Schema Matching

Craig Knoblock

Based on slides by AnHai Doan

Learning Models of the Sources:
Source Modeling vs. Schema Matching

- Schema Matching
  - Align schemas between data sources
  - Assumes static sources and complete access to data
- Source modeling
  - Incrementally build models from partial data (e.g., web services, html forms, programs)
  - Model not just the fields but the source types and even the function of a source
  - Support richer source models (a la Semantic Web)

Road Map

- Schema matching motivation & problem definition
- Representative current solutions: LSD, iMAP, Clio
- Broader picture and conclusions

Motivation: Data Integration

New faculty member

Find houses with 2 bedrooms priced under 200K

Architecture of Data Integration System

Semantic Matches between Schemas
### Schema Matching is Ubiquitous!
- **Fundamental problem in numerous applications**
- **Databases**
  - data integration
  - data translation
  - schema/view integration
  - data warehousing
  - semantic query processing
  - model management
  - peer data management
- **AI**
  - knowledge bases, ontology merging, information gathering agents, ...
- **Web**
  - e-commerce
  - marking up data using ontologies (e.g., on Semantic Web)

### Why Schema Matching is Difficult
- Schema & data never fully capture semantics!
  - not adequately documented
  - schema creator has retired to Florida!
- Must rely on clues in schema & data
  - using names, structures, types, data values, etc.
- Such clues can be unreliable
  - same names => different entities: area => location or square-feet
  - different names => same entity: area & address => location
- Intended semantics can be subjective
  - house-style = house-description?
  - military applications require committees to decide!
- Cannot be fully automated, needs user feedback!

### Current State of Affairs
- Finding semantic mappings is now a key bottleneck!
  - largely done by hand
  - labor intensive & error prone
  - data integration at GTE [Li&Clifton, 2000]
    - 40 databases, 27,000 elements, estimated time: 12 years
- Will only be exacerbated
  - data sharing becomes pervasive
  - translation of legacy data
- Need semi-automatic approaches to scale up!
- Many research projects in the past few years
  - Databases: IBM Almaden, Microsoft Research, BYU, George Mason, U of Leipzig, U Wisconsin, NCSU, UIUC, Washington, ...
  - AI: Stanford, Karlsruhe University, NEC Japan, ...

### Road Map
- Schema matching motivation & problem definition
- Representative current solutions: LSD, iMAP, Clio
- Broader picture and conclusions

### LSD
- Learning Source Description
- Developed at Univ of Washington 2000-2001
  - with Pedro Domingos and Alon Halevy
- Designed for data integration settings
  - has been adapted to several other contexts
- Desirable characteristics
  - learn from previous matching activities
  - exploit multiple types of information in schema and data
  - incorporate domain integrity constraints
  - handle user feedback
  - achieves high matching accuracy (86 -- 97%) on real-world data

### Schema Matching for Data Integration: the LSD Approach
*Suppose user wants to integrate 100 data sources*

1. User
   - manually creates matches for a few sources, say 3
   - shows LSD these matches
2. LSD learns from the matches
3. LSD predicts matches for remaining 97 sources
Learning from the Manual Matches

Schema of realestate.com

Mediated schema

<table>
<thead>
<tr>
<th>listed-price</th>
<th>contact-name</th>
<th>contact-phone</th>
<th>office-phone</th>
<th>description</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>
</tbody>
</table>

If "fantastic" & "great" occur frequently in data instances => description

Mediated schema

<table>
<thead>
<tr>
<th>listed-price</th>
<th>contact-name</th>
<th>contact-phone</th>
<th>office-phone</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>
</tbody>
</table>

Multi-Strategy Learning

- Use a set of base learners
  - each exploits well certain types of information
- To match a schema element of a new source
  - apply base learners
  - combine their predictions using a meta-learner
- Meta-learner
  - uses training sources to measure base learner accuracy
  - weights each learner based on its accuracy

The LSD Architecture

Matching Phase

Base-Learner → Base-Learner

Meta-Learner → Weights for Base Learners

Domain constraints

Prediction

Constraints Handler

Base-Learner

Matching instances

Training Phase

Mediated schema

<table>
<thead>
<tr>
<th>listed-price</th>
<th>contact-name</th>
<th>contact-phone</th>
<th>office-phone</th>
<th>description</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>
</tbody>
</table>

Must Exploit Multiple Types of Information!

