WOMBAT – A Generalization Approach for Automatic Link Discovery

Mohamed Ahmed Sherif\textsuperscript{1}, Axel-Cyrille Ngonga Ngomo\textsuperscript{1,2}, and Jens Lehmann\textsuperscript{3,4}

\textsuperscript{1} R&D Department II, Computing Center, University of Leipzig, 04109 Leipzig, Germany
\textsuperscript{2} Data Science Group, University of Paderborn, Pohlweg 51 33098 Paderborn, Germany
\textsuperscript{3} Computer Science Institute, University of Bonn, Römerstr. 164, 53117 Bonn, Germany
\textsuperscript{4} Fraunhofer IAIS, Schloss Birlinghoven, 53757 Sankt Augustin, Germany

May 30, 2017
Overview

1 Motivation

2 WOMBAT Algorithm

3 Evaluation

4 Conclusion & Future Work
Growing Linked Open Data Cloud
- 12 datasets $\Rightarrow$ 9000+ datasets
- [http://stats.lod2.eu](http://stats.lod2.eu)

Growing size of single knowledge bases
- **DBpedia 2.0** (1997)
  - 103 M triples
  - 1.95 M things
- **DBpedia 2016-04**
  - 9.5 B triples
  - 5.2 M things
Motivation
Why we need link discovery?

1. Complex information needs
   ⇒ Need to consume data across KBs

2. Fourth Linked Data principle

3. Real-time application
   - Structured machine learning
   - Data integration
   - Data enrichment
   - Cross-ontology QA
   - Reasoning
   - Federated queries
   - ...

Sherif et al.
WOMBAT – A Generalization Approach for Automatic Link Discovery
May 30, 2017
4 / 19
Motivation

Why is it difficult?

- Need for automatic LD for evolving datasets
- Mostly positive examples on the Web of Data
- Negative examples rarely to be found
- Missing links cannot be regarded as negative examples (Open World Assumption)
Motivation
Link Discovery (LD)

Given two knowledge bases $S$ and $T$, find links of type $\mathcal{R}$ between $S$ and $T$

Formally
- Find $M = \{(s, t) \in S \times T : \mathcal{R}(s, t)\}$
- Similarity approach: Find $M' = \{(s, t) \in S \times T : \sigma(s, t) \geq \theta\}$
- Distance approach: Find $M' = \{(s, t) \in S \times T : \delta(s, t) \leq \tau\}$

Goal: Find link specification (LS)

- $(\text{euclidean}(x.\text{price}, y.\text{price}), 0.90)$
- $(\text{levenshtein}(x.\text{desc}, y.\text{desc}), 0.50)$
- $(\text{trigrams}(x.\text{name}, y.\text{name}), 0.50)$
- $(\text{cosine}(x.\text{name}, y.\text{name}), 0.52)$
I. Learn atomic LS

II. Combine atomic LS
**WOMBAT Algorithm**

*Algorithm: I. Learning Atomic Link Specifications (LS)*

---

**Goal: Derive a set of initial atomic LS**

1. Compute the subset of properties with sufficient coverage
2. Return as many mappings as property pairs with highest possible F-measure

- \((\text{euclidean}(x.\text{price}, y.\text{price}), 0.90)\)
- \((\text{levenshtein}(x.\text{desc}, y.\text{desc}), 0.50)\)
- \((\text{trigrams}(x.\text{name}, y.\text{name}), 0.50)\)
- \((\text{cosine}(x.\text{name}, y.\text{name}), 0.52)\)
**Goal: Derive a set of complex LS**

1. **Input:** set of atomic LSs
2. Use $\cap$, $\sqcup$, $\setminus$ to append further atomic LS
3. Compute complex LS by using an approach based on generalisation operators
4. Perform an iterative search through a solution space based on a score function
5. **WOMBAT** uses F-measure as the score function

