College of Management Working Papers and Reports

College of Management

9-1-2007

A Survey of Schema Matching Research

Roger Blake
University of Massachusetts Boston, roger.blake@umb.edu

A Survey of Schema Matching Research

Roger Blake

UMBCMWP 1031

September 2007

A Survey of Schema Matching Research

Abstract

Schema matching is the process of developing semantic matches between two or more schemas. The purpose of schema matching is generally either to merge two or more databases, or to enable queries on multiple, heterogeneous databases to be formulated on a single schema (Doan and Halevy 2005). This paper develops a taxonomy of schema matching approaches, classifying them as being based on a combination schema matching technique and the type of data used by those techniques. Schema matching techniques are categorized as being based on rules, learning, or ontology, and the type of data used is categorized as being based on schema elements or instance data. This taxonomy is an extension to previous work, and significant current research efforts are categorized using this taxonomy. Several of these research efforts are profiled and their categorization in the taxonomy is explored. The current research is used to identify the directions in which future research is headed.

Introduction

Schema matching is the process of identifying semantic mappings, or correspondences, between two or more schemas. Schema matching is a first step and critical part of schema integration (Doan and Halevy 2005, Rahm 2001). Schema integration is defined as the integration of local schemas into an enterprise-wide schema (Batini 1986). The resulting schema can then be in the form of a view, or schema, of an integrated database.

Schema integration, in turn, is a step towards data integration (Noy 2005). The need for data integration has been spurred by the recent explosion in data storage and networking

capabilities (Halevy 2005), and data integration has multiple important application areas. Those application areas include, but are not limited to:

Database integration – the need for database integration occurs in response to mergers and acquisitions, as part of integrating two or more systems, and as legacy systems are retired and new systems introduced. Schema matching and integration are necessary steps to integrate two or more databases in each of these cases.

Enterprise information integration – the amount of data available to an enterprise from both internal and external, including online, sources has grown dramatically (Doan and Halevy 2005). But much of this data is heterogeneous, having been developed in different contexts and with different semantics, often even for similar applications. Schema matching is a key to enabling data from these heterogeneous sources to be merged.

E-business – the traditional purpose of data integration for e-business is to share data. There are various applications – a representative example is that of catalog integration wherein product information needs to be combined. XML is a widely used standard to accomplish this, including as used for web services. But even with XML and service-oriented architectures, semantics are often not fully specified. In the case of product catalogs, for example, XML elements from different schemas with the same name – price - might be in dollars in one schema but in Euros in the other. Further, the price in one schema might be a wholesale price, the other a retail price, or each schema may use different units of measure,

and so on. Schema matching approaches need to be able to recognize such semantic differences in order to properly match such schema elements.

<u>Semantic Web</u> – reliable data integration is part of realizing the vision of a semantic web. This is especially true for the 'deep web', that part of the web which can only be accessed through web forms. Many databases are accessible only through filling in and submitting a form on the web. There are many such examples - a listing of cars for sale at a dealership, courses offered at a university, and so on. These web forms can be conceptualized as specific views accessing underlying web databases, and are relevant to schema matching because information required by a form on the web can be considered as a form of schema for the database (Halevy 2005). These underlying databases need to be integrated when data from different sites needs to be combined - for example to combine advertisements for the same items from different sites. This type of information cannot generally be accessed by web search engines precisely because of this difficult, and so is unlikely to be available. If the information on the 'deep web' were to be available through search engines, it has been estimated it would increase the total amount of information available on the web by one or two orders of magnitude (Halevy 2005).

This literature review focuses on research conducted on the first step of data integration, that of schema matching, which has been described as being at the heart of the data integration process (Rahm 2001) and is often used to develop mediated schema between

two or more data sources. A complete data integration solution starts with schema matching, and needs to consider integration architecture and query processing in addition.

Manual schema matching is a time-consuming and labor intensive task. In one example, the time required to manually integrate the 27,000 elements from 40 databases was estimated to be more than 12 person years (Li 2000). At present, schema matching is typically performed manually (Rahm 2001) or in the best case, semi-automatically, with some of the algorithms described here suggesting potential matches and humans often making the final judgment.

None of the schema matching methods reviewed in this paper have yet reached the stage of being completely automatic, and human intervention is still required. In fact, some researchers (for example, Yan, 2001, Ram 2004, Aumueller 2005, and Halevy 2005) do not foresee fully automatic schema matching as a possibility, and orient their research towards assisting human-performed schema matching. Schema matching has been identified as a problem that is "AI complete", meaning that it is as difficult to replicate as human intelligence (Bernstein 2004).

The remainder of this paper first describes the schema matching problem in more detail, develops a classification for schema matching research to date, profiles some of that research, and finally identifies directions in which research on schema matching is headed.

