MPRI Internship Defense
Advances in Holistic Ontology Alignment

Antoine Amarilli
Supervised by Pierre Senellart

Télécom ParisTech
The Web. Lots of information in semi-structured HTML documents.

The semantic Web. An effort to represent information in a *structured* and *semantic* way.

Uses. Interoperability, integration of sources, constraints, complex queries, inference.
Ontologies are the information sources of the Semantic Web.

Vertices are entities or literals.

Edges are facts labeled with a relation.

Sources: manual creation, existing databases, information extraction.
Many ontologies are created independently: different entities and relations express the same things.

Linked Data: integrate existing ontologies in a network structured by equality links between equivalent concepts.

To automatically derive those links, we need to perform ontology alignment.
Sometimes **URIs** do not help us and literals are **ambiguous** or have minor differences...

Sometimes the **structures** of the two ontologies do not match...
Table of Contents

1 Background: the Semantic Web

2 The PARIS System

3 Performance Improvements

4 Join Relations

5 Theoretical Analysis

6 Approximate Literal Matching

7 Application to Information Extraction

8 Conclusion
**PARIS**


- To bootstrap a matching, **PARIS** uses an equality function on literals and applies propagation rules.

\[
\begin{align*}
Pr^{n+1}(x \equiv x') &= 1 - \prod_{r' \subseteq r} \left(1 - Pr^n(r' \subseteq r) \times \text{fun}^{-1}(r) \times Pr^n(y \equiv y')\right) \\
Pr^{n+1}(r \subseteq r') &= \frac{\sum_{r(x,y)} \left(1 - \prod_{r'(x',y')} \left(1 - \left(Pr^n(x \equiv x') \times Pr^n(y \equiv y')\right)\right)\right)}{\sum_{r(x,y)} \left(1 - \prod_{x',y'} \left(1 - Pr^n(x \equiv x') \times Pr^n(y \equiv y')\right)\right)}
\end{align*}
\]

The rules are represented as a system of equations which we iterate until a fixpoint is reached:
PARIS by Example

**Diagram:**
- **a:** Elvis
  - **a:** name → 'Elvis Presley'
  - **a:** birthdate → '1935-01-08'
  - **a:** spouse → Priscilla
    - **a:** name → 'Priscilla Presley'

- **b:** Elvis
  - **b:** name → 'Elvis Presley'
  - **b:** birthdate → '1935-01-08'
  - **b:** spouse → Priscilla
    - **b:** name → 'Priscilla Presley'
**PARIS by Example**

```
<table>
<thead>
<tr>
<th>a:Elvis</th>
<th>a:name</th>
<th>'Elvis Presley'</th>
</tr>
</thead>
<tbody>
<tr>
<td>a:birthdate</td>
<td>'1935-01-08'</td>
<td></td>
</tr>
<tr>
<td>a:spouse</td>
<td>b:Priscilla</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>b:Elvis</th>
<th>b:name</th>
<th>'Elvis Presley'</th>
</tr>
</thead>
<tbody>
<tr>
<td>b:birthdate</td>
<td>'1935-01-08'</td>
<td></td>
</tr>
<tr>
<td>b:spouse</td>
<td>b:Priscilla</td>
<td></td>
</tr>
</tbody>
</table>

| b:Priscilla | b:name | 'Priscilla Presley' |
```
PARIS by Example

- **a:Elvis**
  - a:name: 'Elvis Presley'
  - a:birthdate: '1935-01-08'
  - a:spouse
    - a:priscilla
      - a:name: 'Priscilla Presley'

- **b:Elvis**
  - b:name: 'Elvis Presley'
  - b:birthdate: '1935-01-08'
  - b:spouse
    - b:priscilla
      - b:name: 'Priscilla Presley'
PARIS by Example

a:Elvis
  a:name 'Elvis Presley'
  a:birthdate '1935-01-08'
  a:spouse a:Priscilla
    a:name 'Priscilla Presley'

b:Elvis
  b:name 'Elvis Presley'
  b:birthdate '1935-01-08'
  b:spouse b:Priscilla
    b:name 'Priscilla Presley'
Two instances should be aligned when they share the same values for aligned functional relations.

In theory, the ontology schema should indicate which relations are functional.

In practice, no schema, and no “strict” functionality: compute a fuzzy functionality in \([0, 1]\) from the data.
Paris is implemented in Java.

