Robust Discovery of Positive and Negative Rules in Knowledge-Bases

Paolo Papotti

joint work with S. Ortona (Meltwater) and V. Meduri (ASU)


Lyon – 12 Dec 2017
- RF integrates facts from 1M web sources every day to run analytics
Goal: obtain cleaning programs that

- Effectively detect and fix problems
- Efficiently process large datasets
- Easy to interpret for validation

• Data cleaning rules
Cleaning RF data with Temp FDs

- Nine relations, 4000 manually annotated tuples

<table>
<thead>
<tr>
<th>Relation</th>
<th>N</th>
<th>P</th>
<th>R</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>Company Employees #</td>
<td>24</td>
<td>0.74</td>
<td>0.17</td>
<td>0.27</td>
</tr>
<tr>
<td>Company Meet.</td>
<td>336</td>
<td>0.94</td>
<td>0.5</td>
<td>0.65</td>
</tr>
<tr>
<td>Credit Rating</td>
<td>48</td>
<td>0.6</td>
<td>0.75</td>
<td>0.67</td>
</tr>
<tr>
<td>Employment Change</td>
<td>24</td>
<td>1.0</td>
<td>0.88</td>
<td>0.94</td>
</tr>
<tr>
<td>Natural Disaster</td>
<td>24</td>
<td>0.8</td>
<td>0.5</td>
<td>0.62</td>
</tr>
<tr>
<td>Person Travel</td>
<td>48</td>
<td>0.61</td>
<td>0.82</td>
<td>0.7</td>
</tr>
<tr>
<td>Political Endorsement</td>
<td>48</td>
<td>1.0</td>
<td>0.59</td>
<td>0.74</td>
</tr>
<tr>
<td>Product Recall</td>
<td>177</td>
<td>0.9</td>
<td>0.9</td>
<td>0.9</td>
</tr>
<tr>
<td>Voting Result</td>
<td>24</td>
<td>1.0</td>
<td>0.6</td>
<td>0.75</td>
</tr>
</tbody>
</table>

0.84 0.54

[Abedjan et al, 2015]
Relational Data

<table>
<thead>
<tr>
<th>Name</th>
<th>Location</th>
<th>Timestamp</th>
</tr>
</thead>
<tbody>
<tr>
<td>B. Obama</td>
<td>Rome</td>
<td>8.00 Feb 1</td>
</tr>
<tr>
<td>B. Obama</td>
<td>Rome</td>
<td>8.15 Feb 1</td>
</tr>
<tr>
<td>B. Obama</td>
<td>NYC</td>
<td>11.00 Feb 1</td>
</tr>
<tr>
<td>B. Obama</td>
<td>Paris</td>
<td>18.00 Feb 1</td>
</tr>
<tr>
<td>B. Obama</td>
<td>Paris</td>
<td>18.49 Feb 1</td>
</tr>
</tbody>
</table>

Knowledge Bases

WalMart, KPMG, Amadeus, …
RDF KBs

<Barack Obama> <spouse> <Michelle Obama> .
<Barack Obama> <birthDate> "1961-08-04" .
<Michelle Obama> <birthPlace> <Illinois> .

<table>
<thead>
<tr>
<th>Name</th>
<th># Entity types</th>
<th># Entity instances</th>
<th># Relation types</th>
<th># Confident facts (relation instances)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Knowledge Vault (KV)</td>
<td>1100</td>
<td>45M</td>
<td>4469</td>
<td>271M</td>
</tr>
<tr>
<td>DeepDive [21]</td>
<td>4</td>
<td>2.7M</td>
<td>34</td>
<td>7M</td>
</tr>
<tr>
<td>NELL [6]</td>
<td>271</td>
<td>5.19M</td>
<td>306</td>
<td>0.435M</td>
</tr>
<tr>
<td>PROSPERA [20]</td>
<td>11</td>
<td>N/A</td>
<td>14</td>
<td>0.1M</td>
</tr>
<tr>
<td>YAGO2 [16]</td>
<td>350,000</td>
<td>9.8M</td>
<td>100</td>
<td>4M</td>
</tr>
<tr>
<td>Freebase [4]</td>
<td>1,500</td>
<td>40M</td>
<td>35,000</td>
<td>637M</td>
</tr>
<tr>
<td>Knowledge Graph (KG)</td>
<td>1,500</td>
<td>570M</td>
<td>35,000</td>
<td>18,000M</td>
</tr>
</tbody>
</table>