Schema Matching

In its simplest form, schema matching consists of identifying two elements from two different schemas as semantically equivalent, or matched. In Figure 1 below, for example, most methods would have little difficulty determining Zip from schema A and ZipCode from schema B as matches. This is an example of a match that could have been determined by examining just the element names. It is also an example of a direct match, sometimes also termed a match of 1:1 cardinality, meaning that a single element from one schema is matched to a single element in another schema.

schema A:

First	Last	Address	Zip	Phone
Carol	Frenditte	1102 Washington St, Dover, MA	02051	6174431685
Allen	LeBlanc	42 Union St, Medfield, MA	02053	5089291094
Thomas	Gutierrez	74 Chestnut St, Franklin, MA	02041	5088864218

schema B:

Name	Address	City	State	ZipCode
William J Lyttle III	441 Elm St,	Easton	MA	02356
Mr. Robert Sheridan	65 Georgetown Dr	Framingham	MA	01701
Nancy Langford	891 Dudley St	Providence	RI	02919

Figure 1: Simple schema matching based on element name

Much research has been directed at developing direct matches. But what of the other schema elements? In the example above, the address may require an indirect match, sometimes termed a match of cardinality 1:n, to match Address in schema A with Address, City, and State in schema B. Also, the first and last name elements from schema A would need to be concatenated to match the name element in schema B. Even so, one schema might have additional information not in the other, such as schema B which

includes prefix and suffix as part of the name element. In this case, only a partial match would be possible between the two schemas. This can also be seen with the element Phone in schema A, which has no corresponding match in schema B.

Moreover, some matches may complex matches, also termed matches of m:n cardinality, where multiple elements in one schema must be matched to multiple elements in the other. In the example above, if schema B represented name as prefix, first, middle initial, last, and suffix, then such a match would be required.

Considering matches of m:n cardinality increases the complexity of schema matching exponentially. Further, because data may need to be transformed in addition to only being matched through schema elements, the complexity of schema matching in general can be considered to be unbounded (Doan 2005). Consider the example in Figure 2 in which each schema has

schema C:

Item	Qty	Price	Taxes
1405	5	$110.00	$6.00
1982	3	$45.00	$2.25
2023	1	$18.00	$.90

schema D:

Item	Quantity	Price	Total
A110C	2	$11.00	$22.00
AV99x	4	$9.00	$36.00
AL129	5	$18.00	$18.00

Figure 2: An example of complex schema matching

an element named Price, yet they are not semantically equivalent. In order to match these two schemas, a matcher would have to not only discover this semantic difference, but evaluate Price and also Price + Taxes in schema C as matches for Total in schema D. This example also shows how instance data adds information to the schema matching

process, and how approaches based only on examining the schema itself may not be adequate.

This example also illustrates how data types might be used to improve the accuracy of matches. In this case, both Qty and Quantity have the same data type (integer), and this piece of schema information might be to augment other information in ascertaining matches.

The examples above are considered to be element-based schema matching, in that they determine matches without knowledge of the database structure. However, in many cases a consideration of database structure should improve schema matching.

schema E:

Badge	Name	DeptID	HomePhone
41723	Katherine Baker	172	5082307682
56784	Mark Bharati	189	6179641242
66010	Edward Waters	189	9788917692

DeptID	DeptName
172	Purchasing
189	Marketing

schema F:

EmployeeID	Name	Department
13572	Kevin Li	Accounting
20473	Julie McCormack	HR
33717	Fran Liebowitz	HR

EmployeeID	PhoneType	PhoneNumber
13572	Home	4015629982
13572	Cell	4018849010
13572	Emergency	4012399497
20473	Home	5083431884
20473	Cell	5087898386
33717	Home	6172847702

Figure 3: An example of schema matching using structure

The example in Figure 3 shows how the appropriate elements to match may need to be determined through an examination of each schema's structure. The department names to

be matched would need to come from the department entity in schema E and the employee entity in schema F. Consideration of the database structure would help enable the match of DeptName rather than DeptID from schema E since DeptName is presumably the same data type as Department in schema F, whereas DeptID would not be.

Matching the phone numbers between the two schemas shows how knowledge of the database structure is required. The phone number in schema E is in the table corresponding to employee, but there is a separate table for phone numbers in schema F that is linked to the employee table. Matching the schemas requires knowledge of this correspondence. In addition, the HomePhone element from schema E needs to be matched only to specific rows of PhoneNumber in schema F, also showing the importance of considering database structure in the schema matching process.

These examples provide some illustrations of problems encountered when attempting to match schemas. Ram and Park (Ram 2004) considered these problems to be conflicts between schemas, and separated them into schema or data level conflicts. Schema level conflicts include cases where schemas are created with different names for similar entities or attributes, or where different structures (generalization, aggregation) are used to represent similar concepts. Data level conflicts include cases where different data types or units of measure are used to represent similar data items.

The difficulty of schema matching is partially due to the fact that schemas are designed by many different people. Confronted with identical problems, two individuals may create very different schemas (Halevy 2005). In addition, even when standardized schemas are introduced, the improvement in schema matching has been limited (Halevy 2003). Furthermore, meta-data for schema – such as element name or data type – does not in general allow for a complete representation of semantics. Even seemingly descriptive element names may not capture the nuances needed for matching – such as the case above might be if the home phone being used was a cell phone, which would not be in the meta-data for either schema. Once a schema has been designed, some extent of semantic meaning is lost, and this adds potential difficulty to subsequent matching efforts.

As a result, approaches to schema matching have used many different sources of input to obtain information necessary to perform matching. Matching approaches have been developed that consider structure, data types, constraints, default and allowed values, and primary and secondary keys in addition to element names and instance data. In general, those approaches making use of the most information have had the best results (Doan 2005). The next section describes surveys from past works, and develops a new categorization of schema matching approaches by the type of data they use, and how they use it.

