Data and Schema Matching

“A survey of approaches to automatic schema matching,”
by Rahm and Bernstein, *VLDB Journal* 10(4)

“Reconciling Schemas of Disparate Data Sources:
A Machine Learning Approach,” by Doan et al, SIGMOD 01

Zachary G. Ives
University of Pennsylvania

April 16, 2003

Some slides on LSD courtesy of
Prof. AnHai Doan, UIUC
Administrivia: Schedule Correction

**Wednesday:** wrap-up and discussion of research in data sharing

**Monday 4/21:** your 5-minute project presentations
- We get to hear about the cool projects you’ve been working on!
- Slides are allowed (but not required)
  - What did you do?
  - What were the hard problems?
  - How are you solving them?
  - How are you evaluating your work?

Take-home final exam will be distributed Monday
- Will likely be 3-4 essay questions; open-book, open-notes

Next Friday, 4/24, 11AM (instead of 4/23 lecture):
- Talk on schema matching by Prof. AnHai Doan, UIUC

Deadlines: final exam and project due before 6PM, 5/2
Data Sharing

We’ve been discussing sharing *semantically rich data* across the web:

- Data integration and data warehousing
- Semantic web and peer data management
- Same techniques apply to problems like e-commerce

- In all of these, there are huge challenges addressed by:
  - Data cleaning (very briefly)
  - Schema matching (in more detail)
Data Cleaning

Actually, refers to several possible types of problems in warehousing/integration/DBs:

- Data is “dirty”, i.e., has typos
- Data is ambiguous/imprecise
- Correspondences between objects in different representations is unknown

Two options:

- Offline, find items we think are the same and merge them together
- Or, online or offline, perform “approximate joins” and similar operations
Dirty or Imprecise Data

This is often something like WHIRL

- What are some key attributes of this approach?

Can also use data mining and probabilistic machine-learning approaches here

- Many AI folks are working on this problem
- Often requires multiple passes over the data
  - Look for “close matches” or “closest matches”
Finding Correspondences

Many different methods
  - Most overlap with “imprecise data” category

Challenges:
  - Very expensive to compute such things
  - How do we define mappings in our query language?
    - Generally use “concordance relations”

Better if we can compute correspondences and mappings at the schema level
  (There may be a concordance relation/function)
Schema Matching

A problem that has been the focus of work since the 1970s, in the AI, DB, and knowledge representation communities

- Today, people are realizing that this is a core problem to most of the things they want to do:
  - E-commerce exchanges
  - Data integration/warehousing
  - Semantic web

- Goal: make it (mostly) generic and reusable in different application domains
  Generally use probabilistic, machine-learning-based techniques
What’s the Schema Matching Problem?

Given two schemas, S1 and S2:

Create a mapping between the two:

- Mapping might be directional or symmetric
- Mapping might be in the form of a query, or it might be a set of expressions between items in each schema

Many people simply look at finding correspondences between elements as the first step

Correspondences are often informally justified
# A Matching Example

<table>
<thead>
<tr>
<th>S1 elements:</th>
<th>S2 elements:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cust</td>
<td>Customer</td>
</tr>
<tr>
<td>C#</td>
<td>CustID</td>
</tr>
<tr>
<td>CName</td>
<td>Company</td>
</tr>
<tr>
<td>FirstName</td>
<td>Contact</td>
</tr>
<tr>
<td>LastName</td>
<td>Phone</td>
</tr>
</tbody>
</table>
A Matching Example

S1 elements:
- Cust
- C# =
- CName =
- FirstName +
- LastName

S2 elements:
- Customer
- CustID
- Company
- Contact
- Phone
What Goes into a Match Decision?

- Data values
  - May find common patterns or phrases in data values
- Element names
- Constraint information
- Structural information
- Domain knowledge: Synonyms, related terms, etc.
- Cardinality relationship between elements

*What are implications for instance-level vs. schema-level?*
What Makes Matching Complicated 1/2

How do we deal with partial and composite matches?

Contact
  email
  street
  city
  state
  zip
  hphone
  wphone
  fax

BillAddress
  street
  city
  stateOrProvince
  zipOrRegion
  country
  phone
  fax

ShipAddress
  street
  city
  stateOrProvince
  zipOrRegion
  country
May have different levels of representation:

```
MealsRequested
  breakfast
  lunch
  dinner

MealList: set of {
  time, order
}
```
Approaches People Have Used

Schema Matching Approaches

Individual matcher approaches
- Schema-only based
  - Element-level
    - Linguistic
    - Constraint-based
  - Structure-level
    - Constraint-based
- Instance/contents-based
  - Element-level
    - Linguistic
    - Constraint-based

Combining matchers
- Hybrid matchers
- Composite matchers
  - Manual composition
  - Automatic composition

Further criteria:
- Match cardinality
- Auxiliary information used ...