Paris was evaluated on:
- toy datasets from the OAEI,
- DBpedia and Yago (two ontologies extracted from Wikipedia)
- Yago and IMDb

The evaluation is done in terms of precision, recall and F-measure.

<table>
<thead>
<tr>
<th></th>
<th>Instances</th>
<th>Classes</th>
<th>Relations</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Prec</td>
<td>Rec</td>
<td>F</td>
</tr>
<tr>
<td>OAEI person</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
</tr>
<tr>
<td>OAEI restaurant</td>
<td>95%</td>
<td>88%</td>
<td>91%</td>
</tr>
<tr>
<td>DBpedia–Yago</td>
<td>90%</td>
<td>73%</td>
<td>81%</td>
</tr>
<tr>
<td>IMDb–Yago</td>
<td>94%</td>
<td>90%</td>
<td>92%</td>
</tr>
</tbody>
</table>
Table of Contents

1 Background: the Semantic Web
2 The PARIS System
3 Performance Improvements
4 Join Relations
5 Theoretical Analysis
6 Approximate Literal Matching
7 Application to Information Extraction
8 Conclusion
The original PARIS takes a few hours per iteration.

Ways to improve this:

- Replace BerkeleyDB by an in-memory representation of the ontologies.
- Parallelize the propagation of entity alignment scores over all entities. Aggregate results at the end to avoid races.
- Change the hardware (now that the computation is CPU-bound).
### Performance Improvement Results

<table>
<thead>
<tr>
<th>Iteration</th>
<th>Original PARIS</th>
<th>New PARIS (1 thread)</th>
<th>New PARIS (4 threads)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Startup</td>
<td>0h00</td>
<td>0h27</td>
<td>0h10</td>
</tr>
<tr>
<td>1</td>
<td>4h04</td>
<td>0h40</td>
<td>0h27</td>
</tr>
<tr>
<td>2</td>
<td>5h06</td>
<td>3h00</td>
<td>1h02</td>
</tr>
<tr>
<td>3</td>
<td>5h00</td>
<td>0h34</td>
<td>0h24</td>
</tr>
<tr>
<td>4</td>
<td>5h30</td>
<td>0h29</td>
<td>0h16</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>20h</strong></td>
<td><strong>5h</strong></td>
<td><strong>2h</strong></td>
</tr>
</tbody>
</table>

**Table:** Running times for the DBpedia–YAGO alignment task. The original PARIS was run on an Intel Xeon E5620 CPU clocked at 2.40 Ghz on a machine with 12 GB of RAM. The new PARIS was run on an Intel Core i7-3820 CPU clocked at 3.60 Ghz with 48 GB of RAM.
Table of Contents

1. Background: the Semantic Web
2. The PARIS System
3. Performance Improvements
4. Join Relations
5. Theoretical Analysis
6. Approximate Literal Matching
7. Application to Information Extraction
8. Conclusion
The simplest possible difference in structure between ontologies: relations of one ontology correspond to join relations in the other ontology.

The terminology is motivated by the “join” operator of relational algebra.

We see the join as a binary predicate: the intermediate nodes are existentially quantified but projected away.
We must keep the representation of joins implicit in Paris (memory constraints).

We must recursively enumerate all possible join facts instead of enumerating all possible facts.

We must avoid duplicate facts caused by multiple possible choices for the intermediate nodes.

We cannot afford to enumerate all possible relations anymore (many possible joins).

⇒ New algorithm to compute the entity and relation alignments simultaneously.
Practical Issues

- How to determine the **functionality** of join relations?
- How to select **interesting joins** to perform without exploring all joins?
- How to achieve acceptable **running time** on large ontologies?

⇒ We only perform the join alignment on **small** ontologies.
Table of Contents

1 Background: the Semantic Web
2 The PARIS System
3 Performance Improvements
4 Join Relations
5 Theoretical Analysis
6 Approximate Literal Matching
7 Application to Information Extraction
8 Conclusion
Log-transformation and Product Graph

\[ \Pr^{n+1}(x \equiv x') = 1 - \prod_{r(x,y) \subseteq r', (x', y') \subseteq r'} \left( 1 - \Pr^n(r' \subseteq r) \times \text{fun}^{-1}(r) \times \Pr^n(y \equiv y') \right) \times \left( 1 - \Pr^n(r \subseteq r') \times \text{fun}^{-1}(r') \times \Pr^n(y \equiv y') \right) \]