Schema Matching Survey and Categorization

Several notable literature reviews of schema matching are in existence, as schema matching and integration has been the subject of research efforts for more than 20 years. One of the first was by Batini, Lenzerini, and Navathe (Batini 1986), which contrasted 12 methodologies used for schema matching and integration. Of note is that none of the 12 used the data itself, or instance data, to generate matches; they were each based on analyzing schema structures only. Nor did any of these 12 use either machine learning or ontological techniques.

Rahm and Bernstein developed a taxonomy of schema matching approaches (Rahm 2001), classifying schema matching approaches into those based on analyzing the schema, those analyzing instance data, and those combining the two. The schema analysis schemes were further subdivided by what was analyzed, structure or elements. Element-level approaches were further subdivided by specific technique, being either linguistic-based (such as using element names or description similarity) or constraint-based (such as using data type similarity). Structure-level approaches were all based on constraints (such as those using data type similarity).

The instance-based approaches surveyed by Rahm and Bernstein all used element-level approaches, being either linguistic-based (such as those using word frequencies) or constraint-based (such as those using value patterns and ranges).

Rahm and Bernstein profiled seven prototype schema matchers. All seven employed multiple matching techniques, such as combining element name matching and structure matching, and so each utilized several approaches from their taxonomy. One of the seven prototypes was termed a composite matcher, using primarily schema-based matching techniques with machine learning. This prototype used machine learning to adjust weights given to the underlying matching techniques. Rahm and Bernstein also described the potential of storing results of matches for re-use, but noted that the application would not be universal because of application domain. For example, a match of salary and income might be appropriate in a human resource system but not in a related application such as a tax system.

A literature review by Shvaiko and Euzenat (Shvaiko 2005) extended this classification scheme by incorporating granularity and specific inputs used by matching techniques. The levels of granularity were adopted from those developed by Rahm and Bernstein (Rahm 2001), and inputs were considered to be terminological, structural, or semantic. The categories of matching techniques were strings, language, linguistics, constraints, alignment reuse, formal ontology, graphs, taxonomy, structure repositories, and models; techniques based on instance data were not considered. The survey compared eight schema matching systems; Shvaiko and Euzenat found that most used techniques from several of the categories which they had developed.

A more recent literature review by Doan and Halevy (Doan 2005) classified ongoing research on schema matching into rule-based and learning-based solutions. For example,

a rules-based solution might include a rule that the first and last name elements from one schema should be concatenated to form the name element in a second schema. The discovery of such rules is typically based on the examination of schema information such as element names, data types, structure, or possibly instance data. Rule-based solutions were described Doan and Halevy as relatively inexpensive, since no training step is required as with learning-based solutions.

Learning-based solutions typically use methods such as neural networks or Naïve Bayesian classification to improve the effectiveness of matches. These solutions typically utilize both schema information and instance data to determine matches, increasing the information available with which to generate matches. However, one potential drawback of learning-based solutions is that not only do they require a training data set consisting of correct matches, but they may need a large training data set to find what might be equivalent to simply stated rules, if they find those rules at all. Doan and Halevy advocated for combination rule-based and learning-based solutions as the most effective. This type of solution is similar to the composite matching system described in the survey by Rahm and Bernstein (Rahm 2001).

In keeping with the survey by Doan and Halevy, the taxonomy proposed in this literature review retains the categories of rules and learning-based approaches. However, there has been research that considers schema matching as a form of ontology alignment and uses ontological factors as part of the schema matching process. Especially in recent years, some of the research on ontology mapping has been applied specifically to the schema

matching problem. The use of either domain-specific or general ontologies to facilitate schema matches is considered here as outside of either rules-based or learning-based approaches, and so for this reason, a third category of ontology-based approaches has been added to the classifications by Doan and Halevy.

The taxonomy proposed here also considers, as has previous work such as that by Rahm and Bernstein (Rahm 2001), the classification of schema matching solutions as being either based on the use of schema information, instance data information, or a combination of both. An addition to previous work is that this survey considers two dimensions of schema matching for classification. This survey combines the dimensions of which type of data was used as input (schema or instance) with that of which type of algorithm was used. The intention was to be able to classify research approaches by the techniques and information used.

An extension to previous work is that the taxonomy proposed here classifies schema matching approaches by two dimensions. Approaches are classified as being rules-based, learning-based, or ontology-based, and by using schema information, instance data, or a combination of both.

This classification scheme, or taxonomy, and examples of research that can be considered in these classifications, is shown as Table 1. This is followed by a more detailed description of several research efforts from this classification.

	Analyzes schema	**Analyzes instances**	**Analyzes schema and instances – hybrid/composite**
Rules-based	MGS/DCM (He 2004) XML DTD fragment matching (Rahm 2004) COMA++ (Aumueller 2005) Cupid (Madhavan 2001) Element clustering (Smiljanic 2006) QMatch (Claypool 2005) Similarity Flooding (Melnik 2002)	Clio (Yan 2001) Un-interpreted structure matching (Kang 2003) Instance-based identification (Chua 2003) Matching using duplicates (Bilke 2005)	iMAP (Dhamankar 2004) Clio (Haas 2005)
Learning-based	Adaptive clustering (Cohen 2002) Automatch (Berlin 2002)	SMDD (Li 2005) SEMINT (Li 2000)	LSD (Doan 2003) Corpus-based (Madhavan 2005)
Ontology-based	SCROL (Ram 2004)		Direct and indirect matches (Xu 2003) Data frames (Embley 2004) Ontology-driven matches (Sung 2006)

