Predicting relations between RDF entities by Deep Neural Network

Tsuyoshi Murata[1], Yohei Onuki[1], Shun Nukui[1], Seiya Inagi[2], Xule Qiu[2], Masao Watanabe[2], and Hiroshi Okamoto[2]

[1] Department of Computer Science, School of Computing
Tokyo Institute of Technology, W8-59 2-12-1 Ookayama, Meguro, Tokyo, 152-8552
Japan

[2] Research & Technology Group, Fuji Xerox Co., Ltd.
6-1 Minatomirai, Nishi-ku, Yokohama, Kanagawa, 220-8668 Japan

Abstract. In the process of ontology construction, we often need to find relations between entities described by the Resource Description Framework (RDF). Predicting relations between RDF entities is important for developing large-scale ontologies. The goal of our research is to predict a relation (predicate) of two given entities (subject and object). TransE and TransR have been proposed as the methods for such a prediction. We propose a method for predicting a predicate from a subject and an object by using a Deep Neural Network (DNN), and developed RDFDNN. Experimental results showed that predictions by RDFDNN are more accurate than those by TransE and TransR.

1 Introduction

Ontology learning is one of the important topics for developing the Semantic Web. In general, there are many entity pairs where the relations between them are unknown [7][16]. If we can predict such relations accurately, we can augment a given ontology. Since many Semantic Web data (such as Google's Knowledge Graph) are already available, techniques for predicting relations between entities are important for developing large-scale ontologies.

The goal of our research is to predict relations between two given entities in Resource Description Framework (RDF) accurately. RDF is the framework for representing Web resources, and each triple in RDF is composed of three entities (subject, predicate, and object). Subject and object are entities, and predicate is the relation between the entities. Suppose (Tokyo, is-capital-of, Japan) is an example of such a triple. We would like to predict "is-capital-of" when "Tokyo" and "Japan" are given. For this purpose, we propose a method for predicting a predicate from a subject and an object by using a Deep Neural Network (DNN), and developed RDFDNN.

Freebase and Wordnet are used as the datasets of our experiments. The following experiments are performed: (1) comparison with previous methods (Sect. 5), (2) failure analysis (Sect. 6), (3) embedding dimension and prediction accuracy (Sect. 7), and (4) embedding dimension and computational time (Sect.

8). As the results of our experiments, RDFDNN is more accurate than TransE [1] and TransR [10] for predicting a predicate from a given subject and object. We also propose a method for finding appropriate embedding dimensions in RDFDNN.

2 RDFDNN

RDFDNN predicts the relation between two entities represented as a RDF triple. When h and t of a RDF triple (h, l, t) are given as inputs, RDFDNN will output l.

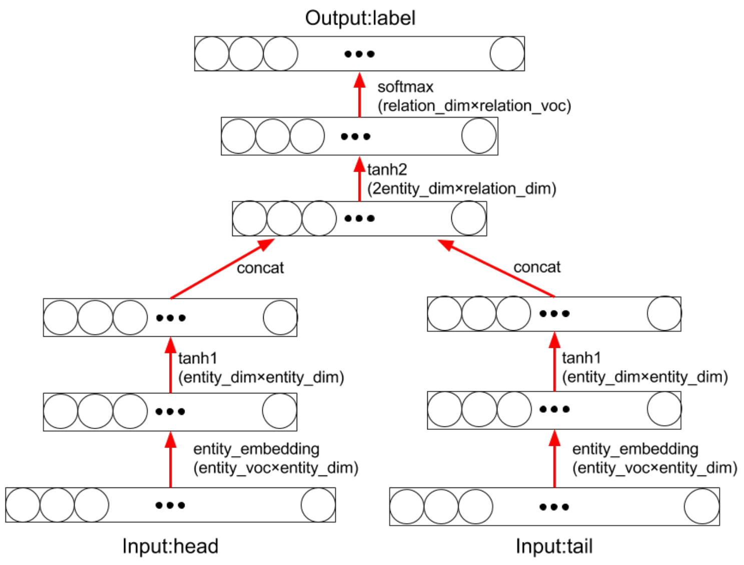

Fig. 1. The structure of RDFDNN

The structure of RDFDNN is shown in Fig.1. Circles are nodes, rectangles are layers of DNN, and red arrows are the transformation of weight matrices. Its inputs are one-hot codes of h and t, and its output is the probability distribution of the one-hot codes of l. One-hot code is a sequence of bits for representing entities. For example, the following vector of length n (whose i-th bit is 1 and others are 0) is the representation of i-th entity among n entities.

