Rule Driven Query Expansion

No Author Given

No Institute Given

Abstract. Empty answers problem exists when we use SPARQL to access RDF knowledge graph data. Put situations that querying facts inexistent to the real world aside, one reason is that users lack enough knowledge for a particular RDF knowledge graph, that leads to SPARQL queries with wrong formats or inaccurate expressions. However, due to the schema-free nature of RDF data and incompleteness of particular RDF knowledge graphs, even a SPARQL query with correct format can reflect users' intentions accurately, it may fail to get any results. Researches going on in translating the natural language to SPARQL help a lot to address the first problem, but these are inconducive for the second problem which was caused by the faults in structure and content of RDF knowledge graphs. We design a rule-driven framework to alleviate the obstacles caused by the structure and content of RDF knowledge graphs. Specifically, given a SPARQL query, we use knowledge graph oriented rule-learning procedure to take reasoning rules, with the help of these rules, our system return possible results. More importantly, our system shows detailed information with similarity score and rules to explain why our system chooses particular possible answers.

Keywords: SPARQL · Empty Answer · Rule Learning.

1 Problem Statement

1.1 Empty Answer Problem for SPARQL query

Users use SPARQL queries reflecting their intentions to access the data from RDF knowledge graph, reasons for Empty Answer Problem are various, three of which are main ones: 1) the facts that users want to query do not exist in the real world, this may be caused by the wrong mapping of entities or relations, the misplace of subjects or objects; 2) the formats of SPARQL is wrong, including the namespace, operators and so on; 3) SPARQL queries are accurate and well-formatted, but they do not have exact matches in particular RDF knowledge graphs.

The problem 1) and 2) can be regarded as users can not use SPARQL language descripe their intentions, there are lots of works being done to solve this problem, including entity linking [21], relation linking [6] and more sophisticated work, translating natural language into SPARQL [20]. These works are done under the situation that users lack the full knowledge of particular RDF knowledge graph, they aim to help users to generate well-formatted SPARQL query to reflect their intentions accurately, however, it is not enough. Even a SPARQL query can avoid the problem 1) and 2), it may still has to face problem 3). For example, when a user want to know *"Who is the advisor of Newton?"*, he makes a SPARQL in table 3 to access Dbpedia, this is a correct SPARQL query, but it gets no answer because there is not an entity that exists explicit relation *"dbo:doctoralAdvisor"* with *"dbr:Isaac_Newton"*. But Newton really has advisors, it is *"dbr:Isaac_Barrow"*, this fact is recorded as *"dbr:Isaac_Newton dbo:academicAdvisor dbr:Isaac_Barrow"* in Dbpeida. This seems that we can make some "magic operations" to link word "advisor" to *"dbo:academicAdvisor"* instead of *"dbo:doctoralAdvisor"*, we also prepare another example in table 1. We construct this query to get Asian directors and their films. It gets no answer, too. Although some directors' birthplaces are connected to *"dbr:China"*, *"dbr:Japan"* and so on, they are Asian directors absolutely, but no one's birthplace is connected to *"dbr:Asia"* directly.

Table 1. SPARQL to get Asian films and their films.

SELECT ?film ?director WHERE	
(1)	?film dbo:director ?director.
(2)	?director dbo:birthPlace dbr:Asia.

In this paper, we assume that our SPARQL queries are not related to problem 1) and 2) and focus on problem 3). Problem 1) and 2) are more related to Natural Language Processing, Problem 3) is different, it is caused by the inherent limits(or features) of RDF knowledge graph. RDF knowledge graph is schema-free, there are often several similar predicates to describe the same relation, this leads to the problem about "Newton's advisors" above; RDF knowledge graph is incomplete, it is also impossible to make a single complete RDF knowledge graph now, so it leads to the problem about "Asian" presented above.

More clearly, Problem 3) has 2 typical kinds of expressions. One is the SPARQL queries have a high level of constraints, we get an instance from the released resource of [23], there are three constraints for the SPARQL query in table 2. This query gets no answers, in fact, the constraints (1) and (2) are redundant, we can infer that "?company" is "dbr:Apple_Inc" only with constraints (3). It seems good to relax or delete the unnecessary constraints. Some works tried to solve it by Query Relaxation techniques, including replacement with similar entities and predicates [7], relaxation with ontology rule [18] and approximation for related results with representation learning in vector space [9, 23, 26]. Some works try to find which parts of a SPARQL query should be deleted, but it is a NP-hard problem and do not get good results. Also, there are works [11] that try to construct this problem as a complete document IR problem, which needs extra text sources. We will detail these methods later.