Table 1: Schema matching research taxonomy and recent research examples

The following profiles several matching systems from the above taxonomy, describing their basic functionality and how they fit into the taxonomy:

iMAP (Dhamankar 2004) is a composite schema matching system using both schema and instance data in order to match a source schema to a target schema. Matches are developed for each element of a target schema, and these matches can be complex matches, involving transformations of data elements from the source database and multiple elements in that transformation. At the core of iMAP are several searchers, each which examines source data for possible matches with target data. There are iMAP

searchers for various types of data: text, numeric, categorical, and date. The data types of each element are determined by iMAP by examining instance data, rather than by examining the data dictionary. An example of a search is a text searcher which examines target schema elements as concatenations of source data elements, and uses a beam search (Russell 2003) to find the best matches. Another example is a numeric searcher, which looks for matches through arithmetic combinations, and can include more expressive functions if they are explicitly stated before hand (such as total = quantity * price + tax, etc.) A categorical searcher uses the Kullback-Leibler measure to determine the distance between source and target categorical elements.

The searchers also include a schema mismatch searcher. The mismatch searcher finds matches where two elements in the source schema may have been represented differently in the target schema. For example, a target schema may have an element describing houses for sale, and the descriptions might often include the word 'fireplace', whereas a source schema might have a separate element for fireplace which is a boolean field. The schema mismatch searcher is designed to uncover this type of situation as a potential match.

iMAP then uses a similarity estimator to combine matches generated by the searchers for each element in the target schema in order to arrive at a score for each potential match. The last step is a match selector, which examines the matches from the similarity estimator and potentially discards them if they violate constraints. An example of such a constraint is one which states that an 'Employee home phone' element in the target

schema can be matched to at most one element in the source schema, or more interestingly, that two elements should not be considered as part of the same match – for example, an element representing price and another representing year might be constrained from ever being part of the same match, even though might be numeric with some overlapping values. To some extent, these constraints could incorporate domain-specific knowledge.

iMAP is intended to be used to suggest matches to a user, with confirmed matches handed to a system such as Clio (Haas 2005) for generating mappings. A unique feature of iMAP is that it provides explanations to the user about the matches it develops. Because it retains the results of the searchers, similarity estimator, and match selector, a user can determine what caused a match to be made, why a particular match ranked higher than another match, or why a specific match was not made. A system based on iMAP was created and tested on some schema matching exercises where the overall accuracy was rated at 43-92%. Even though the schemas being matched were small (2-4 tables with 30-40 elements each), the approaches used by iMAP have promise.

LSD (Doan 2003), for Learning Source Description, is a multi-strategy approach to schema matching that combined rules and machine learning-based techniques. LSD works with XML data for which there is an associated DTD, and starts with a mediated schema to which several schema and data instances have already been mapped. It is these initial mappings to a mediated schema that form the training base for LSD. Several learners are applied to this training base, specifically a name learner, content learner,

Naïve Bayes learner, county-name learner (because of the real-estate domain to which LSD was applied), and an XML learner. Each of these learners was presented with schemas for which there were known matches to the mediated schema. For example, the names learner used a nearest-neighbor approach to determine the best matches for each tag in a learning schema to the mediated schema. The content and Naïve Bayes learners analyzed instance data, while the XML learner analyzed both tags and instance data.

Given a schema to be matched, LSD's matching phase consisted of applying the learners, each of which would generate a confidence score between every element in the source schema with every element in the mediated schema. The highest score was selected from each learner. The name learner generates a score for each DTD tag in the schema to be matched, and the content learner generates a score for the instance data associated with each DTD tag. In the case of the content learner, the scores from the naïve Bayes classifications are averaged for all data instances to produce an overall score from the content learner.

The contributions of LSD include the XML learner, which considers both tag (structure) data and instance data in combination. This was developed because experience with other learners showed that both structure and instance data improved accuracy. Another contribution of LSD is the creation of a meta-learner which combines the results of individual learners, and is capable of adjusting the weight given to each learner as it is shown more 'correct' matches. A limitation of LSD, however, is that it can be used to find only 1:1 matches; finding more complex matches is essential to any practical

matching effort. The learners and meta-learner would appear to require substantive changes to accommodate matches of these more complex matches.

Corpus-based schema matching (Madhavan 2005) uses the outputs from multiple learners as does LSD. Many of the learners are similar, including a name learner which examines fragments and n-grams of element names, a text learner which examines annotations, and an instance learner built on a text classifier. As with LSD, each learner outputs a confidence level for a pair of potential element matches, and a meta-learner combines the results from the individual learners using a sigmoid function to arrive at an overall confidence level of similarity (ranging between 0 and 1).

However, unlike LSD, the learners are applied to find similarities between each element in a schema to be matched and each schema element residing in the corpus. Corpus-based schema matching first transforms a schema to be matched by estimating similarities to elements in the corpus. These similarities are of cardinality 1:1, and so more complex matches are not considered.

The schemas in the corpus are also used for their domain constraints and conceptual representations. For example, the corpus schemas may be used to determine that 50% of the tables representing inventory have a foreign key linking to warehouse location, or that 90% these tables have columns representing quantity. Further, the tables might also be used to determine that the customer table never has a product column, for instance. In

order to establish these statistics, a clustering of schema elements is created prior to schema matching. This clustering is performed on both tables and individual columns.