$$\begin{pmatrix} 0 \, 0 \ldots 0 \, 1 \, 0 \ldots 0 \end{pmatrix} \tag{1}$$

entity_voc and *relation_voc* are the numbers of entities and relations, respectively. *entity_dim* and *relation_dim* are the dimensions of weight matrices. Embedding is the transformation from RDF entities to their vector representations. The length of the transformed vector is called embedding dimension.

There are five weight matrices in RDFDNN: *entity_embedding*, $tanh1$, *concat*, $tanh2$, and $softmax$. $tanh1$ and $tanh2$ are the weight matrices for activation by tanh function, and $softmax$ is the weight matrix for activation by softmax function. Concat is the composition of the embeddings of h and t. We employ simple concatenation of two embedding vectors for the concat. *entity_embedding* is the weight matrix for transforming an entity to its embedding. L2 regularization is used for the weight matrices of *entity_embedding*, $tanh1$, $tanh2$, and $softmax$ in order to suppress overfitting.

Since the output of RDFDNN is the probability distribution of one-hot representation of relation l, the following cross entropy is used as the objective function for training RDFDNN:

$$E = -\sum_{(h,t,l)\in S} \sum_{k \in relation\_voc} l_k log P(h,t)_k, \tag{2}$$

where S is the set of triples in training data, $P(h,t)$ the output of RDFDNN when h and t are given as its inputs, k is the integer index that satisfies $0 \leq k < relation\_voc$. As the optimizer of the above objective function, Adam [8] is used.

3 Previous methods for predicting relations

3.1 TransE

TransE [1] embeds both entities and relations in the same vector space. Based on vector operations of entities and relations, TransE predicts t from given h and l, and predicts l from h and t. It generates the vector space that satisfies the following equation:

$$d(h+l,t) = 0, \tag{3}$$

where d is the function of Euclidean distance between two given vectors.

TransE obtains a vector representation of entities and relations by minimizing the following objective function L by Gradient descent

$$L = \sum_{(h,l,t)\in S} \sum_{(h',l,t')\in S'} max(\gamma + d(h+l,t) - d(h'+l,t'), 0), \tag{4}$$

where γ is the margin for training, S is the set of triples in the dataset, S' is the set of wrong triples, E is the set of entities, and $S'_{(h,l,t)}$ is defined as follows:

$$S'_{(h,l,t)} = \{(h',l,t) | h' \in E\} \cap \{(h,l,t') | t' \in E\}. \tag{5}$$

3.2 TransR

TransR [10] is the extension of TransE, and it has the ability to learn 1-to-N relations, which is not possible for TransE. 1-to-N relation means that there are more than one ts for a given pair of h and l, such as (John, likes, pizza) and (John, likes, hamburger). In the case of TransE, learning 1-to-N relations is not possible because there is only one vector that satisfies $d(h + l, t) = 0$. In the above example, both pizza and hamburger are represented as the same vector, which is the problem of TransE.

In the case of TransR, h and t are mapped by means of the transformation matrix M_l which is unique to l and then vector operation is performed in order to avoid the above problem. TransR generates a vector space that satisfies the following equation:

$$d(hM_l + l, tM_l) = 0, \tag{6}$$

where M_l is the transformation matrix corresponding to relation l. TransR accepts vector representations of the entities obtained by TransE as its initial values, and it minimizes the following objective function L by Gradient descent in order to obtain vectors and matrices corresponding each relation:

$$L = \sum_{(h,l,t)\in S} \sum_{(h',l,t')\in S'} max(\gamma + d(h_l + l, t_l) - d(h'M_l + l, t'M_l), 0), \tag{7}$$

where $h_l = hM_l$ and $t_l = tM_l$.

