Industrial Engineering Journal
ISSN: 0970-2555
Volume : 50, Issue 4, No. 1, April 2021

Action space optimization inspired a new way to deal with dimension problems in big data

[1]D. Lakshmi Narayana Reddy

[1]Assistant Professor Anantha Lakshmi Institute of Technology & Sciences , Anantapuramu, A. P, India.

Department of Computer Science and Engineering

Mail id: lakshmi1217@gmail.com.

Abstract.

This study describes a new method called Action Space Optimization and Joint Reward (AOJR) that helps solve the problems of the curse of dimensionality and reward sparsity in reinforcement learning knowledge graph reasoning. The writers suggest an action space optimization method based on TuckER embedding to deal with the problem of the large action space. This optimization method narrows down the actions that can be searched for and makes it easier for agents to choose the best actions. The process has three steps: relation grouping, relation cluster selection, and relation selection. These steps work together to make the agent better at making decisions. To fix the problem of not having enough rewards, the writers come up with a joint reward system with three types of awards: a basic reward, a single-step reward, and a path reward. When these benefits are paired with the right weights, they give the robot better and more correct information as it learns. In terms of reasoning correctness and path discovery, the testing data show that the proposed AOJR method works better than the standard models. This method is a big step forward in knowledge graph reasoning using reinforcement learning techniques because it solves the problems of complexity and reward shortage.

Keywords: Knowledge graph; knowledge reasoning; reinforcement learning; policy network

1. Introduction

The exponential rise of data has made it critical to find efficient ways to harvest information from large datasets. As a center of interest in the study of AI, the Knowledge Graph[1] (KG) is a data structure for storing and retrieving information about ideas and the connections between things in the real world.

KG has a lot of information, however it has issues like outdated material and missing pieces since it is always being updated. Freebase[2], a popular KG, has billions of information, however it is still not full. Among the three million people entities, 71 percent had no birthplace information. Sixty percent of the person entities in DBpedia[3] are missing their country of origin. KG has to be finished because of these issues, which restrict its usefulness. Knowledge reasoning[4] is the foundational technology of

knowledge graph completion; it may infer new triples from current data or discover erroneous triples and rectify them.

Single-hop reasoning based on triples and multi-hop reasoning based on relational routes are two broad categories into which knowledge reasoning may be broken down. Multi-hop reasoning considers the route information between entity pairs, whereas single-hop reasoning uses representation learning to anticipate direct relations between entities. Therefore, Reinforcement Learning (RL) [6] may be used to execute route search in multi-hop reasoning, which can be seen as a markov decision process[5].

In this research, we take into account the benefits of RL in sequential decision-making and present a new knowledge reasoning approach called Action space Optimization and Joint Reward(AOJR). To achieve superior reasoning of missing data to KG, AOJR enhances the policy network through action selection, route encoding, and reward function. The following are the paper's most significant contributions:

(1) The agent's journey through hierarchical decision-making to alleviate the curse of dimensionality in the large-scale KG is completed with the addition of the action space optimization module for action selection to the policy network of RL.

(2) To address the problem of reward sparsity in the policy network, a novel joint reward mechanism is developed, including single-step reward, path reward, and basic reward. This mechanism rewards agents for their walking behavior at various phases of development.

The experimental findings demonstrate that (3) action space optimization and joint reward mechanism may enhance reasoning ability, as AOJR outperforms the comparison model in both reasoning correctness and route exploration ability.

2. Related Work

Knowledge reasoning for KG, according to related research[7], may be broken down into three distinct subfields: knowledge reasoning grounded in logical principles, knowledge reasoning grounded in representation learning, and knowledge reasoning grounded in RL. Typically, the relevant algorithms of statistics and probability theory are used in knowledge reasoning based on logical rules in order to mine a collection of rules with high confidence, and then these rules are used in reasoning. For learning rules quickly, the Association rule Mining algorithm with Incomplete Evidence[8] (AMIE) is used. AMIE knows how to unearth the closed-loop horn rule, which stipulates that every entity in the chain of rules must be mentioned twice. However, AMIE only mines a small percentage of rules, prompting the proposal of AMIE+[9]. Setting a threshold between 0 and 1 for each rule and adjusting the parameters during rule expansion and pruning allow the algorithm to optimize AMIE. Tensorlog[10] uses sparse matrix multiplication for first-order logic rule reasoning, where entities and relations are represented by one-hot encoding and matrix, respectively, thereby transforming rule-based knowledge reasoning into differentiable matrix computations, thereby solving the non-differentiable problem of rule learning.