**Operators:**
- $\cap$: Intersection
- $\sqcup$: Union
- $\setminus$: Set difference

**Score Functions:**
- $(\text{euclidean}(x.\text{price}, y.\text{price}), 0.90)$
- $(\text{levenshtein}(x.\text{desc}, y.\text{desc}), 0.50)$
- $(\text{trigrams}(x.\text{name}, y.\text{name}), 0.50)$
- $(\text{cosine}(x.\text{name}, y.\text{name}), 0.52)$
Wombat Algorithm
Simple Operator (ϕ)

- Is not a refinement operator
- Allows efficient implementation
- Can not reach all specifications
  e.g., \((A_1 \sqcup A_2) \cap (A_3 \sqcup A_4)\)

\[
\varphi(L) = \begin{cases} 
\bigcup_{i=1}^{n} A_i = A^* & \text{if } L = \bot \\
\left( \bigcup_{i=1}^{n} L \sqcup A_i \right) \cup \left( \bigcup_{i=1}^{n} L \cap A_i \right) \cup \left( \bigcup_{i=1}^{n} L \setminus A_i \right) & \text{otherwise}
\end{cases}
\]

Example: Assume we have only 2 atomic LS \((A_1 \text{ and } A_2)\)

\[
\begin{align*}
A^* \cap A_1, F &= 0.2 \\
A^* \cap A_2, F &= 0.6 \\
A^* \sqcup A_2, F &= 0.4 \\
A^* \sqcup A_1, F &= 0.4 \\
A^* \setminus A_1, F &= 0.8 \\
A^* \setminus A_2, F &= 0.5 \\
\end{align*}
\]
Wombat Algorithm
Complete Operator ($\psi$)

- Uses a more sophisticated expansion strategy
- Allows learning nested LS
- Is an upward complete refinement operator
- Is improved using pruning

$$\psi(L) = \begin{cases} \{A_i \setminus A_j | A_i, A_j \in A \text{ for all } 1 \leq k \leq m\} & \text{if } L = \bot \\ \{L \cup A_i | A_i \in A, A_j \in A\} & \text{if } L \in A \\ \{L_1\} \cup \{L \cup A_i | A_i \in A, A_j \in A\} & \text{if } L = L_1 \setminus L_2 \\ \{L_1 \cap \cdots \cap L_{i-1} \cap L' \cap L_{i+1} \cap \cdots \cap L_n | L' \in \psi(L_i)\} \cup \{L \cup A_i | A_i \in A, A_j \in A\} & \text{if } L = L_1 \cap \cdots \cap L_n (n \geq 2) \\ \{L_1 \cup \cdots \cup L_{i-1} \cup L' \cup L_{i+1} \cup \cdots \cup L_n | L' \in \psi(L_i)\} \cup \{L \cup A_i | A_i \in A, A_j \in A\} & \text{if } L = L_1 \cup \cdots \cup L_n (n \geq 2) \end{cases}$$

Example: Assume we have only 2 atomic LS (A₁ and A₂)

1. A₁ \ (A₂ \ A₁), F = 0.7
2. A₁ \ (A₁ \ A₂), F = 0.6
3. A₁ \ A₂, F = 0.4
4. A₂ \ A₁, F = 0.3
5. (A₁ \ A₂) \ (A₂ \ A₁), F = 0.5
6. A₁ \ (A₁ \ A₂), F = 0.6
7. A₁ \ (A₂ \ A₁), F = 0.5
8. A₂ \ A₁, F = 0.3
9. ⊥
\[ \psi \text{ is an upward refinement operator} \]

The set of links generated by a child node is a superset of or equal to the set of links generated by its parent.

- \( r_{\text{max}} \) is bounded by the most general constructable LS
- \( p_{\text{max}} \) is bounded as false positives cannot disappear during generalisation
- \( F_{\text{max}} = \frac{2p_{\text{max}}r_{\text{max}}}{p_{\text{max}}+r_{\text{max}}} \)
- Prune all nodes in the search tree with \( F_{\text{max}} < F_{\text{best}} \)

**Example: Assume** \( F_{\text{best}} = 0.8 \)

\[
\begin{align*}
F_{\text{max}} &= 0.2 & F_{\text{max}} &= 0.8 & F_{\text{max}} &= 0.5 & F_{\text{max}} &= 0.4 & F_{\text{max}} &= 0.8 \\
\end{align*}
\]
**ψ is an upward refinement operator**

