Semantic Matching across Heterogeneous Data Sources

Huimin Zhao

As our ability to build information systems continues to grow, so does the need to integrate the systems we build. There is currently an urgent need for cooperation among massively distributed information systems for homeland security purposes. Information about a suspected individual needs to be retrieved from many systems maintained nationwide by various organizations, including intelligence agencies, police departments, motor vehicle departments, and airlines. There is also a need for a unified Master Patient Index (MPI) that integrates numerous healthcare information systems and allows authorized care providers to easily access the medical records of all patients [1]. The rapid growth of the Internet, especially the recent development of Web services, continuously amplifies the need for semantic interoperability across heterogeneous data sources. Related data sources accessible via different Web services create new requirements and opportunities for data integration. Such need for integration of information systems is becoming ubiquitous, both within and across organizations, in many domains.

The information systems that need to be integrated are typically heterogeneous in several aspects; these include operating system, data model, database management system (DBMS), application programming language, structural format, and data semantics. Many techniques have been developed to bridge the systematic and structural gaps across heterogeneous information systems. Some examples are heterogeneous DBMS, connectivity middleware (e.g., ODBC, OLE DB, and JDBC), and the emerging XML-based Web services technology [5]. However, semantic alignment across systems is still a resource-consuming process and demands automated support. A particularly critical step in semantic integration is determining semantic correspondences

across heterogeneous data sources. This article presents various techniques that are potentially useful for facilitating such intersystem semantic matching.

Detecting Semantic Correspondences

Semantic correspondences across heterogeneous data sources include schema-level correspondences and instance-level correspondences. Schema-level correspondences consist of tables in different data sources that describe the same real world entity type and attributes in different data sources that describe the same property of some entity type. Instance-level correspondences consist of records in different data sources that represent the same real world entity. The problem of determining schema-level correspondences is referred to as schema matching and interschema relationship identification [2, 11]. The problem of determining instance-level correspondences is referred to as record matching (or linkage) and instance (or entity) identification [4, 10, 12].

For the purposes of illustration, consider an example of two security-related databases owned by different organizations. Table 1 and Table 2 show some made-up entries of two corresponding tables in the databases. The two tables contain some common attributes and overlapping records, although there are various discrepancies and data errors. Restricting the scope to just these two tables, the task is to identify the corresponding attributes (e.g., Suspect.FirstNm and Criminal.FName) and corresponding records (e.g., the first record of Suspect and the first record of Criminal) so that the tables can be effectively linked or integrated together.

Since real-world data sources are often very large—databases with hundreds of tables, thousands of attributes, and millions of records are not uncommon—manually identifying

correspondences from such large data sources tends to be prohibitively expensive. Automated or semi-automated tools are therefore desired to assist human analysts in the semantic matching process. Such tools can be based on expert rules or on learning techniques. In a rule-based approach, domain experts are required to provide the decision rules for claiming correspondences [6]. In a learning-based approach, such decision rules are learned using machine learning techniques [2, 7, 9, 10, 11, 12]. Since the rule-based approach requires a time-consuming knowledge acquisition process to elicit the domain knowledge from human experts, the learning-based approach is preferred, as it bypasses the knowledge acquisition bottleneck.

Both unsupervised learning (i.e., cluster analysis) and supervised learning (i.e., classification) techniques have been developed to automate the discovery of semantic correspondences. In general, cluster analysis techniques are more suitable for identifying schema-level correspondences, while classification techniques are more suitable for detecting instance-level correspondences. Cluster analysis recommends rough groups of similar examples in a data set; it needs to be redone whenever the data set changes. Classification learns a decision model based on a training sample to make specific predictions (i.e., whether two records match or not) on new data. The number of tables and attributes is usually much smaller than the number of records in a data source. While some amount of follow-up manual review of clustering results is affordable for schema matching, this is less likely to be true for record matching. Further, schemas are relatively more stable than instance records. There is no need to repeat schema matching unless many new data sources need to be integrated dynamically, but record matching needs to be regularly performed whenever the data in the underlying data sources are updated.