- The entity alignment equation is justified by a probabilistic model (independent choices).
- If the relation functionalities and alignments are in \( \{0, 1\} \), we can apply a log-transformation:
  \[ \text{LPr}^n(x \equiv x') := -\log(1 - \Pr^n(x \equiv x')) \]
- By looking at propagation in the product graph, we get a nicer equation, for some matrix \( M \) and a constant literal alignment vector \( L \):
  \[ \text{LPr}^{n+1} = M \text{LPr}^n + L \]
Green Measures

\[ LPr^{n+1} = M LPr^n + L \]

- This equation is similar to PageRank \((LPr^{n+1} = M LPr^n)\) except:
  1. The matrix is not stochastic.
  2. Diverging to \(+\infty\) means convergence (because of the log-transformation).
  3. \(L\) is pouring alignment weight to the aligned couples of literals.

- This last point can be linked to the use of Green measures to focus the PageRank computation.

- This interpretation suggests possible changes to the entity alignment equation (but we lose the probabilistic interpretation).
Table of Contents

1. Background: the Semantic Web
2. The PARIS System
3. Performance Improvements
4. Join Relations
5. Theoretical Analysis
6. Approximate Literal Matching
7. Application to Information Extraction
8. Conclusion
Literal Similarity Functions

- The original PARIS uses an exact literal equality function.
- Possible refinements: adjust for case, strip special characters, etc.
- Yet, we would need a better equality function giving \( > 0 \) weight to the alignment of similar literals.
- Approximate dictionary searching problem: given a literal, to find quickly all similar literals in the other ontology.
We use a shingling technique which was implemented by Mayur Garg (who interned in the team from IIT Delhi).

I interfaced his code with PARIS.

The performance of the shingling technique matches ad-hoc normalization on the OAEI restaurants dataset.

<table>
<thead>
<tr>
<th></th>
<th>Precision</th>
<th>Recall</th>
<th>F-measure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Paris with exact equality</td>
<td>95%</td>
<td>88%</td>
<td>91%</td>
</tr>
<tr>
<td>Paris with shingling</td>
<td>96%</td>
<td>95%</td>
<td>96%</td>
</tr>
<tr>
<td>Paris with normalization</td>
<td>98%</td>
<td>96%</td>
<td>97%</td>
</tr>
</tbody>
</table>
Table of Contents

1. Background: the Semantic Web
2. The PARIS System
3. Performance Improvements
4. Join Relations
5. Theoretical Analysis
6. Approximate Literal Matching
7. Application to Information Extraction
8. Conclusion
Many structured databases can only be queried through interfaces designed for humans (Web forms and HTML result pages).

To access this structured information, an automated agent must probe the form and perform wrapper induction on the result pages.

To understand the meaning of the extracted records and attributes, we can use PARIS (with a reference ontology).
Application to Form Understanding

Form

Author: 
Title: 
Publisher: 
Submit

Result page

The following results were found for your search:

Great Expectations
Charles Dickens
Dover Thrift Editions

David Copperfield
by Charles Dickens
Penguin Classics

List of records

The following results were found for your search:

Great Expectations
Charles Dickens
Dover Thrift Editions

David Copperfield
by Charles Dickens
Penguin Classics

Labeled graph

The following results were found for your search:

Great Expectations
Charles Dickens
Dover Thrift Editions

David Copperfield
by Charles Dickens
Penguin Books

RDF triples generation

ontology alignment

ontology enrichment

wrapper induction

input and output schema mapping

new probing terms

form probing

y:hasName

y:created

y:hasName

rdfs:type

?e1

?e2

?class

'Great Expectations'

'Charles Dickens'

'Othello'

'Shakespeare'

'Great Expectations'

'Charles Dickens'

'David Copperfield'

'David Copperfield'

'Penguin Books'

'Great Expectations'

'Charles Dickens'

'Othello'

'Shakespeare'

'Great Expectations'

'Charles Dickens'

'David Copperfield'

'David Copperfield'

'Penguin Books'

'Great Expectations'

'Charles Dickens'

'Othello'

'Shakespeare'

'Great Expectations'

'Charles Dickens'

'David Copperfield'

'David Copperfield'

'Penguin Books'
Application to Form Understanding

Form

- Author: 
- Title: 
- Publisher: 

