Link Prediction based on Tensor Decomposition for the Knowledge Graph of COVID-19 Antiviral Drug

Ting Jia[1], Yuxia Yang[1], Kuo Yang[1,2*], Xuezhong Zhou[1*]

[1] Institute of Medical Intelligence, School of Computer and Information Technology, Beijing Jiaotong University, Beijing 100044, China
[2] BNRIST/Department of Automation, Tsinghua University, Beijing 10084, China
19120365@bjtu.edu.cn

Abstract. Due to the large-scale spread of COVID-19, which has an important impact on human health and social economy, designing effective antiviral drugs for COVID-19 is a key to save lives. Existing relationship knowledge, e.g., drug-virus and viral protein-host protein, can be used for building a reliable knowledge graph. Based on this, large-scale graph and knowledge reasoning algorithms can be used to predict new targeting effects between antiviral drugs and viruses. China Conference on Knowledge Graph and Semantic Computing (CCKS) 2020 set up a link prediction task for the knowledge graph of COVID-19 antiviral drug, requiring prediction of the relationship between two entities in the graph. In this study, we proposed a fusion strategy to integrate the results of two tensor decomposition-based models (i.e., CP-N3 and ComplEx-N3). Sufficient experiments indicated that our method obtained high performance (MRR=0.2328), which ranks third in the fierce competition of CCKS 2020 Challenge Task 1.

Keywords: Link Prediction, Knowledge Graph, COVID-19, antiviral drug prediction

1 Introduction

Knowledge Graph (KG) is a structured representation of real-world information. In KGs, nodes represent entities, such as people and places; edges are specific facts that connect two entities; labels are the types of edges [1,2]. As a basis for upper-level applications, KG plays an extremely important role in the intelligent era. However, it is well known that even state-of-the-art KGs suffer from incompleteness [3,1,4], even the most advanced KGs, such as FreeBase [5], WikiData [6], DBPedia [7] and – in industry – Google KG [8]. Link prediction (LP), which exploits the existing facts in a KG to infer missing ones, is one of the most prospective ways to solve this problem [8].

At the time of the large-scale spread of COVID-19 [9], China Conference on Knowledge Graph and Semantic Computing (CCKS) 2020 set up a link prediction task for the knowledge graph of COVID-19 antiviral drug (ADKG), requiring prediction of the relationship between two entities in the graph. It has great significance and academic value for the research and development of antiviral drugs.

At present, in terms of link prediction tasks, methods that can be considered include the similarity of node attribute, network structure and likelihood analysis, but more and more effective methods are based on machine learning methods. In these methods, some can be based on classification methods, such as decision trees [10], support vector machines [11], multi-layer perceptions [12], etc. and some be based on probability graph models and matrix decomposition methods [13-16]. In recent years, tensor decomposition has attracted much attention [14]. The Canonical Tensor Decomposition (CP) decomposes a high-order tensor into several one-dimensional factor matrices [13,17]. For example, a third-order tensor can be decomposed into three one-dimensional factors. It can be understood as a low-rank approximation or feature extraction of high-dimensional data. ComplEx [14], a complex number-based representation method can grasp the symmetric and asymmetric relationships in the binary relationship of the knowledge base.

In this study, we participated in the CCKS 2020 link prediction task and proposed a fusion strategy, which effectively integrates the results obtained by two tensor decomposition-based models (i.e., CP-N3 and ComplEx-N3). Sufficient experiments indicated that our method obtained high performance (MRR=0.2328), which ranks third in the fierce competition of CCKS 2020 Challenge Task 1.

2 Related work

Representation learning based on tensor decomposition treats each triple as an element in a tensor and performs representation learning through a tensor decomposition algorithm. Tensor decomposition is a combination of low-dimensional vectors, which are used as embeddings of entities and relationships. The core idea is that the model is not excessively applied to the training set, the learned embeddings should be able to generalize and associate high values with real facts that are not visible in the graph adjacency matrix. As the first typical representative algorithm that uses tensor decomposition for the completion of knowledge graphs, RESCAL [18] uses a third-order tensor to represent entities and relationships. However, due to the large number of parameters in this method, the risk of overfitting is more likely to occur, so the DistMult [15] algorithm appears. The method proposed that each relationship should have a corresponding diagonal matrix. The third-order binary tensor trained by this algorithm is symmetric in the head and tail entity embeddings, so it cannot be modeled a symmetric relationship.