Information for Comparing Schema Elements and Instances

Schema elements can be compared based on various characteristics, including names, documents, specifications, data patterns, and usage patterns. As tables and attributes are named to reflect their meanings, string matching methods and linguistic tools, such as thesauri, can be used to measure the similarities among table names or attribute names. Descriptions of tables and attributes in design documents can be compared using document similarity measures developed in the information retrieval field. Attributes representing similar concepts tend to be modeled using similar specifications, including data type, length, and constraints. Similar attributes also tend to exhibit similar data patterns, such as length, formation (i.e., proportions of digits, letters, and white spaces), number of distinct values, and percentage of values that are missing. Similar schema elements are also used in similar manners (e.g., update frequency and number of users or user groups). Such specifications, data patterns, and usage patterns can therefore be used in comparing schema elements.

It is important to carefully select such characteristics for comparing schema elements in each particular application, due to the potential for various problems. Schema elements are frequently named using ad-hoc abbreviations rather than standard words. Design documents are often outdated, incomplete, incorrect, ambiguous, or simply not available. Semantically similar concepts could often be modeled using different structures. For example, “gender” can be defined as a numeric attribute in one data source and a character one in another data source. Data patterns are correlated more with structures than with semantics. Usage data may not be maintained in legacy systems. Other semantics and business rules may simply reside in human minds or may be deeply embedded into hard code. It is therefore necessary to utilize multiple types of available clues and also involve domain experts in the process to capture their domain knowledge.

Schema-level correspondences provide the basis for comparing records across heterogeneous data sources. A pair of records in corresponding tables can be compared based on a set of corresponding attributes to determine whether the records match or not when there is no common key across the tables. If two corresponding attributes are accurately recorded following the same format, they can be compared literally. However, we frequently encounter various kinds of discrepancies across real-world data sources. Semantically corresponding attributes often have different formats in different data sources. For example, both (414)2296524 and 1-414-229-6524 refer to the same phone number. The same attribute may be measured on different scales (e.g., the metric kilogram versus the U.S. pound) in different data sources. There are wrong data, spelling errors, and different abbreviations in most operational databases. Human names are often misspelled, mistaken with similar-sounding names (e.g., Keafer and Keefer), or substituted with nicknames (e.g., Andy and Andrew).

Due to such discrepancies across data sources, different transformation procedures and approximate matching functions need to be used to standardize the format and measure the similarity between corresponding attributes. There are many approximate string-matching methods, some of which (e.g., edit distance) account for spelling errors, such as insertions, deletions, substitutions, and transpositions of characters, while others (e.g., Soundex) account for phonetic errors. There are also special-purpose methods for standardizing and comparing particular types of attributes, such as human names and addresses. Special translators (e.g., the two-letter abbreviation for each U.S. state) can be built to resolve coding differences across databases.

Learning Techniques for Detecting Semantic Correspondences

Figure 1 classifies some widely used learning techniques. Cluster analysis techniques group the examples in a data set into groups (called clusters) of similar examples. Because the groups to be discovered from the data are previously unknown, cluster analysis is characterized as “unsupervised” learning. When applied to schema matching, schema elements are grouped into clusters of similar ones based on their characteristics, such as names, documents, specifications, data patterns, and usage patterns, described earlier. These groups of similar schema elements are then presented to domain experts for further evaluation.

There are many statistical and neural network techniques for cluster analysis [11]. Statistical clustering methods may be hierarchical or non-hierarchical. A nonhierarchical clustering method, such as K-means, requires the user to specify the desired number of clusters. A hierarchical method clusters examples on a series of similarity levels, from very fine to very coarse partitions. Hierarchical methods can be agglomerative or divisive. Agglomerative methods start from the finest partition, in which each individual example is a cluster, and successively merge smaller clusters into bigger ones. Divisive methods start from the coarsest partition, in which all the schema elements are in a single cluster, and successively divide big clusters into smaller ones. Kohonen's Self-Organizing Map (SOM) is an unsupervised neural network that can project high-dimensional data onto a low-dimensional (usually two-dimensional) space. SOM is particularly good at visualizing the proximity among schema elements. Since no clustering method has been found to be the single best choice, several methods can be used together.

