ECAI 2012
Luc De Raedt et al. (Eds.)

doi:10.3233/978-1-61499-098-7-933

Reasoning with Fuzzy-$\mathcal{EL}^+$ Ontologies Using MapReduce

Zhangquan Zhou and **Guilin Qi** [1] and **Chang Liu** [2] and **Pascal Hitzler** and **Raghava Mutharaju** [3]

Abstract. Fuzzy extension of Description Logics (DLs) allows the formal representation and handling of fuzzy knowledge. In this paper, we consider fuzzy-$\mathcal{EL}^+$, which is a fuzzy extension of $\mathcal{EL}^+$. We first present revised completion rules for fuzzy-$\mathcal{EL}^+$ that can be handled by MapReduce programs. We then propose an algorithm for scale reasoning with fuzzy-$\mathcal{EL}^+$ ontologies based on MapReduce.

1 INTRODUCTION

The Web Ontology Language OWL has been designed as one of the major standards for formal knowledge representation and automated reasoning in the Semantic Web. OWL 2 EL, which is essentially based on description logic $\mathcal{EL}^{++}$, stands out for its positive complexity results and the sufficient expressive power for many real ontologies, such as the medical ontology SNOMED-CT. However, description logics are not able to represent fuzzy information, which is available in some applications, such as multimedia and bioinformatics. Fuzzy extension of description logics has been proposed to provide more expressive power. One of the challenging problems of fuzzy description logics is reasoning with large scale fuzzy ontologies. Such ontologies can be extracted from different sources, such as multimedia.

Parallel reasoning is an obvious choice to achieve the scalability goal. One of the most successful attempts is WebPIE [5], an efficient inference engine for large ontologies under pD* semantics using MapReduce. This work is further extended in [2] to handle fuzzy knowledge. In [3] a parallel classification algorithm using MapReduce is given for classical $\mathcal{EL}^+$. However, this algorithm is not optimized for implementation and cannot handle reasoning in fuzzy ontologies.

In this paper, we consider a fuzzy extension of $\mathcal{EL}^+$, called fuzzy-$\mathcal{EL}^+$, which is introduced in [4]. Although a polynomial time algorithm is given to classify fuzzy-$\mathcal{EL}^+$ ontologies, it is not implemented and it may not scale to large ontologies. In order to provide scalable reasoning in fuzzy-$\mathcal{EL}^+$, we consider using parallel reasoning techniques based on MapReduce. Since the completion rules for fuzzy-$\mathcal{EL}^+$ cannot be handled by MapReduce programs, we revise some of them and propose a novel algorithm for scale reasoning with fuzzy-$\mathcal{EL}^+$ ontologies based on MapReduce.

2 PRELIMINARIES

A fuzzy language extending the description logic $\mathcal{EL}^+$ is introduced in [4]. Concepts in fuzzy-$\mathcal{EL}^+$ are defined according to the following grammar:

$$C, D ::= \top | A | C \sqcap D | \exists r.C$$

where A ranges over the set of *concept names* (CN) and r over the set of *role names* (RN). A fuzzy-$\mathcal{EL}^+$ ontology is a finite set of *fuzzy general concept inclusions* (F-GCIs) of the form $\langle C \sqsubseteq D, n\rangle$, where $n \in (0, 1]$, and *role inclusions* (RIs) of the form $r_1 \circ, ..., \circ r_k \sqsubseteq s$, where k is a positive integer. Note that the *role inclusions* axioms are not fuzzified in [4]. A polynomial algorithm is given to perform classification of fuzzy-$\mathcal{EL}^+$ ontologies. The algorithm first transforms the given ontology $\mathcal{O}$ into normal form $\mathcal{O}'$, where all concept inclusions are of the form $\langle A_1 \sqcap ... \sqcap A_k \sqsubseteq B, n\rangle$, $\langle A \sqsubseteq \exists r.B, n\rangle$ or $\langle \exists r.B \sqsubseteq A, n\rangle$, and all role inclusions are of the form $r_1 \circ r_2 \sqsubseteq s$ or $r \sqsubseteq s$. The normalization can be done in linear time. In the following, we assume that an input ontology $\mathcal{O}$ is in normal form. The algorithm is formulated by two mappings S and R, where S ranges over subsets of CN×[0, 1] and R over subsets of CN×CN×[0, 1]. Intuitively, $\langle B, n\rangle \in S(A)$ implies $\langle A \sqsubseteq B, n\rangle$ and $\langle A, B, n\rangle \in R(r)$ implies $\langle A \sqsubseteq \exists r.B, n\rangle$. The two mappings are initialized by setting $S(A) = \{\langle A, 1\rangle, \langle \top, 1\rangle\}$ for each class name A, and $R(r) = \emptyset$ for each role name r in the input $\mathcal{O}$. Then the two sets $S(A)$ and $R(r)$ are extended by applying the completion rules in Table 1 until no more rules can be applied.