These domain constraint and column statistics are applied to the similarities outputted from the meta-learner. The matching process therefore consists of calculating similarities between each schema element in the source and target schemas, and also with the corpus schemas, to which the domain constraint and column statistics are applied in the form of a cost function. This cost function enables each element in the source schema to be paired with the least cost element (or no element) in the target schema. This method was evaluated on schemas from several domains, including some XML schemas for purchase orders and customers from xml.org. Corpus-based matching was found to be an improvement over other techniques, especially when matches are more difficult due to insufficient direct evidence in target or source schemas.

The main contribution of corpus-based schema matching is in the use of previous known matches to guide the schema matching process. Specific weights used by the cost can be modified, or learned, as more schemas are added to the corpus and more correct matches are known. This can be considered a form of replicating how a human schema matcher would build experience and reference past matches. The set of known schema matches can be increased with more known matches, and so corpus-based schema matching is considered to be learning-based. As the set of known schema matches can be viewed as a separate body of domain knowledge, and because of this, corpus-based schema matching is taken to be ontology-based in this taxonomy.

Bilke and Naumann (Bilke 2005) developed a method to match schemas based on instance data, and would be classified in this taxonomy as a rules-based approach. Specifically, this approach develops matches by identifying duplicates. Duplicates are developed based on tuples, rather than by examining and matching characteristics of instance data in single columns, as do many other approaches which use instance data. There are other methods which identify duplicates based on tuples, but these are typically for other purposes. Many of these de-duplication methods, such as merge-purge, assume that schemas have already been matched and are oriented towards a specific domain. For example, merge-purge is oriented toward de-duplicating names and addresses for direct mailings.

Finding duplicate tuples was accomplished by a term-frequency inverse document frequency (TF/IDF) weight scheme, in which each token value in each tuple was weighted, and then a similarity measure calculated between each pair of tuples. The top K tuple matches in similarity were retained and a similarity matrix between each pair of columns in the two schemas to be matched was then constructed. This method considered only direct matches (those of 1:1 cardinality), and complex matches were not considered.

The Bilke and Naumann approach assumes that there are not just schemas but data instances to be matched, which may not always be the case – for example, when a mediated schema is used as a target schema, as may be the case for a data warehouse. A weakness of this approach is that schema elements that have similar data but differing

semantic meanings (such as bill-to and ship-to addresses) may be mistakenly identified as matches. Another weakness is that schema elements with data expressed in different units (kilograms vs. metric tons, for example) will not likely be matched. The contribution of this approach is in the method for measuring the duplication between tuples and columns in two schemas, and in the ability to match schemas when column names may be opaque.

SCROL (Ram 2004) applies ontology in order to detect and resolve conflicts between two or more schema and a federated schema. The ontology which is applied is not specific to a domain, and consists of concepts that are related as either peers or disjoints, or as parent-child relationships through generalization or aggregation. In this ontology, a child concept can have only one parent concept.

SCROL uses this ontology to relate schemas and to detect conflicts between them. Conflicts are in the form of mismatches between data or schema elements. An example of a data conflict would be one in which the same concept, such as square area, would be measured in acres in one schema, and square miles in another. A particular strength of SCROL appears to be in its ability to recognize matches between differently named elements that are semantically similar but require conversions in order to implement the match.

An example of a schema conflict would occur when two attributes are stored in a single table in one schema, but two tables in a second schema. SCROL takes as input a set of

pre-determined schema matches as well as individual schemas, a federated schema, and other schema information, in order to detect and reconcile conflicts.

Xu and Embley (Xu 2003) devised a method for matching XML schemas capable of detecting complex matches, or matches with cardinality m:n, where multiple elements may need to be matched between the source and target schemas. Their approach applies a domain-specific ontology to both schema and instance data. In order to do so, four basic matching techniques are used and then combined. Those four matching techniques are based on terminological relationships, data-value characteristics, domain-specific matches, and structural similarities.

Terminological relationships, based on schema element names, are identified through the use of WordNet (Miller 1995), a freely available lexical database. WordNet is used to train a set of decision rules through a decision tree, and applied to find a confidence level ranging between 0 and 1 for each pair of schema elements. Data-value characteristics are evaluated through the use of a decision tree, although based on string lengths and alphanumeric characteristics. A confidence level for each pair of schema elements is created based on data-value characteristics. The confidence levels arrived at from terminological relationships and data-value characteristics may be quite different. For example, elements from two different schemas in the real-estate domain may have the exact same name – location – but be used for an address in one schema, and a description of sites in another. This would produce a high confidence level from the terminological matcher yet a low confidence level from the data-value matcher.

The domain-specific matcher applies a specific ontology (termed an ontology snippet in other work by Xu and Embley), which describes relationships between concepts. The ontology also includes a set of regular expressions describing the expected data values for each concept (termed data frames in other work by Xu and Embley). These regular expressions are used to evaluate instance data, and matched to concepts in the ontology. From this, a determination can be made if elements from one schema are matches to a concept which is related through the ontology to a concept which has matches in the other schema. This enables complex matches involving composition, decomposition, union, and selection to be detected. The domain-specific matcher also creates a confidence level between each pair of schema elements. The structures of each schema are also evaluated to generate a fourth confidence level.

The four confidence levels from the individual matching techniques are weighted and combined to an overall confidence level. These confidence levels are then considered in combination with two additional measures, a vicinity level and an importance level. The vicinity levels attempt to measure the similarity of each schema element in the source and target schemas to their vicinity elements, and the importance levels attempt to measure context similarity by comparing instance and element nodes for source and target schemas. Finally, potential matches with confidence, vicinity, and importance levels above a specific threshold are outputted as schema matches, possibly for human review.