Prediction is achieved via scoring functions, and knowledge reasoning based on representation learning portrays items and relations as low-dimensional vectors. TransE[11] has the benefits of high accuracy and efficient computation, but it cannot handle complicated relations since it views the relation vector as a

translation from the head entity vector to the tail entity vector. To allow entities under various relations to get distinct embedding vectors, TransH[12] establishes an additional projection plane for each relation. A high-dimensional sparse third-order tensor is used to represent KG in RESACL[13], and this tensor is then broken down into three factor matrices and a core tensor. DistMult[14] simplifies RESACL by restricting the bilinear relation matrix to the diagonal matrix, which increases the efficiency of the reasoning process. ComplEx[15] uses the complex dot product to determine the triplet score by first projecting entities and relations into a complex vector space. Scores for symmetrical triplets might change based on the relative positions of the head and tail entities. Later, features are extracted using neural networks. Using convolution, ConvKB[16] combines the embeddings of "head" entities with those of "relations" and "tail" entities. By using scaled entity significance, which is determined by an attention-based global random walk algorithm, EIGAT[17] facilitates the accurate assimilation of global information into the Graph Attention Network(GAT) family of models.

The DeepPath[18] technique is the first to apply RL to knowledge reasoning using a multi-hop reasoning approach. The generalization performance is significantly improved by sampling relations for extending pathways using RL. An end-to-end paradigm for answering multi-hop KG queries, MINERVA[19] requires just a specified bigram (head entity, relation) as input. Top-down rule learning models like AnyBURL[20] may be used to do RL-based route sampling by agents. To determine the veracity of tail entities, the fundamental rules are built in accordance with the sample route. With the use of a recurrent neural network and a monte carlo tree, M-Walk[21] is able to encode route states and construct more rewarding pathways despite the problem of few rewards.

Knowledge reasoning that is based on representation learning simplifies reasoning by mapping things and relations into low-dimensional vectors. However, these approaches only address direct interactions between entity pairs, and they can't communicate anything about the paths between them. By extending the route via policy choice, knowledge reasoning based on reinforcement learning compensates for the restriction of representation learning. Unfortunately, RL's performance during path search is subpar. One issue is that the action selection and dimensional catastrophe both rise exponentially with the quantity of the dataset and the dimension of the vectors. On the other hand, rewards may be scarce in large-scale KG if just the reasoning outcome is considered and the agent's action input throughout the walking process is disregarded.

In light of the aforementioned dimensional catastrophe and reward sparsity, we suggest a new RL reasoning approach called AOJR that is based on optimizing action spaces and jointly receiving rewards. We augment the policy network with an action space optimization module and progressively refine the action selection to shrink the action space, therefore increasing the method's precision and generalizability. In addition, the new method of action selection is intended to be rewarded jointly. The basic reward is in charge of doling out the ultimate payout, whereas the route reward is used to encourage connection selection and may increase path variety. To reduce incentive scarcity, we blend the three types of rewards according to a predetermined weight**.**

3. Action Space Optimization And Joint Reward for Knowledge Graph Reasoning

In RL, the action selection process is broken up into three stages, namely relation clustering, relation cluster selection, and relation selection. These stages are based on the concept of hierarchical reinforcement learning [22]. Figure 1 depicts AOJR, which may be broken down into its three component elements.

(1) Use the TuckER Embedding format. TuckER is used to do the mapping from the triplets of the KG to the low-dimensional vectors.

(2) There are two components to the Policy Network. The portion on the left is an optimization module for action space that was developed for the agent action selection. The route walking module can be found in the top right corner. This module makes use of a Long Short-Term Memory Network (LSTM) to encode the path. The joint reward module that has been built may be found in the bottom right corner. This module includes the single-step reward, the route reward, and the basic reward.

(3) The use of reasoning that is governed by policy. As a result of the interaction between the policy network and the knowledge graph, the agent is able to get the route between entity pairs in order to finish the knowledge reasoning.

