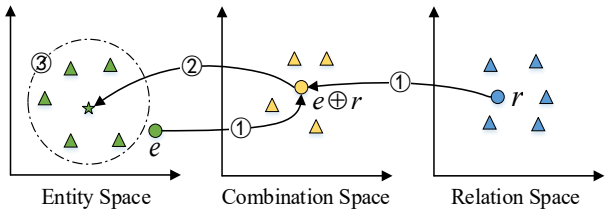

Figure 1: The Spatial Transformations in MultiE. $h \oplus t$ **denotes the combination of** h **and** t**. ①, ② and ③ denote the first, second and third layer of the network.**

Recently, a number of algorithms have been proposed for the KBC problem, and the most successful models use low-dimensional embedding vectors to represent entities and relations [1, 15]. However, most existing models suffer from the following two problems: (1) each relation is embedded into a latent vector separately, and the correlations among different relations are not exploited. Actually, semantically similar relations widely exist in KBs, for example, in FB15k, *'/location/country/languages_spoken'* and *'/location/country/official_language'* have a high similarity. The information from similar relations may be of great value, which can provide meaningful associations among related relations, and supply more training data for less frequent relations; (2) most models embed entities and relations into a unified space. However, we argue that the entities and relations should be better embedded into separate entity space and relation space.

Inspired by multi-task learning (MTL), which conducts multiple related learning tasks simultaneously and promotes the performance of each individual task, we treat the embedding task for each relation as a single task and propose a novel KBC model by *Multi*-Task *E*mbedding, named MultiE. MultiE first clusters all the relations into different groups based on pre-trained relation embeddings, and then relations in the same group are trained simultaneously. In this way, the embedding of each relation can not only leverage the knowledge from itself but also other relations in the same group. Moreover, three semantic spaces are considered in MultiE to find missing values of incomplete triples, including an entity space, a relation space and a combination space, which are shown in Figure 1. Given a query $(h, r, ?)$ or $(?, r, t)$, the first layer combines the input entity e and input relation r from separate entity space and relation space into a combination space, in this way, the input information can be represented in a combined way. The second layer transforms the combination of e and r to the entity

CIKM '18, October 22–26, 2018, Torino, Italy

ACM ISBN 978-1-4503-6014-2/18/10…$15.00
https://doi.org/10.1145/3269206.3269295

space, and the third layer ranks the candidates (denoted as green triangles) based on the combination of e and r in the entity space (denoted as a green star). Experiments demonstrate the expressivity and effectiveness of our model.

In summary, our work is highlighted as follows,

(1) Inspired by MTL, we propose MultiE, which exploits and leverages the correlations among different relations to improve the performance.
(2) We believe entities and relations should be embedded into separate entity space and relation space, and further consider three spaces. In this way, MultiE can embed entities and relations into proper semantic spaces, which guarantees the expressivity of our model.
(3) A three-layer network is adopted by MultiE, which can compute a ranking score for each entity candidate with limited parameters. Finally, experimental results demonstrate the effectiveness of the proposed model.

2 RELATED WORK

A variety of low-dimensional representation-based models have been developed for the KBC task. TransE [1] is one of the most widely used models along this line. TransE treats the relation between two entities as a translation from the head entity to the tail entity, i.e., $\boldsymbol{h} + \boldsymbol{r} \approx \boldsymbol{t}$ when (h, r, t) holds. Following this way, TransH [13] utilizes relation-specific hyperplanes to lay the entities. TransR [4] embeds entities and relations into separate entity space and relation-specific spaces. DistMult [15] adopts a bilinear energy function to compute the scores given (h, r, t) triples. ComplEx [14] embeds entities and relations into complex vectors instead of real-valued ones. HolE [6] employs circular correlations to create compositional representations. Except the above mentioned models which only utilize the KB information, some works also use paths or text as auxiliary information, like R-GCN [7], TEKE_H [12].

MTL conducts multiple related learning tasks simultaneously so that the useful information in one task can be used for other tasks [2]. Some research studies have shown the benefits of applying multi-task learning to NLP-related tasks, like relation extraction [3] and path ranking algorithms in KBs [11]. This paper investigates the possibility of multi-task learning with entity and relation embedding for KBC.

3 METHODOLOGY

In this section, we first introduce how MTL is used in MultiE, and then the architecture of MultiE is explained. Finally, we give detailed descriptions of the ranking loss function and implementation details. First of all, let us introduce some notations. A KB is viewed as a graph $\mathcal{G} = \{(h, r, t)\} \subseteq \mathcal{E} \times \mathcal{R} \times \mathcal{E}$, where $\mathcal{E}$ and $\mathcal{R}$ are the entity (node) set and relation (edge) set respectively.