ComplEX [16] extends DistMult [15] in the complex number domain, uses vector dot multiplication to calculate the triplet score, uses probability to determine the correctness of the triplet, and can also model asymmetric relationships. Among the linear methods, first-order logic rules can be used to learn the relationships in the KG. CP regards the link prediction as a 3rd-order binary tensor completion problem, which embeds the head and tail entities of the unified entity independently to each other.

3 Methods

3.1 Data preprocessing

First, we used regular expressions to match the given officially data files to get the data with triple format we need. Second, we added the list of entity attributes to our knowledge graph to get more attribute information of the entity. Finally, to expand more relationships from the knowledge graph, we used a well-known data augmentation technique [13,19], which added reciprocal relations for every triple, i.e. adding (t; r^{-1}; h) for every (h; r; t), where h is the head entity, r is the relationship and t is the tail entity.

3.2 CP-N3

The canonical decomposition of tensors, also called CANDECOM/PARAFAC (CP) [20], represents a tensor $X \in R^{N1\times N2\times N3}$ as a sum of R rank one tensors $u_r^{(1)} \otimes u_r^{(2)} \otimes u_r^{(3)}$ ($\otimes$ denotes tensor product) where $r \in \{1, \dots, R\}$ and $u_r^{(m)} \in R^{N_m}$ [13]:

$$\boldsymbol{X} = \sum_{r=1}^{R} u_r^{(1)} \otimes u_r^{(2)} \otimes u_r^{(3)}$$

Current CP on the standard benchmarks of Knowledge Base Completion are behind their competitors. Timothee et al. [13] proposed a novel regularizer, based on tensor nuclear *p*-norms, and then presented a reformulation of the problem, which make it beat the current state of the art on several datasets with a CP decomposition (named CP-N3).

3.3 ComplEx-N3

Similar to DistMult [15], ComplEx [16] forces each relation embedding to be a diagonal matrix thus yields a tensor that is symmetric in the object and subject modes, but extends such formulation in the complex space: $h \in C^d, t \in C^d, r \in C^{d\times d}$. In the complex space, the bilinear product becomes a Hermitian product, where in lieu of the traditional $\boldsymbol{t}$, its conjugate-transpose $\bar{t}$ is employed. This allows ComplEx to successfully model asymmetric relations as well. Timothee et al. [13] proposed a ComplEx-based model (named ComplEx-N3) with the novel regularizer and systematic parameter searches, which obtained better performance than ComplEx.

3.4 A fusion strategy of predictive results

Different from the traditional machine learning method with one learner to train, the integrated learning methods usually trains multiple learners and combines them to obtain better results. These methods have been applied to a variety of applications, such as computer vision [21], species distribution prediction [22] and auxiliary diagnosis [23].

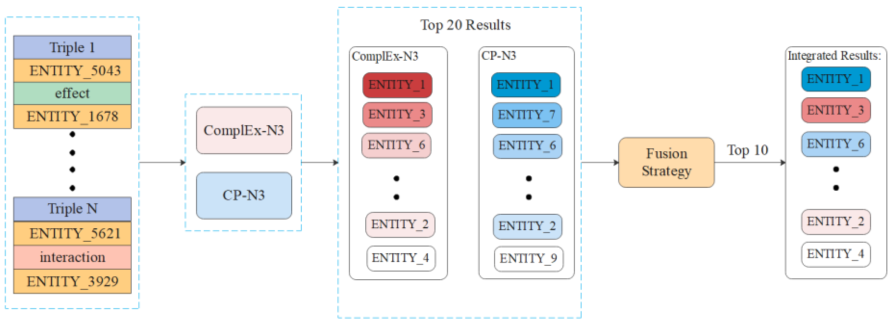

Fig. 2. The framework of a fusion strategy. First, the same knowledge graph is used to train the CP-N3 and ComplEx-N3 models, respectively. Then the two models are used to predict the prediction results separately to obtain top 20 prediction results. Finally, the two top 20 results are weighted integrated by voting to obtain the final top 10 predictive results.

Inspired by the idea of integrated learning, we designed a fusion strategy for this task (Fig. 2). First, we use the same knowledge graph to train the CP-N3 and ComplEx-N3 models, respectively. Second, we use the two models to predict the results separately to obtain the top 20 predictive results. Finally, the two top 20 results are weighted integrated by voting to obtain the final top 10 predictive results. When the results are integrated, we restrict the entity based on the entity type, and eliminate the entities that are not of this type.