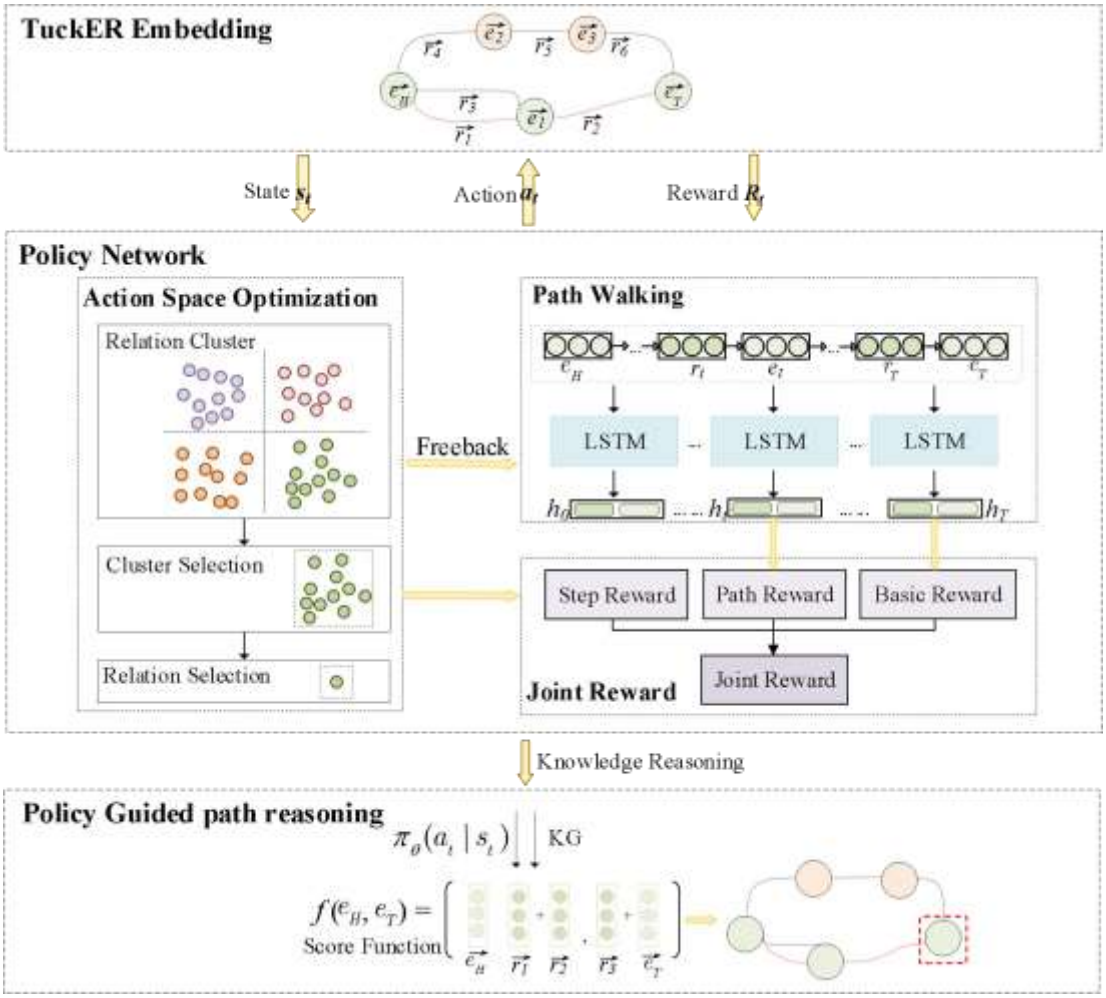
3.1 TuckER Embedding

TuckER (Tensor factorization for Knowledge Graph Embeddings) is a knowledge graph embedding model that combines the advantages of tensor factorization and neural networks. It was proposed in the paper "TuckER: Tensor Factorization for Knowledge Graph Completion" by Balazevic et al. (2019).

The main idea behind TuckER is to leverage tensor factorization techniques to model the complex relationships in a knowledge graph. Knowledge graphs consist of triples (head entity, relation, tail entity) that represent factual information. The goal of knowledge graph embedding models is to learn low-dimensional representations (embeddings) for entities and relations, such that these embeddings can be used to predict missing triples or answer queries about the knowledge graph.

TuckER represents entities and relations as tensors and employs tensor factorization to capture the latent structure of the knowledge graph. Specifically, TuckER uses the Tucker decomposition, which decomposes a tensor into a core tensor and factor matrices for each mode of the tensor. In TuckER, the core tensor captures the interactions between entities and relations, while the factor matrices represent the embeddings for entities and relations.