A classification technique is used to build a general prediction model, called a classifier, which can predict the value of a dependent variable (called class) based on a set of explanatory variables. Because the classifier needs to be derived from a set of training examples whose

classes are given, classification is characterized as “supervised” learning. The learned classifier can then be used to predict the classes of other examples. When applied to record matching, a pair of records from different data sources is classified into one of two classes, match and non-match, based on their similarity scores on corresponding attributes. Domain experts should manually classify some record pairs for training purposes.

There are many statistical, machine learning, and neural network methods for classification [12]. Some widely used statistical methods include naive Bayes, Fellegi and Sunter’s record linkage theory [4], logistic regression, and *k*-nearest neighbor. Naive Bayes estimates the odds ratio (a record pair being match versus non-match) under the assumption that the explanatory variables are conditionally independent. Fellegi and Sunter’s record linkage theory extends naive Bayes specifically for the record linkage problem and allows a record pair to be classified into one of three classes: match, non-match, and unclassified. Logistic regression finds a linear boundary (a weighted sum of the explanatory variables) to separate the two classes, match and non-match. *k*-nearest neighbor simply memorizes the training examples and classifies each new example into the majority class of the *k* closest training examples. Machine learning techniques generate decision tables, trees, or rules. The most widely used techniques include C5 and CART. Back propagation is one of the most widely used neural network techniques for classification. Neural networks are highly interconnected networks, with an input layer, an output layer, and zero or more intermediate layers; they successively adjust the weights of the connections between the nodes on neighboring layers during training.

There are also methods for combining multiple classifiers to further improve classification accuracy. Some examples are bagging, boosting, cascading, and stacking. Bagging and boosting train multiple classifiers of the same type (i.e., with homogeneous base classifiers)

and make the final prediction based on the votes of the base classifiers. In bagging, the base classifiers are trained independently using different training data sets and are given equal weights in the voting. In boosting, base classifiers are learned sequentially–each new classifier focuses more on the examples classified incorrectly by previous classifiers–and are weighted according to their accuracy. Cascading and stacking combine classifiers of different types (i.e., with heterogeneous base classifiers). Cascading combines classifiers vertically, with the outputs of one base classifier used as additional input variables of the next base classifier. Stacking combines classifiers horizontally, with the outputs of several base classifiers used as input variables of a high-level classifier, which is responsible for making the final classification decision.

Combining the Two Tasks into a Comprehensive Procedure

Semantic correspondences on the two levels are obviously related. Schema-level correspondences provide the necessary basis for comparing records. Given identified corresponding records, attribute correspondences can be more accurately evaluated using statistical analysis techniques [3]. It is therefore productive to combine the two tasks into a comprehensive procedure, so that the accuracy of identified correspondences on the two levels can be improved gradually [8].

Figure 2 outlines a general procedure for semantic matching across heterogeneous data sources. The procedure starts from clustering schema elements. Schema-level correspondences suggested by cluster analysis are reviewed and verified by domain experts and then used to determine corresponding records using classification techniques. After some corresponding records have been identified, data from different data sources are linked or integrated together

and further analyzed using statistical analysis techniques. Semantically related attributes tend to be highly correlated and can be revealed by correlation analysis. Regression analysis can further determine the actual relationship (e.g., scaling discrepancy) between correlated attributes. Correspondences between categorical attributes can be analyzed using a more general statistical dependence measure, such as mutual information. A normalized mutual information index between two attributes equals zero if the attributes are statistically independent and 100% if the attributes are one-to-one transformations of each other. If such statistical analysis reveals any new findings, record matching can be redone. Similarly, if new corresponding records are identified, statistical analysis of attribute correspondences can be performed again. This procedure is repeated until no further improvement in the results can be obtained.

