Rule Mining in Knowledge Bases

Luis Galárraga

Télécom ParisTech, DBWeb
September 29th, 2016
Overview

Rule Mining in Knowledge Bases

citizenOf(z, y), hasChild(z, x) \Rightarrow \text{citizenOf}(x, y)

- Knowledge Bases
- Rule Mining
  - Challenges
  - The AMIE system
  - Experimental evaluation

Applications

- Semantifying wikilinks
- Schema alignment
- Canonicalization of open KBs
- Prediction of completeness
Rule Mining in Knowledge Bases
Rule Mining in Knowledge Bases
Knowledge Bases (KBs)
Knowledge Bases (KBs)

Barack Obama

Malia

Sasha

USA

Politician

Person

hasChild(Barack Obama, Malia)

hasChild(Barack Obama, Sasha)

...
Knowledge Bases (KBs)

- hasChild(Barack Obama, Malia)
- hasChild(Barack Obama, Sasha)

Taxonomy

Person -> subclassOf

Politician

USA

citizenOf

Barack Obama

hasChild

Malia

Sasha

citizenOf
KBs in action
KBs in action
Some popular KBs
Rule Mining in Knowledge Bases
Rule Mining in KBs

Barack Obama

Malia

Sasha

United States
Rule Mining in KBs

\[
citizenOf(z, y), \text{hasChild}(z, x) \Rightarrow citizenOf(x, y)
\]
Rule Mining in KBs

citizenOf(z, y), hasChild(z, x) ⇒ citizenOf(x, y)

Body

Head
Applications of Rule Mining

- Fact prediction
Applications of Rule Mining

- Fact prediction

\[
\text{citizenOf}(z, y), \; \text{hasChild}(z, x) \Rightarrow \text{citizenOf}(x, y)
\]
Applications of Rule Mining

- Fact prediction

\[ \text{citizenOf}(z, y), \text{hasChild}(z, x) \Rightarrow \text{citizenOf}(x, y) \]
Applications of Rule Mining

- Fact prediction
- Domain description
Applications of Rule Mining

- Fact prediction
- Domain description
  - Finding trends in KBs
Applications of Rule Mining

- Fact prediction
- Domain description
  - Finding trends in KBs

\[ \text{in}(c, \text{Europe}), \text{president}(x, c) \Rightarrow \text{male}(x) \ [80\%] \]
Applications of Rule Mining

- Fact prediction
- Domain description
  - Finding trends in KBs
- Data engineering and maintenance
Applications of Rule Mining

- Fact prediction
- Domain description
  - Finding trends in KBs
- Data engineering and maintenance
  - Schema mining
Applications of Rule Mining

- Fact prediction
- Domain description
  - Finding trends in KBs
- Data engineering and maintenance
  - Schema mining

\[
\text{marriedTo}(x, y) \Rightarrow \text{marriedTo}(y, x) \\
\text{livesIn}(x, y) \Rightarrow \text{type}(x, \text{Person}) \\
\text{livesIn}(x, y) \Rightarrow \text{type}(y, \text{City})
\]
Applications of Rule Mining

- Fact prediction
- Domain description
  - Finding trends in KBs
- Data engineering and maintenance
  - Schema mining
  - Data correction
Applications of Rule Mining

- Fact prediction
- Domain description
  - Finding trends in KBs
- Data engineering and maintenance
  - Schema mining
  - Data correction
    \[ \text{hasCapital}(x, y) \Rightarrow \text{in}(y, x) \]
Applications of Rule Mining

- Fact prediction
- Domain description
  - Finding trends in KBs
- Data engineering and maintenance
  - Schema mining
  - Data correction

\[ \text{hasCapital}(x, y) \Rightarrow \text{in}(y, x) \]
Applications of Rule Mining

- Fact prediction
- Domain description

Goal: Mine rules that draw concrete and correct conclusions

- Data correction
Challenges
Generate counter-evidence

Counter-examples are required to evaluate the quality of rules
Generate counter-evidence

Counter-examples are required to evaluate the quality of rules

\[ \text{directed}(x, z), \text{actor}(z, y) \Rightarrow \text{partner}(x, y) \]
Generate counter-evidence

Counter-examples are required to evaluate the quality of rules

directed(x, z), actor(z, y) ⇒ partner(x, y)
Generate counter-evidence

Counter-examples are required to evaluate the quality of rules

\[
\text{directed}(x, z), \text{actor}(z, y) \Rightarrow \text{partner}(x, y)
\]
Generate counter-evidence

Counter-examples are required to evaluate the quality of rules

directed(x, z), actor(z, y) ⇒ partner(x, y)
KBs do not store negative evidence
How to generate counter-evidence?

- Closed World Assumption (CWA)

*directed(x, z), actor(z, y) ⇒ partner(x, y)*
How to generate counter-evidence?

• Closed World Assumption (CWA)
  • Missing predictions are used as counter-evidence

\[ \text{directed}(x, z), \text{actor}(z, y) \Rightarrow \text{partner}(x, y) \]
How to generate counter-evidence?

- Closed World Assumption (CWA)
  - Missing predictions are used as counter-evidence

\[
directed(x, z), \text{actor}(z, y) \Rightarrow \text{partner}(x, y)
\]
How to generate counter-evidence?

- Closed World Assumption (CWA)
  - Missing predictions are used as counter-evidence

\[
\text{directed}(x, z), \text{actor}(z, y) \Rightarrow \text{partner}(x, y)
\]
How to generate counter-evidence?

- Closed World Assumption (CWA)
  - Missing predictions are used as counter-evidence

$$\text{conf}_{\text{std}}(B \Rightarrow r(x, y)) = \frac{\text{# predictions in KB}}{\text{# all predictions}}$$

```
directed(x, z), actor(z, y) \Rightarrow \text{partner}(x, y)
```
How to generate counter-evidence?

- Closed World Assumption (CWA)
  - Missing predictions are used as counter-evidence

\[
\text{conf}_{\text{std}}(B \Rightarrow r(x, y)) = \frac{\text{support}(B \Rightarrow r(x, y))}{\#(x, y) : \exists z_1, \ldots, z_n : B}
\]

directed(x, z), actor(z, y) \Rightarrow \text{partner}(x, y)
How to generate counter-evidence?

- Closed World Assumption (CWA)
  - It is too restrictive most of the times

\[
\text{citizenOf}(z, y), \text{hasChild}(z, x) \Rightarrow \text{citizenOf}(x, y)
\]
How to generate counter-evidence?