Mediated schema

<table>
<thead>
<tr>
<th>listed-price</th>
<th>contact-name</th>
<th>contact-phone</th>
<th>office-phone</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>
</tbody>
</table>

If "office" occurs in name => office-phone

Base Learners

- Training

Object

Training examples

Hypothesis

Weights for Base Learners

Constraint Handler

Predictions for instances

Base-Learner

Matching elements

Name Learner

- training: 
  - "location", address
  - "contact name", name
- matching: agent-name (name,0.7), (phone,0.3)

Naive Bayes Learner

- training: 
  - "Seattle, WA", address
  - "250K", price
- matching: "Kent, WA" (address,0.8), (name,0.2)

Training the Base Learners

<table>
<thead>
<tr>
<th>location</th>
<th>price</th>
<th>contact-name</th>
<th>contact-phone</th>
<th>office-phone</th>
<th>description</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>$230K</td>
<td>Mike Doan</td>
<td>(617) 253 1429</td>
<td>(617) 112 2315</td>
<td>Great location</td>
</tr>
</tbody>
</table>

Learning the Manual Matches

Mediated schema

<table>
<thead>
<tr>
<th>listed-price</th>
<th>contact-name</th>
<th>contact-phone</th>
<th>office-phone</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>
</tbody>
</table>

If "fantastic" & "great" occur frequently in data instances => description

Mediated schema

<table>
<thead>
<tr>
<th>listed-price</th>
<th>contact-name</th>
<th>contact-phone</th>
<th>office-phone</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>
</tbody>
</table>

If "office" occurs in name => office-phone

Base Learners

- Training

Object

Training examples

Hypothesis

Weights for Base Learners

Constraint Handler

Predictions for instances

Base-Learner

Matching elements

Name Learner

- training: 
  - "location", address
  - "contact name", name
- matching: agent-name (name,0.7), (phone,0.3)

Naive Bayes Learner

- training: 
  - "Seattle, WA", address
  - "250K", price
- matching: "Kent, WA" (address,0.8), (name,0.2)

Training the Base Learners

<table>
<thead>
<tr>
<th>location</th>
<th>price</th>
<th>contact-name</th>
<th>contact-phone</th>
<th>office-phone</th>
<th>description</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>$230K</td>
<td>Mike Doan</td>
<td>(617) 253 1429</td>
<td>(617) 112 2315</td>
<td>Great location</td>
</tr>
</tbody>
</table>
Meta-Learner: Stacking

[Wolpert 92, Ting&Witten 99]

- 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>Name Learner</th>
<th>Weight (address)</th>
<th>Naive Bayes</th>
<th>Weight (address)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Seattle, WA</td>
<td>0.4</td>
<td>Kent, WA</td>
<td>0.9</td>
</tr>
<tr>
<td>Bend, OR</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Matching: combine predictions of base learners

Area: (address, 0.7), (description, 0.3)
Name Learner 0.4
Naive Bayes 0.9
Meta-Learner 0.2

The LSD Architecture

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

Matching Phase
- Predictions for instances
- Domain constraints
- Predictions for elements
- Constraints Handler
- Mappings

Domain Constraints
- Encode user knowledge about domain
- Specified only once, by examining mediated schema
- Examples
  - at most one source-schema element can match address
  - if a source-schema element matches house-id then it is a key
  - \( \text{avg-value(price)} > \text{avg-value(num-baths)} \)
- Given a mapping combination
  - can verify if it satisfies a given constraint