Submit

Result page

The following results were found for your search:
- Great Expectations
  - Author: Charles Dickens
  - Publisher: Dover Thrift Editions
- David Copperfield
  - Author: Charles Dickens
  - Publisher: Penguin Classics

List of records

The following results were found for your search:
- Great Expectations
  - Author: Charles Dickens
  - Publisher: Dover Thrift Editions
- David Copperfield
  - Author: Charles Dickens
  - Publisher: Penguin Classics

Yago

- Othello
  - y:hasName: 'Othello'
  - y:created: 'Charles Dickens'
- Shakespeare
  - y:hasName: 'Shakespeare'
- Great Expectations
  - y:hasName: 'Great Expectations'
  - y:created: 'Charles Dickens'
- David Copperfield
  - y:hasName: 'David Copperfield'
  - y:created: 'Charles Dickens'

Labeled graph

- ?class
- rdf:type
- rdfs:type
- ?e1
- rdfs:type
- rdfs:type
- ?e2
- rdf:type
- rdfs:type

Form probing

new probing terms

input and output schema mapping

ontology alignment

ontology enrichment

wrapper induction

RDF triples generation

27/32
Application to Form Understanding

**Form**
- Author: 
- Title: 
- Publisher: 

Submit

**Result page**
The following results were found for your search:
- Great Expectations
  - Author: Charles Dickens
  - Publisher: Dover Thrift Editions
- David Copperfield
  - Author: Charles Dickens
  - Publisher: Penguin Classics

**List of records**
The following results were found for your search:
- Great Expectations
  - Author: Charles Dickens
  - Publisher: Dover Thrift Editions
- David Copperfield
  - Author: Charles Dickens
  - Publisher: Penguin Classics

**Labeled graph**
- Othello
  - y:hasName: 'Othello'
  - y:created: 'Shakespeare'
- Great Expectations
  - y:hasName: 'Great Expectations'
  - y:created: 'Charles Dickens'
- David Copperfield (novel)
  - y:hasName: 'David Copperfield'
  - y:created: 'Charles Dickens'

**RDF triples generation**
- ?class
- ?e1
- ?e2

**ontology alignment**
- 'Great Expectations'
- 'Charles Dickens'
- 'Dover Thrift Editions'

**ontology enrichment**
- 'Penguin Books'

**wrapper induction**
- input and output schema mapping
- new probing terms

---

27/32
Application to Form Understanding

Form

Author: 
Title: 
Publisher: 
Submit

Result page

The following results were found for your search:

- Great Expectations
  - Charles Dickens
  - Dover Thrift Editions

- David Copperfield
  - by Charles Dickens
  - Penguin Classics

List of records

The following results were found for your search:

- Great Expectations
  - Charles Dickens
  - Dover Thrift Editions

- David Copperfield
  - by Charles Dickens
  - Penguin Classics

Jago

'Great Expectations'
'Shakespeare'
'Charles Dickens'
'David Copperfield'

RDF triples generation

ontology alignment

ontology enrichment

Labeled graph

wrapper induction

input and output schema mapping

new probing terms

form probing
Application to Form Understanding

Form

Author: 
Title: 
Publisher: 
Submit

Result page

The following results were found for your search:

- Great Expectations
  - Author: Charles Dickens
  - Publisher: Dover Thrift Editions

- David Copperfield
  - Author: Charles Dickens
  - Publisher: Penguin Classics

List of records

The following results were found for your search:

- Great Expectations
  - Author: Charles Dickens
  - Publisher: Dover Thrift Editions

- David Copperfield
  - Author: Charles Dickens
  - Publisher: Penguin Classics

Yago

Labeled graph

RDF triples generation

ontology alignment

ontology enrichment

Wrapper induction
Application to Form Understanding

Form

Author: 
Title: 
Publisher: 
Submit

Result page

The following results were found for your search:
- Great Expectations
  - Charles Dickens
  - Dover Thrift Editions
- David Copperfield
  - by Charles Dickens
  - Penguin Classics

Yago

Book
- rdfs:type
  - Othello
    - y:hasName: 'Othello'
    - y:created: 'Shakespeare'
  - Great Expectations
    - y:hasName: 'Great Expectations'
    - y:created: 'Charles Dickens'
  - David Copperfield (novel)
    - y:hasName: 'David Copperfield'
    - y:created: 'Charles Dickens'