The set of links generated by a child node is a superset of or equal to the set of links generated by its parent

- $r_{max}$ is bounded by the most general constructable LS
- $p_{max}$ is bounded as false positives cannot disappear during generalisation
- $F_{max} = \frac{2p_{max}r_{max}}{p_{max} + r_{max}}$
- Prune all nodes in the search tree with $F_{max} < F_{best}$

**Example: Assume $F_{best} = 0.8$**

- $F_{max} = 0.2$
- $F_{max} = 0.8$
- $F_{max} = 0.5$
- $F_{max} = 0.4$
- $F_{max} = 0.8$
8 benchmark datasets (5 real-world, 3 synthetic)

2.80 GHz PC running OpenJDK 64-Bit Server 1.7.0_75 on Ubuntu 14.04.2 LTS

7 GB RAM

**WOMBAT**

- Similarity measures: jaccard, trigrams, cosine and qgrams
- Termination: $F = 1$ or max number of refinement tree depth of 10
- Properties coverage threshold = 0.6
## Evaluation

10-Fold Cross Validation F-Measure

<table>
<thead>
<tr>
<th>Dataset</th>
<th><strong>WOMBAT</strong></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th><strong>EAGLE</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Simple</td>
<td>Complete</td>
<td>Linear</td>
<td>Conjunction</td>
<td>Disjunction</td>
<td></td>
</tr>
<tr>
<td>Person 1</td>
<td>1.00</td>
<td>1.00</td>
<td>0.64</td>
<td>0.97</td>
<td>1.00</td>
<td>0.99 ± 0.004</td>
</tr>
<tr>
<td>Person 2</td>
<td>1.00</td>
<td>0.99</td>
<td>0.22</td>
<td>0.78</td>
<td>0.96</td>
<td>0.94 ± 0.032</td>
</tr>
<tr>
<td>Restaurants</td>
<td>0.98</td>
<td>0.97</td>
<td>0.97</td>
<td>0.97</td>
<td>0.97</td>
<td>0.97 ± 0.024</td>
</tr>
<tr>
<td>DBLP-ACM</td>
<td>0.97</td>
<td><strong>0.98</strong></td>
<td><strong>0.98</strong></td>
<td><strong>0.98</strong></td>
<td><strong>0.98</strong></td>
<td><strong>0.98 ± 0.007</strong></td>
</tr>
<tr>
<td>Abt-Buy</td>
<td>0.60</td>
<td>0.61</td>
<td>0.06</td>
<td>0.06</td>
<td>0.52</td>
<td><strong>0.65 ± 0.025</strong></td>
</tr>
<tr>
<td>Amazon-GP</td>
<td>0.70</td>
<td>0.67</td>
<td>0.59</td>
<td>0.71</td>
<td><strong>0.73</strong></td>
<td>0.71 ± 0.033</td>
</tr>
<tr>
<td>DBP-LMDB</td>
<td>0.99</td>
<td><strong>1.00</strong></td>
<td>0.99</td>
<td>0.99</td>
<td>0.99</td>
<td>0.99 ± 0.004</td>
</tr>
<tr>
<td>DBLP-GS</td>
<td><strong>0.94</strong></td>
<td><strong>0.94</strong></td>
<td>0.90</td>
<td>0.91</td>
<td>0.91</td>
<td>0.93 ± 0.006</td>
</tr>
<tr>
<td><strong>Average</strong></td>
<td><strong>0.90</strong></td>
<td><strong>0.90</strong></td>
<td>0.67</td>
<td>0.80</td>
<td>0.88</td>
<td><strong>0.90 ± 0.017</strong></td>
</tr>
</tbody>
</table>
### Evaluation