Besides high level constraints query, there are many simple and plain SPARQL queries, like SPARL in table 3, relaxation techniques are not proper for these simple queries, because relaxation will change the meaning of simple SPARQL queries easily.

Table 2. SPARQL with high constraints.

SELECT ?company WHERE			
(1)	?company	rdfs:type	dbo:Company.
(2)	?company	dbo:industry	dbr:Electronics.
(3)	dbr:IPhone	dbp:developer	?company.

Table 3. SPARQL to get the advisors of Newton.

SELECT ?advisor WHERE	
(1)	dbr:Isaac_Netwon dbo:doctoralAdvisor ?advisor.

1.2 Semantic Parsing

1.3 Paraphrasing

1.4 Similarity Based Method

The idea behind similarity based method is straight and simple, it wants to replace certain entities or relations with similar ones to get approximated results. For SPARQL in table 1, replacing *"dbr:Electronics"* with its similar one *"dbr:Computer_hardware"* will make this SPARQL return *"dbr:Apple_Inc"* successfully.

This kind of works [7] are often constructed by two stages: 1)design approaches to calculate the similarity between entities and relations; 2)design stategies to replace entities and relations in original SPARQL queries.

We want to point out 2 drawbacks of this method.

1. It is hard to definen "similar". Take SPARQL in table 4 for example, users want to query Newton's students and advisors. For the first constraint, if we want to try to replace entity *"dbr:Isaac_Newton"*, we should choose Newton's classmates; for the second constraints, Newton's colleagues are more better. Similarity resides in aspects. Similar entites should be choosen according to the situations, but existing methods just choose entites under a fixed similarity ranking.
2. To our best of knowledge, existing entities similarity or relatedness methods [16,17] often neglect the influence of predicates, they regard RDF knowledge graph as social networks to calculate similarity between entites. Triples relatedness [22] may be a good direction to get high quality and fine-grained entities relatedness. Predicates similarity or relatedness is rarely studied.

Table 4. SPARQL to get the advisors and students of Newton.

SELECT ?advisor ?student WHERE			
(1)	dbr:Isaac_Netwon	dbo:doctoralAdvisor	?advisor.
(2)	?student	dbo:doctoralAdvisor	dbr:Isaac_Netwon.

1.5 Ontology Rule Based Method

This kind of method [18] follows rules in table 5. Given an ontology K, this method computes the *extended reduction* $extRed(K)$ of K as follows: (i) compute the closure of K as $cl(K)$; (ii) apply the rules of table 6 until no longer applicable; and (iii) apply rules (1) and (3) of table 5 in reverse until no longer applicable. (Applying a rule in reverse means deleting the triple deduced by the rule.) Then for rules in table 5, it assigns cost of applying (2) and (4) to be β, and the cost of applying (5) and (6) to be γ. Finally, it uses thest four rules to relax some constraints and calculate the costs to rank the results.

This approach has two drawbacks.

1. This is a very limiting relaxation. After constructing *extended reduction* $extRed(K)$, every uri only has its nearest super class, super property, range and domain. According to original paper [18], it is convient to calculate final costs, but it also leads to the a uri has to relax in a small scope.
2. This is inaccurate in many situations.

Table 5. RDFS Inference Rules.

Group A (Subproperty)	(1) $\frac{(a,sp,b)(b,sp,c)}{a,sp,c}$	(2) $\frac{(a,sp,b)(X,a,Y)}{(X,b,Y)}$
Group B (Subclass)	(3) $\frac{(a,sc,b)(b,sc,c)}{(a,sc,c)}$	(4) $\frac{(a,sc,b)(X,type,a)}{(X,type,b)}$
Group C (Typing)	(5) $\frac{(a,dom,c)(X,a,Y)}{(X,type,c)}$	(6) $\frac{(a,range,c)(X,a,Y)}{(Y,type,c)}$

Table 6. Rules used to compute the extended reduction of an RDFS ontology.

