Enhanced Rule-based OWL Reasoning on Spark

Zhihui Liu[1,3], Weimin Ge[1,3], Xiaowang Zhang[1,3,*], and Zhiyong Feng[1,2]

[1] School of Computer Science and Technology, Tianjin University, Tianjin, China
[2] School of Computer Software, Tianjin University, Tianjin, China
[3] Tianjin Key Laboratory of Cognitive Computing and Application, Tianjin, China
∗Corresponding author: xiaowangzhang@tju.edu.cn

Abstract. The rule-based OWL reasoning is to compute the deductive closure of an ontology by applying RDF/RDFS and OWL entailment rules. In previous work, we present an approach to enhancing the performance of the rule-based OWL reasoning on Spark based on a locally optimal executable strategy. However, some key optimizations that based on LUBM do not generalize to more diverse datasets. In this paper, we analyze these problems, and demonstrate the inference engine. We have evaluated the approach using the real-world datasets. The experimental results show that our approach also achieve better performance as compared to Kim & Park's algorithm (2015).

1 Introduction

The Web Ontology Language [1] (OWL) is a semantic web language designed to represent rich and complex knowledge about things, groups of things, and relations between things. It provides the means for ontology to verify the consistency and to make implicit knowledge explicit. In [5] we already presented an method to enhancing the performance of the rule-based OWL reasoning on Spark.

In this paper, we extend our approach to deal with more diverse datasets and demonstrate the high efficiency. The major contributions and novelties of our work are summarized as follows: the OWL-Horst rules are classified into four classes and we provide the optimal executable strategies for each class. The rest of this paper is organized as follows. Section 2 presents our locally optimal strategies. Section 3 implements our proposed strategy on Spark and Section 4 evaluates our method on the standard dataset. In Section 5, we summarize this paper.

2 Locally optimal strategy

OWL-Horst [2] has a powerful expression and reasoning ability. There exists interdependency among the rules. If the input to a rule R_i depends on the output of another rule R_j, then the rule R_i is dependent on R_j. The rule R_j should be executed before the rule R_i. A rule dependency graph can be constructed based on the interdependency among the rules. There are millions of executable strategies among the rules that adjacent rules satisfy dependency. And the longest strategies contain 22 rules that cannot fully execute all rules. The number is so huge that it is challenge to test every strategy. The OWL reasoning has close relationship with the characteristic of datasets. Therefore, we prefer to make a analysis of dataset.

LUBM [3] is a widely used standard benchmark for evaluating the performance of ontology reasoning. We divide the dataset into three classes as follows:

– Triples whose predicate is *rdf:type*. We call this class as type.
– Triples whose predicate is *owl:sameAs*. We call this class as *sameAs* .
– The remainder is classified as a class. We call this class as SPO.

The data generator (UBA) uses specific rules to generate data so that all datasets generated by it have same data characteristic. Here we use the LUBM-50 as the sample and analyze the proportion of each class. The result is listed in Table 1.

Dataset	type	sameAs	SPO
LUBM-50	20.055%	0	79.945%

Table 1: The proportion of each type in LUBM-50

From the Table 1, we can see that the SPO triples account for absolute proportion, about 80% of the total. The type triples are in the second place, but the number of *sameAs* triples is zero. Therefore, the OWL reasoning should focus on the reasoning of SPO triples and type triples. Based on the statistics of dataset, we divide the OWL-Horst rules[2] into four classes as follow:

– Rules whose condition or consequences have triples whose predicates are rdf:type, including R4, R5, R6, O13, O14, O15, O16. The dependency graph of type rules is shown in Figure 2.
– Rules whose condition or consequences have triples whose predicates are owl:sameAs, including O1, O2, O5, O6, O8, O9, O10. The dependency graph of sameAs rules is shown in Figure 3.
– Rules whose condition or consequences have triples of SPO class, including R3, O3, O4, O7a, O7b. The dependency graph of SPO rules is shown in Figure 1, which the rule O7a and O7b are classified as O7.
– The remainder is classified as a class, including R1, R2, O11a, O11b, O11c, O12a, O12b, O12c. The dependency graph of schema rules is shown in Figure 4.

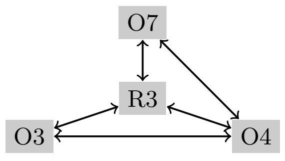

Fig. 1: SPO rules dependency graph

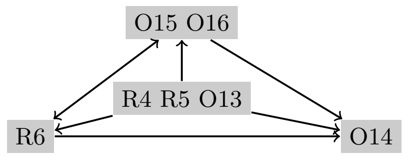

Fig. 2: type rules dependency graph

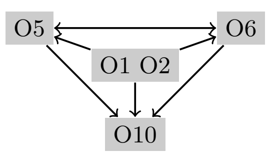

Fig. 3: sameAs rules dependency graph

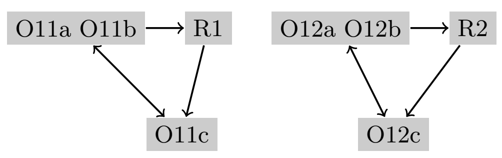

Fig. 4: schema rules dependency graph

For each class, based on the dependency of rules, we use the *depth-first-search* (DFS) algorithm to find all possible executable orders among the rules. Through the experiments, there is a large number of orders for each class, but we choose the longest orders that contain rules as many as possible. We pick out the orders that spend the shortest time to infer same dataset as the optimal executable orders. The optimal executable orders for each class are listed in Table 2.