A particular strength of this approach is the ability to detect matches between schema elements represented as names and values. For instance, it is capable of matching a

schema where the concept of waterfront location is represented as a boolean element in one real estate schema, and as part of comments in the instances of a text field in another.

When tested on several sample schema matching exercises, Xu and Embley found it to detect over 90% of the correct matches (defined as recall), and that over 90% of the matches it detected were correct (defined as precision).

Directions and Open Issues

<u>Combining approaches</u>

A common thread through the research profiled above and acknowledged by researchers is that there is no single technique or method that can effectively match schemas, and that most often a combination of techniques produces superior results. Examples of this are found in iMap's searchers, LSD's learners, and Xu and Embley's method, each of which use multiple techniques.

As the need for data integration has grown, the need to implement a variety of schema matching methods has also increased. As a consequence, there has been an emphasis on evaluating schema matching methods. To this end, several systems have been created to act as implementation shells for schema matching approaches. These systems have been typically designed to handle the tasks surrounding the actual schema matching, which include inputting, transforming, and possibly preparing source, target, and/or mediated

schemas, passing these schemas to a matching algorithm, and interpreting the results, often as schema mappings.

One such system supporting multiple matching methods is COMA++ (Aumeuller 2005). This system supports a variety of languages for inputting schemas, including SQL, XSD, and OWL, and outputs schema mappings. The architecture of COMA++ includes a schema pool, match customizer, and mapping pool. It is the match customizer which can implement different matching methods. An execution engine is used to combine the results from multiple matchers, and the results are presented to the user through a graphical user interface.

As with many other matching algorithms, COMA++ supports approaches which can generate direct, or 1:1, matches, and develops a similarity measure between each pair of elements in a target and source schema. There are currently more than 15 matchers in COMA++, using both schema and instance information to develop matches. A contribution of COMA++, in addition to specific matching techniques, is in its ability to act as a repository for past schemas and matches, and to add additional matching techniques modularly.

Clio (Haas 2005) is a system capable of generating mappings between either XML or relational schemas, and of producing XQuery, XSLT, or SQL queries based on those mappings. Clio transforms queries through those mappings to access query fragments from source databases, and is capable of combining those fragments accurately through

de-duplication and a process termed 'deep-union'. The architecture of Clio includes a user interface with which a user can direct the schema matching process, choosing to create matches manually or to select from those automatically generated. A particular contribution of Clio is that it can be used to incorporate many different schema matching algorithms, and evaluate each of those on real world problems starting with input schemas and ending with actual query execution.

Standardization and Benchmarking

Some research profiled here included an evaluation of a schema matching method on sample data. The test schemas were not consistent with each other, and varied widely in the number of tables, elements, complexity, and application domain. There were also no uniformly consistent metrics reported, with the possible exception of recall, precision, and an F-measure which were often reported. Recall is typically defined as the percentage of correct (as determined by a human matcher, sometimes used as the 'gold standard' (Madhavan 2005)) matches found, precision as the percentage of total matches found that are correct, and the F-measure as the harmonic mean of the two. The F-measure has been used because neither the recall nor precision measures are fully meaningful when used independently. The recall measure can be inflated when a matching algorithm detects a large number of schema matches (up to a cross product of two schemas for direct 1:1 matchers), since more correct matches would be identified. The precision measure can be inflated if only a small number of matches are reported based on the matcher using high confidence levels.

Research by Do, Melnik, and Rahm (Do, 2002) used the recall, precision, and F-measure to compare six matching systems. They noted that evaluations of matching systems should include more than these measures. Specifically, a matching system needs to be evaluated on the degree to which the input needs to be prepared or transformed before it is used, the degree to which the output can be readily used (for example, if it produces matches or complete mappings), and how much effort is required for the system to be used. In some cases, the effort required to fully train a learning based system might outweigh the effort for a knowledgeable user to manually ascertain the matches. Do, Melnik, and Rahm underscore the growing importance of being able to compare and benchmark alternative approaches to schema matching.

THALIA (Test Harness for the Assessment of Legacy Information Integration) is a publicly available set of scenarios derived from university course catalogs for benchmarking data integration approaches (it is currently available at http://www.cise.ufl.edu/project/thalia.html). THALIA has been used to benchmark SMWRA (Topsakal 2005), which is a matching system that takes advantage of report headers and other report information in addition to schema information in order to find matches.

The Ontology Alignment Evaluation Initiative of the Knowledge Web Consortium in Europe has recently proposed a set of benchmarking methods to evaluate ontology alignments, including those of schema matching (Euzenat 2005 and Castro 2005). At

least one recently developed matching system, COMA++ (Aumeller 2005), has been evaluated by using the scenarios proposed for evaluation by these guidelines (http://co4.inrialpes.fr/align/Contest) by this group.

Scalability

Three factors are influencing the need for scalable schema matching solutions. The first factor is the trend towards schema matching solutions that use multiple techniques, from the recognition that there simply is no one single dominant technique. The second is the recognition that complex matches, those requiring data transformation and possibly matching multiple elements, are often required in real-world scenarios. Both of these factors push towards compute-intensive schema matching; for example, the iMAP matching system (Dhamankar 2004) can require execution times in excess of five minutes to generate matches for one pair of schemas. The third factor influencing the need for scalability is the growing demand for schema matching as a step towards data integration.