(e1) $\frac{(b,dom,c)(a,sp,b)}{a,dom,c}$	(e2) $\frac{(b,range,c)(a,sp,b)}{(a,range,c)}$
(e3) $\frac{(a,dom,b)(b,sc,c)}{(a,dom,c)}$	(e4) $\frac{(a,range,b)(b,sc,a)}{(a,range,c)}$

1.6 Embedding Based Method

The basic idea behind this kind of method is to approximate results using the projection in vector space. This is a good blueprint in approximating SPARQL, but we do not think Embedding method is suitable for this problem. Take SPARQL in table 1 for example. The *"?director"* is approximated by *"dbr:Asia"* and *"dbo:birthPlace"*. We can note that this method only restrain *"?director"* to be a "thing" who was born in Asia, it has nothing to do with "Film Director".

For straight and simple SPARQL in table 3, this method looks great in theory but have a bad performace in practicing. Under this situation, this can be regarded as a link prediction problem with embedding [13]. We survey the preformance of embedding method [2,12,19] in link prediction problem and have 2 conclusions: 1) the size of test dataset is much smaller than the size of the whole Dbpedia; 2) even in test dataset, preformance is not good. Our experiments also show this method may not work.

1.7 Conclusions about existing methods.

The three methods above are all designed for SPARQL queries with over constraints. They all have high degree of relaxation. The first two methods make very inaccurate relaxations to reduce the influence of some redundant constraints like the second constraint in table 2. Embedding method needs over constraints to approximate final results in vector space because the ability of link prediction is stable now.

These method all have fatal problems when dealing with simple and straight query like SPARQL in table 3.

They also do not have explicit design to display why some approximated results are choosen. Before we introduce our model, we want to make serveral requirements for a query relaxation system.

1. The ability to get accurate results, especially for the simple query like SPARQL in table 3.
2. Explicit explanation why some results are choosen.

2 Our model

We can see from Fig. 3 that they can not handle simple query properly. We use the reasong rules in RDF knowledge graph to edit SPQARQL query with no answers, like query graph in Fig. 2. The ideas behind our model is simple, if we do not know who is the advisor of Newton, we can go to find someone whose student is Newton. Take SPARQL query in table 3 for example. We use RDF knowledge graph oriented rule learning techniques [15] to mine inference rules for predicate "dbo:doctoralAdvisor", then we find the "?advisor" candidates that can fulfil one of the inference rules. Then we rank candidates and design a form to display the rules to explain the reason we choose these candidates. We will detail them in subsequent sections.

2.1 Rule Learning Components.

In our problem, rule learning is used to search the inference rules of a particular predicate. Rule learning is a problem with high complexity. In recent years, there have been some machine learning methods [4,5,15,24,25] which try to improve the efficiency and accuracy of rule learning. But the datasets they use are FB15K-237, NELL-995, WN18RR, FB15K and WN18. We can see from table 7 that their size is much smaller than Dbpedia [1] and their underlying technologies, Reinforcement Learning, RNN, Attention, are are all trying to map the whole knowledge graph to a vector space. The time and space complexity will be very high if we apply these methods directly to Dbpedia. We choose Path Ranking Algorithm [15] as our main algorithm and make some adjustments with refering to some ideas from Inductive Logic Programming [3] and AMIE [8], we want to describe Path Ranking Algorithm first and then show the adjustments and results.

Path Ranking Algorithm(PRA) Given knowledge graph G and a relation R, a inference rule p_i of R is a sequence of relations $R_0, R_1, R_2, ...R_l$, where $domain(R_0) = domain(p_i)$ and $range(R_l) = range(p_i)$. For example, the triple $(BobDylan, birthPlace, USA)$ can be expressed by the inference rule $(birthPlace \rightarrow birthPlace, country)$ as follows:

$$(BobDylan, birthPlace, Duluth,\_Minnesota, country, USA)$$

A relation R usually has several inference rules as a inference rule set $P = \{p_0, p_1, ..., p_n\}$. PRA [14,15] can give proper weights to every p_i of P, which is regarded as $\Theta = \{\theta_0, ..., \theta_n\}$. For two entities (h, t), we can get $score_{(h,t)}p_i$ as Eq.(2). Then we can get the probability $prob(h, R, t)$ whether (h, t) has relation R as Eq.(1).