- Closed World Assumption (CWA)
  - It is too restrictive most of the times

\[\text{citizenOf}(z, y), \text{hasChild}(z, x) \Rightarrow \text{citizenOf}(x, y)\]
How to generate counter-evidence?

- Closed World Assumption (CWA)
  - It is too restrictive most of the times

\[
\text{citizenOf}(z, y), \text{hasChild}(z, x) \Rightarrow \text{citizenOf}(x, y)
\]
How to generate counter-evidence?

- Closed World Assumption (CWA)
  - It is too restrictive most of the times

\[ \text{CWA assumes } \text{Thomas does not have a nationality} \]

\[ \text{citizenOf}(z, y), \text{hasChild}(z, x) \Rightarrow \text{citizenOf}(x, y) \]
How to generate counter-evidence?

- Closed World Assumption (CWA)
  - It is too restrictive most of the times

KBs operate under the **Open World Assumption**

\[
\text{citizenOf}(z, y), \text{hasChild}(z, x) \Rightarrow \text{citizenOf}(x, y)
\]
How to generate counter-evidence?

- Partial Completeness Assumption (PCA)

$$\text{citizenOf}(z, y), \text{hasChild}(z, x) \Rightarrow \text{citizenOf}(x, y)$$
How to generate counter-evidence?

- **Partial Completeness Assumption (PCA)**
  - If we know at least one object, we know them all

\[
\text{citizenOf}(z, y), \text{hasChild}(z, x) \Rightarrow \text{citizenOf}(x, y)
\]
How to generate counter-evidence?

- Partial Completeness Assumption (PCA)
  - If we know at least one object, we know them all

\[
\text{citizenOf}(z, y), \text{hasChild}(z, x) \Rightarrow \text{citizenOf}(x, y)
\]
How to generate counter-evidence?

- Partial Completeness Assumption (PCA)
  - If we know at least one object, we know them all

\[
\text{citizenOf}(z, y), \text{hasChild}(z, x) \Rightarrow \text{citizenOf}(x, y)
\]
How to generate counter-evidence?

- Partial Completeness Assumption (PCA)
  - If we know at least one object, we know them all

\[
citizenOf(z, y), \text{ hasChild}(z, x) \Rightarrow citizenOf(x, y)
\]
How to generate counter-evidence?

- Partial Completeness Assumption (PCA)
  - If we know at least one object, we know them all

$$\text{citizenOf}(z, y), \text{hasChild}(z, x) \Rightarrow \text{citizenOf}(x, y)$$
How to generate counter-evidence?

- Partial Completeness Assumption (PCA)
  - If we know at least one object, we know them all

\[
\text{conf}_{\text{pca}}(B \Rightarrow r(x, y)) = \frac{\# \text{ positives}}{\# \text{ positives} + \# \text{ PCA negatives}}
\]

\[
\text{citizenOf}(z, y), \text{hasChild}(z, x) \Rightarrow \text{citizenOf}(x, y)
\]
PCA Confidence

\[
\text{conf}_{\text{pca}}(B \Rightarrow r(x, y)) = \frac{\# \text{ positives}}{\# \text{ positives} + \# \text{ PCA negatives}}
\]

citizenOf(y, z), hasChild(y, x) \Rightarrow citizenOf(x, z)
PCA Confidence

\[ \text{conf}_{\text{pca}}(B \Rightarrow r(x, y)) = \frac{\# \text{ positives}}{\# \text{ positives} + \# \text{ PCA negatives}} = \frac{1}{1} \]

citizenOf(y, z), hasChild(y, x) \Rightarrow citizenOf(x, z)
PCA Confidence

\[
\text{conf}_{\text{pca}}(B \Rightarrow r(x, y)) = \frac{\# \text{ positives}}{\# \text{ positives} + \# \text{ PCA negatives}} = \frac{1}{1}
\]

citizenOf(y, z), hasChild(y, x) \Rightarrow citizenOf(x, z)
PCA Confidence

\[
\text{conf}_{\text{pca}}(B \Rightarrow r(x, y)) = \frac{\# \text{ positives}}{\# \text{ positives} + \# \text{ PCA negatives}} \times \frac{1}{1}
\]

\[\text{citizenOf}(y, z), \text{hasChild}(y, x) \Rightarrow \text{citizenOf}(x, z)\]
PCA Confidence

\[\text{conf}_{\text{pca}}(B \Rightarrow r(x, y)) = \frac{\# \text{ positives}}{\# \text{ positives} + \# \text{ PCA negatives}} \]

\[\text{citizenOf}(y, z), \text{hasChild}(y, x) \Rightarrow \text{citizenOf}(x, z)\]
PCA Confidence

citizenOf(y, z), hasChild(y, x) \Rightarrow citizenOf(x, z)

\[
\text{conf}_{\text{pca}}(B \Rightarrow r(x, y)) = \frac{\text{support}(B \Rightarrow r(x, y))}{\#(x, y) : \exists z_1, \ldots, z_n, y' : B \land r(x, y')} \frac{1}{2}
\]
KBs are large

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Facts</th>
<th>Entities</th>
</tr>
</thead>
<tbody>
<tr>
<td>YAGO</td>
<td>120M</td>
<td>10M</td>
</tr>
<tr>
<td>Dbpedia</td>
<td>6.9B</td>
<td>38M</td>
</tr>
<tr>
<td>Wikidata</td>
<td>100M</td>
<td>20M</td>
</tr>
</tbody>
</table>

State-of-the-art approaches do not scale

<table>
<thead>
<tr>
<th>Dataset</th>
<th># facts</th>
<th>WARMR</th>
<th>ALEPH</th>
</tr>
</thead>
<tbody>
<tr>
<td>YAGO core</td>
<td>1M</td>
<td>-</td>
<td>5s to 1d</td>
</tr>
<tr>
<td>YAGO (sample)</td>
<td>47K</td>
<td>18h</td>
<td>0.05s to 1d</td>
</tr>
</tbody>
</table>
How to mine rules efficiently?
How to mine rules efficiently?

