The Schema Matching Problem

- The problem of generating correspondences between elements of two schemas
### Basic Inputs to Matching Techniques

- **Element names**
- **Schema structure**
- **Constraints: data type, keys, nullability**

#### BookInfo
- ID: char(15) key
- AuthorID: integer references AuthorInfo
- BookTitle: varchar(150)
- ListPrice: float
- DiscountPrice: float

<table>
<thead>
<tr>
<th>Authors</th>
</tr>
</thead>
<tbody>
<tr>
<td>LastName: varchar(25)</td>
</tr>
<tr>
<td>FirstName: varchar(25)</td>
</tr>
</tbody>
</table>

#### Books
- ISBN: char(15) key
- Title: varchar(100)
- Author: varchar(50)
- MarkedPrice: float

### Other Inputs to Basic Matching

- **Synonyms**
  - Code = Id = Num = No
  - Zip = Postal [code]
  - Node = Server

- **Acronyms**
  - PO = Purchase Order
  - UOM = Unit of Measure
  - SS# = Social Security Number

- **Data instances**
  - Elements match if they have similar instances or value distributions
Many Apps Need Correspondences

- Data translation
- Data integration
- ER design tools
- Schema evolution
- Object-to-relational mapping
- XML message translation
- Data warehouse loading (ETL)

Semantics of Correspondences

- A correspondence is just a relationship, with no semantics
- Correspondences can be directly useful
  - Schema merging, impact analysis, ...
- Or they can be semantically enriched
  - Clio project [Miller et al., VLDB 2000]
  - Translate correspondences into constraints on instances
  - Then translate constraints into an executable mapping
Example

\[ \pi_{ISBN, Title, MarkedPrice} (Books) = \pi_{ID, BookTitle, ListPrice} (BookInfo) \]
\[ \pi_{Author} (Books) = \pi_{FirstName + LastName} (AuthorInfo) \]

Example (continued)

Books

\[ = \pi_{ID, BookTitle, FirstName + LastName, ListPrice} (BookInfo \bowtie AuthorInfo) \]
History

- 1994–98, I worked on Microsoft Repository

- I talked to many tool developers
  - They were all working with models of software artifacts and mappings between them

- This led me to propose Model Management
  - Bulk operators to manipulate models & mappings
    - Match, Merge, Diff, Compose, Invert, ModelGen, …
  - [Bernstein, Halevy, Pottinger, SIGMOD Record ’00]

Model Management Scenarios

- They’re all multi-step
  - The first step usually generates a mapping: $S \rightarrow \text{map} \rightarrow T$
  - Then merge($S, T$), diff($S, T$), compose($S' \rightarrow \text{map} \rightarrow S, S \rightarrow \text{map} \rightarrow T$)

- So the Match operator was the place to start.
  - Survey the literature
  - Develop new match algorithms

- We found existing work on schema matching was embedded in other multi-step solutions
Schema Matching is an Independent Problem

- It was one of our contributions
- There are now hundreds of papers on the topic
- The problem can’t be solved perfectly because
  - It depends on the available information
  - It depends on the required accuracy
  - It depends on the application and usage scenario
- So it’s no wonder our paper is highly cited!

Outline

- Problem definition
- History – what led us to the problem
  - Summary of our 2001 paper (Jayant Madhavan)
  - Approaches since 2001 & Future trends (Erhard Rahm)
Goals and Contributions

Our original goals
- Introduce schema matching as an independent problem and independent component
- Provide a credible candidate algorithm and implementation as a basis for future work
- Generic: independent of data model and target application

Our contributions
- Taxonomy of schema matching algorithms
- Schema-based hybrid matching algorithm
- Evaluation that compared multiple approaches
Cupid overview

Schema-based hybrid matching algorithm
- Combines multiple approaches that use only schema (no instances)

Input: Two schema graphs
Output: Similarity matrix and candidate mapping

- Linguistic matching: compare elements based on names
- Structure matching: compare elements based on relationships

\[ W_{\text{sim}} = w \times L_{\text{sim}} + (1 - w) \times S_{\text{sim}} \]

- Not the first to propose either linguistic or structure matching

Example from VLDB’01

PO
- POShipTo
  - Name
  - Street
- City

POLines
- Item
  - Line
  - UoM
  - Qty

PurchaseOrder
- Items
  - Item
    - ItemNumber
    - UnitOfMeasure
  - Quantity
- DeliverTo
  - Name
    - Address
  - City
  - Street
Linguistic Matching

- Tokenization of names
  - PurchaseOrder \rightarrow purchase + order
- Expansion of acronyms
  - UOM \rightarrow unit + of + measure
- Clustering based on keywords and data-types
  - Street, City, POAddress \rightarrow Address

- Linguistic similarity
  - Pair-wise comparison of elements that belong to the same cluster
  - Token similarity = f(string matching, synonymy score)
  - Token set similarity = average (best matching token similarity)

