X BenchMatch: a Benchmark for XML Schema Matching Tools

Fabien Duchateau¹, Zohra Bellahsene¹ and Ela Hunt²

¹LIRMM, Univ. Montpellier 2-CNRS, ²ETH Zurich
X BenchMatch uses as

• Input: the result of a schema matching algorithm (set of mappings and/or an integrated schema).

• Output: statistics about the quality of this input and the performance of the matching tool.

• A demo version of the prototype is available at


GOALS:
extensibility, portability, simplicity (ease of use), scalability, genericity, completeness
Extensibility. The benchmark should be able to be extended to include new measures and new format.

Portability. The benchmark should be OS-independent.

Simplicity. since both end-users and schema matching experts are targeted by this benchmark tool.

Scalability on two aspects creating new benchmark scenarii is an easy task. And a benchmark composed of many scenarii should be easy to build and evaluate.

Genericity. It should work with most of the available matchers.
KIND OF EVALUATION

• Quality of Mappings
  - Measures (precision, recall, f-measure)

• Quality of Integrated Schema
  - based on the use of the metrics

• Performance of Matching Algorithms
  (time)
MAPPING QUALITY MEASURES

- **Given** $T_{map}$ a set of derived mappings
- **Given** $T_{ex}$ a set of expert mappings

\[
\text{Precision} = \frac{|T_{map} \cap T_{ex}|}{|T_{map}|}
\]

\[
\text{Recall} = \frac{|T_{map} \cap T_{ex}|}{|T_{ex}|}
\]

\[
\text{Fmeasure} = \frac{(2 \cdot \text{precision} \cdot \text{recall})}{(\text{precision} + \text{recall})}
\]
Integrated Schema Quality Measures

• Given an integrated schema $S_i$, and an input schema $S_g$:

• **Backbone measure, BM,**
  – computes the size of the largest common subtree of $S_g$ and $S_i$ (measured in nodes), seen against the background of the integrated schema $S_i$.

  $$BM = \frac{| \text{LCS}(S_i, S_g) |}{| S_i |}$$

• **Structural overlap**
  – computes the number of nodes shared by $S_i$ and $S_g$ and included in a common subtree. $\text{Sub}$ is the set of all disjoint subtrees (each containing a minimum of two nodes) common to $S_i$ and $S_g$.
  – $k_{\text{Sub}}$ is the total number of elements of all subtrees in $\text{Sub}$.

  $$\text{StructuralOverlap} = \frac{k_{\text{Sub}}}{|S_i|}$$

• **Structural proximity**
  • computes the number of subtrees common to $S_i$ and $S_g$.
  • $o$ is the number of elements in $S_i$ that are not included in any common subtree, $o = | S_i | - k_{\text{Sub}}$.

  $$\text{StructuralProximity} = \frac{k_{\text{Sub}}}{\sqrt{|S_i| \times |\text{Sub}| + o}}$$
XBenchMatch Prototype

INPUT

Ideal File

Matcher File

Ideal schema

Matcher schema

Ideal mappings

Matcher mappings

XBenchMatch

XML Parser

Wrapper

Ideal tree internal structure

Matcher tree internal structure

Ideal list internal structure

Matcher list internal structure

Schema Benchmark Engine

Mapping Benchmark Engine

OUTPUT

schema quality measures

mapping quality measures

statistics
Scenarii of schemas

**SCHEMAS**

- Person schemas are small and strongly heterogeneous.

- Purchase orders, XCBL collection 3, demonstrate matching of a large schema to a smaller one.

- University course schemas are from Thalia [4].

- Biological schemas correspond to Uniprot protein DB, and GeneCards integrate data from over 100 databases.

**TESTED MATCHERS**

- Porsche, COMA++ and Similarity Flooding.
Similarity Flooding (SF)

- Based on structural approaches.
- Input schemas are converted into directed labeled graphs and the aim is to find relationships between those graphs.
- Structural rule: two nodes from different schemas are considered similar if their adjacent neighbours are similar.
- When similar nodes are discovered, this similarity is then propagated to the adjacent nodes until there is no changes anymore.
- This algorithm mainly exploits the labels with some semantic-based algorithms, like String Matching, to determine the nodes to which it should propagate.
- Similarity Flooding does not give good results when labels are often identical, especially for polysemic terms. Thus involving wrong mappings to be discovered by propagation.
COMA/COMA++

- A generic, composite matcher

- It can process the relational, XML, RDF schemas as well as ontologies. Internally it converts the input schemas as trees for structural matching.

- For linguistic matching, it utilizes a user defined synonym and abbreviation tables like CUPID, along with n-gram name matchers.

- Similarity of pairs of elements is calculated into a similarity matrix.

- Uses 17 element level matchers. For each source element, elements with similarity higher than than threshold are displayed to the user for final selection.
<table>
<thead>
<tr>
<th></th>
<th>Person</th>
<th>University</th>
<th>Order</th>
<th>Biology</th>
</tr>
</thead>
<tbody>
<tr>
<td>NB nodes ( (S_1 / S_2) )</td>
<td>11 / 10</td>
<td>18 / 18</td>
<td>20 / 844</td>
<td>719 / 80</td>
</tr>
<tr>
<td>Avg NB of nodes</td>
<td>11</td>
<td>18</td>
<td>432</td>
<td>400</td>
</tr>
<tr>
<td>Max depth ( (S_1 / S_2) )</td>
<td>4 / 4</td>
<td>5 / 3</td>
<td>3 / 3</td>
<td>7 / 3</td>
</tr>
<tr>
<td>NB of Mappings</td>
<td>5</td>
<td>15</td>
<td>10</td>
<td>57</td>
</tr>
</tbody>
</table>

Table 1: Summary of four evaluation scenarios.
Comparison of different matching tools on the matching quality
Comparison of different matching tools on the matching quality
## Performances Results

<table>
<thead>
<tr>
<th>Person</th>
<th>University</th>
<th>Order</th>
<th>Biology</th>
</tr>
</thead>
<tbody>
<tr>
<td>NB nodes (S1/S2)</td>
<td>11/10</td>
<td>18/18</td>
<td>20/844</td>
</tr>
<tr>
<td>BMatch</td>
<td>&lt; 1</td>
<td>&lt;1</td>
<td>&lt;1</td>
</tr>
<tr>
<td>COMA++</td>
<td>&lt; 1</td>
<td>&lt;1</td>
<td>3</td>
</tr>
<tr>
<td>SF</td>
<td>&lt;1</td>
<td>&lt;1</td>
<td>2</td>
</tr>
<tr>
<td>PORSCHE</td>
<td>&lt;1</td>
<td>&lt;1</td>
<td>&lt;1</td>
</tr>
</tbody>
</table>