4 Experiments

4.1 Datasets

We use the ADKG of CCKS 2020 to evaluate our approach. The dataset is published by the CCKS 2020 task 1, which includes three types of entities (i.e., proteins, viruses, and drugs) and four relationship types (i.e., effect, interaction, produce and binding). The knowledge graph includes 7,844 entities, 40,000 triples, and the first batch of predictive data has 4,256 triples and the second batch of predictive data has 5,000 triples.

In experiments, to evaluate the performance of models, we randomly selected 10% of each relationship type from the training set as the test set, a total of 4000 triples. In order to get more attribute information of the entity, we add the list of entity attributes to our training set and add the involved entities to the entity list simultaneously. In total, the knowledge graph has 8,494 entities and 51,131 triples, of which 47,131 triples are in the training set and 4,000 triples are in the test set.

4.2 Experimental setup

We tuned all hyper-parameters based on predictive performance to find the best parameter settings on ADKG. Table 1 showed the best parameter settings of the two methods. In experiments, we used the well-tuned parameters to obtain the top 20 results of the two methods respectively, and integrated the two results according to our fusion strategy to obtain the final results.

Table 1. Best parameter settings of CP-N3 and ComplEx-N3 on ADKG.

Hyper-parameters	CP-N3	ComplEx-N3
Dimension	2000	2500
Learning rate	0.06	0.07
Batch size	256	128
Regularization coefficient	0.01	0.005
Initialization	0.001	0.001
Optimizer	Adagrad	Adagrad

5 Results

5.1 Comparison results with baselines

For each test triple we generate 8,494 candidate triples by combining the test entity-relation pair with all possible entities E, ranking the scores obtained. We used the filtered setting [24], i.e. all known true triples are removed from the candidate set except for the current test triple. Since the top 10 candidate entities were finally submitted, we still cared about the prediction results after the top 10 to integrate the results. Therefore, we used mean reciprocal rank of top 10 (MRR_10) and hits@k, $k \in \{10; 20; 50; 100\}$ to evaluate the performance. Mean reciprocal rank of top 10 is the average of the inverse of the mean rank assigned to the true triple over the top 10 candidate triples. Hits@k measures the percentage of times a true triple is ranked within the top k candidate triples. Comparison results of link prediction on ADKG were shown in Table 2.

In addition to the two given baselines (i.e., TransE [24] and R-GCN [25]), we also showed the performance of other six classic methods, i.e., DistMult [15], ComplEx [16], TuckER [14], HypER [26], CrossE [27] and ConvE [19]. Results showed that CP-N3 (MRR_10=0.185) and ComplEx-N3 (MRR_10=0.183) not only achieved better than some linear models (e.g., DistMult, ComplEx and TuckER), but also better than some deep neural network, e.g. RGCN, HypER, CrossE and ConvE. The performance of CP-N3 and ComplEx-N3 far exceeds other methods, therefore, we chose the top 20 results of these two models to integrate the results on the prediction set. In the fierce competition of CCKS 2020 Challenge Task 1, it got 19.785 (ranking first) in the first batch of predictive data and got 23.275 (ranking third) in the second batch of data.

Table 2. Comparison results of link prediction methods.

Model	Hits@10	Hits@20	Hits@50	Hits@100	MRR_10
TransE	0.17	0.24	0.34	0.42	0.076
R-GCN	0.20	0.26	0.38	0.49	0.079
DistMult	0.301	0.384	0.506	0.596	0.140
ComplEx	0.308	0.391	0.504	0.588	0.145
TuckER	0.337	0.428	0.545	0.640	0.166
HypER	0.328	0.418	0.546	0.638	0.161
CrossE	0.251	0.332	0.440	0.531	-
ConvE	0.311	0.402	0.524	0.615	0.150
ComplEx-N3	**0.375**	**0.462**	**0.589**	**0.674**	**0.183**
CP-N3	**0.375**	**0.467**	**0.589**	**0.676**	**0.185**
Integrated					

5.2 Parameter sensitivity

To investigate the influence of hyper-parameters of CP-N3 and ComplEx-N3, we tuned all related hyper-parameters of them to find best parameter settings on ADKG. We selected MRR and MRR of top 10 as evaluation metrics. There are four parameters that need to be tuned. The first parameter is the entity and relation *dimension* of the embedding vectors learned in the model, tuning from {500, 1000, 1500, 2000, 2500}. The second parameter is *learning rate*, which determines the rate of decrease of parameters during optimization, tuning from {0.05, 0.06, 0.07, 0.08, 0.09, 1.0}. The third parameter is *batch size*, indicating the sample size for training at the same time, tuning from {128, 256, 512, 1024}. The final parameter is *regularization coefficient*, tuning from {0.005, 0.01, 0.05, 0.1}. In turn, we fixed each possible value of the three parameters, while tuning the remaining parameter.