4 Evaluation

4.1 Dataset

In our experiments, we have used the FB15k and WN18, which are the samples of the following two datasets. Details of FB15k and WN18 are shown in Table 1. The datasets are the same as the ones used in the experiments of previous research [1][10]. Original datasets of FB15k and WN18 are as follows:

Freebase [2]
a large collaborative online knowledge base
Wordnet [11]
a large lexical database of English

4.2 Criteria for Evaluation

We have implemented RDFDNN using keras and TensorFlow. Keras (https://keras.io) is a Python-based library executable on TensorFlow and Theano. Training of DNN by keras is done using CuDNN library on GPU. We use Python, keras and TensorFlow for the implementation. The CPU used in our experiments is Intel Xeon CPU E5-2609, and GPU is GeForce GTX 1080.

Table 1. Details of FB15k and WN18

	FB15k	WN18
original data	Freebase	Wordnet
number of entities (entity_voc)	14,951	40,943
number of relations (relation_voc)	1,345	18
number of triples for training	483,142	141,442
number of triples for testing	59,071	5,000

We evaluated the results by *top-k accuracy*. After 10 times of training, h and t of a triple in the test data are given to RDFDNN as input, and its output l is evaluated by *top-k accuracy*, whose value is one if the correct answer is included in top-k plausible outputs, and is zero otherwise. This evaluation is done using all test data and results are averaged.

For the comparison of accuracy with previous methods, we set the parameters as $(entity\_dim, relation\_dim) = (30, 30)$ for FB15k and WN18. For failure analysis, we set the parameter as $(entity\_dim, relation\_dim) = (30, 30)$ for FB15k. For the experiments of embedding dimension and accuracy, parameters *entity_dim* and *relation_dim* are set as each of 2, 4, 6, 8, 10 for FB15k. For the experiments of relations between embedded dimension and computational time, *entity_dim* is set to 60, 120, 180, and 240, and *relation_dim* is set to 20, 40, 60, and 80 for FB15k.

4.3 Setting for comparison

We have compared RDFDNN with TransE and TransR implemented by previous approach [10]. For TransE, the learning rate is set to 0.01, γ is set to 1, and embedding dimensions are set to 50 for FB15k, and 100 for WN18, respectively. For TransR, the learning rate is set to 0.001, γ is set to 1, and embedding dimensions are set to 50 for FB15k, and 100 for WN18, respectively.

When h and t are given, TransE computes $d(t-h, l)$ for all possible relations and selects the relation l of its minimum value as its prediction. This is because the vector space satisfying $d(t-h, l) = 0$ is generated in TransE, so the relation l that takes the minimum value of $d(t-h, l)$ for given h and t is expected to constitute a valid triple (h, l, t) rather than other relations.

When h and t are given, TransR computes $d(tM_l - hM_l, l)$ for all possible relations and select the relation l of its minimum value as its prediction. This is because the vector space satisfying $d(tM_l - hM_l, l) = 0$ is generated in TransR, so the relation l that takes the minimum value of $d(tM_l - hM_l, l)$ for given h and t is expected to constitute valid triple (h, l, t) rather than other relations.

5 Comparison with previous methods

For the dataset FB15k, we set parameters as $(entity\_dim, relation\_dim) = (30, 30)$ and compare the accuracy of RDFDNN with previous methods. For the dataset WN18, we set parameters as $(entity\_dim, relation\_dim) = (30, 30)$ and compare the accuracy of RDFDNN with previous methods.