3.1 Multi-Task Learning

Large-scale KBs always have a large number of relations, and there exist some semantically similar relations among them. Inspired by MTL, which conducts multiple related learning tasks simultaneously so that the useful information in one task can be used for other tasks, we treat the embedding task for each relation as a single task and adopt the MTL method to solve the related tasks together. In our model, semantically similar relations are trained in a collective way.

First of all, we cluster relation embeddings learned from the previous model into different groups. Since previous studies have shown that the embeddings of semantically similar relations locate near each other in the latent space [15], we cluster the relations based on the results of ProjE [8] (In this paper, we use the k-means algorithm). In this way, similar relations can be clustered into the same group.

In MultiE, the embedding of each relation is comprised of two parts, which is defined as follow,

$$\boldsymbol{r} = \boldsymbol{r}_0 + \boldsymbol{r}', \tag{1}$$

where $\boldsymbol{r}_0$ is the common model parameter for all the relations in the same group, and $\boldsymbol{r}'$ is the specific parameter for each individual relation, which represents the relation-specific characteristic.

3.2 Model Architecture

As introduced in Section 1, the entity, relation, and combination are embedded into separate semantic spaces in MultiE, i.e., $\boldsymbol{e} \in \mathbb{R}^k$, $\boldsymbol{r} \in \mathbb{R}^d$, $\boldsymbol{e} \oplus \boldsymbol{r} \in \mathbb{R}^p$, where $\boldsymbol{e}$ and $\boldsymbol{r}$ represent entity and relation embeddings, $\oplus$ is the combination operator, which will be introduced later, and k, d, p are the dimensions of the entity space, relation space and combination space, respectively.

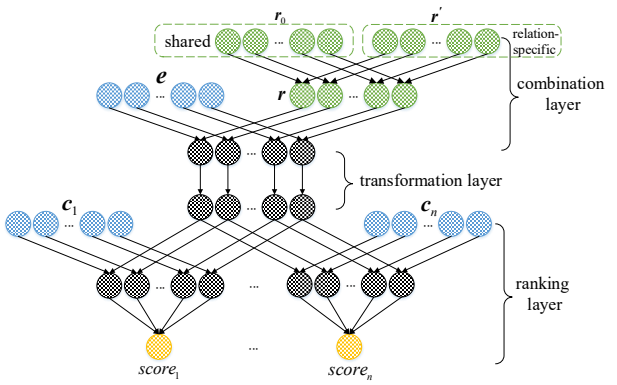

Figure 2: The Network Framework of MultiE.

MultiE adopts a three-layer network as shown in Figure 2 to rank entity candidates in an appropriate order, that is, positive candidates should precede negative ones. In Figure 2, the entity nodes, relation nodes and score nodes are colored in blue, green and yellow respectively. Given an input query $(h, r, ?)$ or $(?, r, t)$, the first layer combines the input entity and relation into a joint vector. The combination operator is defined as follow,

$$\boldsymbol{e} \oplus \boldsymbol{r} = f\left(\boldsymbol{W}_e \boldsymbol{e} + \boldsymbol{W}_r \boldsymbol{r} + \boldsymbol{b}_c\right), \tag{2}$$

where f is the tanh activation function, $\boldsymbol{W}_e \in \mathbb{R}^{p \times k}$ and $\boldsymbol{W}_r \in \mathbb{R}^{p \times d}$ are weight matrices and $\boldsymbol{b}_c \in \mathbb{R}^p$ is the bias vector. Entities and relations can be regarded as being embedded into separate spaces since they use different weight matrices $\boldsymbol{W}_e$ and $\boldsymbol{W}_r$. In this way, the input information can be obtained by $\boldsymbol{e} \oplus \boldsymbol{r} \in \mathbb{R}^p$ in the combination space.

The second layer is the transformation layer, which transforms the combination of the input entity and relation from the combination space to the entity space. The operation in the second layer is

defined as

$$(\boldsymbol{e} \oplus \boldsymbol{r})' = f\left(\boldsymbol{W}_t\left(\boldsymbol{e} \oplus \boldsymbol{r}\right) + \boldsymbol{b}_t\right), \tag{3}$$

where $\boldsymbol{W}_t \in \mathbb{R}^{k \times p}$ is the transformation matrix, $\boldsymbol{b}_t \in \mathbb{R}^k$ is the bias of the transformation operation, and $(\boldsymbol{e} \oplus \boldsymbol{r})' \in \mathbb{R}^k$ is the transformed vector in the entity space.