- Start with all possible rules of the form $\Rightarrow r(x,y)$
- Refine the rules iteratively by means of mining operators:
  - Add dangling atom ($O_D$)
  - Add closing atom ($O_C$)
  - Add instantiated atom ($O_I$)
How to mine rules efficiently?

citizenOf(z, y), hasChild(z, x) ⇒ citizenOf(x, y)

⇒ citizenOf(x, y)
How to mine rules efficiently?

\[
\text{citizenOf}(z, y), \hspace{1em} \text{hasChild}(z, x) \Rightarrow \text{citizenOf}(x, y)
\]

\[
\Rightarrow \text{citizenOf}(x, y)
\]

Add dangling atom

\[
?r(z, x) \Rightarrow \text{citizenOf}(x, y)
\]
How to mine rules efficiently?

```
citizenOf(z, y), hasChild(z, x) \Rightarrow citizenOf(x, y)
⇒ citizenOf(x, y)
⇒ citizenOf(x, y)

Add dangling atom

?r(z, x) ⇒ citizenOf(x, y)
```

\[ \text{hasChild}\]
\[ \text{influences}\]
\[ \text{....}\]
How to mine rules efficiently?

\[
\text{citizenOf}(z, y), \text{hasChild}(z, x) \Rightarrow \text{citizenOf}(x, y) \\
\Rightarrow \text{citizenOf}(x, y) \\
\Rightarrow \text{citizenOf}(x, y)
\]

Add dangling atom

\[?r(z, x) \Rightarrow \text{citizenOf}(x, y)\]
How to mine rules efficiently?

citizenOf(z, y), hasChild(z, x) ⇒ citizenOf(x, y)

⇒ citizenOf(x, y)

?r(z, x) ⇒ citizenOf(x, y)

hasChild(z, x) ⇒ citizenOf(x, y)

Add dangling atom
How to mine rules efficiently?

citizenOf(z, y), hasChild(z, x) ⇒ citizenOf(x, y)

⇒ citizenOf(x, y)

Add dangling atom

?r(z, x) ⇒ citizenOf(x, y)

hasChild(z, x) ⇒ citizenOf(x, y)

Add closing atom

?r(z, y) hasChild(z, x) ⇒ citizenOf(x, y)

citizenOf
livesIn
...

67
How to mine rules efficiently?

\[ \text{citizenOf}(z, y), \text{hasChild}(z, x) \Rightarrow \text{citizenOf}(x, y) \]

\[ \Rightarrow \text{citizenOf}(x, y) \]

Add dangling atom

\[ ?r(z, x) \Rightarrow \text{citizenOf}(x, y) \]

\[ \text{hasChild}(z, x) \Rightarrow \text{citizenOf}(x, y) \]

Add closing atom

\[ \Rightarrow \text{citizenOf}(x, y) \]

\[ ?r(z, y) \Rightarrow \text{citizenOf}(x, y) \]

\[ \text{hasChild}(z, x) \Rightarrow \text{citizenOf}(x, y) \]

\[ \Rightarrow \text{citizenOf}(x, y) \]

\[ \text{citizenOf} \]

\[ \text{livesIn} \]

...
How to mine rules efficiently?

citizenOf(z, y), hasChild(z, x) ⇒ citizenOf(x, y)

⇒ citizenOf(x, y)

Add dangling atom

?r(z, x) ⇒ citizenOf(x, y)

hasChild(z, x) ⇒ citizenOf(x, y)

Add closing atom

?r(z, y) hasChild(z, x) ⇒ citizenOf(x, y)

citizenOf(z, y), hasChild(z, x) ⇒ citizenOf(x, y)
How to mine rules efficiently?

citizenOf(z, y), hasChild(z, x) ⇒ citizenOf(x, y)

⇒ citizenOf(x, y)

Add dangling atom

?r(z, x) ⇒ citizenOf(x, y)

hasChild(z, x) ⇒ citizenOf(x, y)

Add closing atom

?r(z, y) hasChild(z, x) ⇒ citizenOf(x, y)

citizenOf(z, y), hasChild(z, x) ⇒ citizenOf(x, y)

Add instantiated atom adds atoms of the form ?r(x, C) where C is a constant, e.g., livesIn(x, USA)
How to mine rules efficiently?

- Prune the search space
How to mine rules efficiently?

- Prune the search space
  - Using monotonic definition of support and a threshold

$$\text{support}(B \Rightarrow r(x, y)) = \#(x, y) : \exists z_1, \ldots, z_n : B \land r(x, y)$$
How to mine rules efficiently?

- Prune the search space
  - Using monotonic definition of support and a threshold
    \[
    \text{support}(B \Rightarrow r(x, y)) = \#(x, y) : \exists z_1,...z_n : B \land r(x, y)
    \]
    \[
    \text{Support} = 3 \quad \text{citizenOf}(z, y), \text{hasChild}(z, x) \Rightarrow \text{citizenOf}(x, y)
    \]
How to mine rules efficiently?

• Prune the search space
  • Using monotonic definition of support and a threshold
    \[
    \text{support}(B \Rightarrow r(x, y)) = \#(x, y) : \exists z_1, \ldots, z_n: B \land r(x, y)
    \]

\text{Support} = 3 \quad \text{citizenOf}(z, y), \text{hasChild}(z, x) \Rightarrow \text{citizenOf}(x, y)

\text{bornIn}(z, y), \text{citizenOf}(z, y), \text{hasChild}(z, x) \Rightarrow \text{citizenOf}(x, y)

\text{Support} = 2
How to mine rules efficiently?

• Apply a language bias that complies with our goal
How to mine rules efficiently?

- Apply a language bias that complies with our goal
  - Goal: rules that make correct and concrete predictions
How to mine rules efficiently?

• Apply a language bias that complies with our goal
  • Goal: rules that make correct and concrete predictions
  • Avoid existentially quantified conclusions

\[
\text{citizenOf}(z, y), \text{hasChild}(z, x) \Rightarrow \text{citizenOf}(x, w)
\]
How to mine rules efficiently?

- Apply a language bias that complies with our goal
  - Goal: rules that make correct and concrete predictions
  - Avoid existentially quantified conclusions

\[
\text{citizenOf}(z, y), \text{hasChild}(z, x) \implies \text{citizenOf}(x, w)
\]
How to mine rules efficiently?

- Apply a language bias that complies with our goal
  - Goal: rules that make correct and concrete predictions
  - Avoid existentially quantified conclusions

\[ \text{citizenOf}(z, y), \text{hasChild}(z, x) \Rightarrow \exists w : \text{citizenOf}(x, w) \]
How to mine rules efficiently?

• Apply a language bias that complies with our goal
  • Goal: rules that make correct and concrete predictions
  • Avoid existentially quantified conclusions
    \[
    \text{citizenOf}(z, y), \text{hasChild}(z, x) \Rightarrow \exists w : \text{citizenOf}(x, w)
    \]
  • Focus on \textbf{closed} Horn rules
How to mine rules efficiently?

