Introduction to Schema Matching Problem

- The Goal is to map certain schema elements to other schema
- Applications:
  - Schema Integration / mediated schemas
  - Translate data between multiple databases
  - Databases Consolidation

Why schema matching is difficult?

- Imprecise wording, e.g. contact-info
- Different Ontology, e.g. 'Load' in electrical and mechanical contexts
- Schema and data maybe insufficient.
- Documentations and original schema designers are usually no available
- Matching decisions are highly subjective
Approaches

- Learning-based Approach
- Rules-based Approach
- Information Retrieval Approach

Most solutions require user intervention either at the beginning (in case of learning) or after creating a mapping (correcting).

Similarity Flooding

- Based on schema elements matching. (vs. schema + instances matching)
- Based on schema structure (vs. elements level matching)
- Each Schema is represented as a directed graph
- Based on the assumption:
  - Whenever any two elements in the graphs $G_1$ and $G_2$ are similar, their neighbors tend to be similar.

```sql
CREATE TABLE Personnel

| Pno int, |
| Pname string, |
| Dept string, |
| Born date, |
| UNIQUE pkey(Pno) |

Schema 1

CREATE TABLE Employee

| EmpNo int PRIMARY KEY, |
| EmpName varchar(50), |
| DeptNo int REFERENCES Department, |
| Salary dec(15,2), |
| Birthdate date |
|

CREATE TABLE Department

| DeptNo int PRIMARY KEY, |
| DeptName |

Schema 2
```
**Equivalent Graph Representation**

**SF algorithm**

1- Initial Mapping is made by simple string matching between graphs nodes.

<table>
<thead>
<tr>
<th>Line</th>
<th>Similarity</th>
<th>Node in $G_1$</th>
<th>Node in $G_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.0</td>
<td>Column</td>
<td>Column</td>
</tr>
<tr>
<td>2</td>
<td>0.66</td>
<td>ColumnType</td>
<td>Column</td>
</tr>
<tr>
<td>3</td>
<td>0.66</td>
<td>&quot;Dept&quot;</td>
<td>&quot;Dept&quot;</td>
</tr>
<tr>
<td>4</td>
<td>0.66</td>
<td>&quot;DeptName&quot;</td>
<td>&quot;DeptName&quot;</td>
</tr>
<tr>
<td>5</td>
<td>0.5</td>
<td>UniqueKey</td>
<td>PrimaryKey</td>
</tr>
<tr>
<td>6</td>
<td>0.26</td>
<td>&quot;EmpName&quot;</td>
<td>&quot;EmpName&quot;</td>
</tr>
<tr>
<td>7</td>
<td>0.26</td>
<td>&quot;date&quot;</td>
<td>&quot;Birthdate&quot;</td>
</tr>
<tr>
<td>8</td>
<td>0.22</td>
<td>&quot;dept&quot;</td>
<td>&quot;Department&quot;</td>
</tr>
<tr>
<td>9</td>
<td>0.11</td>
<td>&quot;m&quot;</td>
<td>&quot;Department&quot;</td>
</tr>
<tr>
<td>10</td>
<td>0.06</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**SF algorithm (cont’d)**

2- Similarity Flooding

Model A  
Model B  

Pairwise connectivity graph
**Assumptions**

1. Each edge type has the same contribution = 1.0
2. Similarity contribution for edges with the same type outgoing from one node is evenly distributed

**SF algorithm (cont’d)**

- Normalize similarity values after each iteration
- Stopping condition: \( \Delta (\sigma^n, \sigma^{n-1}) < \varepsilon \)
- Convergence can be guaranteed when all initial similarities for all pairs > 0

**Propagation Rule**:

\[
\sigma^{i+1}(x,y) = \sigma^i(x,y) + \sum_{(a \rightarrow p \rightarrow x,y) \in A} \sigma^i(a_x, b_y) \cdot w((a_x, b_y), (x,y)) + \sum_{(x \rightarrow p \rightarrow b,y) \in B} \sigma^i(a_x, b_y) \cdot w((a_x, b_y), (x,y))
\]
SF algorithm (cont’d)

- 3- Filters
The goal of filtering is to choose the best match candidates from the output list

- Application-Specific Constraints filter : e.g. cardinality constraint

- Selection Metrics : e.g. stable marriage, maximal matching, assignment problem

- Selection Threshold (0 < t_{rel} ≤ 1)