Table 1. Completion rules for fuzzy-$\mathcal{EL}^+$

R1	If $\langle A_1, n_1\rangle \in S(X), ..., \langle A_l, n_l\rangle \in S(X)$, $\langle A_1 \sqcap ... \sqcap A_l \sqsubseteq B, k\rangle \in \mathcal{O}$ and $\langle B, m\rangle \notin S(X)$, where $m = \min(n_1, ..., n_l, k)$ then $S(X) := S(X) \cup \{\langle B, m\rangle\}$, where $m = \min(n_1, ..., n_l, k)$
R2	If $\langle A, n\rangle \in S(X), \langle A \sqsubseteq \exists r.B, k\rangle \in \mathcal{O}$, and $\langle X, B, m\rangle \notin R(r)$, where $m = \min(n, k)$ then $R(r) := R(r) \cup \{\langle X, B, m\rangle\}$, where $m = \min(n, k)$
R3	If $\langle X, Y, n_1\rangle \in R(r), \langle A, n_2\rangle \in S(Y)$, $\langle \exists r.A \sqsubseteq B, n_3\rangle \in \mathcal{O}$, and $\langle B, m\rangle \notin S(X)$, where $m = \min(n_1, n_2, n_3)$ then $S(X) := S(X) \cup \{\langle B, m\rangle\}$, where $m = \min(n_1, n_2, n_3)$
R4	If $\langle X, Yn\rangle \in R(r), r \sqsubseteq s \in \mathcal{O}$, and $\langle X, Yn\rangle \notin R(s)$ then $R(s) := R(s) \cup \{\langle X, Y, n\rangle\}$
R5	If $\langle X, Y, n_1\rangle \in R(r), \langle Y, Z, n_2\rangle \in R(s), r \circ s \sqsubseteq t \in \mathcal{O}$, and $\langle X, Z, m\rangle \notin R(t)$, where $m = \min(n_1, n_2)$ then $R(t) := R(t) \cup \{\langle X, Z, m\rangle\}$, where $m = \min(n_1, n_2)$

MapReduce is a programming model for parallel processing over huge data sets [1]. A MapReduce task consists of two phases: a map phase and a reduce phase. In map phase a user-defined map function receives a key/value pair and outputs a set of key/value pairs. All pairs sharing the same key are grouped and passed to reduce phase. Then a user-defined reduce function is set up to process the grouped pairs. The grouping procedure between map and reduce phase is called *shuffle* that is the key factor to determine the efficiency of a task. The tradeoff of load overhead, number of tasks and burden over shuffle leads us to design and optimize our algorithms in following work.

[1] University of Southeast, China, email: 220111394, gqi@seu.edu.cn
[2] University of Shanghai Jiaotong, China, email: liuchang@apex.sjtu.edu.cn
[3] University of Wright State, America, email: pascal.hitzler, mutharaju.2@wright.edu

3 A REASONING ALGORITHM FOR Fuzzy-$\mathcal{EL}^+$ USING MAPREDUCE

Since R2 and R4 have only one joint in their preconditions, they can be directly handled by MapReduce programs. The rest of rules R1, R3 and R5 have more than one joints in the preconditions, so they need to be modified. We give revised fuzzy-$\mathcal{EL}^+$ rules in Table 2.