- Apply a language bias that complies with our goal
  - Goal: rules that make correct and concrete predictions
  - Avoid existentially quantified conclusions
  - Focus on **closed** Horn rules

```plaintext
\text{citizenOf}(z, y), \text{hasChild}(z, x) \Rightarrow \exists w : \text{citizenOf}(x, w)
\Rightarrow \text{citizenOf}(x, y)
```

![Diagram showing relationships between x, z, y, and the predicates citizenOf and hasChild]
How to mine rules efficiently?

- Do not specialize rules with 100% confidence
How to mine rules efficiently?

- Do not specialize rules with 100% confidence
- Use efficient confidence approximation
How to mine rules efficiently?

- Do not specialize rules with 100% confidence
- Use efficient confidence approximation
  - To discard rules with low confidence in advance
Confidence approximation

- Bad rules make a lot of false predictions per entity
Confidence approximation

- Bad rules make a lot of false predictions per entity

\[ \text{director}(x, z), \text{actor}(z, y) \Rightarrow \text{partner}(x, y) \]
Confidence approximation

- Bad rules make a lot of false predictions per entity
- Director is partnered with all actors of his movies

\[
director(x, z), \ actor(z, y) \Rightarrow \text{partner}(x, y)
\]
Confidence approximation

- Bad rules make a lot of false predictions per entity
  - They are counted in the denominator of the confidence

\[ \text{conf}_{\text{pca}}(B \Rightarrow r(x, y)) = \frac{\text{support}(B \Rightarrow r(x, y))}{\#(x, y) : \exists z_1, \ldots, z_n, y' : B \land r(x, y')} \]

director(x, z), actor(z, y) \Rightarrow partner(x, y)
Confidence approximation

- Bad rules make a lot of false predictions per entity
- They are counted in the denominator of the confidence

$$\text{director}(x, z), \text{actor}(z, y) \Rightarrow \text{partner}(x, y)$$

$$\text{conf}_{\text{pca}}(B \Rightarrow r(x, y)) = \frac{\text{support}(B \Rightarrow r(x, y))}{\#(x, y) : \text{director}(x, z) \land \text{actor}(z, y) \land \text{partner}(x, y') }$$
Confidence approximation

- Use statistics to estimate $\text{conf}_{\text{pca}}$ denominator

\[
\text{director}(x, z), \text{actor}(z, y) \Rightarrow \text{partner}(x, y)
\]

\[
\text{conf}_{\text{pca}}(B \Rightarrow r(x, y)) = \frac{\text{support}(B \Rightarrow r(x, y))}{\#(x, y) : \text{director}(x, z) \land \text{actor}(z, y) \land \text{partner}(x, y')}
\]
Confidence approximation

- Use statistics to estimate $\text{conf}_{\text{pca}}$ denominator
  - $(\# \text{ of actors per director}) \times (\# \text{ of partnered directors})$
  - $\text{director}(x, z), \text{actor}(z, y) \Rightarrow \text{partner}(x, y)$

$$\text{conf}_{\text{pca}}(B \Rightarrow r(x, y)) = \frac{\text{support}(B \Rightarrow r(x, y))}{\#(x, y) : \text{director}(x, z) \land \text{actor}(z, y) \land \text{partner}(x, y') \land ...}$$
Confidence approximation

- Use statistics to estimate $\text{conf}_{\text{pca}}$ denominator
  - ($\#$ of actors per director) $\times$ ($\#$ of partnered directors)

$\text{director}(x, z), \text{actor}(z, y) \Rightarrow \text{partner}(x, y)$

$$\text{conf}_{\text{pca}}(B \Rightarrow r(x, y)) = \frac{\text{support}(B \Rightarrow r(x, y))}{\#(x, y) : \text{director}(x, z) \land \text{actor}(z, y) \land \text{partner}(x, y')}$$
Confidence approximation

- Use statistics to estimate \( \text{conf}_{\text{pca}} \) denominator
  - \((\# \text{ of actors per director}) \times (\# \text{ of partnered directors})\)

\[
\text{director}(x, z), \text{actor}(z, y) \Rightarrow \text{Avg \# movies per director} \times \text{Avg \# actors per movie}
\]

\[
\text{conf}_{\text{pca}}(B \Rightarrow r(x, y)) = \frac{\text{support}(B \Rightarrow r(x, y))}{\#(x, y) : \text{director}(x, z) \land \text{actor}(z, y) \land \text{partner}(x, y')}
\]

Actors can play in several movies of the same director.
Confidence approximation

- Use statistics to estimate $\text{conf}_{\text{pca}}$ denominator
  - $(\# \text{ of actors per director}) \times (\# \text{ of partnered directors})$

\[
director(x, z), \ actor(z, y) \Rightarrow \text{partner}(x, y)
\]

\[
\text{conf}_{\text{pca}}(B \Rightarrow r(x, y)) = \frac{\text{support}(B \Rightarrow r(x, y))}{\#(x, y) : \ director(x, z) \ \& \ actor(z, y) \ \& \ \text{partner}(x, y')}
\]
Confidence approximation

- Use statistics to estimate $\text{conf}_{\text{pca}}$ denominator
  - $(\# \text{ of actors per director}) \times (\# \text{ of partnered directors})$

$$\text{director}(x, z), \text{actor}(z, y) \Rightarrow \text{partner}(x, y)$$

$$\text{conf}_{\text{pca}}(B \Rightarrow r(x, y)) = \frac{\text{support}(B \Rightarrow r(x, y))}{\#(x, y) : \text{director}(x, z) \land \text{actor}(z, y) \land \text{partner}(x, y')}$$

overlap between director and partner relation
Confidence approximation

- If estimation is below threshold, discard the rule.
- $(\#$ of actors per director) × $(\#$ of partnered directors)

$\text{director}(x, z), \text{actor}(z, y) \Rightarrow \text{partner}(x, y)$
AMIE: Association Rule Mining Under Incomplete Evidence

Tailored In-memory DB

Concurrent mining implementation
AMIE's runtime

AMIE is 3 order of magnitude faster than state-of-the-art approaches.

