Matching Schemas From Disparate Data Sources

Eiman Elnahrawy
(Rutgers University, eiman@cs.rutgers.edu)
Under supervision of
Philip Bohannon & Michael Flaster
Data..Data.. and more Data !!

- Explosion of information and data sources
- Need to ultimately benefit from each piece of data
What is Semantic Matching?

- Mapping schema elements in one information repository to corresponding elements in a second repository

  - Integrating diverse set of queryable info sources over
    - The web (e.g., Inventory)
    - Enterprise data
### Schema T

<table>
<thead>
<tr>
<th>Area</th>
<th>Price</th>
<th>Agent-Address</th>
<th>Agent-Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Denver, CO</td>
<td>550,000</td>
<td>Boulder, CO</td>
<td>Laura Smith</td>
</tr>
<tr>
<td>Atlanta, GA</td>
<td>370,800</td>
<td>Athens, GA</td>
<td>Mike Brown</td>
</tr>
</tbody>
</table>

### Schema S

<table>
<thead>
<tr>
<th>Location</th>
<th>Price($)</th>
<th>AgentId</th>
</tr>
</thead>
<tbody>
<tr>
<td>Atlanta, GA</td>
<td>360,000</td>
<td>32</td>
</tr>
<tr>
<td>Raleigh, NC</td>
<td>430,000</td>
<td>15</td>
</tr>
</tbody>
</table>

### Agent Information

<table>
<thead>
<tr>
<th>Id</th>
<th>Name</th>
<th>City</th>
<th>State</th>
<th>Fee-Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>32</td>
<td>Mike Brown</td>
<td>Athens</td>
<td>GA</td>
<td>0.03</td>
</tr>
<tr>
<td>15</td>
<td>Jean Laup</td>
<td>Raleigh</td>
<td>NC</td>
<td>0.04</td>
</tr>
</tbody>
</table>
fantastic house
list-price 250,000
agent-id 15
phone 729-0831
location
city Atlanta
state GA

agent-id 15
price 250,000 $
agent-address Atlanta, GA
agent-name Laura Smith
Why do we care?

- **Our main concern is network data**
  - *E.g., how to map data from SNMP into NetML?*

- Not quite a new problem...!
Manual integration is not practical

- Schemas can be quite large (tedious, error-prone, ..)
- No documentation or meta data available sometimes
- Poor design: exact semantics not known or fully captured in the schema
- Schema structure may repeatedly change over the time

Cannot be fully automated but at least to a large extent... (Semi-Automatic)
Outline

- Introduction and motivations
- What is Schema Matching?
- State of the Art in Matching Schemas from Diverse Sources
- Our Prototype.. Our Goals..
- Conclusions and Future Work
Key Challenges

- Matching Accuracy!
- Smart ideas to exclude unnecessary matching
- How to match (semi)structured data or data with different designs (column name may be a data field!)
- Reflect/guarantee domain constraints in the match
- Complex matches 1-to-M (fns of multiple columns)
- Time complexity (schemas may have thousands attr)
Related Work

- Description of data in column (data-type):
  - Categorize attributes according to field specifications and data values
  - Train a neural network to recognize similar attributes (attribute category)

[Li and Clifton VLDB94]
- Training/domain expert:
  - Ask users to match samples
  - Use these to train a set of learners
  - Find mappings using trained learners
  - Combine predictions using a meta learner
    - [Doan, Domingos and Halevy SIGMOD01]

- Something somewhat similar
  - [Berlin and Motro CAiSE02]

- Building a matching corpus:
  - Explore evidence from other matching problems in a matching corpus
    - [Madhaven, Bernstein et al ICDE05]
• **Information theory-based:**
  - Column names can be opaque!
  - Construct dependency graph using mutual information between columns
  - Reduces to a graph matching problem
    [Kang and Naughton SIGMOD03]

• **Rule-based (iMAP):**
  - Search space of all possible matches “efficiently”
  - Apply rules that reflect domain constraints
    [Dhamanker, Lee, Doan et al SIGMOD04]
- **Fragment-oriented:**
  - Decompose XML trees into much smaller sub-trees and match from bottom up
    
    [Rahm et al ACM SIGMOD Records04]

- **Matching different designs -- values as attribute names:**
  - Extends SQL to promote and demote attribute names
    
    [Wyss, Fletcher et al. ACM trans on DB05]
Our Prototype.. Our Goal..