Table 2. Revised fuzzy-$\mathcal{EL}^+$ rules

Key	Completion Rule For MapReduce	
A_1	**R1-1**	If $\langle A_1, n_1\rangle \in S(X)$ and $\langle A_1 \sqcap A_2 \sqsubseteq B, n_2\rangle \in \mathcal{O}$ then $P(X) := P(X) \cup \{\langle A_2, B, m\rangle\}$, where $m = \min(n_1, n_2)$
A	**R1-2**	If $\langle A, n_1\rangle \in S(X)$ and ($\langle A, B, n_2\rangle \in P(X)$ or $\langle A \sqsubseteq B, n_2\rangle \in \mathcal{O}$) then $S(X) := S(X) \cup \{\langle B, m\rangle\}$, where $m = \min(n_1, n_2)$
A	**R2**	If $\langle A, n_1\rangle \in S(X)$ and $\langle A \sqsubseteq \exists r.B, n_2\rangle \in \mathcal{O}$ then $R(r) := R(r) \cup \{\langle X, B, m\rangle\}$, where $m = \min(n_1, n_2)$
r	**R3-1***	If $\langle X, Y, n_1\rangle \in R(r)$ and $\langle \exists r.A \sqsubseteq B, n_2\rangle \in \mathcal{O}$ then $Q(X) := Q(X) \cup \{\langle Y, A, B, m\rangle\}$, where $m = \min(n_1, n_2)$
Y(or A)	**R3-2***	If $\langle A, n_1\rangle \in S(Y)$ and $\langle Y, A, B, n_1\rangle \in Q(X)$ then $S(X) := S(X) \cup \{\langle B, m\rangle\}$, where $m = \min(n_1, n_2)$
r	**R4**	If $\langle X, Y, n\rangle \in R(r)$ and $r \sqsubseteq s \in \mathcal{O}$ then $R(s) := R(s) \cup \{\langle X, Y, n\rangle\}$
Z	**R5**	If $\langle X, Z, n_1\rangle \in R(r)$, $\langle Z, Y, n_2\rangle \in R(s)$ and $r \circ s \sqsubseteq t \in \mathcal{O}$ then $R(t) := R(t) \cup \{\langle X, Y, m\rangle\}$, where $m = \min(n_1, n_2)$

We adopt the mapping P introduced in [3] to split R1 into R1-1 and R1-2. $\langle A, B, n\rangle \in P(X)$ means that if $\langle A, m\rangle \in S(X)$ then $\langle B, k\rangle \in S(X)$, where $k = \min(n, m)$. In other words, $P(X)$ contains the intermediate or incomplete derived information that is used in R1-2 to complete the inference task of R1. Initially, $P(X)$ is set to $\emptyset$ for each X. For R1-1, we should take another normalization step: each axiom of the form $\langle A_1 \sqcap ... \sqcap A_n \sqsubseteq A, k\rangle$ with $n > 2$ is replaced by $\langle A_1 \sqcap A_2 \sqsubseteq N_1, k\rangle$, $\langle N_1 \sqcap A_3 \sqsubseteq N_2, k\rangle$,..., $\langle N_{n-2} \sqcap A_n \sqsubseteq A, k\rangle$, where all $N_i (i = 1, ..., n-2)$ are newly introduced concept names which will be ignored in the final knowledge base and does not change the subsumption hierarchy between concept names of the original ontology. The axiom of the form $\langle A \sqsubseteq B, k\rangle$, that is when $n = 1$, is covered by R1-2 alone.

The modification of R3 given in [3] results in expansion of the original ontology $\mathcal{O}$, which is not expected since the added information are trivial in final results. Inspired by the introduction of mapping P, we introduce a new mapping Q and replace R3 with R3-1* and R3-2* (here we use the symbol $*$ to show the difference with modified rules of R3 given in [3]). This treatment avoids modification of input ontology $\mathcal{O}$. Q records the intermediate information of R3. $\langle Y, A, B, n\rangle \in Q(X)$ means that if $\langle A, m\rangle \in S(Y)$ holds then $\langle B, k\rangle \in S(X)$ should be added into the final knowledge base, where $k = \min(n, m)$. $Q(X)$ is Initially set to $\emptyset$ for each X.