<table>
<thead>
<tr>
<th>Dataset</th>
<th># facts</th>
<th>WARMR</th>
<th>ALEPH</th>
<th>AMIE</th>
</tr>
</thead>
<tbody>
<tr>
<td>YAGO core</td>
<td>1M</td>
<td>-</td>
<td>5s to 1d</td>
<td>3.17min</td>
</tr>
<tr>
<td>YAGO (sample)</td>
<td>47K</td>
<td>18h</td>
<td>0.05s to 1d</td>
<td>2.59s, 2.90s</td>
</tr>
</tbody>
</table>
# AMIE's runtime

AMIE can mine rules in large ontologies up to 11M facts and more than 1500 relations.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Facts</th>
<th>Relations</th>
<th>Runtime</th>
<th>Rules</th>
</tr>
</thead>
<tbody>
<tr>
<td>YAGO2</td>
<td>948K</td>
<td>32</td>
<td>28.19s</td>
<td>138</td>
</tr>
<tr>
<td>YAGO2 (const)</td>
<td></td>
<td></td>
<td>9.93min</td>
<td>18K</td>
</tr>
<tr>
<td>YAGO2 (l=4)</td>
<td></td>
<td></td>
<td>8.35min</td>
<td>6.9K</td>
</tr>
<tr>
<td>YAGO2s</td>
<td>4.12M</td>
<td>37</td>
<td>59.38min</td>
<td>94</td>
</tr>
<tr>
<td>Dbpedia 2.0</td>
<td>6.7M</td>
<td>1595</td>
<td>46.88min</td>
<td>113K</td>
</tr>
<tr>
<td>Dbpedia 3.8</td>
<td>11.02M</td>
<td>650</td>
<td>7h 6min</td>
<td>2.47K</td>
</tr>
<tr>
<td>Wikidata</td>
<td>8.4M</td>
<td>431</td>
<td>25.50min</td>
<td>889</td>
</tr>
</tbody>
</table>
AMIE's output quality

PCA confidence suitable at ranking predictive rules.

![Graph showing the output quality of AMIE's predictions]

- **Std. Confidence (rules 1-45)**
- **PCA Conf. (rules 1-30)**
- **PCA Conf. + types + joint-prediction**

**Aggregated predictions (beyond the initial KB)**
Some rules found by AMIE

- **YAGO**
  - hasWonPrize(x, Leibniz Prize) ⇒ livesIn(x, Germany)
  - hasAdvisor(x, y), graduatedFrom(x, z) ⇒ worksAt(y, z)

- **DBpedia**
  - countySeat(x, y) ⇒ largestCity(x, y)

- **Wikidata**
  - relative(y, z), sister(z, x) ⇒ relative(x, y)
Summary

- Pruning strategies in combination with custom DB implementation allow for scalable rule mining
- PCA more suitable at generating counter-evidence
Summary

- Pruning strategies in combination with custom DB implementation allow for scalable rule mining
- PCA more suitable at generating counter-evidence
Applications of Rule Mining
Semantifying wikilinks
Semantifying wikilinks

• KBs store the hyperlinks structure of Wikipedia articles
Semantifying wikilinks

• KBs store the hyperlinks structure of Wikipedia articles
Semantifying wikilinks

- KBs store the hyperlinks structure of Wikipedia articles
- Usually the semantics of the relation are unknown

Barack Obama \(\text{linksTo}\) Nobel Peace Prize
Semantifying wikilinks

- KBs store the hyperlinks structure of Wikipedia articles
- Usually the semantics of the relation are unknown
  - These are the unsemantified wikilinks

Barack Obama \( \text{linksTo} \) Nobel Peace Prize
Semantifying wikilinks

- KBs store the hyperlinks structure of Wikipedia articles

**Goal:** Find the relations that hold between the endpoints of wikilinks.
Approach

Use semantified wikilinks to learn rules.
Approach

Use semantified wikilinks to learn rules.
Approach

Use semantified wikilinks to learn rules.

\[
\text{linksTo}(x, y), \text{type}(x, \text{Politician}), \text{type}(y, \text{Prize}) \Rightarrow \text{wonPrize}(x, y)
\]
Approach

Apply rules to predict the meaning of wikilinks

\[
\text{linksTo}(x, y), \; \text{type}(x, \text{Politician}), \; \text{type}(y, \text{Prize}) \; \Rightarrow \; \text{wonPrize}(x, y)
\]
Approach

Apply rules to predict the meaning of wikilinks

\[ \text{linksTo}(x, y), \text{type}(x, \text{Politician}), \text{type}(y, \text{Prize}) \Rightarrow \text{wonPrize}(x, y) \]
Experimental evaluation
Wikilinks semantification

- Experiments on DBpedia 3.8
Wikilinks semantification

- Experiments on DBpedia 3.8
  - 2M+ wikilinks, 18M facts in total
Wikilinks semantification

- Experiments on DBpedia 3.8
  - 2M+ wikilinks, 18M facts in total
- AMIE for rule mining
Wikilinks semantification

- Experiments on DBpedia 3.8
  - 2M+ wikilinks, 18M facts in total
- AMIE for rule mining
  - 3500+ semantification rules
Wiki links semantification

- Experiments on DBpedia 3.8
  - 2M+ wikilinks, 18M facts in total
- AMIE for rule mining
  - 3500+ semantification rules
- 180K unsemantified wikilinks
Wikilinks semantification

• Experiments on DBpedia 3.8
  • 2M+ wikilinks, 18M facts in total
• AMIE for rule mining
  • 3500+ semantification rules
• 180K unsemantified wikilinks
  • For each wikilink we generated a ranking of possible relations.
  • 77% precision @top1, 67% @top3
Summary: Wikilinks

Rule Mining is an effective method for the semantification of wikilinks

Luis Galárraga, Danai Symeonidou, Jean-Claude Moissinac.
Rule Mining for Semantifying Wikilinks.
In LOWD, 2015
Schema Alignment
Schema Alignment

KBs in the semantic web speak in different “languages” about the same things.
Schema Alignment

Some instances has been aligned.

Barack Obama
sameAs
President Obama

bornInState
Hawaii
sameAs

bornInCity
Honolulu
locatedIn
State of Hawaii
This does not suffice for a full data integration
This does not suffice for a full data integration.

Goal: Find schema alignments between two KBs

President Obama \(\text{sameAs} \) Barack Obama

bornInCity: Honolulu \(\text{locatedIn} \) State of Hawaii

bornInState: Hawaii
Approach

Use instance alignments to “coalesce” the KBs.

Barack Obama

bornInState

Hawaii

sameAs

President Obama

bornInCity

Honolulu

locatedIn

State of Hawaii
Approach

Use instance alignments to “coalesce” the KBs.

