Leveraging Data and Structure in Ontology Integration

Octavian Udrea$^1$
Lise Getoor$^1$
Renée J. Miller$^2$

$^1$University of Maryland College Park
$^2$University of Toronto
Contents

- Motivation and goals
- Short overview of OWL Lite
- The ILIADS method
- Experimental evaluation
ILIADS

Goal:
- Produce high-quality integration via a flexible method able to adapt to a wide variety of ontology sizes and structures (aka integrate ontologies)

Method:
- Combining statistical and logical inference

Solution:
- Integrated Learning In Alignment of Data and Schema (ILIADS)
Contents

- Motivation and goals
- Short overview of OWL Lite
- The ILIADS method
- Experimental evaluation
Example OWL Lite ontologies

(discoveredBy, owl:inverseOf, discoverer); (discoveredBy, owl:type, owl:FunctionalProperty)
(discoveredBy, owl:inverseOf, discoverer); (associatedWith, owl:type, owl:TransitiveProperty)
(resultsFrom, rdfs:subPropertyOf, associatedWith)
Example OWL Lite ontologies

An **entity** can be a:
- **Class**

(discoveredBy, owl:inverseOf, discoverer); (discoveredBy, owl:type, owl:FunctionalProperty)
(discoveredBy, owl:inverseOf, discoverer); (associatedWith, owl:type, owl:TransitiveProperty)
(resultsFrom, rdfs:subPropertyOf, associatedWith)
Example OWL Lite ontologies

An entity can be a:
- Class
- Instance

(discoveredBy, owl:inverseOf, discoverer); (discoveredBy, owl:type, owl:FunctionalProperty)
(discoveredBy, owl:inverseOf, discoverer); (associatedWith, owl:type, owl:TransitiveProperty)
(resultsFrom, rdfs:subPropertyOf, associatedWith)
Example OWL Lite ontologies

An entity can be a:
- Class
- Instance
- Property

(discoveredBy, owl:inverseOf, discoverer); (discoveredBy, owl:type, owl:FunctionalProperty)
(discoveredBy, owl:inverseOf, discoverer); (associatedWith, owl:type, owl:TransitiveProperty)
(resultsFrom, rdfs:subPropertyOf, associatedWith)
Example OWL Lite ontologies

(discoveredBy, owl:inverseOf, discoverer)
(discoveredBy, owl:type, owl:FunctionalProperty)
(discoveredBy, owl:inverseOf, discoverer)
(associatedWith, owl:type, owl:TransitiveProperty)
(resultsFrom, rdfs:subPropertyOf, associatedWith)
The integration problem
The integration problem
The integration problem
The integration problem
Contents

- Motivation and goals
- Short overview of OWL Lite
- The ILIADS method
- Experimental evaluation
The ILIADS algorithm

repeat until no more candidates
1. Compute local similarities
2. Select *promising* candidates
3. For each candidate
   a. Select relationship
   b. Perform logical inference
   c. Update score with the inference similarity
4. Select the candidate with the best score
end
The ILIADS algorithm

repeat until no more candidates

1. **Compute local similarities**
2. Select *promising* candidates
3. **For each** candidate
   a. Select relationship
   b. Perform logical inference
   c. Update score with the inference similarity
4. Select the candidate with the best score
end
Computing local similarities

\[ \text{sim}(e, e') = \lambda_x \text{sim}_{\text{lexical}}(e, e') + \lambda_s \text{sim}_{\text{structure}}(e, e') + \lambda_e \text{sim}_{\text{extension}}(e, e') \]

- **Parameters:** \( \lambda_x, \lambda_s, \lambda_e \)
- Different for classes, instances and properties

<table>
<thead>
<tr>
<th></th>
<th>( \text{sim}_{\text{ex}} )</th>
<th>( \text{sim}_{\text{struct}} )</th>
<th>( \text{sim}_{\text{ext}} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Properties</td>
<td>Jaro-Winkler/Wordnet.</td>
<td>Jaccard on.rdf:s:subPropertyOf neighbors multiplied by ( \text{sim}_{\text{features}} ).</td>
<td>Jaccard on set of ((X, Y)) pairs in property extension.</td>
</tr>
<tr>
<td>Instances</td>
<td>Jaro-Winkler.</td>
<td>Jaccard on sets of incident ((p,Y)) pairs.</td>
<td>0</td>
</tr>
</tbody>
</table>
Jaro-Winkler distance

- Ideas similar to edit distance
  - excellent for short strings (e.g. person names)
- Normalized to $[0,1]$
- Example: MARTHA and MARHTA
  - $JW=0.961$
- Example DIXON and DICKSONX
  - $JW=0.813$
The ILIADS algorithm

repeat until no more candidates
1. Compute local similarities
2. Select *promising* candidates
3. For each candidate
   a. Select relationship
   b. Perform logical inference
   c. Update score with the inference similarity
4. Select the candidate with the best score
end
Selecting promising candidates