- Thesaurus: acronyms, synonyms, stop words and categories

Structure Matching

\[ \text{PO} \rightarrow \text{POLines} \rightarrow \text{Item} \rightarrow \text{Line} \rightarrow \text{Qty} \rightarrow \text{UoM} \rightarrow \text{UnitOfMeasure} \rightarrow \text{Quantity} \]
\[ \text{PurchaseOrder} \rightarrow \text{Items} \rightarrow \text{Item} \rightarrow \text{Name} \rightarrow \text{Address} \rightarrow \text{City} \rightarrow \text{Street} \]
Tree Match Algorithm

- Atomic elements (leaves) are similar
  - Linguistically and data-type similar
  - Their contexts, i.e., ancestors, are similar

- Compound elements (non-leaves) are similar if
  - Linguistically similar
  - Elements in their context, i.e., subtrees rooted at the elements, are similar

- Mutually dependent formulation
  - Leaves determine internal node similarity
  - Similarity of internal nodes leads to increase in leaf similarity

- Bottom-up traversal of trees

Tree Match: Mutually Reinforcing Similarity

Extensions for shared types, referential integrity, views, etc.
Evaluation

- Cupid compared with MOMIS/ARTEMIS @ Modena/Milano, DIKE @Calabria
- Canonical tasks and real world examples

Technical conclusions
- Linguistic matching with attention to detail does help
- Structure matching can identify non-linguistic matches
- Structure matching can disambiguate between seemingly identical structures in different contexts
- Ability to match across relational schemas, XML variants, possibly others

What we learned?
- Schema Matching Taxonomy
  - Provided a framework to describe future solutions and place them in comparison to other work
- Quantitative evaluation
  - Set a precedent for future papers
  - Very thankful to MOMIS/ARTEMIS and DIKE teams
- Making software available helps a lot
  - Possible even when developed in industry
  - We get requests for software even to this day
Follow up Techniques

- Using schema matching results as is: possible when matches only contribute implicitly end-user task
- For example, building a deep-web crawler [Madhavan+, VLDB’08]

- Design mediated schema
- Extract schemas of web forms
- Match web forms to mediated schemas
- Generate URLs for interesting subset of form submissions
- Add generated pages to the corpus of indexed pages

Collective Schema Matching

Schema matching is almost never an isolated task
It ought to get easier over time!

- [Doan+, SIGMOD’01]: Learn to match sources to a mediated schema
- [Do+, ICDE’02]: Compose known matches to discover new ones
Collective Schema Matching

- [He+, SIGMOD’03]: Build mediated schema for a domain by clustering elements in multiple schemas

- [Madhavan+, ICDE’05]: Learn to map between new schemas based on other schemas and mappings in the same domain

Progress in many areas

- Match workflows
- New match techniques
- User interaction for Match
- Semantic matching
- Match techniques for large schemas
- Self-tuning match workflows
- Reuse-oriented matching
- Holistic (collective) schema matching
- Numerous match prototypes
- Evaluation of match tools
- Commercial tools
Schema matching is a multi-step process

**General workflow** *(COMA, …)*

1. **Input schemas**
2. **Pre-processing**
3. **Matcher Execution**
4. **Combination of matcher results**
5. **Selection of correspondences**
6. **Result Mapping**

**Matcher sub-workflows**

- **Sequential matchers**
- **Parallel (independent) matchers**
- **Mixed strategy**

**New match techniques**

- **Graph matching**
  - e.g., similarity flooding [Melnik et al, ICDE 2002]

- **Instance-based ontology matching**
  - concepts with similar instances should match
  - consider all instances of a concept as a document and utilize document similarity (e.g., TF/IDF) to find matching concepts

- **Usage-based matching**
  - utilize query logs for hints about related schema elements (e.g., in join clauses) [Elmeleegy et al., ICDE 2008]
  - Hamster approach for taxonomy matching [Nandi et al, VLDB 2009]
Instance-based ontology matching

- Concepts with most similar instances should match
  - requires shared/similar instances for most concepts
- Mutual treatment of entity resolution (instance matching) and ontology matching
- Promising for link discovery in the Linked Open Web of Data

User interaction for Match

- **GUI support** to inspect and correct computed correspondences [Falconer et al., ISWC 2007]
- **Incremental schema matching** [Bernstein et al., VLDB 2006]
  - focused matching on user-selected element / subtree
- Provision of **top-k matches** per element for selection [Gal, J Data Semantics 2006]
- **Collaborative schema matching** using a wiki-like infrastructure to provide and improve mappings [McCann et al., ICDE 2008]
Semantic matching

- Correspondences with semantic relationships
  - equality, more general, less general, disjointness
  - e.g. PortableComputers \supseteq Tablets

- Discovery of mapping expressions
  - e.g., room-price = room-rate * (1 + tax-rate)
  - iMAP [Dhamankar et al., SIGMOD 2004]

- Conditional correspondences [Bohannon et al., VLDB 2006]
  - e.g., if productType = "book"
    then S1.Invoice.Code = S2.ISBN

Match techniques for large schemas

- Low-level optimizations
  - Optimized string matching
  - Space-efficient similarity matrices
  - Database-based matching

- Parallel matching
  - Inter-matcher and intra-matcher parallelism

- Partition-based matching (COMA++, Falcon-AO)
  - Reduced search space by matching only similar schema partitions/fragments
  - Light-weight search for similar schema fragments
Initially determine highly similar element pairs called “anchors”
Only partitions that share at least one anchor are matched

Self-tuning match workflows

- Semi-automatic configuration
  - Selection of promising matchers
  - Ordering of different matchers
  - Combination of match results
  - Selection of correspondences (top-k, threshold, ...)