Barack Obama

Hawaii

bornInCity

locatedIn

bornInState
Approach

Mine alignment rules on the coalesced KB
Approach

Mine alignment rules on the coalesced KB

\[ \text{bornInCity}(x, z), \text{locatedIn}(y, z) \Rightarrow \text{bornInState}(x, y) \]
<table>
<thead>
<tr>
<th>Rule</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>[d:artist(x, y) \Rightarrow y:created(x, y)]</td>
<td>R-subsumption</td>
</tr>
<tr>
<td>[d:nationality(x, y) \leftrightarrow y:citizenOf(x, y)]</td>
<td>R-equivalence</td>
</tr>
<tr>
<td>[\text{type}(x, \text{Athlete}_d) \Rightarrow \text{type}(x, \text{Person}_y)]</td>
<td>C-subsumption</td>
</tr>
<tr>
<td>[y:\text{bornIn}(x, y), y:\text{label}(y, z) \Rightarrow i:\text{bornIn}(x, z)]</td>
<td>2-hops translation</td>
</tr>
<tr>
<td>[y:\text{child}(x, y), y:\text{child}(x, z) \Rightarrow f:\text{sibling}(y, z)]</td>
<td>Triangle alignment</td>
</tr>
<tr>
<td>[y:\text{bornIn}(x, y), \text{type}(x, \text{City}_y) \Rightarrow f:\text{birthPlace}(x, y)]</td>
<td>Specific R-subsumption</td>
</tr>
<tr>
<td>[y:\text{locatedIn}(x, \text{Italy}) \Rightarrow d:\text{timeZone}(x, \text{CET})]</td>
<td>Attribute-Value translation</td>
</tr>
<tr>
<td>[\text{type}(x, \text{Royal}_f), f:\text{gender}(x, \text{female}) \Rightarrow \text{type}(y, \text{Princess})]</td>
<td>2-values translation</td>
</tr>
</tbody>
</table>
### ROSA rules

<table>
<thead>
<tr>
<th>Rule</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>d:artist(x, y) ⇒ y:created(x, y)</code></td>
<td>R-subsumption</td>
</tr>
<tr>
<td><code>d:nationality(x, y) ⇔ y:belongsTo(x, y)</code></td>
<td>R-equivalence</td>
</tr>
<tr>
<td><code>type(x, Athlete_d) ⇒ type(x, Person_y)</code></td>
<td>C-subsumption</td>
</tr>
<tr>
<td><code>y:bornIn(x, y), y:label(y, z) ⇒ i:bornIn(x, z)</code></td>
<td>2-hops translation</td>
</tr>
<tr>
<td><code>y:child(x, y), y:child(x, z) ⇒ f:sibling(y, z)</code></td>
<td>Triangle alignment</td>
</tr>
<tr>
<td><code>y:bornIn(x, y), type(x, City_y) ⇒ f:birthPlace(x, y)</code></td>
<td>Specific R-subsumption</td>
</tr>
<tr>
<td><code>y:locatedIn(x, Italy) ⇒ d:timeZone(x, CET)</code></td>
<td>Attribute-Value translation</td>
</tr>
<tr>
<td><code>type(x, Royal_f), f:gender(x, female) ⇒ type(y, Princess)</code></td>
<td>2-values translation</td>
</tr>
</tbody>
</table>

Complex alignments suffer from low precision and the presence of soft-rules.
## ROSA rules

<table>
<thead>
<tr>
<th>Rule</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>(d: \text{artist}(x, y) \Rightarrow y: \text{created}(x, y))</td>
<td>R-subsumption</td>
</tr>
<tr>
<td>(d: \text{nationality}(x, y) \Leftrightarrow y: \text{citizenOf}(x, y))</td>
<td>R-equivalence</td>
</tr>
<tr>
<td>(\text{type}(x, \text{Athlete}_d) \Rightarrow \text{type}(x, \text{Person}_y))</td>
<td>C-subsumption</td>
</tr>
<tr>
<td>(y: \text{bornIn}(x, y), y: \text{label}(y, z) \Rightarrow i: \text{bornIn}(x, z))</td>
<td>2-hops translation</td>
</tr>
<tr>
<td>(y: \text{child}(x, y), y: \text{child}(x, z) \Rightarrow f: \text{sibling}(y, z))</td>
<td>Triangle alignment</td>
</tr>
<tr>
<td>(y: \text{bornIn}(x, y), \text{type}(x, \text{City}_y) \Rightarrow f: \text{birthPlace}(x, y))</td>
<td>Specific R-subsumption</td>
</tr>
<tr>
<td>(y: \text{locatedIn}(x, \text{Italy}) \Rightarrow d: \text{timeZone}(x, \text{CET}))</td>
<td>Attribute-Value translation</td>
</tr>
<tr>
<td>(\text{type}(x, \text{Royal}_f), f: \text{gender}(x, \text{female}) \Rightarrow \text{type}(y, \text{Princess}))</td>
<td>2-values translation</td>
</tr>
</tbody>
</table>

Luis Galárraga, Nicoleta Preda, Fabian Suchanek.  
Mining Rules to Align Knowledge Bases.  
AKBC 2013.
Canonicalizing Open Knowledge Bases
Open Knowledge Bases

- Normally extracted from text
Open Knowledge Bases

• Normally extracted from text
  • Entities and relations are not **canonical**
Open Knowledge Bases

- Normally extracted from text
  - Entities and relations are not canonical

![Diagram showing connections between entities: President Obama, Barack Obama, Harvard Law School]
Open Knowledge Bases

- Normally extracted from text
  - Entities and relations are not **canonical**

**Goal:** Take an open KB and rewrite its entities and relations in a canonical form
Approach

- Two stages process
Approach

- Two stages process
  - Canonicalize the noun phrases, then the verbal phrases

```
Barack Obama
  earned degree from
    Harvard Law School

President Obama
  earned degree from
    Harvard Law School

Barack Obama
  is a graduate of
    Harvard Law School
```
Noun phrase canonicalization

• Use a group of signals of synonymy
Noun phrase canonicalization

• Use a group of signals of synonymy
  • Example: attributes overlap

President Obama
earned degree from
Harvard Law School

Barack Obama
earned degree from
Harvard Law School

is a graduate of
Noun phrase canonicalization

- Use a group of signals of synonymy
  - Example: attributes overlap, tokens overlap
Noun phrase canonicalization

- Simple signals
  - Attribute overlap
  - String equality or similarity
  - IDF Tokens Overlap
- Source signals
  - Words overlap
  - Entities overlap
  - Types overlap
- Combined signal
Noun phrase canonicalization