Experimental results (Fig. 3A-D for CP-N3 and Fig. 3E-H for ComplEx-N3) show that in CP-N3, when the *batch size* is too small (128) or too large (512 or 1024), it showed poor performance, but good performance is obtained when the level is equal to 256. The *dimension* between 1000 and 2500 does not have much influence on the model, and the small-scale adjustment of *learning rate* has little influence. In terms of predictive performance, it shows that the model has strong robustness in predicting candidate entities. Similarly, in ComplEx-N3, the best performance is achieved when the *batch size* is 128, the *dimension* is 2500, the *regularization coefficient* in {0.005, 0.01, 0.1} and the *learning rate* in {0.07, 0.09}.

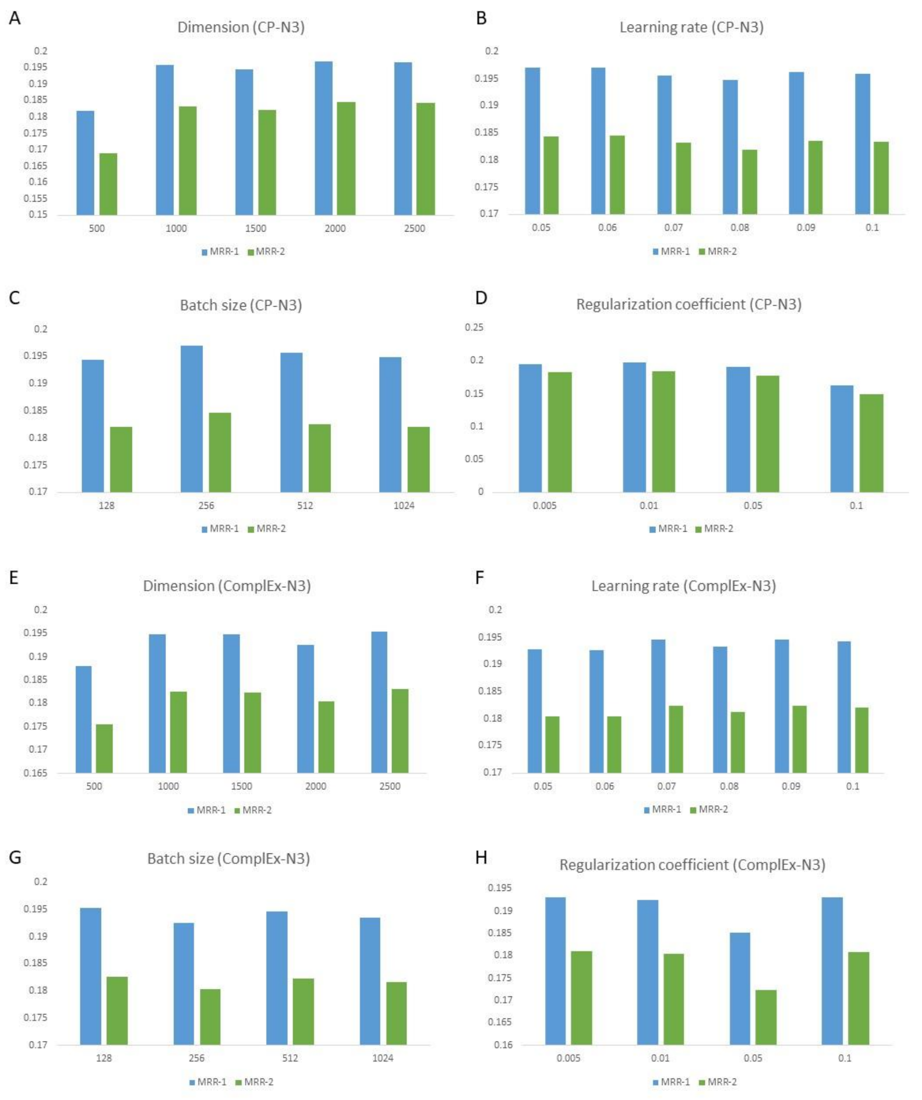

Fig. 3. The prediction performance with tuning the hyper-parameters. The first parameter is the entity and relation *dimension* of the embedding vectors, tuning from {500, 1000, 1500, 2000, 2500}. The second parameter is *learning rate*, tuning from {0.05, 0.06, 0.07, 0.08, 0.09, 1.0}; The third parameter is *batch size*, tuning from {128, 256, 512, 1024}; The final parameter is *regularization coefficient*, tuning from {0.005, 0.01, 0.05, 0.1}. A-D sub figures are for CP-N3 and E-H sub figures are for ComplEx-N3. MRR-1 denotes MRR and MRR-2 denotes MRR of top 10.