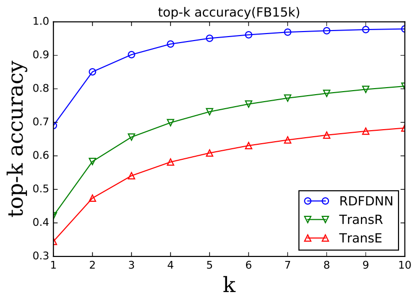

Fig. 2. Top-k accuracy (FB15k)

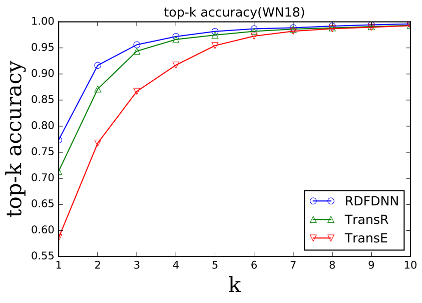

Fig. 3. Top-k accuracy (WN18)

Fig.2 and Fig.3 are the results of comparison with the FB15k and WN18 datasets, respectively. The X-axis is k, and the Y-axis is *top-k accuracy*. For RDFDNN, we set parameters as $(entity\_dim, relation\_dim) = (30, 30)$ because its accuracy is the best when these values are used. As shown in both figures, RDFDNN is more accurate than TransE and TransR in both datasets. The results with FB15k dataset are a clear victory for RDFDNN. In the following discussion, we will discuss the results with FB15k data. The reason for the clear victory is that the number of relations in FB15k (1,345) is much higher than that in WN18 (18). Prediction of relations is harder when the number of relations is much higher.

6 Failure analysis

Failure analysis is done in the experiments with FB15k when parameters are set as $(entity\_dim, relation\_dim) = (30, 30)$. RDFDNN's failed predictions can be classified to the following four categories:

- A: deceived by majority cases
- B: too abstract / too concrete
- C: structurally similar
- D: complete failure

For this failure analysis, parameters are set as $(entity\_dim, relation\_dim) = (30, 30)$ and 100 triples of RDFDNN failures are randomly sampled. Then the triples are manually evaluated and classified into the above four categories in order to obtain the results in Table 2.

As shown in Table 2, the most frequent failure is type A. As an example of type A, RDFDNN's prediction of relation between "Leslie Dilley" and "Raiders

Table 2. Types and the number of failed predictions

type	number of failures
A	49
B	14
C	5
D	32

of the Lost Ark" is "performer", while its correct answer is "art director". This is because the relation "performer" is the most frequent one for the relation between people and movies. The second most frequent failure is type D, complete failure. One of the examples is the prediction of the relation between "Iron Man" and "Stan Lee". The correct answer should be "creator", but the prediction by RDFDNN was "cause of death". The third most frequent failure is type B, too abstract or too concrete compared with correct answers. As an example of this type, RDFDNN predicts the relation between "Park Chu-yong" and "South Korea" as "citizenship", while its correct answer is "Olympic representative". The least frequent failure is type C, but this failure means that RDFDNN recognizes structural similarity between relations. As an example of this type, RDFDNN predicts the relation between "Washington Wizards" and "Michael Jordan" as "belonging states", while its correct answer is "team member".

7 Embedding dimension and prediction accuracy

For the dataset FB15k, we set parameters *entity_dim* and *relation_dim* as each of 2, 4, 6, 8, 10, and observed the *top-k accuracy* of RDFDNN.

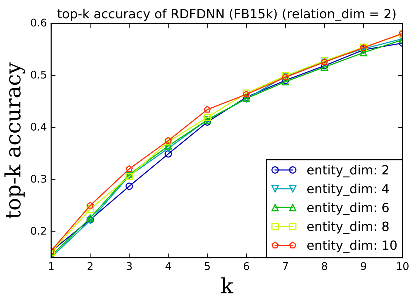

Fig. 4. top-k accuracy of RDFDNN (FB15k) (relation_dim=2)

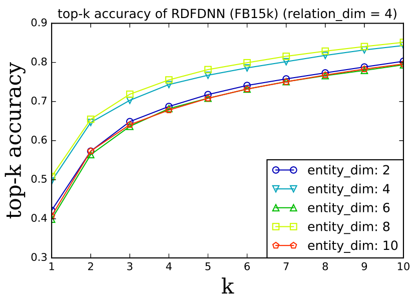

Fig. 5. top-k accuracy of RDFDNN (FB15k) (relation_dim=4)