- Preliminary test-bed in Java with a reasonable collection of matchers
- To be extended by integrating more matching approaches
- Tool to investigate/evaluate different approaches using “appropriate” performance metrics
  - Speed, Accuracy (Precision versus Recall)
Design

- A collection of matchers that works on data summaries
- Numeric, Categorized and Text-based data
Go over the source and create all summaries in the “summary factory”

Repeat for target

Match columns using summaries and compute matching confidence

If [confidence > threshold] -> columns match

Filter overall match results and display Top-k matches for each target element
- **Name Matcher** (Exact, Levenshtein distance)
- **Histogram Matcher** (Kullback-Leibler distance)
- **Bayesian Classifier** (NN to be integrated) [http://classifier4j.sourceforge.net/usage.htm]
- **Value Set Matcher**
- **Different Tokenizers** (Qgrams, non-word break, white-space break)
- **Type Matcher** (calendar, phone #, zip code, general strings, numbers, ..)
Supported Features

- One-to-one mapping
- Try-everything strategy with type checking
- Fixed threshold, although doesn’t really matter
- Filter results by choosing top-k columns from source that match target
<table>
<thead>
<tr>
<th>lineno</th>
<th>area</th>
<th>price</th>
<th>agent-address</th>
<th>agent-name</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>Denver, CO</td>
<td>580,000</td>
<td>Boulder, CO</td>
<td>Laura Smith</td>
</tr>
<tr>
<td>3</td>
<td>Atlanta, GA</td>
<td>370,000</td>
<td>Athens, GA</td>
<td>Mike Brown</td>
</tr>
<tr>
<td>4</td>
<td>New Brunswick, NJ</td>
<td>500,000</td>
<td>North Brunswick, NJ</td>
<td>John Smith</td>
</tr>
<tr>
<td>AgentID</td>
<td>location</td>
<td>city</td>
<td>state</td>
<td>fee-rate</td>
</tr>
<tr>
<td>---------</td>
<td>------------------</td>
<td>---------</td>
<td>-------</td>
<td>----------</td>
</tr>
<tr>
<td>32</td>
<td>Atlanta, GA</td>
<td>Athens</td>
<td>GA</td>
<td>0.03</td>
</tr>
<tr>
<td>5</td>
<td>Raleigh, NC</td>
<td>Raleigh</td>
<td>NC</td>
<td>0.04</td>
</tr>
<tr>
<td>1</td>
<td>North Brunswick, NJ</td>
<td>New Brunswick</td>
<td>NJ</td>
<td>0.05</td>
</tr>
<tr>
<td>00</td>
<td>Boulder, CO</td>
<td>Denver</td>
<td>CO</td>
<td>0.035</td>
</tr>
</tbody>
</table>
Highest Confidence = 0.99
bayesian classifier returns this attr with highest probability (Confidence = 0.99)
and QgramMatcher noted that 65% of the values on left were found on right (Confidence = 0.65)
and Levenshtein distance between names is less than threshold (Confidence = 0.583333333333334)
and ValueSetMatcher noted that 58% of the values are common between left and right (Confidence = 0.583333333333333)
Highest Confidence = 0.375  
ValueSetMatcher noted that 87% of the values are common between left and right. (Confidence = 0.075)  
and QgramMatcher noted that 77% of the values on left were found on right. (Confidence = 0.775)
Highest Confidence = 0.8695652173913043
QgramMatcher noted that 86% of the values on left were found on right. (Confidence = 0.8695652173913043)
and ValueSetMatcher noted that 50% of the values are common between left and right. (Confidence = 0.5833333333333333)
Conclusion

- Schema matching is the problem of identifying corresponding elements in different schemas
- A challenging problem
- Very active area of research with various market applications
- Working prototype
- Different matchers, combine results using some strategies
Future Work

- More work on the prototype
  - Study performance (metrics) of different matchers (*what works best on which data*)
  - Better *filtering* of matching results
  - Large test data
- Opaque column matching!
- Learning over time to better match, set thresholds
Thanks! 😊

Any Q?