Area: address
Sold-at: price
Contact-agent: agent-phone
Extra-info: address

The Current LSD System
- Can also handle data in XML format
- matches XML DTDs
- Base learners
  - Naive Bayes [Duda & Hart, Domingos & Pazzani, 1997]
  - exploits frequencies of words & symbols
  - WHIRL Nearest-Neighbor Classifier [Cohen & Hersh KDD-98]
    - employs information-retrieval similarity metric
  - Name Learner [SIGMOD-01]
    - matches elements based on their names
  - County-Name Recognizer [SIGMOD-01]
    - stores all U.S. county names
  - XML Learner [SIGMOD-01]
    - exploits hierarchical structure of XML data
Empirical Evaluation

- Four domains
  - Real Estate I & II, Course Offerings, Faculty Listings
- For each domain
  - created mediated schema & domain constraints
  - chose five sources
  - extracted & converted data into XML
  - mediated schemas: 14 - 66 elements, source schemas: 13 - 48
- Ten runs for each domain, in each run:
  - manually provided 1-1 matches for 3 sources
  - asked LSD to propose matches for remaining 2 sources
  - accuracy = % of 1-1 matches correctly identified

High Matching Accuracy

LSD’s accuracy: 71 - 92%
- Best single base learner: 42 - 72%
- + Meta-learner: 5 - 22%
- + Constraint handler: 7 - 13%
- + XML learner: 0.8 - 6%

Contribution of Schema vs. Data

LSD Summary

- LSD
  - learns from previous matching activities
  - exploits multiple types of information
  - by employing multi-strategy learning
  - incorporates domain constraints & user feedback
  - achieves high matching accuracy
- LSD focuses on 1-1 matches
- Next challenge: discover more complex matches!
  - iMAP (illinois Mapping) system [SIGMOD-04]
  - developed at Washington and Illinois, 2002-2004
  - with Robin Dhamanka, Yoonkyong Lee, Alon Halevy, Pedro Domingos

The iMAP Approach

- For each mediated-schema element
  - searches space of all matches
  - finds a small set of likely match candidates
  - uses LSD to evaluate them
- To search efficiently
  - employs a specialized searcher for each element type
  - Text Searcher, Numeric Searcher, Category Searcher, ...

The iMAP Architecture [SIGMOD-04]
### An Example: Text Searcher

- **Beam search** in space of all concatenation matches
- Example: find match candidates for address

<table>
<thead>
<tr>
<th>listed-price</th>
<th>agent-id</th>
<th>full-baths</th>
<th>half-baths</th>
<th>city</th>
<th>zipcode</th>
</tr>
</thead>
<tbody>
<tr>
<td>320K</td>
<td>532a</td>
<td>2</td>
<td>1</td>
<td>Seattle</td>
<td>98105</td>
</tr>
<tr>
<td>240K</td>
<td>115c</td>
<td>1</td>
<td>1</td>
<td>Miami</td>
<td>23591</td>
</tr>
</tbody>
</table>

**Mediated-schema**

<table>
<thead>
<tr>
<th>price ($)</th>
<th>num-baths</th>
<th>address</th>
</tr>
</thead>
<tbody>
<tr>
<td>320,000</td>
<td>532a</td>
<td>98105</td>
</tr>
<tr>
<td>240,000</td>
<td>115c</td>
<td>23591</td>
</tr>
</tbody>
</table>

**Best match candidates for address**

- (agent-id,0.7), (concat(agent-id,city),0.75), (concat(city,zipcode),0.9)

### Empirical Evaluation

- **Current iMAP system**
  - 12 searchers
- **Four real-world domains**
  - real estate, product inventory, cricket, financial wizard
  - target schema: 19 -- 42 elements, source schema: 32 -- 44
- **Accuracy**: 43 -- 92%
- **Sample discovered matches**
  - agent-name = concat(first-name,last-name)
  - area = building-area / 43560
  - discount-cost = (unit-price * quantity) * (1 - discount)

More detail in [Dhamanka et. al. SIGMOD-04]

### Observations

- Finding complex matches much harder than 1-1 matches!
  - require gluing together many components
  - e.g., num-rooms = bath-rooms + bed-rooms + dining-rooms + living-rooms
  - if missing one component => incorrect match
- However, even partial matches are already very useful!
  - so are top-k matches => need methods to handle partial/top-k matches
- Huge/infinite search spaces
  - domain knowledge plays a crucial role!
- Matches are fairly complex, hard to know if they are correct
  - must be able to explain matches
- Human must be fairly active in the loop
  - need strong user interaction facilities
- Break matching architecture into multiple "atomic" boxes!