After the transformation operation of the second layer, the combination of the input entity and relation is in the entity space, which gives an opportunity to rank entity candidates in a collective way. The third layer is a ranking layer, which computes a ranking score for each candidate. Given the i-th candidate entity c_i, the ranking score is defined as follow,

$$s(\boldsymbol{e}, \boldsymbol{r}, \boldsymbol{c}_i) = g(\boldsymbol{c}_i^\top\left(\boldsymbol{e} \oplus \boldsymbol{r}\right)' + b_r), \tag{4}$$

where b_r is the bias of the third layer, and g is the softmax function.

In order to reduce the total number of parameters, we set the entity space, relation space and combination space to share the same dimension, i.e., $k = d = p$ and set all the matrices to be diagonal matrices. Experiments show that even this simple setting can help to improve the training efficiency without damaging the results. Compared to the basic parameters with the size of $k|\mathcal{E}| + d|\mathcal{R}|$ that turn entities and relations into embedding vectors, MultiE increases the number of parameters by $p|C| + 10k + 2$, where $|C|$ ($|C| < |\mathcal{R}|$) is the number of relation clusters, $5k$ are for the $\boldsymbol{W}_e, \boldsymbol{W}_r, \boldsymbol{b}_c, \boldsymbol{W}_t$ and $\boldsymbol{b}_t$ and 1 for b_r. Note that the matrices and biases above are not shared when predicting heads and predicting tails. MultiE has a smaller number of parameters compared with TransR, which uses a total number of $k|\mathcal{E}| + d|\mathcal{R}| + kd|\mathcal{R}|$ parameters to embed entities and relations separately into an entity space and relation-specific spaces.

3.3 Loss Function and Implementation Details

The KBC problem is viewed as a ranking problem by MultiE. That is, positive candidates should be ranked above the negative ones. Following ProjE, we believe the scores of candidate entities should be computed collectively rather than using a margin-based loss and computing the scores independently.

For the input entity e and relation r in a given query, we first use the candidate sampling algorithm introduced in ProjE [8] to sample positive and negative candidates, and minimize the following loss function,

$$\mathcal{L}(\boldsymbol{e}, \boldsymbol{r}) = -\sum_i^{|\boldsymbol{y}|} 1(y_i = 1)\log(s(\boldsymbol{e}, \boldsymbol{r}, \boldsymbol{c}_i)) + \lambda_1 \left\|\boldsymbol{r}_0\right\|_2^2 + \lambda_2 \left\|\boldsymbol{r}'\right\|_2^2, \tag{5}$$

where $\boldsymbol{y}$ is a binary label vector and $y_i = 1$ means the i-th candidate is positive, $\boldsymbol{c}_i$ denotes the embedding of the i-th candidate. λ_1 and λ_2 are the trade-off parameters. Large value of λ_1 will result in the separate training of each relation, while large value of λ_2 will lead to all relations in the same group sharing the same embedding vector. Since the score function Equation (4) uses a softmax function, the scores of each candidate can be computed collectively.

Following the same protocol as TransE, we divide all the relations into 4 categories: 1-to-1, 1-to-M, M-to-1 and M-to-M relations where M stands for 'Many'. To reduce the weights of the M-to-1, 1-to-M and M-to-M relations, we also introduce a normalized version of MultiE called N-MultiE, where N stands for 'Normalized'. N-MultiE has the following loss function,

$$\mathcal{L}(\boldsymbol{e}, \boldsymbol{r}) = -\sum_i^{|\boldsymbol{y}|} \frac{1(y_i = 1)}{\sum_j^{|\boldsymbol{y}|} 1(y_j = 1)} \log(s(\boldsymbol{e}, \boldsymbol{r}, \boldsymbol{c}_i)) + \lambda_1 \left\|\boldsymbol{r}_0\right\|_2^2 + \lambda_2 \left\|\boldsymbol{r}'\right\|_2^2. \tag{6}$$

For both MultiE and N-MultiE, we introduce the single-task learning (STL) version of the two models by setting $\boldsymbol{r}_0$ in Equation (1) as $\boldsymbol{0}$, i.e., all the relations don't share embedding parameters, which are denoted as MultiE-STL and N-MultiE-STL.

The learning process of our model is carried out using the Adam optimizer. A L_1 regularizer is applied to all the parameters during the training phase. We also apply dropout with a rate of 0.5 to the combination layer and transformation layer. Following TransE, we initialize all the parameters with a uniform distribution $U\left[-\frac{6}{\sqrt{k}}, \frac{6}{\sqrt{k}}\right]$. In the test phase, given an input query, the three-layer network can calculate a ranking score for each candidate entity. The larger the score is, the more likely the candidate is true.