Fig.4, Fig.5, Fig.6, Fig.7, and Fig.8 are the results when *entity_dim* and *relation_dim* are set to each of 2, 4, 6, 8, 10, respectively. The X-axis is k, and

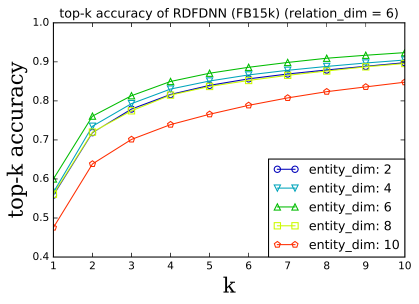

Fig. 6. top-k accuracy of RDFDNN (FB15k) (relation_dim=6)

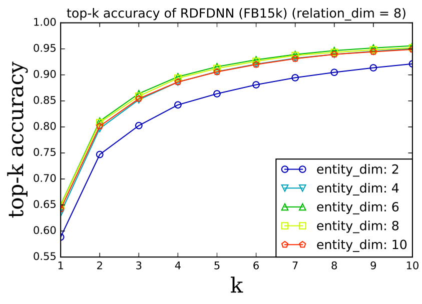

Fig. 7. top-k accuracy of RDFDNN (FB15k) (relation_dim=8)

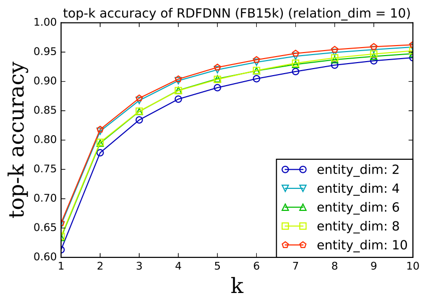

Fig. 8. top-k accuracy of RDFDNN (FB15k) (relation_dim=10)

the Y-axis is *top-k accuracy*. Results of the same *relation_dim* with five different values of *entity_dim* are drawn in each figure. The best accuracy is obtained in Fig.8 when parameters are set as $(entity\_dim, relation\_dim) = (10, 10)$, and its *top*-10 *accuracy* is 0.963. The best result in Sect. 5 is *top*-10 *accuracy* $= 0.979$ when parameters are set as $(entity\_dim, relation\_dim) = (30, 30)$, which is comparable to the above result. Except Fig. 4 ($relation\_dim = 2$), the results are the best when $relation\_dim = entity\_dim$ among all parameter settings. When $relation\_dim > entity\_dim$, the dimension of the weight matrix of *entity_embedding* is too small, and when $relation\_dim < entity\_dim$, the dimension of the weight matrix of *entity_embedding* is too big, which causes overfitting. Fig. 4 ($relation\_dim = 2$), the dimensions of weight matrices of *activation_2* and $softmax$ are too small so overfitting is avoided. Therefore, more embedding dimensions of weight matrices of *entity_embedding* and *entity_dim* are more accurate.

Fig.9 aggregates all the results of Fig.4, Fig.5, Fig.6, Fig.7, and Fig.8. The X-axis is k, and the Y-axis is *top-k accuracy*. Results with the same *relation_dim* are drawn with the same color, so five curves are drawn in each color. The difference of *top-k accuracy* with different *entity_dim* is 0.1 at most, which are

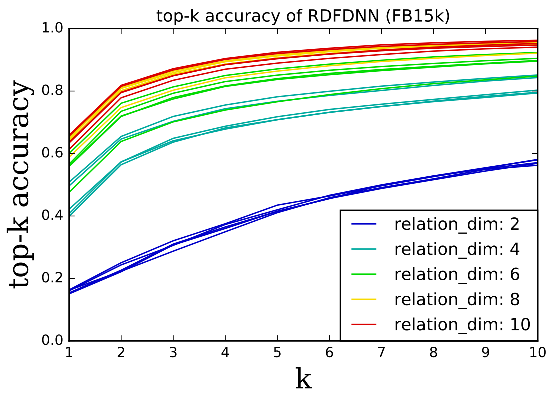

Fig. 9. top-k accuracy of RDFDNN (FB15k)

quite small compared with the difference with different *relation_dim* values. In Fig.9, curves of the same color (results with the same *relation_dim* value) are almost the same. *relation_dim* is more important for the accuracy of RDFDNN than *entity_dim*.