#### Pruning Procedure

<table>
<thead>
<tr>
<th>Dataset</th>
<th>10%</th>
<th>20%</th>
<th>30%</th>
<th>40%</th>
<th>50%</th>
<th>60%</th>
<th>70%</th>
<th>80%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Person 1</td>
<td>1.57</td>
<td>2.13</td>
<td>1.85</td>
<td>2.13</td>
<td>2.13</td>
<td>2.13</td>
<td>2.13</td>
<td>2.13</td>
</tr>
<tr>
<td>Person 2</td>
<td>1.29</td>
<td>1.29</td>
<td>1.57</td>
<td>1.57</td>
<td>1.57</td>
<td>1.57</td>
<td>1.57</td>
<td>1.57</td>
</tr>
<tr>
<td>Restaurant</td>
<td>1.17</td>
<td>1.45</td>
<td>1.17</td>
<td>1.45</td>
<td>1.45</td>
<td>1.45</td>
<td>1.45</td>
<td>1.45</td>
</tr>
<tr>
<td>Abt-Buy</td>
<td>3.38</td>
<td>3.00</td>
<td>3.00</td>
<td>3.39</td>
<td>3.39</td>
<td>3.39</td>
<td>3.39</td>
<td>3.39</td>
</tr>
<tr>
<td>Amazon-GP</td>
<td>1.14</td>
<td>1.38</td>
<td>1.33</td>
<td>1.37</td>
<td>1.38</td>
<td>1.45</td>
<td>1.54</td>
<td>1.59</td>
</tr>
<tr>
<td>DBP-LMDB</td>
<td>1.00</td>
<td>1.86</td>
<td>2.86</td>
<td>1.86</td>
<td>1.86</td>
<td>2.33</td>
<td>2.36</td>
<td>2.36</td>
</tr>
<tr>
<td>DBLP-GS</td>
<td>1.79</td>
<td>1.93</td>
<td>2.01</td>
<td>2.36</td>
<td>2.45</td>
<td>1.66</td>
<td>2.44</td>
<td>2.26</td>
</tr>
</tbody>
</table>

**Pruning factor:** \[
\frac{\text{number of searched nodes (search tree size + pruned nodes)}}{\text{Max. size of the search tree (2000 nodes in this set of experiments)}}
\]
### Evaluation

**Training with only 2%**

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Pessimistic</th>
<th>Re-weighted</th>
<th>Simple</th>
<th>Complete</th>
</tr>
</thead>
<tbody>
<tr>
<td>Persons 1</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
</tr>
<tr>
<td>Persons 2</td>
<td>0.97</td>
<td>1.00</td>
<td>0.80</td>
<td>0.84</td>
</tr>
<tr>
<td>Restaurants</td>
<td>0.95</td>
<td>0.94</td>
<td>0.98</td>
<td>0.88</td>
</tr>
<tr>
<td>DBLP-ACM</td>
<td>0.93</td>
<td>0.95</td>
<td>0.94</td>
<td>0.94</td>
</tr>
<tr>
<td>Amazon-GP</td>
<td>0.39</td>
<td>0.43</td>
<td>0.53</td>
<td>0.45</td>
</tr>
<tr>
<td>Abt-Buy</td>
<td>0.36</td>
<td>0.37</td>
<td>0.37</td>
<td>0.36</td>
</tr>
<tr>
<td><strong>Average</strong></td>
<td>0.77</td>
<td>0.78</td>
<td>0.77</td>
<td>0.74</td>
</tr>
</tbody>
</table>
Conclusion & Future Work

Conclusion

- Presented **WOMBAT**, the first approach to learn LS from positive examples
- **WOMBAT** is based on generalisation over the space of LS
- Presented 2 operators to achieve this goal
- Evaluated **WOMBAT** against SOTA
- **WOMBAT** outperforms SOTA by 11% on average

Future work

- Parallelize **WOMBAT**
- Try more aggressive pruning techniques for better scalability
- Apply active learning strategies
- Unsupervised **WOMBAT**
Thank you for your Attention!

Mohamed Ahmed Sherif
Sherif@informatik.uni-leipzig.de
http://aksw.org/MohamedSherif
http://aksw.org/Projects/LIMES

This work has been supported by H2020 projects SLIPO (GA no. 731581) and HOBBIT (GA no. 688227) as well as the DFG project LinkingLOD (project no. NG 105/3-2) and the BMWI Project GEISER (project no. 01MD16014).