$$prob(h, R, t) = \frac{1}{1 + e^{-value(h,R,t)}} \tag{1}$$

$$score_{(h,t)}p_i = \begin{cases} 1, \text{if}(h, p_i, t) \in G \\ 0, \text{if}(h, p_i, t) \notin G \end{cases} \tag{2}$$

$$value(h, R, t) = \theta_0 score_{(h,t)}p_0 + \theta_i score_{(h,t)}p_i + ... + \theta_n score_{(h,t)}p_n \tag{3}$$

The flow of the model is as follows:

1. For a relation R, it gets entity pair set $Posi$ and $Nege$ as Eqs.(4) and (5), and then it records the paths between every (h_i, t_i) of $Posi$ as $P = \{p_0, p_1, ...p_n\}$.
2. Given a relation R and a set of entity pairs $\{(h_i, t_i)\}$, it constructs a training set $\{(x_i, r_i)\}$, where x_i is the path feature of (h_i, t_i), the j-th component of x_i is $x_{ij} = score_{(h_i,t_i)}p_j$, r_i is defined as Eq.(9).
3. Given training set $\{(x_i, r_i)\}$, it uses Logistic Regression [10] to estimate weights Θ. The loss function is as Eq.(8).

$$Posi = \{(h_i, t_i) | (h_i, R, t_i) \in G\} \tag{4}$$

$$Nege = \{(h_i, t_i) | (h_i, R, t_i) \notin G\} \tag{5}$$

$$p_i = p(r_i = 1 | \mathbf{x}_i; \Theta) = \frac{\exp(\Theta^T \mathbf{x}_i)}{1 + \exp(\Theta^T \mathbf{x}_i)} \tag{6}$$

$$l_i(\Theta) = r_i \ln p_i + (1 - y_i) \ln(1 - p_i) \tag{7}$$

$$L(\Theta) = \sum_i l_i(\Theta) - \lambda_1 |\Theta|_1 - \lambda_2 |\Theta|_2 \tag{8}$$

$$r_j = \begin{cases} 1, \text{if}(h_i, R, t_i) \in G \\ 0, \text{if}(h_i, R, t_i) \notin G \end{cases} \tag{9}$$

Compared with rule learning methods [4, 25] that project the whole knowledge graph into vector space, PRA can generate explicit and more generic rules, that is the main reason why we choose PRA to mine rules. To our best of knowledge, there is not work that evaluates the performance of methods that project the whole KG into vector space in big knowledge graph like Dbpedia. Methods like PRA give a direction that we can learn rules and models in a small KG and use what we have learned in a larger KG. DeepPath [24] can also get explicit rules but it still uses PRA to learn weights for different rules, and more importantly, it uses Reinforcement Learning to mine rules, so it costs much more time than PRA and the imporvements are very limited according the expriments from the paper [24], so we think PRA is more proper.

PRA also has some limits, we present thest limits and adjust them as follows:

1. The knowledge graph trained and tested by PRA is much smaller than Dbpeida which is showed in Table 7. It is also difficult to train a Logistic Regression Model in the whole Dbpedia, we want to give an example to explain the ideas that we try to solve this problem. For example, if we want to get inference rules of relation "*birthPlace*", we do not extract the entity pairs (h_i, t_i) that hold "*birthPlace*" from the whole Dbpeida, we extract these related to a specific country, for example, our experiments choose USA. We think that the whole Dbpedia represents the world, the inference rules of "*birthPlace*" are the same to either the whole world and a small city, so we can get high-quality rules in a rather smaller knowledge graph.
2. The mined rules need a preliminary filter. Due to we record every pathes hold by entity pairs in $Posi$, we usually get many rules and most of them are redundant and wrong. We borrowed concept "Partial Completeness Assumption(PCA)" used by rule learning tool AMIE [8]. We use this concept to calculate the precision of a inference rule. The concept and algorithm will be displayed later.