### Finding Matches is only Half of the Job!

- To translate data/queries, need mappings, not matches

<table>
<thead>
<tr>
<th>location</th>
<th>price ($)</th>
<th>agent-id</th>
</tr>
</thead>
<tbody>
<tr>
<td>Raleigh, NC</td>
<td>410,000</td>
<td>15</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>area</th>
<th>price ($)</th>
<th>agent-address</th>
<th>agent-name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Atlanta, GA</td>
<td>380,000</td>
<td>Mike Brown</td>
<td></td>
</tr>
</tbody>
</table>

**Mappings**

- area = SELECT location FROM HOUSES
- agent-address = SELECT concat(city, state) FROM AGENTS
- list-price = price * (1 + fee-rate)

To translate data/queries, need mappings, not matches

### Clio: Elaborating Matches into Mappings

- Developed at Univ of Toronto & IBM Almaden, 2000-2003
  - by Renee Miller, Laura Haas, Mauricio Hernandez, Lucian Popa, Howard Ho, Ling Yan, Ron Fagin

**Given a match**

- list-price = price * (1 + fee-rate)

**Refine it into a mapping**

- list-price = SELECT price * (1 + fee-rate) FROM HOUSES (FULL OUTER JOIN) AGENTS WHERE agent-id = id

**Need to discover**

- the correct join path among tables, e.g., agent-id = id
- the correct join, e.g., full outer join? inner join?

**Use heuristics to decide**

- when in doubt, ask users
  - employ sophisticated user interaction methods [VLDB-00, SIGMOD-01]
Road Map

- Schema matching motivation & problem definition
- Representative current solutions: LSD, iMAP, Clio
- Broader picture and conclusions

Broader Picture: Find Matches

Hand-crafted rules
Exploit schema
1:1 matches
- TRANSCM [Milo & Zahor98]
- MATEMS [Zaitsev & Antenucci09]
- CLUSA [Palagoni et al. 08]
- CUPID [Mudur et al. 01]

Single learner
Exploit data
1:1 matches
- SEMINT [Liu & Clifton04]
- ILA [Perkowitz & Etzioni95]
- DELTA [Clifton et al. 97]
- AutoMatcher, Amply
- [Blei & Lafferty, 04-05]

Learners + rules, use multi-strategy learning
Exploit schema + data
1:1 + complex matches
Exploit domain constraints
- LSD [Doan et al., SIGMOD-01]
- iMAP [Dhamanka et al., SIGMOD-04]
- SEMINT [Liu & Clifton04]
- DELTA [Clifton et al. 97]
- AutoMatcher, Amply
- [Blei & Lafferty, 04-05]

Broader Picture: From Matches to Mappings

Rules
- Exploit data
- Powerful user interaction

Learners + rules
- Exploit schema + data
- 1:1 + complex matches
- Automate as much as possible

iMAP [Dhamanka et al., SIGMOD-04]

Need Much More Domain Knowledge

- Where to get it?
  - past matches (e.g., LSD, iMAP)
  - other schemas in the domain
  - holistic matching approach by Kevin Chang group [SIGMOD-02]
  - corpus-based matching by Alon Halevy group [IJCAI-03]
  - clustering to achieve bridging effects by Clement Yu group [SIGMOD-04]
  - external data (e.g., iMAP at SIGMOD-04)
  - mass of users (e.g., MOBS at WebDB-03)
- How to get it and how to use it?
  - no clear answer yet

Summary

- Schema/ontology matching:
  - key to numerous data management problems
  - much attention in the database, AI, Semantic Web communities
- Simple problem definition, yet very difficult to do
  - no satisfactory solution yet
  - AI complete?
- We now understand the problems much better
  - still at the beginning of the journey
  - will need techniques from multiple fields