• Use signals to cluster synonym noun phrases
Noun phrase canonicalization

- Use signals to cluster synonym noun phrases
- Agglomerative Clustering

President Obama = Barack Obama
- earned degree from Harvard Law School
- is a graduate of Harvard Law School
- is a graduate of Harvard Law School
Noun phrase canonicalization

- Use signals to cluster synonym noun phrases
  - Agglomerative Clustering

- Barack Obama
  - President Obama
  - Barack Hussein Obama
  - is a graduate of
  - earned degree from
  - earned degree from

- Harvard Law School
Noun phrase canonicalization

Pick one noun phrase to canonicalize all mentions
Noun phrase canonicalization

Pick one noun phrase to canonicalize all mentions

Barack Obama earned degree from Harvard Law School is a graduate of
Verbal phrase clustering

Rely on canonicalization of the noun phrases
Verbal phrase clustering

Apply rule mining on the semi-canonicalized KB

Barack Obama earned degree from Harvard Law School is a graduate of AMIE
Verbal phrase clustering

Apply rule mining on the semi-canonicalized KB

earned degree from ⇔ is a graduate of

Barack Obama

earned degree from

is a graduate of

Harvard Law School
Verbal phrase clustering

Pick one verbal phrase for the canonicalization

earned degree from ⇔ is a graduated of

AMIE

Barack Obama

earned degree from

is a graduate of

Harvard Law School
Verbal phrase clustering

Pick one verbal phrase for the canonicalization

Barack Obama

earned degree from

Harvard Law School
Experimental evaluation
Canonicalization of noun phrases

F1 measure of the signals on an open KB (with polysemy) constructed with Reverb on Clueweb09

<table>
<thead>
<tr>
<th>Signal</th>
<th>Macro</th>
<th>Micro</th>
<th>Pairwise</th>
</tr>
</thead>
<tbody>
<tr>
<td>Str identity</td>
<td>51%</td>
<td>84%</td>
<td>71%</td>
</tr>
<tr>
<td>Str. Similarity</td>
<td>51%</td>
<td>83%</td>
<td>67%</td>
</tr>
<tr>
<td>IDF token overlap</td>
<td>57%</td>
<td>88%</td>
<td>79%</td>
</tr>
<tr>
<td>Attr. Overlap</td>
<td>15%</td>
<td>28%</td>
<td>5%</td>
</tr>
<tr>
<td>Entity overlap</td>
<td>63%</td>
<td>78%</td>
<td>61%</td>
</tr>
<tr>
<td>Type overlap</td>
<td>62%</td>
<td>76%</td>
<td>56%</td>
</tr>
<tr>
<td>Word overlap</td>
<td>55%</td>
<td>76%</td>
<td>56%</td>
</tr>
<tr>
<td>Simple ML</td>
<td>55%</td>
<td>86%</td>
<td>78%</td>
</tr>
<tr>
<td>Full ML</td>
<td>61%</td>
<td>79%</td>
<td>46%</td>
</tr>
</tbody>
</table>
Canonicalization of verbal phrases

Performance of verbal phrase clustering on Reverb dataset taken from Clueweb09.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Precision</th>
<th>Coverage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reverb</td>
<td>94%</td>
<td>15%</td>
</tr>
<tr>
<td>Reverb (types)</td>
<td>98%</td>
<td>21%</td>
</tr>
</tbody>
</table>
Canonicalization of verbal phrases

Some clusters of verbal phrases could be linked to Freebase relations

<table>
<thead>
<tr>
<th>Cluster</th>
<th>Freebase</th>
</tr>
</thead>
<tbody>
<tr>
<td>be spoken in, be the official language of</td>
<td>location.country.official_language</td>
</tr>
<tr>
<td>be the national language of</td>
<td></td>
</tr>
<tr>
<td>be bought, acquire</td>
<td>organization.organization.acquired_by</td>
</tr>
</tbody>
</table>
Summary

• Simple signals such as the tokens overlap are effective at identifying synonym noun phrases.

• Rule mining plus instance information can find clusters of close verbal phrases with high precision.

Luis Galárraga, Geremy Heitz, Kevin Murphy, Fabian Suchanek.
Canonicalizing Open Knowledge Bases.
In CIKM, 2014
Predicting Completeness
Predicting completeness in KBs

- KBs are highly incomplete
Predicting completeness in KBs

- KBs are highly incomplete
  - 2% of people have a father in Wikidata
Predicting completeness in KBs

- KBs are highly incomplete
  - 2% of people have a father in Wikidata
- We do not know where the incompleteness lies
Predicting completeness in KBs

- KBs are highly incomplete
  - 2% of people have a father in Wikidata
- We do not know where the incompleteness lies
  - A person without spouse in the KB could be incomplete or single
Predicting completeness in KBs

- KBs are highly incomplete
  - 2% of people have a father in Wikidata
- We do not know where the incompleteness lies
  - A person without spouse in the KB could be incomplete or single
- Problems for data producers and consumers
Predicting completeness in KBs

• KBs are highly incomplete
  • 2% of people have a father in Wikidata

• We do not know where the incompleteness lies
  • A person without spouse in the KB could be incomplete or single

• Problems for data producers and consumers
  • Consumers: no completeness guarantees for queries
  • Producers: which parts of the KB need to be populated?
Completeness

- Defined with respect to a **query q** via a complete hypothetical KB K*
Completeness

- Defined with respect to a query $q$ via a complete hypothetical KB $K^*$
  - A query $q$ is complete in $K$, iff $q(K^*) \subseteq q(K)$
Completeness

- Defined with respect to a query $q$ via a complete hypothetical KB $K^*$
  - A query $q$ is complete in $K$, iff $q(K^*) \subseteq q(K)$
- We focus on queries of the form
  
  ```sql
  SELECT ?x { subject relation ?x }
  ```
Completeness

- Defined with respect to a query \( q \) via a complete hypothetical KB \( K^* \)
  - A query \( q \) is complete in \( K \), iff \( q(K^*) \subseteq q(K) \)
- We focus on queries of the form
  $$\text{SELECT } ?x \{ \text{subject relation } ?x \}$$

Does the KB know all the children of Barack Obama?
Completeness

- Defined with respect to a query \( q \) via a complete hypothetical KB \( K^* \)

**Goal:** Study different signals to predict if a query of the form \( \{ o : r(s, o) \} \) is complete in a KB
Completeness oracles