Partial Completeness Assumption(PCA) PCA holds the ideas that a fact does not exsit in a KG, it is just unknown to this KG, it may be right, so a fact can be categorized to KBtrue, KBfalse, Newtrue and Newflase as show in Fig.1. The precision of a inference rule $B \Rightarrow R(h,t)$ is calculated as Eq.(11).

$$\text{supp}(\boldsymbol{B} \Rightarrow r(x,y)) := \#(x,y) : \exists z_1, \ldots, z_m : \boldsymbol{B} \wedge r(x,y) \tag{10}$$

$$\text{pcaconf}(\boldsymbol{B} \Rightarrow r(x,y)) := \frac{\text{supp}(\boldsymbol{B} \Rightarrow r(x,y))}{\#(x,y) : \exists z_1, \ldots, z_m, y' : \boldsymbol{B} \wedge r\left(x, y'\right)} \tag{11}$$

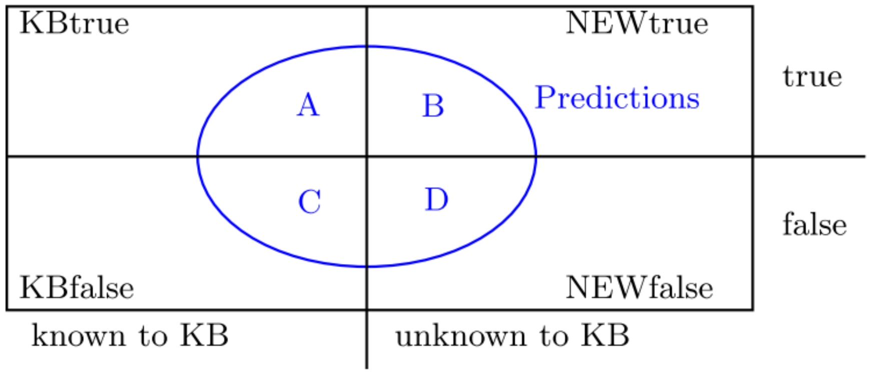

Fig. 1. Prediction under Incompleteness.

Table 7. Data size comparison of different datasets.

Dataset	*#entities*	*#relations*	*#facts*
WN18RR	40945	15	86835
NELL-995	75492	200	154213
FB15K-237	14504	237	272115
WN18	40943	18	106088
FB15K	14951	1345	362538
Dbpeida	**5900558**	**1322**	**18746174**

2.2 Candidates Searching Components.

Candidate searching is showed in Fig.(2).