Table 1: Comparison of knowledge bases [9]. KV, DeepDive, NELL, and PROSPERA rely solely on extraction, Freebase and KG rely on human curation and structured sources, and YAGO2 uses both strategies. Confident facts means with a probability of being true at or above 0.9.

[Dong and Srivastava, 2015]
"ML did not reach the required 92% precision threshold, [with 20,459 rules] precision consistently in the range 92-93%"

[Suganthal et al, 2015]

"[to build Kosmix KB —> WalmartLabs KB] analysts have written several thousands of rules"

[Deshpande et al, 2013]
Data Quality issues in KBs

Incomplete data
DBPedia: 1.7M Person, birth dates reported only for 1M

Errors
Yago: 9K cases where child is born before parent
Horn Rules Discovery

Body (conjunction of atoms) \[ \Rightarrow \] Head (atom)

\[
\text{child}(x,z) \land \text{child}(y,z) \Rightarrow \text{spouse}(x,y)
\]

Atom = predicate from KB

AMIE:

- memory based
- language bias
  - max body size 2, equality comparison only, no literals
- “positive” rules \[ \Rightarrow \] Incomplete data only

[Galarraga et al, 2013]
Negative Rules

Positive Rule:
child(x,z) \land child(y,z) \Rightarrow spouse(x,y)

Negative Rule:
birthDate(x,w) \land birthDate(y,z) \land w \leq z \Rightarrow \neg child(y,x)

Atom = \{ - positive/negative predicate from KB
- value comparison (<, \leq, =, >, \geq)\}

Constants allowed for conditional rules
(e.g., rule applies only in US)
Negative Rules

Positive Rule:
\[
\text{child}(x,z) \land \text{child}(y,z) \implies \text{spouse}(x,y)
\]

Negative Rule:
\[
\text{birthDate}(x,w) \land \text{birthDate}(y,z) \land w \leq z \land \text{child}(y,x) \implies \text{error}
\]

Denial constraints [Chu et al, 2013]
Problem Definition

Given a pair of entities \((a,b)\) in the KB, rule \(r\): body \(\rightarrow p(x,y)\) covers \((a,b)\) if there exists an instantiation in KB such that the body of \(r\) holds

Input:
- target predicate \(p\) \((spouse)\)
- generation set \(G\) \((examples of married couples)\)
- validation set \(V\) \((examples of unmarried couples)\)

Output:
a set of positive \((negative)\) rules covering all elements in \(G\), and none of \(V\) \((none of G, all of V)\)
Exact Solution May Fail

\[ \text{child}(x,z) \land \text{child}(y,z) \Rightarrow \text{spouse}(x,y) \]
Exact Solution May Fail

? ⇒ spouse(x,y)

+ <Barack Obama> <spouse> <Michelle Obama> .
+ <Beyonce’> <spouse> <Jay-Z> .
- <Tom Cruise> <spouse> <Serena Williams> .
- <Leonardo Da Vinci> <spouse> <Hillary Clinton> .

Miss valid rules because do not hold on all examples in G
- rule does not apply for all (couples w/out children)
- for missing values in the data - OWA (missing children)
- for mistakes in the data (wrong parent)

→ failure or overfitting
Exact Solution May Fail

? ⇒ ¬spouse(y,x)

- <Barack Obama> <spouse> <Michelle Obama> .
- <Beyonce’> <spouse> <Jay-Z> .
+ <Tom Cruise> <spouse> <Serena Williams> .
+ <Leonardo Da Vinci> <spouse> <Hillary Clinton> .