SPO rules	type rules
	R4 → R6 → O14 → O13 → O15 → O16
O3 → R3 → O7 → O4	R4 → R6 → O14 → O13 → O16 → O15
O7 → R3 → O3 → O4	R5 → R6 → O14 → O13 → O15 → O16
	R5 → R6 → O14 → O13 → O16 → O15
schema rules	sameAs rules
O11a, O11b → R1 → O11c	O1 → O10 → O2 → O6 → O5
O12a, O12b → R2 → O12c	O2 → O10 → O1 → O6 → O5

Table 2: The optimal strategies of each class

3 Distributed rule-based OWL reasoning on Spark

In this section, we design the overall strategy of the OWL reasoning. The reasoning of instance triples will use the output of schema triples, so the schema reasoning should be executed firstly. There are several orders for each class of instance triples. We can choose any one order and then combine an new executable strategy. In this paper, we select the first order for each class to perform reasoning. The general workflow of the parallel OWL reasoning is described in figure 5.

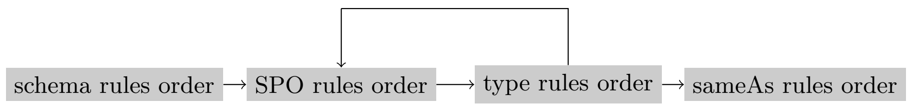

Fig. 5: The reasoning strategy

4 EVALUATION

In this section, we conduct a series of experiments to compare the performance of the proposed approach with the KP [4] under the same environment. We set up a cluster with one master and four worker nodes. Each node has 48 Xeon E5_4607 2.20GHz processors, 64GB memory and 10TB 7200 RPM SATA hard disk. The nodes are connected with Gigabit Ethernet. All the nodes run on a 64-bit Ubuntu 12.04 LTS operating system. The version of the Spark is 1.0.2. And the corresponding Hadoop v2.2 with Java 1.7 is installed on this cluster. We use both the synthetic and real-world benchmarks.

DBpedia is a widely used standard benchmark for semantic programs. We used four sets of DBpedia data: DBpedia-10(10 million triples), DBpedia-15(15 million triples), DBpedia-20(20 million triples), DBpedia-25(25 million triples) in our experiments. All experiments run three times and the average value is listed as follows.

Experimental results are shown in figure 6. We can see that the running time increases almost linearly with the growth of data size, our approach performs better than KP on DBpedia, which is improved by about 30%.

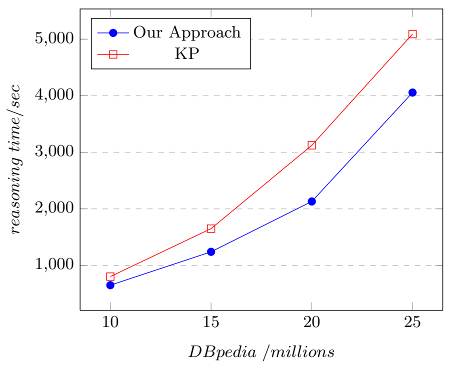

Fig. 6: The running time of our approach and KP

5 Conclusions

In this paper, we have shown an approach to enhancing the performance of the rule-based OWL reasoning on Spark and demonstrated inference over both standard and real-world dataset. In terms of reasoning time, Our method performs much better than KP in the reasoning strategy.

6 Acknowledgments

This work is supported by the program of the National Natural Science Foundation of China (NSFC) under grant number 61502336. Xiaowang Zhang is supported by Tianjin Thousand Young Talents Program.

References

1. L. McGuinness D., van Harmelen F. (2004) Web Ontology Language. *W3C Recommendation.*
2. Liu C., Qi G., Wang H. & Yu Y. (2011) Large scale fuzzy pD* reasoning using MapReduce. In: *Proc. of ISWC 2011*, Springer, pp. 405–420.
3. Guo Y., Pan Z., & Heflin J.(2005) LUBM: A benchmark for OWL knowledge base systems. *J. Web Sem.*, 3(2-3):158–182.
4. Kim J. & Park Y. (2015) Scalable OWL-Horst ontology reasoning using Spark. In: *Proc. of BigComp 2015*, pp. 79–86.
5. Liu Z., Feng Z., Zhang X., Wang X., & Rao G.(2016) RORS: Enhanced Rule-based OWL Reasoning on Spark In: *Proc. of APWeb 2016*, pp. 1–4.