4 EXPERIMENTS

4.1 Datasets and Experimental Settings

In this paper, we evaluate our model on three popular benchmarks FB15k [1], FB15k-237 [10] and WN18 [1]. The statistics of the three datasets are summarized in Table 2.

The number of relation clusters C, learning rate r, embedding size k, batch size b, regularizer weight γ, the probability for negative sampling p and the trade-off parameters λ_1, λ_2 are set as follows: for FB15k, we set $C = 300, r = 0.0005, k = 250, b = 200, \gamma = 1e-5, p = 0.5, \lambda_1 = 1e-6, \lambda_2 = 1e-5$ for MultiE, N-MulitE, MultiE-STL and N-MultiE-STL; for FB15k-237, we set $C = 120, r = 0.01, k = 200, b = 200, \gamma = 1e-5, p = 0.5, \lambda_1 = 1e-5, \lambda_2 = 1e-4$; while for WN18, we set $C = 10, r = 0.01, k = 250, b = 200, \gamma = 1e-5, p = 0.5, \lambda_1 = 1e-5, \lambda_2 = 1e-5$.

In the test phase, we follow the same protocol as TransE: for each triple in the test data, we replace the head/tail entity by all the entities in the KB. Then, the scores of the candidate entities are calculated by the three-layer network. All the candidates are ranked in an descending order according to the scores. We compare our model with various kinds of baselines, including TransE [1], TransR [4], DistMult [15], HolE [6], ComplEx [14] and the most recent competitors , ProjE[8] and R-GCN+[7] using three measures as our evaluation metrics: (1) Mean Rank (MR); (2) Mean Reciprocal Rank (MRR); (3) Hits@n (the proportion of ranks not larger than n). We report all the results in a filtered setting as introduced in TransE.

4.2 Experimental Results

Evaluation results are shown in Table 1. In Table 1, Hits@n are denoted as Hn for short. From Table 1, we have the following findings: (1) Our model can outperform other baselines on most metrics with a large margin. On the more challenging dataset FB15k-237, N-MultiE outperforms the best baseline R-GCN+ on the metric of Hits@10 with a margin as large as 7.4%. Experiments demonstrate the expressivity and effectiveness of our model; (2) For FB15k and FB15k-237, MultiE and N-MultiE outperform the corresponding STL models. While for WN18, the results are opposite. We think the reason is that the relevance of relations in FB15k and FB15k-237 is

Table 1: KBC results. For FB15k and WN18, we take the results of TransE, TransR and HolE from [6], take the results of DistMult and ComplEx from [14] and take the results of R-GCN+ from [7]. For FB15k-237, we take the results of all the baselines except ProjE from [7]. For both datasets, we download the code of ProjE from https://github.com/bxshi/ProjE and run ourselves. In this table, ProjE denotes ProjE_wlistwise, which achieves the best results in [8].

	FB15k					FB15k-237					WN18				
	MR	MRR	H10	H3	H1	MR	MRR	H10	H3	H1	MR	MRR	H10	H3	H1
TransE [1]	125	0.463	0.749	0.578	0.297	-	0.233	0.398	0.263	0.147	251	0.495	0.943	0.888	0.113
TransR [4]	77	0.346	0.582	0.404	0.218	-	-	-	-	-	225	0.427	0.940	0.876	0.335
DistMult [15]	-	0.654	0.824	0.733	0.546	-	0.191	0.376	0.207	0.106	-	0.532	0.936	0.914	0.728
HolE [6]	-	0.524	0.739	0.613	0.402	-	0.222	0.391	0.253	0.133	-	0.938	0.949	0.945	0.930
ComplEx [14]	-	0.692	0.840	0.759	0.599	-	0.201	0.388	0.213	0.112	-	0.941	0.947	0.945	0.936
ProjE [8]	**34**	0.727	0.884	0.772	0.646	237	0.241	0.410	0.258	0.160	271	0.817	0.948	0.928	0.913
R-GCN+ [7]	-	0.696	0.842	0.760	0.601	-	0.249	0.417	0.264	0.151	-	0.819	**0.964**	0.929	0.697
MultiE-STL	56	0.756	0.883	0.817	0.668	240	0.262	0.441	0.287	0.179	257	0.940	0.949	0.945	0.939
MultiE	42	**0.775**	**0.887**	**0.832**	**0.681**	247	0.284	0.463	0.312	0.199	248	0.925	0.942	0.938	0.931
N-MultiE-STL	59	0.713	0.836	0.771	0.616	**148**	0.287	0.470	0.322	0.203	229	**0.949**	0.956	**0.949**	**0.941**
N-MultiE	48	0.736	0.859	0.792	0.637	183	**0.309**	**0.491**	**0.339**	**0.219**	**223**	0.938	0.949	0.939	0.933

Table 2: Detailed Information of the Three Datasets.