- Initial tuning frameworks: Apfel, eTuner, YAM
- Use of supervised machine learning
  - need previously solved match problems for training
  - difficult to support large schemas
- Heuristic approaches
  - Use linguistic and structural similarity of input schemas to select matchers and their weights (RiMOM)
  - Favor matchers giving higher similarity values in the combination of matcher results (QOM, PRIOR+, OpenII)
Reuse–oriented Matching

- Many similar match tasks → reuse previous matches
  - Schema and mapping repository needed
- Example: reuse match results after schema evolution
  - compose previous match result $S \rightarrow T$ with mapping $T \rightarrow T'$ to solve new match task $S \rightarrow T'$

First proposals for reuse at 3 mapping granularities

- Reuse complete schema mappings, e.g. after schema evolution
- Reuse individual element correspondences, e.g. synonyms
- Reuse mappings between schema fragments

Fragment–level reuse most sophisticated

- Populate repository by most relevant fragments and their mappings
- Analyze schemas to be matched for fragment pairs in the repository
- Assemble and complement fragment mappings
Holistic (collective) schema matching

- Matching between N schemas, e.g. web forms
  - mostly simple schemas
- Typical use case: creation of a mediated schema

- Holistic matching based on clustering of similar attributes (Wise-Integrator, DCM, HSM, …)
  - utilize high name similarity between schemas
  - similar names within a schema are mismatches

- Probabilistic mediated schemas
  [Das Sarma et al., SIGMOD 2008]
  - Ranking of several clustering alternatives based on probabilistic mappings
  - Fully automatic approach

Research match prototypes
Bencharking Initiative OAEI*

- Yearly ontology matching contests since 2005
- Up to 17 participating systems per year
- Simple tests (Benchmark) and larger test cases (Anatomy, Directory)
- Improvements for Benchmark and Anatomy, but not for Directory

<table>
<thead>
<tr>
<th>System</th>
<th>2007</th>
<th>2008</th>
<th>2009</th>
<th>2010</th>
</tr>
</thead>
<tbody>
<tr>
<td>AFlood</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>AgrMaker</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>AROMA</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>AOAS</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>ASMOV</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>BLOOMS</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>CODI</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>DSSim</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>El2Match</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Falcon AO</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>GeRMeSMB</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Kosimap</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Lily</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>NBILM</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Prior+</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>RIMOM</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>SAMBO</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>SOBOM</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>TaxoMap</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>X SOM</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

Avg. F-measure: 0.598, 0.718, 0.764, 0.785

Anatomy test case


Match Prototype Comparison*

<table>
<thead>
<tr>
<th></th>
<th>Cupid</th>
<th>COMA++</th>
<th>Falcon</th>
<th>Rimom</th>
<th>Asmov</th>
<th>Agr.Maker</th>
<th>Oil Harmony</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input relational</td>
<td>✓</td>
<td>✓</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>✓</td>
</tr>
<tr>
<td>schemas XML</td>
<td>✓</td>
<td>✓</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>✓</td>
</tr>
<tr>
<td>ontologies linguistic</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>OAEI participation compreh. GUI</td>
<td>-</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>-</td>
</tr>
<tr>
<td>Matchers structure Instance</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>-</td>
</tr>
<tr>
<td>use of ext.dictionaries</td>
<td>✓</td>
<td>✓</td>
<td>-</td>
<td>-</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>schema partitioning</td>
<td>-</td>
<td>✓</td>
<td>✓</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>parallel matching</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>dyn. matcher selection</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>✓</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>mapping reuse</td>
<td>-</td>
<td>✓</td>
<td>✓</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Commercial schema matching tools

- Many GUI-based mapping editors to manually specify correspondences and mappings

- Initial support for automatic matching, in particular linguistic matching
  - Altova MapForce
  - MS BizTalk Server 2010
  - SAP Netweaver
  - IBM Infosphere

- Many further improvements possible
  - Structural / instance-based matching
  - Advanced techniques for large schemas

BizTalk 2010 Screenshot

Indicative match result for selected node PO403
Books

Google Scholar: Paper counts

more than 5000 publications for keyword “schema matching” since year 2000
Remaining research challenges (1)

- Joint treatment of entity resolution and schema matching, e.g. for Linked Data
- More comprehensive mapping reuse
- Self-Tuning
- Improvements for
  - user interaction
  - Large-scale schema matching
  - Semantic matching
  - Holistic/collective schema matching ...

Remaining research challenges (2)

- Fully automatic schema matching for web applications
- More match-based approaches for
  - Ontology/schema merging
  - Ontology/schema evolution
  - ...