8 Embedding dimension and computational time

For the dataset FB15k, we set *entity_dim* to 60, 120, 180, and 240, and *relation_dim* as 20, 40, 60, 80, and observed the computational time of RDFDNN.

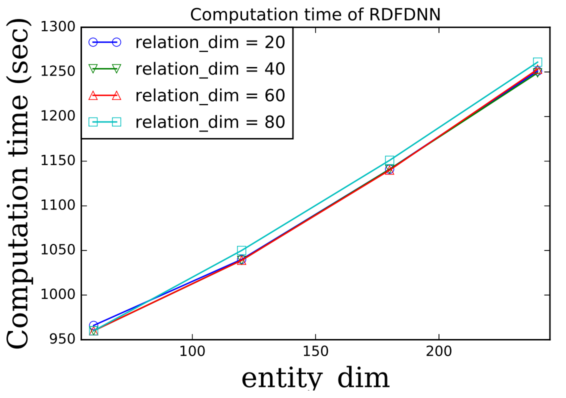

Fig. 10. Computational time of RDFDNN for different embedding dimensions

Fig.10 shows the computational times of RDFDNN for different embedding dimensions. The X-axis is *entity_dim*, and the Y-axis is the computational time (seconds). As shown in the figure, the value of *relation_dim* is not relevant to the computational time of RDFDNN. Only the value of *entity_dim* is relevant to the computational time. As shown in Table 1, *entity_voc* >> *relation_voc* is true in our experiments, which is the reason that *entity_dim* is relevant to the computational time of RDFDNN.

9 Discussion

As shown in Sect. 5, RDFDNN is more accurate than previous methods for the prediction of relations between two given entities. The difference of accuracy with FB15k is bigger than that with WN18, so we can conclude that prediction is harder when there are more possible relations.

The accuracy of RDFDNN is fairly good even when the embedding dimension is only one-fifth of the embedding dimension used in previous methods. In the case of TransR, the embedding dimension is set to 50 and its *top-k accuracy* is 0.808. On the other hand, *top-k accuracy* of RDFDNN is 0.963 even when $(entity\_dim, relation\_dim) = (10, 10)$. This means that the accuracy of RDFDNN is better than TransR even when its embedding dimension is only one-fifth of that of TransR.

As shown in Sect. 6, RDFDNN is more accurate when *relation_dim* is bigger, while the value of *entity_dim* is irrelevant to its accuracy. If *entity_dim* is bigger than *relation_dim*, accuracy is not good because of overfitting. If *entity_dim* is smaller than *relation_dim*, accuracy is not good because the dimension of the DNN weight matrix is not enough. Therefore, the values of *relation_dim* and *entity_dim* should be the same as initial setting.

As shown in Sect. 7, computational time of RDFDNN depends on *entity_dim*, not *relation_dim*, which is contrastive to the result in Sect. 6 that the accuracy depends on *relation_dim*. Therefore, if we keep *entity_dim* small and make *relation_dim* bigger, better accuracy with less computational time will be achieved. However, in this case, *entity_dim* is smaller compared with *relation_dim*, which causes less accuracy because the dimension of DNN weight matrix is not enough.

To summarize, RDFDNN achieves high accuracy with less embedding dimension with the following procedure. The reason for setting parameters as halves of previous values is that the range of possible embedding dimensions is fairly wide, so repeated bipartitioning will be desirable for finding better parameters with less trials.

1. Set the parameters *entity_dim* and *relation_dim* as those of previous methods as initial setting.
2. Set them as halves of previous values, keeping $entity\_dim = relation\_dim$.
3. Fix *relation_dim* and set *entity_dim* as the half of previous value.

10 Conclusion

In this paper, we propose RDFDNN for predicting relations of RDF from two given entities. RDFDNN is more accurate compared with previous methods (TransE and TransR). The following are the characteristics of RDFDNN.