Unlike the work presented in [3], we keep R5 unmodified and just parallelize the axioms of the form $\langle X, Y, n\rangle \in R(r)$ into different nodes and load the axioms of property chain into memory to complete the application of rule R5. This is based on the observation that the number of role inclusion axioms of the form $r \sqsubseteq s$ or $r \circ s \sqsubseteq t$ is much less than that of the concept inclusion axioms in real ontologies such as SNOMED-CT. This treatment helps reduce the numbers of MapReduce tasks.

We further discuss the rationales behind these rules. Rules R2, R4 and R5 are almost unchanged except the preconditions like $\langle B, m\rangle \notin S(X)$ or $\langle X, B, m\rangle \notin R(r)$ are omitted, as they are only used for termination judgment. Since we will consider the termination condition in our reasoning algorithm, there is no difference between these rules in Table 1 and Table 2. Rule R1 (resp. rule R3) is replaced by R1-1 and R1-2 (resp. R3-1* and R3-2*). The outputs of R1-1 (resp. R3-1*) are only used in the precondition of R1-2 (resp. R3-2*), so it does not have any effect on final results. The algorithm terminates when there is no more conclusions obtained from any rule, that is a fix point is reached.

We give our reasoning algorithm based on the revised fuzzy-$\mathcal{EL}^+$ rules. The algorithm first transforms all input axioms to normal forms and initializes S, R, P and Q. The main part of the algorithm consists of two phases:

- Computes the complete *role inclusion closure* (RIC), which stands for the reflexive transitive closure of the axiom $r \sqsubseteq s$ in $\mathcal{O}$. This work can be done in memory.
- Iteratively applies the fuzzy-$\mathcal{EL}^+$ rules until a fix point is reached. a MapReduce task is used to delete the duplicates and get the greatest fuzzy value for an axiom obtained from completion rules.

The application of each rule can be handled by a MapReduce task. In map phase, each axiom which satisfies one of the preconditions of the rule is given as output in form of a key/value pair , where key is concept or role as shown in the left part of Table 2. All axioms having the same key are grouped from different map nodes and passed to one reduce node. The conclusions of the rule can be achieved in reduce phase. Since we can load the axioms of property chain into different nodes, the application of R5 can be done in one MapReduce task. We use RIC in the reduce phases of R2 and R5 to complete the inference task of R4, thus R4 can be omitted from iteration.

4 CONCLUSION

In this paper, we proposed a MapReduce algorithm of classification for large-scale fuzzy-$\mathcal{EL}^+$ ontologies. For this purpose, we revised completion rules for fuzzy-$\mathcal{EL}^+$ that can be handled by MapReduce programs. We introduced a new mapping Q for R3 to keep the input ontology unmodified and this algorithm needs less MapReduce tasks than that of [3] due to the treatments for R5 and R4, which may result in a better performance. Some optimizations are also introduced in our work for the two mappings P and Q. Moreover, our algorithm can handle fuzzy knowledge. Our next step is to implement this algorithm using Hadoop framework for evaluation and practical use.

REFERENCES

[1] Jeffrey Dean and Sanjay Ghemawat, 'MapReduce: Simplified Data Processing on Large Clusters', in *OSDI*, pp. 137–150, (2004).
[2] Chang Liu, Guilin Qi, Haofen Wang, and Yong Yu, 'Large Scale Fuzzy pD* Reasoning Using MapReduce', in *International Semantic Web Conference (1)*, pp. 405–420, (2011).
[3] Raghava Mutharaju, Frederick Maier, and Pascal Hitzler, 'A Mapreduce Algorithm for $\mathcal{EL}^+$', in *Description Logics*, (2010).
[4] Giorgos Stoilos, Giorgos B. Stamou, and Jeff Z. Pan, 'Classifying Fuzzy Subsumption in Fuzzy-$\mathcal{EL}^+$', in *Description Logics*, (2008).
[5] Jacopo Urbani, Spyros Kotoulas, Jason Maassen, Frank van Harmelen, and Henri E. Bal, 'OWL Reasoning with WebPIE: Calculating the Closure of 100 Billion Triples', in *International Semantic Web Conference (1)*, pp. 213–227, (2010).