Large Scale Fuzzy $pD^*$ Reasoning using MapReduce

Chang Liu¹, Guilin Qi², Haofen Wang¹, Yong Yu¹
¹Shanghai Jiao Tong University
²Southeast University, China
Agenda

• Motivation
• Background knowledge
• Challenges and solutions
• Experiment result
• Conclusion and future work
Motivation

- There is a large amount of uncertain semantic data on the web

- We have proposed fuzzy $pD^*$ semantics to represent the uncertain information in the Web
  - No efficient reasoning algorithm for fuzzy $pD^*$ semantics

- MapReduce has been proved to be an efficient framework to do $pD^*$ reasoning

- Can we apply MapReduce framework to deal with large scale fuzzy semantic data?
Background knowledge: fuzzy $pD^*$ semantics

• Fuzzy Logic
  – A fuzzy statement is in form of $\phi[n]$
    • $\phi$ is a statement
    • $n$ is called the fuzzy degree ($n \in [0,1]$)
  – T-norm operator
    • Lukasiewicz Logic
      \[ a \otimes b = \max(a + b - 1, 0) \]
    • Godel Logic
      \[ a \otimes b = \min(a, b) \]
    • Product Logic
      \[ a \otimes b = a \cdot b \]

• Fuzzy RDF triple
  – (Tom, like, pizza)[0.8]
Background knowledge: fuzzy $pD^*$ rules

- **Fuzzy $D^*$ rule**
  - E.g. rule f-rdfs2:
    - $(p, \text{domain}, u)[n], (v, p, w)[m] \Rightarrow (v, \text{type}, u)[n \otimes m]$

- **Fuzzy $P$ rules**
  - E.g. rule f-rdfsp4
    - $(p, \text{type}, \text{TransitiveProperty})[n], (a, p, b)[m], (b, p, c)[k] \Rightarrow (a, p, c)[n \otimes m \otimes k]$

- **Best Degree Bound**
  - $(a, \text{type}, u)[0.5], (a, p, b)[0.9], (p, \text{domain}, u)[1]$
  - Since $(a, p, b)[0.9], (p, \text{domain}, u)[1] \Rightarrow (a, \text{type}, u)[0.9]$
  - The BDB of $(a, \text{type}, u)$ is 0.9
Challenges

• Ordering the rule applications
  – Bad orders will generate more non-BDB fuzzy triples

• The shortest path calculation
  – Some rules essentially calculates the all-pair shortest paths

• Sameas rules
  – Canonical representation technique is not applicable to handle the semantics of vague sameas triples
Ordering the rule applications

Control flow of the reasoning algorithms

Applying fuzzy $D^*$ rules

New fuzzy triples derived?

Applying fuzzy $P$ rules

New fuzzy triples derived?
Ordering the rule applications (cont’d)

- Applying fuzzy $D^*$ rules
  - 1. Property Hierarchy
    - EquivalentProperty rules
  - 2. Domain and range rules
  - 3. Class Hierarchy
    - EquivalentClass rules
  - 4. Other rules
Ordering the rule applications (cont’d)

• Applying fuzzy P rules
  – Non recursive rules
  – Transitive rule
  – Sameas rules
  – hasValue, someValuesFrom and allValuesFrom rules
Shortest path calculation

- Some rules are essentially calculating the shortest path between instances in the fuzzy RDF graph

- Class and property hierarchy rules
  - E.g. rule f-rdfs11,
    \[(u, \text{subClassOf}, v)[n], (v, \text{subClassOf}, w)[m] \Rightarrow (u, \text{subClassOf}, w)[n \otimes m]\]

- Transitive property rules
  - Rule f-rdfp4,
    \[(p, \text{type}, \text{TransitiveProperty})[l], (a, p, b)[n], (b, p, c)[m] \Rightarrow (a, p, c)[n \otimes m \otimes l]\]
Shortest path calculation: Class and Property Hierarchy

• Loading all schema triples into the memory
  – Use edge matrix \( w(u, v) \) to represent there is a fuzzy triple \( (u, \text{subClassOf}, v)[n] \)

• Use a Floyd-Warshall style algorithm to compute the closures of class hierarchy and property hierarchy
  – \( O(|N|^3) \) computational complexity
  – Optimize the algorithm by ignoring the zero \( w(u, v) \)

• EquivalentClass and EquivalentProperty
  – If \( (u, \text{equivalentClass}, v)[n] \) exists, set \( w(u, v) \) to be \( \max(w(u, v), n) \)
  – Emit \( (u, \text{equivalentClass}, v)[w(u, v)] \) and \( (v, \text{equivalentClass}, w)[w(u, v)] \)
Shortest path calculation: Transitive Property

• The whole instance graph is too large to be loaded into memory
  – Employ MapReduce programs to calculate the closure
  – The essential problem is a variant of the all-pairs shortest path calculation