Sample approaches

- Name similarity
- Description similarity
- Global namespaces
- Type similarity
- Key properties
- Graph matching
- IR techniques (word frequencies, key terms)
- Value pattern and ranges
An Example Matcher: 
LSD (Doan, Domingos, Halevy)

A “composite matcher” for mapping data sources to a mediated schema

- Train with mappings from a few sources; let it run on the rest
- Tries to combine information from many different approaches – “multi-strategy learning”
- Favors the approaches that give the best results
  - Uses a machine learning approach called “stacking”
Example Matching Problem

Mediated schema

<table>
<thead>
<tr>
<th>price</th>
<th>agent-name</th>
<th>agent-phone</th>
<th>office-phone</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Schema of realestate.com

<table>
<thead>
<tr>
<th>listed-price</th>
<th>contact-name</th>
<th>contact-phone</th>
<th>office</th>
<th>comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>$250K</td>
<td>James Smith</td>
<td>(305) 729 0831</td>
<td>(305) 616 1822</td>
<td>Fantastic house</td>
</tr>
<tr>
<td>$320K</td>
<td>Mike Doan</td>
<td>(617) 253 1429</td>
<td>(617) 112 2315</td>
<td>Great location</td>
</tr>
<tr>
<td>$350K</td>
<td>(206) 634 9435</td>
<td>Beautiful yard</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$230K</td>
<td>(617) 335 4243</td>
<td>Close to Seattle</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

homes.com

<table>
<thead>
<tr>
<th>sold-at</th>
<th>contact-agent</th>
<th>extra-info</th>
</tr>
</thead>
<tbody>
<tr>
<td>$350K</td>
<td>(206) 634 9435</td>
<td>Beautiful yard</td>
</tr>
<tr>
<td>$230K</td>
<td>(617) 335 4243</td>
<td>Close to Seattle</td>
</tr>
</tbody>
</table>

“fantastic” & “great” occur frequently in data instances => description

“office” occurs in name => office-phone
The LSD Architecture

**Training Phase**

- Mediated schema
- Source schemas
- Training data for base learners

- Base-Learner₁
- Base-Learner₂
- Base-Learnerₖ

- Hypothesis₁
- Hypothesis₂
- Hypothesisₖ

- Meta-Learner

- Weights for Base Learners

**Matching Phase**

- Base-Learner₁
- Base-Learner₂
- Base-Learnerₖ

- Meta-Learner

- Prediction Combiner

- Constraint Handler

- Mappings

- Domain constraints

- Predictions for instances

- Predictions for elements
Training the Base Learners

Mediated schema

<table>
<thead>
<tr>
<th>location</th>
<th>price</th>
<th>contact-name</th>
<th>contact-phone</th>
<th>office</th>
<th>comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Miami, FL</td>
<td>$250K</td>
<td>James Smith</td>
<td>(305) 729 0831</td>
<td>(305) 616 1822</td>
<td>Fantastic house</td>
</tr>
<tr>
<td>Boston, MA</td>
<td>$320K</td>
<td>Mike Doan</td>
<td>(617) 253 1429</td>
<td>(617) 112 2315</td>
<td>Great location</td>
</tr>
</tbody>
</table>

Name Learner

- (“location”, address)
- (“price”, price)
- (“contact name”, agent-name)
- (“contact phone”, agent-phone)
- (“office”, office-phone)
- (“comments”, description)

Naive Bayes Learner

- (“Miami, FL”, address)
- (“$250K”, price)
- (“James Smith”, agent-name)
- (“(305) 729 0831”, agent-phone)
- (“(305) 616 1822”, office-phone)
- (“Fantastic house”, description)
- (“Boston, MA”, address)
Stacking

Training
- uses training data to learn weights
- one for each (base-learner, mediated-schema element) pair
- weight (Name-Learner, address) = 0.2
- weight (Naive-Bayes, address) = 0.8

Matching: combine predictions of base learners
- computes weighted average of base-learner confidence scores

<table>
<thead>
<tr>
<th>area</th>
<th>Name Learner</th>
<th>Naive Bayes</th>
<th>Meta-Learner</th>
</tr>
</thead>
<tbody>
<tr>
<td>Seattle, WA</td>
<td>— (address,0.4)</td>
<td>— (address,0.9)</td>
<td>— (address, 0.4<em>0.2 + 0.9</em>0.8 = 0.8)</td>
</tr>
<tr>
<td>Kent, WA</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bend, OR</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Combining Info from the Learners

homes.com schema

<table>
<thead>
<tr>
<th>area</th>
<th>sold-at</th>
<th>contact-agent</th>
<th>extra-info</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Name Learner

Naive Bayes

Meta-Learner

(address,0.8), (description,0.2)

(address,0.6), (description,0.4)

(address,0.7), (description,0.3)

Prediction-Combiner

(address,0.7), (description,0.3)

(price,0.9), (agent-phone,0.1)

(agent-phone,0.9), (description,0.1)

(address,0.6), (description,0.4)
Does It Work?

LSD’s accuracy: 71 - 92%

This is pretty good but far from perfect:

- Sometimes, even a human may not do better:
  - Some matches need an expert to determine
  - Some things are inherently ambiguous
- But sometimes a human can do better!
- This helps simplify the process of finding matches, but it’s not a panacea

Current hot topic: how to use previous mappings to “bootstrap” new mapping creation