List of records

The following results were found for your search:
- Great Expectations
  - Charles Dickens
  - Dover Thrift Editions
- David Copperfield
  - by Charles Dickens
  - Penguin Classics

Labeled graph

RDF triples
- ?class
  - rdfs:type
- ?e1
  - rdfs:type
  - 'Great Expectations'
  - 'Charles Dickens'
  - 'Dover Thrift Editions'
- ?e2
  - rdfs:type
  - 'David Copperfield'
  - 'by Charles Dickens'
  - 'Penguin Books'

new probing terms
input and output schema mapping
wrapper induction
ontology alignment
ontology enrichment

27/32
Application to Form Understanding

Form

Author: 
Title: 
Publisher: 
Submit

Result page

The following results were found for your search:
- Great Expectations
  Charles Dickens
  Dover Thrift Editions
- David Copperfield
  by Charles Dickens
  Penguin Classics

List of records

The following results were found for your search:
- Great Expectations
  Charles Dickens
  Dover Thrift Editions
- David Copperfield
  by Charles Dickens
  Penguin Classics

Labeled graph

RDF triples generation

ontology alignment
ontology enrichment

new probing terms

input and output schema mapping

wrapper induction

form probing
Application to Form Understanding

**Form**
- Author: 
- Title: 
- Publisher: 

Submit

**Result page**
The following results were found for your search:
- Great Expectations
  - Author: Charles Dickens
  - Publisher: Dover Thrift Editions
- David Copperfield
  - Author: Charles Dickens
  - Publisher: Penguin Classics

**List of records**
The following results were found for your search:
- Great Expectations
  - Author: Charles Dickens
  - Publisher: Dover Thrift Editions
- David Copperfield
  - Author: Charles Dickens
  - Publisher: Penguin Classics

**Yago**
- Othello
  - y:hasName: 'Othello'
  - y:created: 'Shakespeare'
  - rdfs:type: Book
- Great Expectations
  - y:hasName: 'Great Expectations'
  - y:created: 'Charles Dickens'
  - rdfs:type: Book
- David Copperfield
  - y:hasName: 'David Copperfield'
  - y:created: 'Charles Dickens'
  - rdfs:type: Book

**Labeled graph**
- ?class
  - rdfs:type
  - y:hasName: 'David Copperfield'
  - y:created: 'Charles Dickens'
- ?e1
  - rdfs:type
  - y:hasName: 'Great Expectations'
  - y:created: 'Charles Dickens'
- ?e2
  - rdfs:type
  - y:hasName: 'Dover Thrift Editions'
  - y:created: 'Penguin Books'

new probing terms
form probing
input and output schema mapping
wrapper induction
ontology alignment
ontology enrichment
RDF triples generation

27/32
We experimented the approach on the Amazon book search form.

The entity alignments with the best confidence were indeed books aligned through their title and author.

The system identified relations: \( \textit{y:hasPreferredName} \) and \( (\textit{y:created}, \textit{y:hasPreferredName}) \).

It linked them to the result page DOM paths and form fields.

The support for join relations and approximate string matching is required in this setting.

The approach was presented as a vision paper in the VLDS workshop of VLDB.
Table of Contents

1 Background: the Semantic Web
2 The PARIS System
3 Performance Improvements
4 Join Relations
5 Theoretical Analysis
6 Approximate Literal Matching
7 Application to Information Extraction
8 Conclusion
Summary of Contributions

- **Performance improvements** resulting in an 10-fold speedup over the original implementation.
- Support of **join relation** alignments on small ontologies.
- Insights on the relation between **PARIS** and **PageRank**-inspired techniques.
- Integration of **approximate string matching** to improve the literal alignment.
- Application of **PARIS** for **deep Web analysis**.
Further Work

**Performance.** Further gains to be made, perform more complete benchmarks.

**Join relations.** Performance improvements, especially ways to only select interesting joins. Arbitrary patterns?

**Theory.** Study the possible alternative choices and benchmark them. Understand the full model (we *still* have no proof of overall convergence!) and the effects of implementation tweaks. Find links with Max-SAT or Markov Logic Networks?

**Literal matching.** Support of various datatypes such as numbers and dates (engineering work). Fix performance issues to perform larger experiments.

**Information Extraction.** Try with more sources. Find links with named entity disambiguation techniques such as AIDA? Intensional use for large-scale integration.
Thanks for your attention!

Questions?