Matching Quality

- Accuracy : how much effort is needed to convert the output matching pairs to the intended one, i.e. the number of removing false positives and adding false negatives

\[\text{Accuracy} = 1 - \frac{(n - c) + (m - c)}{m}\]

where:
- \(n\) : number of returned matches
- \(m\) : number of intended matches
- \(c\) : number of returned intended matches

Intended Results Specification

Intended match result is categorized into 3 types:

- Sparse
- Expected
-Verbose
Results (cont’d)

Convergence Speed

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Export formula</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>$\sigma^{t+1} = \text{normalize}(\sigma^t + \mathcal{B}(\sigma^t))$</td>
</tr>
<tr>
<td>A</td>
<td>$\sigma^{t+1} = \text{normalize}(\sigma^t + \mathcal{A}(\sigma^t))$</td>
</tr>
<tr>
<td>C</td>
<td>$\sigma^{t+1} = \text{normalize}(\mathcal{C}(\sigma^t) + \mathcal{D}(\sigma^t))$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Formula</th>
<th>$T_1$</th>
<th>$T_2$</th>
<th>$T_3$</th>
<th>$T_4$</th>
<th>$T_5$</th>
<th>$T_6$</th>
<th>$T_7$</th>
<th>$T_8$</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>A (no is)</td>
<td>18</td>
<td>40</td>
<td>122</td>
<td>76</td>
<td>13</td>
<td>25</td>
<td>25</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>A (strongly connected)</td>
<td>15</td>
<td>50</td>
<td>80</td>
<td>81</td>
<td>100</td>
<td>10</td>
<td>25</td>
<td>25</td>
<td></td>
</tr>
<tr>
<td>B (no is)</td>
<td>8</td>
<td>124</td>
<td>17</td>
<td>39</td>
<td>8</td>
<td>13</td>
<td>13</td>
<td>24</td>
<td>21</td>
</tr>
<tr>
<td>B (strongly connected)</td>
<td>7</td>
<td>269</td>
<td>33</td>
<td>32</td>
<td>13</td>
<td>15</td>
<td>16</td>
<td>23</td>
<td>23</td>
</tr>
<tr>
<td>C (no is)</td>
<td>7</td>
<td>9</td>
<td>11</td>
<td>7</td>
<td>9</td>
<td>10</td>
<td>10</td>
<td>5</td>
<td>7</td>
</tr>
<tr>
<td>C (strongly connected)</td>
<td>7</td>
<td>9</td>
<td>11</td>
<td>7</td>
<td>5</td>
<td>9</td>
<td>9</td>
<td>12</td>
<td></td>
</tr>
</tbody>
</table>

Strongly connected: $\mathcal{B}(x,y) > 0$, for all $x \in G_1$, $y \in G_2$

Pros

- Innovative method for quality matching.
- General Model, provided that there is a straightforward method to map the used schema to a graph.
- No learning phase is required before use
- Flexibility in filters to suit specific application constrains
Cons

- Weak basis for similarity propagation
- Estimation errors can also be propagated to neighbors
- Flooding techniques are usually slow. Not practical for large number of elements
- Initial similarities have huge impact on the output quality and the convergence speed. Which returns us to the first square: how to get good matching?
- Heterogeneous sources can be problematic when mapped to graphs
- Does not utilize data instances in the graph
- Unable to detect complex relations between elements

Recommendations

- Extension to N graph
- Adding more powerful more enhanced matcher to initialize similarities
- Adding sample of data to the graph and use instance matchers (e.g. format matcher) as initial similarity
- More useful in case of hierarchical schemas (e.g. XML)

Recommendations (cont’d)

- Restrict the graph to a hierarchical structure
  - The similarity will propagate in bottom-up and top-down fashions
  - Similarity propagation is much reasonable in case of parent/child relations
  - The performance and the convergence speed is improved due to limited propagation paths
  - Can fit most practical schemas, e.g., SQL/XML
Open Questions

- How to improve convergence speed?
- Can a directed (ordered) flooding affect the convergence or matching quality?
- How to extend the model to N graph?
- Will using sample instances increase the matching quality? to what extent? at what cost?

Conclusion

- Similarity Flooding is a structural based approach that fits various data sources
- Generality is chosen over performance
- Useful for specific contexts (e.g. no complex relations)
- Further investigation is required to improve matching quality and convergence speed