To combine tensor factorization with neural networks, TuckER introduces a neural network layer that maps the factor matrices to low-dimensional embeddings. This layer consists of a series of linear transformations, non-linear activation functions, and dropout regularization. By incorporating neural networks, TuckER can capture more complex patterns and non-linear relationships in the knowledge graph. During training, TuckER learns the embeddings and factor matrices by optimizing a suitable objective function, such as maximizing the likelihood of observed triples or minimizing the margin-based ranking loss. Once trained, TuckER can be used for tasks such as knowledge graph completion, link prediction, and entity classification.



3.2 Action space optimization

Action space optimization for knowledge reasoning involves improving the efficiency and effectiveness of the actions taken in the process of reasoning with knowledge. It focuses on reducing the number of possible actions or decision options while maintaining or enhancing the quality of reasoning outcomes. Here are some approaches for optimizing the action space in knowledge reasoning:

- Knowledge pruning: Analyze and prune irrelevant or redundant knowledge from the knowledge base. By removing unnecessary or low-quality information, you can streamline the reasoning process and improve the efficiency of knowledge retrieval and utilization.
- Inference rule selection: Identify the most relevant and effective inference rules for the reasoning task at hand. Inference rules specify how to derive new knowledge based on existing knowledge. By selecting and prioritizing the most suitable rules, you can reduce the search space and improve the efficiency and accuracy of knowledge inference.
- Heuristics and search strategies: Develop heuristics and search strategies that guide the reasoning process towards relevant and promising paths. These strategies can help focus on relevant knowledge sources, prioritize certain types of reasoning, or efficiently explore the knowledge

graph or network. By intelligently navigating the action space, you can optimize the reasoning process.

- Context-aware reasoning: Take into account the context and specific problem at hand when selecting actions. By considering the contextual information, such as the goals, constraints, and available resources, you can prune irrelevant actions and prioritize those that are most likely to lead to desired reasoning outcomes.
- Machine learning and optimization: Utilize machine learning techniques to learn from past reasoning experiences and optimize the action space. This can involve training models to predict the relevance or effectiveness of different reasoning actions, and then using these models to guide the decision-making process.
- Feedback and evaluation: Collect feedback on the reasoning outcomes and evaluate the effectiveness of different actions. By analyzing the results and measuring the quality of reasoning outputs, you can iteratively refine and optimize the action space over time.
- Integration of human expertise: Incorporate human domain experts into the reasoning process. Experts can provide valuable insights and heuristics to guide action selection and improve the quality of reasoning. By combining human expertise with automated reasoning techniques, you can achieve more effective knowledge reasoning.

Pseudocode for Action Space Optimization

```
function optimizeActionSpace(knowledgeBase, goals, constraints):

    relevantKnowledge = pruneIrrelevantKnowledge(knowledgeBase)

    relevantInferenceRules = selectInferenceRules(knowledgeBase, goals, constraints)

    actionSpace = [ ]

    for each knowledge in relevantKnowledge:

        for each inferenceRule in relevantInferenceRules:

            newKnowledge = applyInferenceRule(knowledge, inferenceRule)

            if satisfiesConstraints(newKnowledge, constraints):

                actionSpace.append(newKnowledge)

    return actionSpace
```

In the above pseudo code, we assume the following functions:

- pruneIrrelevantKnowledge(knowledgeBase): This function prunes irrelevant or redundant knowledge from the knowledge base, returning a subset of relevant knowledge.

- selectInferenceRules(knowledgeBase, goals, constraints): This function selects the most suitable and relevant inference rules based on the knowledge base, goals, and constraints. It returns a set of inference rules to be used for reasoning.

- applyInferenceRule(knowledge, inferenceRule): This function applies an inference rule to a given knowledge, generating new knowledge based on the rule and existing knowledge. It returns the derived knowledge.

- satisfiesConstraints(knowledge, constraints): This function checks whether a given knowledge satisfies the specified constraints. It returns true if the constraints are satisfied, and false otherwise.

The optimizeActionSpace function iterates over the relevant knowledge and inference rules, applying the rules to the knowledge and checking if the resulting knowledge satisfies the given constraints. If a derived knowledge satisfies the constraints, it is added to the action space. Once the action space is populated, it can be used for further reasoning or decision-making processes.

3.3 Policy Guided Path Reasoning

Policy Guided Path Reasoning is an approach used to extract information from structured data, such as knowledge graphs, by leveraging the concept of reinforcement learning and graph traversal.

In the context of knowledge graphs, information extraction involves finding relevant paths between entities to infer additional knowledge or answer specific queries. Policy Guided Path Reasoning combines the power of graph traversal algorithms with reinforcement learning techniques to guide the search process and optimize the extraction of valuable information.