5.3 Case study

To illustrate the predictive performance of our model, we took the two triples (?, interaction, ENTITY_6303) and (ENTITY_3946, binding, ?) in the first batch of prediction data as an example and obtained the top 10 candidate entities (Table 3).

In the top 10 candidate entities for (?, interaction, ENTITY_6303), we found that there are five entities, namely ENTITY_7358, ENTITY_5261, ENTITY_6303, ENTITY_3101, ENTITY_2000 and ENTITY_1379 are actually recorded in the benchmarks. In the top 10 candidate entities for (ENTITY_3946, binding, ?), there are four entities (i.e., ENTITY_761, ENTITY_2386, ENTITY_6309 and ENTITY_3333) that are recorded in benchmarks. This demonstrated that an reliable integrated results are obtained by our model.

Table 3. Top 10 candidate entities for two predictive triples. (Bold entities are the hitting entities in benchmarks.)

(**?**, interaction, ENTITY_6303)			(ENTITY_3946, binding, **?**)		
Rank	Candidate entities	Hit	Rank	Candidate entities	Hit
1	**ENTITY_7358**	√	1	ENTITY_2675	
2	**ENTITY_5261**	√	2	**ENTITY_761**	√
3	**ENTITY_6303**	√	3	ENTITY_6527	
4	ENTITY_3101		4	ENTITY_3416	
5	**ENTITY_2000**	√	5	**ENTITY_2386**	√
6	**ENTITY_1379**	√	6	ENTITY_425	
7	ENTITY_4654		7	**ENTITY_6309**	√
8	ENTITY_7365		8	**ENTITY_3333**	√
9	ENTITY_5750		9	ENTITY_1835	
10	ENTITY_2459		10	ENTITY_224	

6 Discussion

In this study, we provided a high-performance model of link prediction on knowledge graph. Of note, we proposed a fusion strategy to integrate the results of two tensor decomposition-based models to obtain a more reliable predictive result. We added the list of entity attributes to our knowledge graph to get more attribute information of the entity and added reciprocal relations [19,13] for every triple to expand the knowledge graph. Full experiments indicated that the proposed method obtained obviously performance improvements. In addition, there are still challenges in the link prediction task, including the need to design a knowledge graph embedding algorithm that integrates more ontology features, and analyze the equivalence between knowledge graph embedding and ontology reasoning. Future research directions might include: (i) Combining logical rules which contain rich background information and KG triples in a unified KG completion framework [28,29]. (ii) Consider the path relationship between triples in the knowledge graph [30]. In the following work, we will continue to try new models and find the limitations of existing models to improve the performance.