• Function that assigns a completeness value to pairs subject-relation \((s, r)\)
Completeness oracles

• Function that assigns a completeness value to pairs subject-relation \((s, r)\)
  
  • PCA oracle: \((s, r)\) is complete if the KB knows at least one object \(o\)
Completeness oracles

- Function that assigns a completeness value to pairs subject-relation \((s, r)\)
- PCA oracle: \((s, r)\) is **complete** if the KB knows at least one object \(o\)

Complete instances in the domain of \textit{hasChild}
Completeness oracles

- Function that assigns a completeness value to pairs subject-relation \((s, r)\)
  - PCA oracle: \((s, r)\) is **complete** if the KB knows at least one object \(o\)

Complete instances in the domain of \textit{hasChild}
Completeness oracles

Oracles have certain precision and recall

Complete instances in the domain of \texttt{hasChild}

PCA oracle
Completeness oracles

Oracles have certain precision and recall

PCA oracle

\[ \text{Precision} = \frac{2}{3} \]
\[ \text{Recall} = \frac{2}{5} \]

Complete instances in
the domain of \textit{hasChild}
Completeness oracles

• CWA: \( cwa(s, r) = \text{true} \)
• PCA: \( pca(s, r) = \exists o : r(s, o) \)
• Cardinality: \( \text{card}(s, r) = \#(o : r(s, o)) \geq k \)
• Popular entities: \( \text{popularity}_{pop}(s, r) = \text{pop}(s) \)
• No-chg over time: \( \text{nochange}_{chg}(s, r) = \neg \text{chg}(s, r) \)
• Star: \( \text{star}_{r_1, \ldots, r_n}(s, r) = \forall i \in \{1, \ldots, n\} : \exists o : r_i(s, o) \)
• Class: \( \text{class}_{c}(s, r) = \text{type}(s, c) \)
• AMIE
Completeness oracles

- CWA: $cwa(s, r) = \text{true}$
- PCA: $pca(s, r) = \exists o : r(s, o)$
- Cardinality: $\text{card}(s, r) = \#(o : r(s, o)) \geq k$
- Popular entities: $\text{popularity}_{\text{pop}}(s, r) = \text{pop}(s)$
- No-chg over time: $\text{nochange}_{\text{chg}}(s, r) = \sim \text{chg}(s, r)$
- Star: $\text{star}_{r_1,\ldots,r_n}(s, r) = \forall i \in \{1,\ldots,n\} : \exists o : r_i(s, o)$
- Class: $\text{class}_c(s, r) = \text{type}(s, c)$
- AMIE

Learned oracles
Learned oracles

- Based on completeness rules
Learned oracles

• Based on completeness rules
  • Learned with AMIE from a set of completeness annotations $\text{complete}(s, r)$ and $\text{incomplete}(s, r)$
Learned oracles

• Based on completeness rules
  
  • Learned with AMIE from a set of completeness annotations $complete(s, r)$ and $incomplete(s, r)$

  $notype(x, \text{Adult}), type(x, \text{Person}) \Rightarrow complete(x, \text{hasChild})$

  $dateOfDeath(x, y), lessThan_1(x, placeOfDeath) \Rightarrow incomplete(x, placeOfDeath)$
Learned oracles

• Based on completeness rules
  
  • Learned with AMIE from a set of completeness annotations $\text{complete}(s, r)$ and $\text{incomplete}(s, r)$
    
    \[
    \text{notype}(x, \text{Adult}), \text{type}(x, \text{Person}) \Rightarrow \text{complete}(x, \text{hasChild})
    \]
    \[
    \text{dateOfDeath}(x, y), \text{lessThan}_1(x, \text{placeOfDeath}) \Rightarrow \text{incomplete}(x, \text{placeOfDeath})
    \]

  • Annotations obtained by two means:
    
    – Automatic: e.g., everyone must have a nationality
    – Crowd-sourcing: ask mechanical turks for more objects in the web
AMIE Oracle

- It uses learned rules to predict completeness
AMIE Oracle

• It uses learned rules to predict completeness
• In case of contradictions, predictions with higher confidence and support prevail
Experimental evaluation
Evaluating oracles

F1 measure of the oracles in YAGO3

<table>
<thead>
<tr>
<th>Relation</th>
<th>CWA</th>
<th>PCA</th>
<th>Class</th>
<th>AMIE</th>
</tr>
</thead>
<tbody>
<tr>
<td>diedIn</td>
<td>60%</td>
<td>22%</td>
<td>99%</td>
<td>96%</td>
</tr>
<tr>
<td>directed</td>
<td>40%</td>
<td>96%</td>
<td>0%</td>
<td>100%</td>
</tr>
<tr>
<td>graduatedFrom</td>
<td>89%</td>
<td>4%</td>
<td>92%</td>
<td>87%</td>
</tr>
<tr>
<td>hasChild</td>
<td>71%</td>
<td>1%</td>
<td>78%</td>
<td>78%</td>
</tr>
<tr>
<td>hasGender</td>
<td>78%</td>
<td>100%</td>
<td>95%</td>
<td>100%</td>
</tr>
<tr>
<td>hasParent</td>
<td>1%</td>
<td>54%</td>
<td>0%</td>
<td>100%</td>
</tr>
<tr>
<td>isCitizenOf</td>
<td>4%</td>
<td>98%</td>
<td>5%</td>
<td>100%</td>
</tr>
<tr>
<td>isConnectedTo</td>
<td>87%</td>
<td>34%</td>
<td>88%</td>
<td>89%</td>
</tr>
<tr>
<td>isMarriedTo</td>
<td>55%</td>
<td>7%</td>
<td>57%</td>
<td>46%</td>
</tr>
<tr>
<td>wasBornIn</td>
<td>28%</td>
<td>100%</td>
<td>0%</td>
<td>100%</td>
</tr>
</tbody>
</table>
Summary

- It is possible to predict completeness in KBs with 100% precision
  - By combining different simple oracles (signals)
- Future work:
  - More signals of completeness, completeness predictions for rule mining.

Final conclusion
Final conclusion

- Rule Mining is about making sense out of semantic knowledge.
- Rule Mining can:
  - Produce insights about the data (AMIE)
  - Predict missing data (wikilinks)
  - Align the schemas of KBs
  - Cluster synonym verbal phrases
  - Predict completeness
- With the goal of making computers even smarter and more helpful to humans.
Credits

Icons made by Freepik from www.flaticon.com

Icon made by Business Dubai from www.flaticon.com