Research has been directed at scalable schema matching methods. He and Chang (He 2004) proposed two statistically-based approaches which take multiple schemas as input. The first approach assumes an underlying hidden model as the source for the schemas to be matched, and attempts to generate that hidden model in order to match schemas to it. The second approach uses correlation mining to attempt to identify co-occurrence of attributes, and indirect matches of cardinality 1:n. The focal point of this research is to

find techniques which can greatly reduce the amount of time needed to perform matching between many related schemas.

For large XML schemas, Rahm, Do, and Massmann (Rahm 2004) proposed a method to partition the schemas to be matched and to match fragments of each. The partitioning is done for similar types in each schema, basing the development of schema fragments on a comparison of schema structures. Identifying schema fragments enables more productive searches for matches between fragments, reducing execution times. This approach was implemented in a version of COMA (Aumeller 2005).

Automated Matching

While many researchers take a general-purpose fully automatic schema matching solution as infeasible, there is a recognized need for the automation of as much schema matching as possible, and for completely automated schema matching to handle specific cases. There have been attempts to more completely automate schema matching, particularly for XML data. QMatch (Claypool 2005) uses XML schema information on elements and structure in order to quantify the similarity between elements in two schemas. SMWRA (Topsakal 2005) augments XML schema information with report text, especially headers from a specific application, and was tested on data available on the Web.

There does not appear as yet to be a schema matching system capable of using both XML schema information and instance data to improve matching. In some cases, an XML

document has a complete set of instance data contained within it. In others, data is presented in an XML document but only through a web form and for a small number of data instances at one time. The vision of automated schema matching for XML schemas on the Web might come closer to being realized if a schema matching system could query a web database through repeated post backs in order to ascertain specific instance data.

The general view seems to be that some while data integration activities, such as integrating legacy databases, may always need human involvement, there are applications such as e-commerce and intelligent agents on the Web for which automatic schema matching must be developed. The access and utilization of instance data to match XML schemas on the Web would be helpful to this end.

Summary

Schema matching has been an area of active research for over 20 years. The increase in the volume of available data, heterogeneous databases, and dramatic proliferation of structured and unstructured data on the Web, have each increased the importance of developing effective schema matching solutions.

There has been considerable success in developing schema matching methods, with some research profiled here reporting the ability to correctly match 70-90% of the elements in a target and source schema. The most successful methods use the most available

information, including structure data, element data, instance data, and past schema matches. These methods can be divided into those which are rules-based, which typically use heuristics to develop similarity measures between target and source schema elements and then select matches from those measures, learning-based, which employ learning techniques such as neural networks to improve rules-based matching with known correct matches, and ontology-based, which often use domain specific information to augment the matching process. There have been successes reported within each of these categories.

However, several considerable challenges remain. The most successful schema matching has been accomplished on schemas from a very specific domain, or from applications which are not fully representative of large-scale, real world schemas. As a result, there is an increasing emphasis on standardizing and benchmarking schema matching methods and improving the capability of matching larger and dynamic schemas. Considerable research is still needed if more fully automated schema matching is to become a reality.

References

Aumueller, D. Do, H., Massmann, S., and Rahm, E. (2005) "Schema and Ontology Matching with COMA++." *Proceedings of the 2005 ACM SIGMOD International Conference on Management of Data,* pp. 906-908.

Batini, C., Lenzerini, M. and Navathe, S.B. (1986) "A Comparative Analysis of Methdologies for Database Schema Integration," <u>ACM Computing Surveys,</u> **18**(4), pp. 323-364.

Berlin, J. and Motro, A. (2002) "Database Schema Matching Using Machine Learning with Feature Selection," *Advanced Information Systems Engineering: 14th International Conference, CAiSE 2002 Toronto, Canada*.

Bernstein, P., Melnik, S., Petropoulos, M. and Quic, C. (2004) "Industrial-Strength Schema Matching." *SIGMOD Record,* **33**(4), pp. 38-43.

Bilke, A. and Naumann, F. (2005) "Schema Matching using Duplicates," *Proceedings of the twenty-first International Conference on Data Engineering*.

Castro, R. and Ehrig, (2005) "Specification of a benchmarking methodology for alignment techniques" from the Knowledge Web Consortium <http://www.inrialpes.fr/exmo/cooperation/kweb/heterogeneity/deli/kweb-222.pdf>.

Chua, C., Chiang, R., and Lim, E. (2003). "Instance-based Attribute Identification in Database Integration," *The VLDB Journal,* **(12),** pp. 228-243.

Claypool, K. and Hegde, V. (2005) "QMatch - A Hybrid Match Algorithm for XML Schemas," *Proceedings of the twenty-first International Conference on Data Engineering*.

Cohen, W., and Richman, J. (2002). "Learning to Match and Cluster Large High-Dimensional Data Sets for Data Integration," *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 475-480.

Dhamankar, R., Lee, Y., Doan, A., Halevy, A., and Domingos, P. (2004). "iMAP: Discovering Complex Semantic Matches between Database Schemas," *Proceedings of the 2004 ACM SIGMOD International Conference on Management of Data*, pp. 383-394.

Do, H., Melnik, S., and Rahm, E. (2002) "Comparison of Schema Matching Evaluations," *Proceedings of the 2nd Int. Workshop on Web Databases.*

Doan, A., Domingos, P., and Halevy, A. (2003) "Learning to Match the Schemas of Data Sources: A Multistrategy Approach," *Machine Learning,* (**50)**, pp. 279-301.