References

1. Rossi, A., Firmani, D., Matinata, A., Merialdo, P., Barbosa, D.: Knowledge Graph Embedding for Link Prediction: A Comparative Analysis, (2020).
2. Wang, Q., Mao, Z., Wang, B., Guo, L.: Knowledge Graph Embedding: A Survey of Approaches and Applications. IEEE T KNOWL DATA EN, **29; 29**(12; 12), 2724-2743 (2017). doi: 10.1109/TKDE.2017.2754499
3. Lin, Y., Liu, Z., Sun, M., Liu, Y., Zhu, X.: Learning Entity and Relation Embeddings for Knowledge Graph Completion, AAAI, 2181-2187 (2015).
4. Nathani, D., Chauhan, J., Sharma, C., Kaul, M.: Learning Attention-based Embeddings for Relation Prediction in Knowledge Graphs. In Korhonen, A., Traum, D., Marquez, L. (eds.),4710-4723 (2019).
5. Bollacker, K.D., Evans, C., Paritosh, P., Sturge, T., Taylor, J.: Freebase: a collaboratively created graph database for structuring human knowledge. Sigmod Conference. ACM, (2008).
6. Vrandecic, D., Kroetzsch, M.: Wikidata: A Free Collaborative Knowledgebase. COMMUN ACM, **57**(10), 78-85 (2014). doi: 10.1145/2629489
7. Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R., Ives, Z.: DBpedia: A nucleus for a web of open data. In Aberer, K., Choi, K.S., Noy, N., Allemang, D., Lee, K.I., Nixon, L., Golbeck, J., Mika, P., Maynard, D., Mizoguchi, R., Schreiber, G., CudreMauroux, P. (eds.), Lecture Notes in Computer Science , Vol. 4825, 722. (2007).
8. Singhal, A.: Official Google Blog: Introducing the Knowledge Graph: things, not strings. Official Google Blog, (2012).
9. Jernigan, D.B.: Update: Public Health Response to the Coronavirus Disease 2019 Outbreak - United States, February 24, 2020. MMWR-MORBID MORTAL W, **69**(8), 216-219 (2020).
10. Quinlan, JR.: C4.5: Programms for Machine Learning. C4.5: programs for machine learning. Morgan Kaufmann Publishers Inc, (1995).
11. Asahara M., Matsumoto Y.: Japanese Named Entity extraction with redundant morphological analysis. ACL, (2003).
12. Rumelhart, D.E., Hinton, G.E., Williams, R.J.: Learning representations by back-propagating errors. Nature, 323(6088), 533-536 (1986).
13. Lacroix, Timothée, Usunier, N., Obozinski, G.: Canonical tensor decomposition for knowledge base completion, (2018).
14. Balaevi, I., Allen, C., Hospedales, T.M.: TuckER: Tensor Factorization for Knowledge Graph Completion, (2019).
15. Yang, B., Yih, W.T., He, X., Gao, J., Deng, L.: Embedding entities and relations for learning and inference in knowledge bases. ICLR, (2014).
16. Trouillon, Théo, Welbl J., Riedel S.: Complex Embeddings for Simple Link Prediction[J]. 2016. .: Complex embeddings for simple link prediction. ICML, (2016).
17. Hitchcock, F.L.: The expression of a tensor or a polyadic as a sum of products.

STUD APPL MATH, **6**(1-4), 164-189 (1927).

18. Nickel, M., Tresp, V., Kriegel, H.P.: A Three-Way Model for Collective Learning on Multi-Relational Data. International Conference on International Conference on Machine Learning, (2011).
19. Dettmers, T., Minervini, P., Stenetorp, P., Riedel, S.: Convolutional 2D Knowledge Graph Embeddings, 1811-1818 (2018).
20. Hitchcock, F.L.: The expression of a tensor or a polyadic as a sum of products. STUD APPL MATH, 164-189 (1927).
21. Avidan, S.: Ensemble Tracking. IEEE PAMI, **29**(2), 261-271 (2007).
22. Miguel B. Araújo, New, M.: Ensemble forecasting of species distributions. Trends in Ecology & Evolution, 22(1), 42-47 (2007).
23. Zhou X .: Predicting distant metastasis in breast cancer using ensemble classifier based on context-specific miRNA regulation modules. IEEE International Conference on Bioinformatics & Biomedicine. IEEE, (2012).
24. A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, and O. Yakhnenko.: Translating embeddings for modeling multi-relational data. In Advances in neural information processing systems. NIPS, 2787–2795 (2013).
25. Schlichtkrull, M., Kipf, T. N., Bloem, P., Berg, R. V., Welling, M .: Modeling Relational Data with Graph Convolutional Networks. In European Semantic Web Conference, (2018).
26. Balazevic, I., Allen, C., Hospedales, T.M.: Hypernetwork Knowledge Graph Embeddings. In Tetko, I.V., Kurkova, V., Karpov, P., Theis, F. (eds.), Lecture Notes in Computer Science , Vol. 11731, 553-565. (2019)
27. Zhang, W., Paudel, B., Zhang, W., Bernstein, A., Chen, H.: Interaction Embeddings for Prediction and Explanation in Knowledge Graphs96-104. (2019)
28. Liu, H., Wu, Y., Yang, Y.: Analogical Inference for Multi-relational Embeddings. ICML, 2168-2178 (2017)
29. Shu, G., Quan, W., Lihong, W., Bin, W., Li, G.: Jointly embedding knowledge graphs and logical rules. EMNLP, 192-202 (2016).
30. Meng, Q., Junkun, C., Louis-Pascal, X., Yoshua. B., Jian, T.: RNNLogic: Learning Logic Rules for Reasoning on Knowledge Graphs, (2020).