2.3 Reasons Displaying Components.

1. Eperiment 1:rule learning in a country and test in whole world
2. Query test.

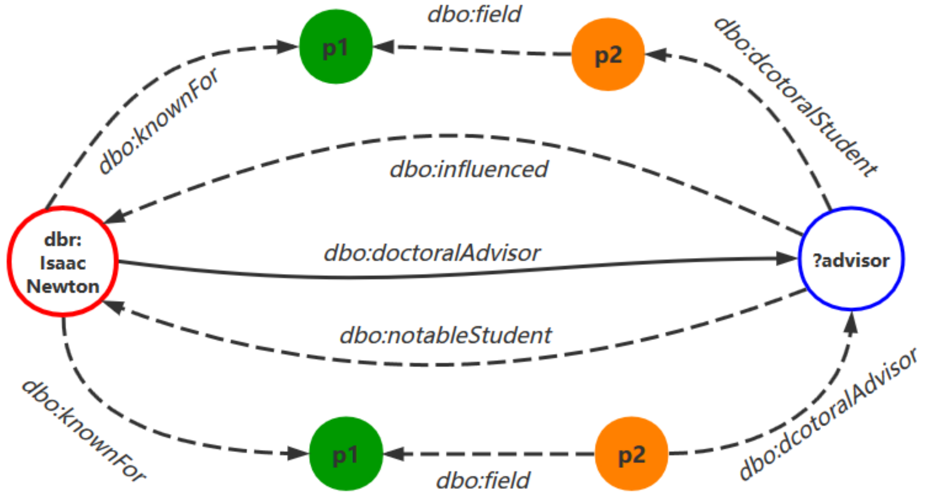

Fig. 2. Query Graph to get the advisors of Newton

References

1. Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R., Ives, Z.: Dbpedia: A nucleus for a web of open data. In: The semantic web, pp. 722–735. Springer (2007)
2. Bordes, A., Usunier, N., Garcia-Duran, A., Weston, J., Yakhnenko, O.: Translating embeddings for modeling multi-relational data. In: Advances in neural information processing systems. pp. 2787–2795 (2013)
3. Camacho, R., King, R., Srinivasan, A.: Inductive Logic Programming: 14th International Conference, ILP 2004, Porto, Portugal, September 6-8, 2004, Proceedings, vol. 3194. Springer Science & Business Media (2004)
4. Das, R., Dhuliawala, S., Zaheer, M., Vilnis, L., Durugkar, I., Krishnamurthy, A., Smola, A., McCallum, A.: Go for a walk and arrive at the answer: Reasoning over paths in knowledge bases using reinforcement learning. arXiv preprint arXiv:1711.05851 (2017)
5. Das, R., Neelakantan, A., Belanger, D., McCallum, A.: Chains of reasoning over entities, relations, and text using recurrent neural networks. arXiv preprint arXiv:1607.01426 (2016)

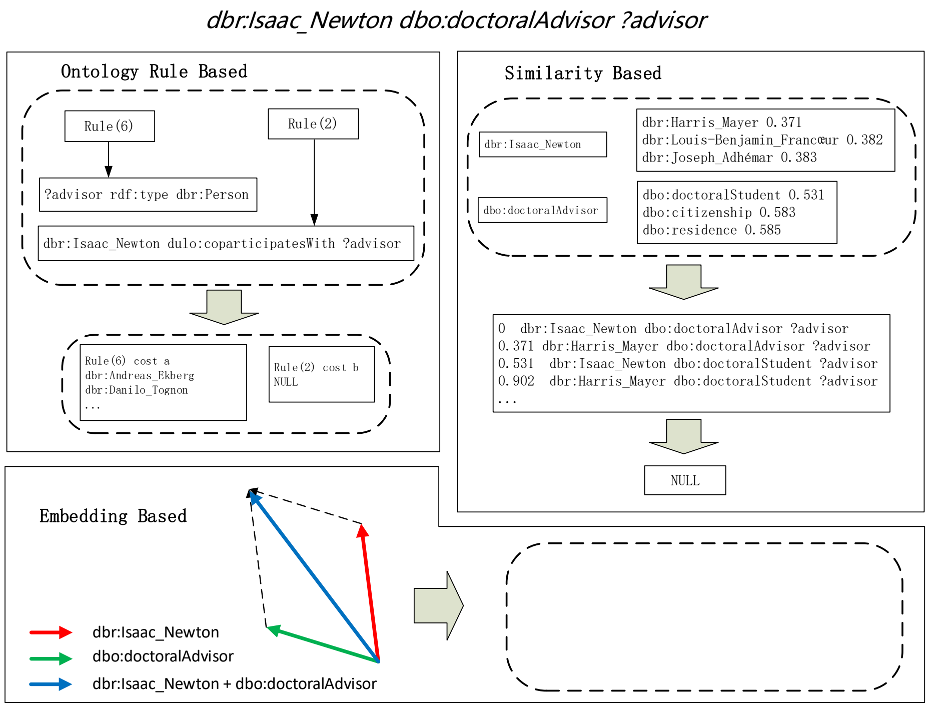
dbr:Isaac_Newton dbo:doctoralAdvisor ?advisor
Ontology Rule Based
Rule(6)
Rule(2)
?advisor rdf:type dbr:Person
dbr:Isaac_Newton dulo:coparticipatesWith ?advisor
Rule(6) cost a
dbr:Andreas_Ekberg
dbr:Danilo_Tognon
...
Rule(2) cost b
NULL
Similarity Based
dbr:Isaac_Newton
dbr:Harris_Mayer 0.371
dbr:Louis-Benjamin_Francœur 0.382
dbr:Joseph_Adhémar 0.383
dbo:doctoralAdvisor
dbo:doctoralStudent 0.531
dbo:citizenship 0.583
dbo:residence 0.585
0 dbr:Isaac_Newton dbo:doctoralAdvisor ?advisor
0.371 dbr:Harris_Mayer dbo:doctoralAdvisor ?advisor
0.531 dbr:Isaac_Newton dbo:doctoralStudent ?advisor
0.902 dbr:Harris_Mayer dbo:doctoralStudent ?advisor
...
NULL
Embedding Based
dbr:Isaac_Newton
dbo:doctoralAdvisor
dbr:Isaac_Newton + dbo:doctoralAdvisor

Fig. 3. Summary of Three Baselines.