Miss valid rules because do not hold on all examples in G

- for missing values in the data - OWA (missing DOB)
- for mistakes in the data (wrong DOB)

—> failure or overfitting
Problem Revised

Input:
- target predicate $p$ (spouse)
- generation set $G$ (examples of married couples)
- validation set $V$ (examples of not married couples)

Output (positive rules):
weighted set cover: $G$ universe of elements, each rule is an element of $S$, and $V$ used for computing weights

$$w(r) = \alpha \cdot (1 - \frac{|C_r(G)|}{|G|}) + \beta \cdot \left( \frac{|C_r(V)|}{|U_r(V)|} \right)$$

max coverage $G$  
min coverage $V$
Generation and Validation Sets

- Generation set $G$: straightforward from KB (all people connected by a spouse predicate)

- Validation Set $V$: all negative example of spouse relationship
Negative Examples

- A true negative, only if the entities have no missing relationships
  - Cannot assume that what is not in KB is false (OWA)
- Naïve creation method: Cartesian product very small fraction of pairs are semantically related → miss meaningful paths!
  - Always true for positive examples: they have target predicate in common
- New method using Local-Closed World Assumption
Negative Examples

- For predicate \textit{child}, negative example is a pair \(x, y\) s.t.
  - \(x\) has some children in the KB who are not \(y\), or \(y\) is the child of someone who is not \(x\) (LCWA on subject and predicate)
  - \(x, y\) are connected via a predicate that is different from the target predicate (semantically related)
Generation and Validation Sets

- Generation set $G$: straightforward from KB (all people connected by a spouse predicate)

- Validation Set $V$: all pairs of people \((x,y)\) where either \(x\) or \(y\) are in a spouse relationship with someone else

- At least another predicate between \((x,y)\): crucial when $V$ is the generation set, size comparable to $G$

Local Closed World Assumption

⇒ M. Obama does not have another spouse
Naive Rule Generation

- Translate rules to graph traversal

\[
\text{child}(x,z) \land \text{child}(y,z) \Rightarrow \text{spouse}(x,y)
\]

- Generate all possible rules (body size 3) from G, compute weights from G\&V, compute weighted set cover
Greedy Algorithm

- Greedy traversal: at each iteration follow the **most promising path** according to marginal weight
- Build graph **incrementally**: query the KB only when needed to follow a given path
- **Prune** paths that do not lead to good solutions

Advantages:
- A* guarantees optimal if estimation is admissible
- No need to generate all possible rules
- Load in memory only the needed portion of the graph
  - Lexical values: more expressive rules
- Running time in seconds/minutes
Experiments

Java with any SPARQL endpoint (Virtuoso)
i5 CPU at 2.80GHz and 16GB RAM

<table>
<thead>
<tr>
<th>KB</th>
<th>Version</th>
<th>Size</th>
<th>#Triples</th>
<th>#Predicates</th>
</tr>
</thead>
<tbody>
<tr>
<td>DBPEDIA</td>
<td>3.7</td>
<td>10.06GB</td>
<td>68,364,605</td>
<td>1,424</td>
</tr>
<tr>
<td>YAGO 3</td>
<td>3.0.2</td>
<td>7.82GB</td>
<td>88,360,244</td>
<td>74</td>
</tr>
<tr>
<td>WIKIDATA</td>
<td>20160229</td>
<td>12.32GB</td>
<td>272,129,814</td>
<td>4,108</td>
</tr>
</tbody>
</table>

5 most popular predicates for every KB
Output triples manually checked (30) for every rule

Experiments

Positive Rules: new triples

- notableWork(y,x) ⇒ creator(x,y) (Wikidata)
- hasChild(z,y) ∧ isMarriedTo(x,z) ⇒ hasChild(x,y) (Yago)

Negative Rules: erroneous triples

- foundingYear(x,z) ∧ birthYear(y,w) ∧ (z≤w) ⇒ ¬ founder(x,y) (34 errors DBPedia)
- isMarriedTo(x,y) ⇒ ¬ hasChild(x,y) (200 errors Yago)
Experiments