Dataset	$\|\mathcal{E}\|$	$\|\mathcal{R}\|$	#triples in Train/Valid/Test
FB15k	14,951	1,345	483,142 / 50,000 / 59,071
FB15k-237	14,541	237	272,115 / 17,535 / 20,466
WN18	40,943	18	141,442 / 5,000 / 5,000

much stronger than that of WN18 which only contains 18 relations. For each relation, the information learned from semantically similar relations are useful, while the information learned from unrelated relations may damage the results. The experimental results indicate MultiE and N-MultiE are especially useful for KBs which have dense semantic distributions over relations.

Table 3: Examples of Relation Clusters in FB15k.

	relations
1	/location/country/languages_spoken /location/country/official_language
2	/film/producer/film, /film/writer/film /film/director/film, /film/cinematographer/film
3	/sports/sports_team/roster./soccer/football_roster_position/player /sports/sports_team/roster./sports/sports_team_roster/player /soccer/football_team/current_roster./soccer/football_roster_position/player /soccer/football_team/current_roster./sports/sports_team_roster/player

Table 3 gives some examples of relation clusters in FB15k. Relations in Cluster 1 are language-related relations, in Cluster 2 are film-related relations while in Cluster 3 are sports-related relations. It's obvious that by clustering, semantically similar relations can be clustered into the same group.

5 CONCLUSION

In this paper, we propose MultiE, a MTL-based model for the KBC task. MultiE uses a three-layer network and a ranking-based loss function to predict the missing values of incomplete knowledge triples. In MultiE, three semantic spaces are considered, which guarantees the expressivity of our model. In addition, we propose a normalized version of MultiE to reduce the weights of 1-to-M, M-to-1 and M-to-M relations. Experiments show that our model can outperform other baselines with a large margin, especially on the data sets which have dense semantic distributions over relations.

ACKNOWLEDGEMENTS

The research work is supported by the National Key Research and Development Program of China under Grant No. 2018YFB1004300, the National Natural Science Foundation of China under Grant No. 61773361, 61473273, 91546122, Guangdong provincial science and technology plan projects under Grant No. 2015 B010109005, the Project of Youth Innovation Promotion Association CAS under Grant No. 2017146.

REFERENCES

[1] Antoine Bordes, Nicolas Usunier, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *NIPS*.
[2] Rich Caruana. 1997. Multitask Learning. *Machine Learning*.
[3] Jing Jiang. 2009. Multi-task transfer learning for weakly-supervised relation extraction. In *ACL*.
[4] Yankai Lin, Zhiyuan Liu, Xuan Zhu, Xuan Zhu, and Xuan Zhu. 2015. Learning entity and relation embeddings for knowledge graph completion. In *AAAI*.
[5] George A Miller. 1995. WordNet: a lexical database for English. *Commun. ACM*.
[6] Maximilian Nickel, Lorenzo Rosasco, and Tomaso Poggio. 2016. Holographic embeddings of knowledge graphs. In *AAAI*.
[7] Michael Schlichtkrull, Thomas N. Kipf, Peter Bloem, Rianne Van Den Berg, Ivan Titov, and Max Welling. 2017. Modeling Relational Data with Graph Convolutional Networks. In *arXiv*.
[8] Baoxu Shi and Tim Weninger. 2017. ProjE: Embedding Projection for Knowledge Graph Completion. In *AAAI*.
[9] Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. 2007. Yago:a core of semantic knowledge. In *WWW*.
[10] Kristina Toutanova and Danqi Chen. 2015. Observed versus latent features for knowledge base and text inference. In *Workshop on CVSC*.
[11] Quan Wang, Jing Liu, Yuanfei Luo, Bin Wang, and Chin-Yew Lin. 2016. Knowledge base completion via coupled path ranking. In *ACL*.
[12] Zhigang Wang and Juan-Zi Li. 2016. Text-Enhanced Representation Learning for Knowledge Graph. In *IJCAI*.
[13] Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014. Knowledge Graph Embedding by Translating on Hyperplanes. In *AAAI*.
[14] Johannes Welbl, Sebastian Riedel, and Guillaume Bouchard. 2016. Complex embeddings for simple link prediction. In *ICML*.
[15] Bishan Yang, Wentau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. 2015. Embedding Entities and Relations for Learning and Inference in Knowledge Bases. In *ICLR*.