Here's a high-level overview of how Policy Guided Path Reasoning works:

- Knowledge Graph Representation: The knowledge graph is represented as a graph structure, where entities are represented as nodes and relations as edges connecting the nodes. Each node and edge typically have associated attributes or properties.
- Path Exploration: The reasoning process begins with an initial entity or a set of entities. Starting from these entities, different paths are explored in the knowledge graph by traversing through the available relations and nodes. Various graph traversal algorithms can be used, such as depth-first search (DFS) or breadth-first search (BFS), to systematically explore the graph.
- Reinforcement Learning: Reinforcement learning is employed to learn a policy that guides the path exploration process. The policy can be represented as a neural network or another suitable function approximator. The policy network takes the current state (e.g., current node, visited nodes, path) as input and outputs a probability distribution over available actions (e.g., possible relations to follow).

- Reward Signal: A reward signal is defined to evaluate the quality or usefulness of the paths explored. The reward signal depends on the specific information extraction task. For example, it can be based on the relevance of extracted facts, the accuracy of predictions, or other task-specific criteria. The reward signal provides feedback to the reinforcement learning agent and is used to update the policy network through techniques like policy gradient methods.
- Policy Optimization: The policy network is trained using reinforcement learning algorithms to optimize the extraction of valuable information. The objective is to learn a policy that maximizes the expected cumulative reward over multiple path explorations. This process involves iteratively exploring paths, collecting rewards, and updating the policy network parameters to improve the extraction performance.
- By iteratively updating the policy based on the reward signal, Policy Guided Path Reasoning can learn to focus on informative paths and discard irrelevant or uninformative ones. This approach allows for efficient and targeted information extraction from knowledge graphs, enabling the discovery of new knowledge or the fulfillment of specific queries.

4. Experiments

4.1 Dataset and Settings

To evaluate the effectiveness of the proposed method for KG reasoning, experiments are]conducted on three commonly used datasets, NELL-995, WN18RR and FB15K-237. Table 1 shows the statistics of the three datasets

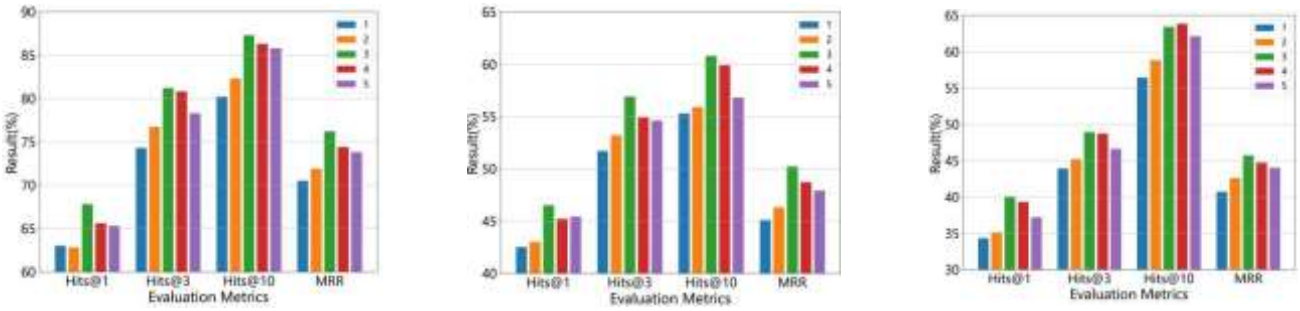
Table 1

Information statistics of the datasets

Datasets	#Ent.	#Rel.	#Train	#Test	#Mean in-degree
NELL-995	75, 492	200	154, 213	15, 838	4.07
WN18RR	40, 943	18	141, 442	5, 000	2.19
FB15K-237	14, 541	237	272, 115	20, 466	19.74

We use Mean Reciprocal Rank (MRR) and Hits@N as evaluation metrics, while N is 1, 3, 10. For the missing relation in the triplet, the model predicts the relation r = {*r1, r2, …, rn*} of the entity pair (*e1*, *e2*) in the test set according to the scoring function. The higher the correct relation is ranked in the relation set, the better the results of MRR and Hits@N. Adam is used for iterative optimization. The number of relation clusters k is set to 7, 3 and 10 on NELL-995, WN18RR and FB15K-237 respectively. The reward function coefficients δ, η, and λ is {0.5, 0.2, 0.3}. Dimensions of entity and

relation embedding and LSTM hidden dimension are both set to 100. The learning rate is set to 0.001, the path length is set to 3 and batch size is set to 128. To evaluate the effect of different path lengths and obtain the most suitable path length, Figure 5 shows the experimental results of different path lengths on three datasets. When the path length is less than 3, the results of Hits@N and MRR improve significantly with the increase of the path length. When the path length is equal to 3, the reasoning result reaches the highest value. As the path length increases, the results of Hits@N and MRR have different degrees of decline. Therefore, the path length is uniformly set to 3.



