CS548: Schema matching tools & homework#6

Outline

• Schema matching in general
• Schema matching tools
  – Clio & IBM Rational Data Architect
  – Microsoft Biztalk
  – COMA++
• Homework#6

Motivation

• If Microsoft takes over Yahoo! successfully
  
  ![Microsoft Yahoo!](image)

  • Tons of DB schemas will be mediated! Integration would take several weeks or months if done manually.

What does schema matching/mapping do?

• Schema matching:
  – Find correspondences between elements in the two schemas
  – They can be 1-1, 1-many, ...

• Schema mapping:
  – Create mapping expressions from the matches (post matching)

What does schema matching do?

• Given 2 schemas
• Returns how each element from each schema is related (\(=\), \(<\), is-a, part-of, overlap (set), contain (set) .. etc)

• It is impossible to determine fully automatically all matches. At best, what we can do is to infer match candidates which users can accept, reject or change.
Outline

• Schema matching in general
• Schema matching tools
  – Clio & IBM Rational Data Architecture
  – Microsoft Biztalk
  – COMA++
• Homework#6

Schema matching tool#1: Clio & IBM Rational Data Architect

• Clio project (http://www.cs.toronto.edu/db/clio/)
  2000-2003

• Aims
  – GUI for users to map schema elements easily
  – Semi-automatic schema matching
  – Automatic schema transformation (mapping)

Clio: Schema Mapping & Data Translation

Clio’s mapping construction algorithm

Illustration: Clio Schema Mapping

Generate Queries (XQuery) of the target schema
From Clio to IBM Rational Data Architect

- "Clio technology has been transferred into IBM’s product lines and forms a core component of IBM’s Rational Data Architect" (Renee J. Miller)

IBM Rational Data Architect
Enterprise data modeling and integration design tool

Rational Data Architecture

- Short demo to generate matches between 2 tables.

Schema Matching using Rational Data Architect (1)

Step 1: select tables to the project workspace

Schema Matching using Rational Data Architect (2)

Step 2: Identify source and target tables

Schema Matching using Rational Data Architect (3)

Matching manually or use "Discovery Relationships" to automatically match
Configuring matching strategies & thresholds.

Match results, which users can accept or reject them.

Specify transformation rule for each map.


Translate XML messages between e-business applications.

BizTalk Mapper
- Maps are huge, complex
- Data mapping is a tedious, labor-intensive task

BizTalk Mapper (2)
- Rich editor for message transformation
- Scales to even the most complex message schema
- “Assisted mapping”
BizTalk Mapper (3)

- Novelty: action-based matching
  - Take into account history of the user’s prior matching actions to bias the ranking computation.
  - Recent matches
  - Implicit scope e.g. if all neighbors of the element E mapped to the same region of the target schema, it’s likely that E is also mapped to that region too.

Bernstein+ vldb'06


COMA++

- Developed by Database Group at Leipzig (currently active!)  http://dbs.uni-leipzig.de/Research/coma.html

COMA++ matching process

COMA++ matcher library

<table>
<thead>
<tr>
<th>Matcher Type</th>
<th>Matcher</th>
<th>Schema Info</th>
<th>Auxiliary Info</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simple</td>
<td>AFS</td>
<td>Element Names</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>Optimal</td>
<td>Element Names</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>Uniform</td>
<td>Element Names</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>FullFace</td>
<td>Element Names</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>NVG</td>
<td>Element Names</td>
<td>Extent dictionaries</td>
</tr>
<tr>
<td></td>
<td>Exact</td>
<td>Data types</td>
<td>Data type compatibility table</td>
</tr>
<tr>
<td></td>
<td>Exact</td>
<td>User-specified</td>
<td>User-specified matches</td>
</tr>
<tr>
<td></td>
<td>Hybrid</td>
<td>Element Names</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>Hybrid</td>
<td>Names</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>Hybrid</td>
<td>Data types</td>
<td>Data types=Names</td>
</tr>
<tr>
<td></td>
<td>Hybrid</td>
<td>Overlap</td>
<td>Child elements</td>
</tr>
<tr>
<td></td>
<td>REN</td>
<td>Schema</td>
<td>Existing schema-level match results</td>
</tr>
</tbody>
</table>

COMA++ discussions

- Implement several matching strategies. Most of them are based on information retrieval techniques. Weights for aggregating matching scores from all matchers need to be adjusted by hand.
- Not only support schema matching but ontology alignment as well.
- However it does not have any instance-based matcher yet. Only schema metadata are taken into account. -> class project ???
- Have no ability to differentiate different kinds of matches – what relation type a match is. (contain, overlap, part-of ??) -> class project ???

Outline

- Schema matching in general
- Schema matching tools
  - Clio & IBM Rational Data Architect
  - Microsoft Biztalk
  - COMA++

Homework#6
Homework# 6

- It's time to create your own schema matching tool, which can match elements in Web forms!

More details in the homework description.

2 fields are matched if they are semantically similar e.g. "departureDate" and "depDate" are the same but "arrivalDate" and "depDate" are not (although they have the same datatype)

Homework# 6

- These data are in the format like:

```
# left-right order should be retained
# nodes (node#, name, label, instances)
1, "vfrom", "from", "1900"
2, "vto", "to", "2000"
3, "make", "Make", "Alfa Romeo, AMC, Aston Martin, Audi, .. Not Listed"
4, "vtype", "Type", "Coupe, Sedan, Limousine, Convertible, Pickup, Van, .."
5, "\"", "Year"
6, "\"", "\"

# end
```

Your code should be able to parse data the format like this. Then execute your schema matching algorithm and return match results.

Homework# 6

- Unlike database tables, this dataset has no data instances!
- However, some fields have useful auxiliary metadata e.g. predefined values in a selection box.

These predefined values were also extracted in this dataset.

Evaluate your match results

- We will randomly pick some extracted forms in the dataset. Then see how your code performance using F-measure.

- F-measure: one measure of performance that takes into account both recall and precision.
  - Harmonic mean of recall and precision:
  
  \[
  F = \frac{2PR}{P + R} = \frac{2}{\frac{1}{P} + \frac{1}{R}}
  \]
  
  - Compared to arithmetic mean, both need to be high for harmonic mean to be high.

Precision and Recall

- Single matcher may not work well for all cases!
  Consider systems we have learned so far how they can handle this.
- Here is the list of possible algorithms for finding if 2 strings are similar.
  - Affix (check if prefix or suffix of words are the same)
  - Soundex (both "Robert" and "Rupert" return the same string "R163" while "Rubin" yields "R150")
  - Edit distance (e.g. Levenshtein distance – Levenshtein("sitting","kitten") = 3)
  - Abbreviation
  - Regular expression
  - Etc.

There are some java libraries available e.g. http://secondstring.sourceforge.net/ and http://www.dcs.shef.ac.uk/~sam/stringmetrics.html. You are welcome to use them but make sure to include these library files in your submission.

Suggestions & Hints

Precision (P) = 1/ (1+2)
Recall (R) = 1/ (1+3)
Suggestions & Hints

• Start early! It would take some times for coding & trying several matchers.
• Do not "hardcode" for all possible cases. Try to see some regularity across data and use them instead.
• Good luck!