6. Dubey, M., Banerjee, D., Chaudhuri, D., Lehmann, J.: Earl: Joint entity and relation linking for question answering over knowledge graphs. arXiv preprint arXiv:1801.03825 (2018)
7. Elbassuoni, S., Ramanath, M., Weikum, G.: Query relaxation for entity-relationship search. In: Extended Semantic Web Conference. pp. 62–76. Springer (2011)
8. Galárraga, L.A., Teflioudi, C., Hose, K., Suchanek, F.: Amie: association rule mining under incomplete evidence in ontological knowledge bases. In: Proceedings of the 22nd international conference on World Wide Web. pp. 413–422. ACM (2013)
9. Hamilton, W., Bajaj, P., Zitnik, M., Jurafsky, D., Leskovec, J.: Embedding logical queries on knowledge graphs. In: Advances in Neural Information Processing Systems. pp. 2030–2041 (2018)
10. Hosmer Jr, D.W., Lemeshow, S., Sturdivant, R.X.: Applied logistic regression, vol. 398. John Wiley & Sons (2013)
11. Huang, H., Liu, C., Zhou, X.: Approximating query answering on rdf databases. World Wide Web **15**(1), 89–114 (2012)
12. Ji, G., He, S., Xu, L., Liu, K., Zhao, J.: Knowledge graph embedding via dynamic mapping matrix. In: Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers). vol. 1, pp. 687–696 (2015)
13. Kazemi, S.M., Poole, D.: Simple embedding for link prediction in knowledge graphs. arXiv preprint arXiv:1802.04868 (2018)
14. Lao, N., Cohen, W.W.: Fast query execution for retrieval models based on path-constrained random walks. In: Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining. pp. 881–888. ACM (2010)
15. Lao, N., Mitchell, T., Cohen, W.W.: Random walk inference and learning in a large scale knowledge base. In: Proceedings of the Conference on Empirical Methods in Natural Language Processing. pp. 529–539. Association for Computational Linguistics (2011)
16. Perozzi, B., Al-Rfou, R., Skiena, S.: Deepwalk: Online learning of social representations. In: Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining. pp. 701–710. ACM (2014)
17. Ponza, M., Ferragina, P., Chakrabarti, S.: A two-stage framework for computing entity relatedness in wikipedia. In: Proceedings of the 2017 ACM on Conference on Information and Knowledge Management. pp. 1867–1876. ACM (2017)
18. Poulovassilis, A., Wood, P.T.: Combining approximation and relaxation in semantic web path queries. In: International Semantic Web Conference. pp. 631–646. Springer (2010)
19. Qian, W., Fu, C., Zhu, Y., Cai, D., He, X.: Translating embeddings for knowledge graph completion with relation attention mechanism. In: IJCAI. pp. 4286–4292 (2018)
20. Sander, M., Waltinger, U., Roshchin, M., Runkler, T.: Ontology-based translation of natural language queries to sparql. In: NLABD: AAAI Fall Symposium (2014)
21. Shen, W., Wang, J., Han, J.: Entity linking with a knowledge base: Issues, techniques, and solutions. IEEE Transactions on Knowledge and Data Engineering **27**(2), 443–460 (2015)
22. Voskarides, N., Meij, E., Reinanda, R., Khaitan, A., Osborne, M., Stefanoni, G., Kambadur, P., de Rijke, M.: Weakly-supervised contextualization of knowledge graph facts. arXiv preprint arXiv:1805.02393 (2018)

23. Wang, M., Wang, R., Liu, J., Chen, Y., Zhang, L., Qi, G.: Towards empty answers in sparql: Approximating querying with rdf embedding. In: International Semantic Web Conference. pp. 513–529. Springer (2018)
24. Xiong, W., Hoang, T., Wang, W.Y.: Deeppath: A reinforcement learning method for knowledge graph reasoning. arXiv preprint arXiv:1707.06690 (2017)
25. Yang, F., Yang, Z., Cohen, W.W.: Differentiable learning of logical rules for knowledge base reasoning. In: Advances in Neural Information Processing Systems. pp. 2319–2328 (2017)
26. Zhang, L., Zhang, X., Feng, Z.: Trquery: An embedding-based framework for recommanding sparql queries. arXiv preprint arXiv:1806.06205 (2018)