• An iterative algorithm
  – Load the schema triple \((p, \text{type}, \text{TransitiveProperty})[k]\) into memory
  – Compute the join \((a, p, b)[n]\) and \((b, p, c)[m]\)
  – The algorithm halts if no new triples are derived and no BDBs are updated
  – There will be at most \(O(\log N)\) iterations, where \(N\) is the number of all instances
Sameas rules

- Traditional Method
  - Canonical representation
- Drawback
  - Vague sameas triples
    - \((a, \text{sameas}, b)[0.8] (b, \text{sameas}, c)[0.1] (c, \text{sameas}, d)[0.8]\)
    - \((a, \text{range}, r)[0.9] (u, b, v)[0.9] (c, \text{domain}, e)[1] (u', d, v')[0.9]\)
- There is no canonical representation!
  - If we choose \(c\) as the representation
  - the RDF graph will be converted into
    - \((c, \text{range}, r)[0.1] (u, c, v)[0.1] (c, \text{domain}, e)[1] (u', c, v')[0.8]\)
    - The BDB of \((v, \text{type}, r)\) is 0.1
    - However the BDB of \((v, \text{type}, r)\) in the original graph is 0.8
Sameas rules (cont’d)

• Handle sameas triples
  – Use canonical representation to remove all certain sameas triples
  – Calculate the closure for vague sameas triples
  – Revise the MapReduce algorithms for each rule, to consider the sameas triples
Sameas rules (cont’d)

The original MapReduce program for rule f-rdfs2

\[(s, p, o)[n]\] map \{p, (L, s, n)\}

\[(p, \text{domain}, u)[m]\] map \{p, (R, u, m)\}

reducer

emit

\[(s, \text{type}, u)[n \otimes m]\]
Sameas rules (cont’d)

The revised MapReduce program for rule f-rdfs2

\[(p, \text{sameas}, p')[k]\]

\[(s, p, o)[n]\] map \[\{p', (L, s, n \times k)\}\]

\[\{p', (R, u, m)\}\] map \[
\]

 reducers

\[(s, \text{type}, u)[n \times k \times m]\]
Experiment setup

• Dataset
  – fpdLUBM 1000, 2000, 4000, 8000

• Cluster
  – 25 machine with 75 mapper/reducer slots
## Experiment result

### Experimental results of FuzzyPD and WebPIE

<table>
<thead>
<tr>
<th>Number of Universities</th>
<th>Time of FuzzyPD (minutes)</th>
<th>Time of WebPIE (minutes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000</td>
<td>38.8</td>
<td>41.32</td>
</tr>
<tr>
<td>2000</td>
<td>66.97</td>
<td>74.57</td>
</tr>
<tr>
<td>4000</td>
<td>110.40</td>
<td>130.87</td>
</tr>
<tr>
<td>8000</td>
<td>215.48</td>
<td>210.01</td>
</tr>
</tbody>
</table>
## Experiment result (cont’d)

Scalability over number of units

<table>
<thead>
<tr>
<th>Number of units</th>
<th>Time (minutes)</th>
<th>Speedup</th>
</tr>
</thead>
<tbody>
<tr>
<td>128</td>
<td>38.80</td>
<td>4.01</td>
</tr>
<tr>
<td>64</td>
<td>53.15</td>
<td>2.93</td>
</tr>
<tr>
<td>32</td>
<td>91.58</td>
<td>1.70</td>
</tr>
<tr>
<td>16</td>
<td>155.47</td>
<td>1.00</td>
</tr>
</tbody>
</table>
Experiment result (cont’d)

Scalability over number of units

![Graph showing scalability over number of units]
## Experiment result (cont’d)

### Scalability over data volume

<table>
<thead>
<tr>
<th>Number of universities</th>
<th>Input (Mtriples)</th>
<th>Output (Mtriples)</th>
<th>Time (minutes)</th>
<th>Throughput (Ktriples/second)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000</td>
<td>155.51</td>
<td>92.01</td>
<td>38.8</td>
<td>39.52</td>
</tr>
<tr>
<td>2000</td>
<td>310.71</td>
<td>185.97</td>
<td>66.97</td>
<td>46.28</td>
</tr>
<tr>
<td>4000</td>
<td>621.46</td>
<td>380.06</td>
<td>110.40</td>
<td>57.37</td>
</tr>
<tr>
<td>8000</td>
<td>1243.20</td>
<td>792.54</td>
<td>215.50</td>
<td>61.29</td>
</tr>
</tbody>
</table>
Conclusion

• Conclusion
  – This is the first reasoning engine considering fuzzy $pD^*$ semantics
  – Identify the unique challenges to build efficient fuzzy $pD^*$ reasoning engine and work out the solutions for these challenges
  – The experiment results show that our engine is comparable with the state-of-the-art crisp $pD^*$ reasoning engine WebPIE
  – The scalability is good in both the dimensions of machines and data volume

• Future work
  – Extends the current engine to support fuzzy OWL 2 RL semantics
  – Apply the methods to handle other annotations
THANK YOU