Doan, A. and Halevy, A. (2005) "Semantic-Integration Research in the Database Community," *AI Magazine,* Spring 2005, pp. 83-94.

Embley, D., Jackman, D., and Xu, L. (2002) "Attribute Match Discovery in Information Integration: Exploiting Multiple Facets of Metadata," *Journal of the Brazilian Computer Society,* 8(2), pp. 32-43.

Embley, D., Xu, L., and Ding, Y. (2004) "Automatic Direct and Indirect Schema Mapping: Experiences and Lessons Learned," *ACM SIGMOD Record,* **33**(4), pp.14-19.

Euzenat, J. (2005) "Towards a Methodology for Evaluating Alignment and Matching Algorithms," from the website of the Ontology Alignment Evaluation Initiative at <http://oaei.inrialpes.fr/doc/oaei-automation.1.pdf>.

Haas, L., Hernandez, M., Ho, H., Popa, L., and Roth, M. (2005) "Clio Grows Up: From Research Prototype to Industrial Tool." Proceedings of the 2005 ACM SIGMOD International Conference on Management of Data, pp. 805-810.

Halevy, A. (2005) "Why Your Data Won't Mix." *ACM Queue,* October 2005, pp. 50-58.

Halevy, A., Franklin, M., and Maier, D. (2006) "Principles of Dataspace Systems." *Proceedings of the twenty-fifth ACM SIGMOD-SIGACT-SIGART symposium on Principles of Database Systems*, Chicago.

Hammer, J., Stonebraker, M., and Topsakal, O. (2005) "THALIA: Test Harness for the Assessment of Legacy Information Integration Approaches," *Proceedings of 21st Int'l Conf. on Data Engineering (ICDE)*, pp. 485-486.

He, B. and Change, K. (2004) "A Holistic Paradigm for Large Scale Schema Matching", *SIGMOD Record*, 33(4), pp. 20-25.

Lenzerini, M. (2002) "Data Integration: A Theoretical Perspective," *Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART symposium on Principles of Database Systems*, New York.

Kang, J. and Naughton, J. (2003), "On Schema Matching with Opaque Column Names and Data Values," *Proceedings of the 2003 ACM SIGMOD international conference on Management of data,* pp. 205-216.

Li, W. and Clifton, C. (2000), "Semint: A Tool for Identifying Attribute Correspondence in Hetergeneous Databases Using Neural Networks," *Data and Knowledge Engineering*, **33**(1), pp. 49-84.

Li, Y., Liu, D., and Zhang, W. (2005), "Schema Matching Using Neural Network," *Proceedings of the 2005 IEEE/WIC/ACM International Conference on Web Intelligence.*

Madhavan, J., Bernstein, P., Doan, A., and Halevy, A. (2005) "Corpus-based Schema Matching," *Proceedings of the twenty-first International Conference on Data Engineering.*

Madhavan, J., Bernstein, P., and Rahm, E. (2001) "Generic Schema Matching with Cupid," *Proceedings of the 27th International Conference on Very Large Data Bases*, pp. 49 – 58.

Melnik, S., Garcia-Molina, H., and Rahm, E. (2002) "Similarity Flooding: A Versatile Graph Matching Algorithm," *Proceedings of the eighteenth International Conference on Data Engineering*, p 117.

Miller, G. (1995). "WordNet: A Lexical Database for English", *Communications of the ACM,* **28**(11), pp. 39-41.

Noy, N., Doan, A., and Halevy, A. (2005) "Semantic Integration," *AI Magazine* ,Spring 2005, pp. 7-9.

Rahm, E. and Bernstein, P. (2001) "A survey of approaches to automatic schema matching," *The VLDB Journal*, **10**, pp. 335-350.

Rahm, E., Do, H., and Massmann, S. (2004) "Matching Large XML Schemas," *ACM SIGMOD Record,* **33**(4), pp.26-31.

Ram, S. and Park, J. (2004) "Semantic Conflict Resolution Ontology (SCROL): An Ontology for Detecting and Resolving Data and Schema-Level Semantic Conflicts," *IEEE Transactions on Knowledge and Data Engineering*, **16**(2), pp. 189-202.

Russell, S. and Norvig, P. (2003) Artificial Intelligence: A Modern Approach, Prentice Hall 2003.

Shvaiko, P. and Euzenat, J. (2005) "A Survey of Schema-based Matching Approaches," *Journal on Data Semantic,* 4, pp. 146-171.

Smiljanic, M., Keulen, M., and Jonker, W. (2006) "Using Element Clustering to Increase the Efficiency of XML Schema Matching."

Sung, S. and McLeod, D. (2006) "Ontology-Driven Semantic Matches between Database Schemas," *Proceedings of the twenty-second International Conference on Data Engineering.*

Topsakal, O., and Hammer, J. (2005) "Schema Matching with Report Analysis," *Proceedings of the 27th International Conference on Very Large Data Bases*, pp. 49 – 58.

Xu, L. and Embley, D. (2003) "Discovering Direct and Indirect Matches for Schema Elements," *Proceedings of the Eight International Conference on Database Systems for Advanced Applications*, pp. 39-46.

Yan, L., Miller, R., Haas, L, and Fagin, R. (2001) "Data-Driven Understanding and Refinement of Schema Mappings," *SIGMOD Record*,**30**(2), pp. 485-496.