(a) NELL-995 (b) WN18RR (c) FB15K-237

Fig.2 Influence of different path lengths on reasoning results

5. Conclusion

In this article, we present a knowledge reasoning model called AOJR that is based on the optimization of action space and joint rewards. Constructing an action optimization space module with the huge agent's action search space in mind allows for better results. A new joint reward mechanism has been created, and it will be used for the sparse reward that the policy network offers. This new mechanism will deliver more accurate and diversified reward feedback. The suggested technique demonstrates a high degree of accuracy for relation reasoning between items in the KG, as shown by the fact that the experiment results on three datasets are superior to those obtained using previous methods. In light of the fact that the logic rule-based method is open to interpretation, we want to make an effort, for the sake of next research, to include logical rules into the reasoning model in an effort to improve the policy network's capacity for decision-making.

6. References

[1] Paulheim H. 2017. Knowledge graph refinement: A survey of approaches and evaluation methods. Semantic Web, 8(3), 489-508.

[2] Bollacker K, Evans C, Paritosh P, et al. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data. Vancouver, BC, Canada: ACM, 1247-1250. DOI:https:// dl.acm.org/doi//10.1145/1376616.1376746.

[3] Auer S, Bizer C, Kobilarov G, et al. 2007. Dbpedia: A nucleus for a web of open data. The semantic web. Busan, Korea: Lecture Notes in Computer Science, 722-735.

[4] Zhang J, Chen B, Zhang L, et al. 2021. Neural, symbolic and neural-symbolic reasoning on knowledge graphs. AI Open, 2, 14-35.

[5] Bellman R. A Markovian decision process. 1957. Journal of mathematics and mechanics, 6(5), 679-684.

[6] Pateria S, Subagdja B, Tan A, et al. 2021. Hierarchical reinforcement learning: A comprehensive survey. ACM Computing Surveys, 54(5), 1-35. DOI:https://dl.acm.org/doi/10.1145/3453160.

[7] Ji S, Pan S, Cambria E, et al. 2022. A survey on knowledge graphs: Representation, acquisition, and applications.IEEE Transactions on Neural Networks and Learning Systems, 494-514.

[8] Galárraga L A, Teflioudi C, Hose K, et al. 2013. AMIE: association rule mining under incomplete evidence in ontological knowledge bases. Proceedings of the 22nd International Conference on World Wide Web. Rio de Janeiro, Brazil: International World Wide Web Conferences Steering Committee, 413-422.

[9] Galárraga L, Teflioudi C, Hose K, et al. 2015. Fast rule mining in ontological knowledge bases with AMIE+. TheVLDB Journal, 24(6), 707-730.

[10] Bordes A, Usunier N, Garcia-Duran A, et al. 2013. Translating embeddings for modeling multi-relational data. Advances in Neural Information Processing Systems, 2787-2795.

[11] Cohen W W. 2016. Tensorlog: A differentiable deductive database. Computing Research Repository, 1605, 06523.

[12] Wang Z, Zhang J, Feng J, et al. 2014. Knowledge graph embedding by translating on hyperplanes. Proceedings of the AAAI Conference on Artificial Intelligence. Québec City, Canada: AAAI, 1112-1119.

[13] Nickel M, Tresp V, Kriegel H P. 2011. A three-way model for collective learning on multi-relational data. International Conference on Machine Learning. Bellevue, Washington, USA: Omnipress, 809-816.

[14] Yang B, Yih W, He X, et al. 2014. Embedding entities and relations for learning and inference in knowledge bases. International Conference on Learning Representations(Poster), 1-12.

[15] Trouillon T, Welbl J, Riedel S, et al. 2016. Complex embeddings for simple link

prediction. Proceedings of the 33nd International Conference on Machine Learning. New York, NY, USA: JMLR, 2071-2080.

[16] Dettmers T, Minervini P, Stenetorp P, et al. 2018. Convolutional 2d knowledge graph embeddings. Proceedings of the AAAI Conference on Artificial Intelligence. New Orleans, Louisiana, USA: AAAI, 1811-1818.