1. Select candidates with $\text{sim}(e, e') > \lambda_t$

2. Use a policy based on entity type to order, e.g.:
   - Class alignments first
   - Instance alignments first
   - Alternate between classes and instances
The ILIADS algorithm

repeat until no more candidates

1. Compute local similarities
2. Select *promising* candidates
3. For each candidate
   a. Select relationship
   b. Perform logical inference
   c. Update score with the inference similarity
4. Select the candidate with the best score

end
Selecting relationship

- Must decide on relation type
  - subClassOf vs. equivalentClass
  - subPropertyOf vs. equivalentProperty

- Determination is difficult, especially under the OWL open-world semantics

- Use a simple extension-based technique based on a threshold $\lambda_r$
Selecting relationship
Selecting relationship

1. If $\frac{|\epsilon(c_1) - \epsilon(c_2)|}{|\epsilon(c_1)|} < \lambda_r$ and $\frac{|\epsilon(c_2) - \epsilon(c_1)|}{|\epsilon(c_2)|} \geq \lambda_r$, then consider the axiom $(c_1, \text{rdfs:subClassOf}, c_2)$.

2. If $\frac{|\epsilon(c_1) - \epsilon(c_2)|}{|\epsilon(c_1)|} \geq \lambda_r$ and $\frac{|\epsilon(c_2) - \epsilon(c_1)|}{|\epsilon(c_2)|} < \lambda_r$, then consider the axiom $(c_2, \text{rdfs:subClassOf}, c_1)$.

3. Otherwise consider the axiom $(c_1, \text{owl:equivalentClass}, c_2)$. 

18,73,000
Selecting relationship

No instances that belong to FoodPoisoning only / Total number of instances = 1/3

No instances that belong to FoodBorneDisease only / Total number of instances = 1/2
The ILIADS algorithm

repeat until no more candidates
1. Compute local similarities
2. Select *promising* candidates
3. For each candidate
   a. Represent candidate relationship
   b. Perform logical inference
   c. Update score with the inference similarity
4. Select the candidate with the best score
end
Performing logical inference

For the candidate pair (e,e’):

- Select an axiom to apply
- The *logical consequences* are the pairs of entities \((e^{(i)}, e^{(j)})\) that have just become equivalent
- Repeat a small number of times (5) to maintain tractability
Performing logical inference

E-Coli Poisoning

discoveredBy

Theodor Escherich

E-Coli

discoverer

T. S. Escherich
Performing logical inference
Performing logical inference

(Theodor Escherich, owl:sameAs, T.S. Escherich) is a logical consequence of the candidate (E-Coli Poisoning, owl:sameAs, E-Coli)
The ILIADS algorithm

repeat until no more candidates
1. Compute local similarities
2. Select *promising* candidates
3. For each candidate
   a. Represent candidate relationship
   b. Perform logical inference
   c. Update score with the inference similarity
4. Select the candidate with the best score
end
Updating score

Similar idea to weight flow in pairwise connectivity graphs. Details omitted.
Updating score

E-Coli Poisoning
Theodor Escherich

E-Coli
T. S. Escherich

0.5

P = 0.6 / (1 – 0.6) = 0.6 / 0.4 = 1.5
Updating score

E-Coli Poisoning

Theodor Escherich

T. S. Escherich

\[ P = \frac{0.6}{1 - 0.6} = \frac{0.6}{0.4} = 1.5 \]
Consistency

- The constructed alignment is not guaranteed to be consistent
  - ILIADS can only detect inconsistencies that appear in the few logical inference steps
  - Pellet used to check consistency after ILIADS

- Experimentally, inconsistent ontologies in less than .5% of runs
Contents

- Motivation and goals
- Short overview of OWL Lite
- The ILIADS method
- Experimental evaluation
Experimental framework

- 30 pairs of real-world ontologies
  - From 194 to over 20,000 triples
  - From a variety of domains: medical, geographical, economical, biological

- Ground truth provided by human reviewers
  - Multiple iterations to ensure the best human-provided alignment

- Datasets available:
Number of inference steps

- **Running time**
  - X-axis: Number of steps (N)
  - Y-axis: Running time [s]
  - Graph shows an exponential increase in running time with increasing number of steps.

- **F1 quality**
  - X-axis: Number of steps (N)
  - Y-axis: F1 quality
  - Graph shows an increase in F1 quality with increasing number of steps, approaching a plateau.

Note: The specific values for running time and F1 quality are not clearly visible in the image.
## ILIADS parameters

<table>
<thead>
<tr>
<th></th>
<th>Lexical parameters</th>
<th>Structural parameters</th>
<th>Extensional parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$\lambda^c_x$</td>
<td>$\lambda^i_x$</td>
<td>$\lambda^p_x$</td>
</tr>
<tr>
<td><strong>ILIADS-fixed</strong></td>
<td>.2</td>
<td>.4</td>
<td>.1</td>
</tr>
<tr>
<td><strong>Min</strong></td>
<td>.15</td>
<td>.4</td>
<td>0</td>
</tr>
<tr>
<td><strong>ILIADS-tailored</strong></td>
<td>.25</td>
<td>.45</td>
<td>.1</td>
</tr>
<tr>
<td><strong>Max</strong></td>
<td>.25</td>
<td>.45</td>
<td>.1</td>
</tr>
<tr>
<td><strong>ILIADS-tailored</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>