- Bigger *relation_dim* for better accuracy
- Smaller *entity_dim* for less computational time
- Even when RDFDNN failed, more than half of its failed prediction are valid in some sense

In addition to TransE and TransR, there are some other related approaches [14][5][15]. The weaknesses of RDFDNN are as follows: (1) prediction of t from given h and l is not easy for RDFDNN, while it is possible for TransE and TransR. This is because RDFDNN cannot obtain embedded representation of relations, and (2) RDFDNN always predicts relations even when two given entities are completely irrelevant. These are left for our future work.

Acknowledgement

This work was supported by Tokyo Tech - Fuji Xerox Cooperative Research (Project Code KY260195), JSPS Grant-in-Aid for Scientific Research(B) (Grant Number 17H01785) and JST CREST (Grant Number JPMJCR1687).

References

1. Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, Oksana Yakhnenko, "Translating Embeddings for Modeling Multi-relational Data", Part of Advances in Neural Information Processing Systems 26, pp.2787-2795. (2013)
2. Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, Jamie Taylor, "Freebase: a collaboratively created graph database for structuring human knowledge", Proceedings of the 2008 ACM SIGMOD international conference on Management of data (SIGMOD '08), pp.1247-1250. (2008)
3. Google, "Freebase Data Dumps", https://developers.google.com/freebase/data.
4. Google, "Google Knowledge Graph Search API - Google Developers", https://developers.google.com/knowledge-graph/.
5. Shu Guo, Boyang Ding, Quan Wang, Lihong Wang, Bin Wang, "Knowledge Base Completion via Rule-Enhanced Relational Learning", China Conference on Knowledge Graph and Semantic Computing (CCKS 2016), pp.219-227. (2016)
6. Geoffrey E. Hinton, Simon Osindero, Yee-Whye Teh, "A fast learning algorithm for deep belief nets" Neural Computation, Volume 18, Issue 7, pp.1527-1554. (2006)
7. Martin Kavalec, Vojtech Svatek, "A Study on Automated Relation Labelling in Ontology Learning", Ontology Learning from Text: Methods, Evaluation and Applications, IOS Press. (2003)
8. Diederik Kingma, Jimmy Ba, "Adam: A Method for Stochastic Optimization", International Conference for Learning Representations, Published as a conference paper at the 3rd International Conference for Learning Representations, San Diego (2015)

9. Ora Lassila, Ralph R. Swick, “Resource Description Framework(RDF) Model and Syntax Specification”, https://www.w3.org/TR/1999/REC-rdf-syntax-19990222/.
10. Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, Xuan Zhu, “Learning Entity and Relation Embeddings for Knowledge Graph Completion”, Twenty-Ninth AAAI Conference on Artificial Intelligence, pp.2181-2187. (2015)
11. George A. Miller, “WordNet: A Lexical Database for English”, Communications of the ACM, Volume 38, Number 11, pp.39-41. (1995)
12. Maximilian Nickel, Volker Tresp, Hans-Peter Kriegel, “A Three-Way model for collective learning on Multi-Relational data”, In Proceedings of the 28th International Conference on Machine Learning, ICML 2011, pp.809-816, Bellevue, WA, USA. (2011)
13. Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg and Li Fei-Fei., “ImageNet Large Scale Visual Recognition Challenge”, International Journal of Computer Vision, Volume 115, Issue 3, pp.211-252. (2015)
14. Quan Wang, Bin Wang, Li Guo, “Knowledge Base Completion Using Embeddings and Rules”, Proceedings of the 24th International Conference on Artificial Intelligence (IJCAI’15), pp.1859-1865. (2015)
15. Quan Wang, Jing Liu, Yuanfei Luo, Bin Wang, Chin-Yew Lin, “Knowledge Base Completion via Coupled Path Ranking”, Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL), pp.1308-1318. (2016).
16. Albert Weichselbraun, Gerhard Wohlgenannt, Arno Scharl, “Refining non-taxonomic relation labels with external structured data to support ontology learning”, Data and Knowledge Engineering, Volume 69, Issue 8, pp.763-778. (2010)