### TABLE II. RuDiK Positive Rules Accuracy.

<table>
<thead>
<tr>
<th>KB</th>
<th>Avg. Run Time</th>
<th>Avg. Precision over Predicates with Rules (All)</th>
<th># Labeled Triples</th>
</tr>
</thead>
<tbody>
<tr>
<td>DBPEDIA</td>
<td>35min</td>
<td>97.14% (63.99%)</td>
<td>139</td>
</tr>
<tr>
<td>YAGO 3</td>
<td>59min</td>
<td>84.44% (62.86%)</td>
<td>150</td>
</tr>
<tr>
<td>WIKIDATA</td>
<td>141min</td>
<td>98.95% (73.33%)</td>
<td>180</td>
</tr>
</tbody>
</table>

### TABLE III. RuDiK Negative Rules Accuracy.

<table>
<thead>
<tr>
<th>KB</th>
<th>Avg. Run Time</th>
<th># Pot. Errors</th>
<th>Precision</th>
</tr>
</thead>
<tbody>
<tr>
<td>DBPEDIA</td>
<td>19min</td>
<td>499 (84)</td>
<td>92.38%</td>
</tr>
<tr>
<td>YAGO 3</td>
<td>10min</td>
<td>2,237 (90)</td>
<td>90.61%</td>
</tr>
<tr>
<td>WIKIDATA</td>
<td>65min</td>
<td>1,776 (105)</td>
<td>73.99%</td>
</tr>
</tbody>
</table>
AMIE as baseline

<table>
<thead>
<tr>
<th>KB</th>
<th>Size</th>
<th>#Triples</th>
<th>#Predicates</th>
<th>#rdf:type</th>
</tr>
</thead>
<tbody>
<tr>
<td>DBPEDIA</td>
<td>551M</td>
<td>7M</td>
<td>10,342</td>
<td>22.2M</td>
</tr>
<tr>
<td>YAGO 2</td>
<td>48M</td>
<td>948.3K</td>
<td>38</td>
<td>77.9M</td>
</tr>
</tbody>
</table>

Fig. 3. Accuracy for new facts identified by executing rules in descending AMIE’s score on YAGO 2 (no literals).

[Galarraga et al, 2013]
AMIE as baseline

- Modified KBs to use AMIE for negative rule discovery
- Added notSpouse predicate for each negative example

<table>
<thead>
<tr>
<th>KB</th>
<th>AMIE</th>
<th>RuDiK (no literals)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td># Errors</td>
<td>Precision</td>
</tr>
<tr>
<td>DBPEDIA</td>
<td>457 (157)</td>
<td>38.85%</td>
</tr>
<tr>
<td>YAGO 2</td>
<td>633 (100)</td>
<td>48.81%</td>
</tr>
</tbody>
</table>

[Galarraga et al, 2013]
Directions

- Rule discovery combined with other signals
- How to involve users in monitoring/evolution
- Applications beyond error detection
Robust Discovery of Positive and Negative Rules in Knowledge-Bases

Relational Data

<table>
<thead>
<tr>
<th>Name</th>
<th>Location</th>
<th>Timestamp</th>
</tr>
</thead>
<tbody>
<tr>
<td>B. Obama</td>
<td>Rome</td>
<td>8.00 Feb 1</td>
</tr>
<tr>
<td>B. Obama</td>
<td>Rome</td>
<td>8.15 Feb 1</td>
</tr>
<tr>
<td>B. Obama</td>
<td>NYC</td>
<td>11.00 Feb 1</td>
</tr>
<tr>
<td>B. Obama</td>
<td>Paris</td>
<td>18.00 Feb 1</td>
</tr>
<tr>
<td>B. Obama</td>
<td>Paris</td>
<td>18.49 Feb 1</td>
</tr>
</tbody>
</table>

Knowledge Bases

Paolo Papotti  
EURECOM