Human analysts should keep in mind that the procedure is not totally automated and that human intervention is important in every step. Cluster analysis is highly empirical in nature and requires careful evaluation of the results. Classifiers may not be able to classify all record pairs with sufficient accuracy, leaving some hard cases for analysts to review manually. Highly correlated attributes detected by statistical analysis techniques may describe related but not identical properties about some entity type, thus requiring analysts to verify whether the correlated attributes are indeed corresponding ones in light of domain knowledge. Tools can help human analysts, but cannot totally replace them.

Application in the Security Example

Now consider how the procedure and the various techniques can be applied in matching the two security-related databases (Table 1 and Table 2). First, the attributes need to be clustered, as the scope is restricted to just the two corresponding tables. The attribute names can be

compared using a string matching method, such as edit distance, to account for different abbreviations (e.g., FirstNm and FName) and a thesaurus to account for different synonyms (e.g., Gender and Sex). If descriptions of the attributes are available, they can be compared using a string or document similarity measure. Data patterns, such as summary statistics (average, standard deviation, and range) about the lengths of attribute values, can be computed for each attribute. Specifications and usage patterns can also be used if available. If there are too many such characteristics for cluster analysis, a dimensionality reduction technique, such as principal component analysis (PCA), can be used first to reduce the number of input variables. PCA produces a few linear combinations of the original variables, called principal components, which can roughly represent the original data set. Cluster analysis techniques, such as K-means, hierarchical clustering, and SOM, can then be applied to cluster the attributes based on these principal components.

After some corresponding attributes have been identified, various classification techniques can be used to identify corresponding records. If some of the records have a common key, such as Social Security Number or Driver's License Number, training examples can be easily generated using this key. Otherwise, the analyst needs to manually classify some record pairs for training classifiers. Different transformation and matching procedures can be built to compare corresponding attributes. Attributes measured on different scales (e.g., Suspect.Weight is measured in U.S. pounds while Criminal.Weight is measure in metric kilograms) require appropriate re-scaling. Categorical attributes coded differently (e.g., Suspect.Gender is coded using "Male" and "Female" while Criminal.Sex is coded using 1 and 2, respectively) require special translators. Human names can be compared by combining several matching methods, such as Soundex for matching similar-sounding names (e.g., Keafer and Keefer), nickname

dictionary for matching different nicknames (e.g., Andy and Andrew), and edit distance for handling spelling errors (e.g., Carol and Carole).

After some corresponding records have been identified, they can be integrated into a single data set, so that statistical analysis can be used to further analyze the relationships among attributes. Correlation analysis can be used to find highly correlated attributes. Regression can be used to discover transformation formulae between corresponding attributes (e.g., Suspect.Weight = 2.2 × Criminal.Height). Mutual information can be used to detect categorical attributes coded differently in the two tables (e.g., Suspect.Gender and Criminal.Sex). At this stage, such differences can be more rigorously analyzed than during the previous cluster analysis. Note that some related but different attributes may be found correlated to some extent. For example, Weight and Height may be somewhat correlated. The analyst should carefully evaluate the analysis results and cross out such spurious correspondences. The overall procedure can be repeated for a few rounds until the analyst is satisfied with the results.

Conclusion

Many techniques have been developed for determining semantic correspondences across heterogeneous data sources, which is a key problem in the semantic integration of such data sources. After over two decades of extensive research, it is now time to harvest some of the results. We need to combine these techniques together, incorporate them into comprehensive tools, and validate and improve them in real-world, large-scale data integration applications. In the meantime, we still need a better understanding of what the real problems, difficulties, and issues are and how well the techniques perform in real applications. Such real insights gained from practice are crucial for assuring the relevance of theoretical research.

REFERENCES

1. Bell, G. and Sethi, A. Matching records in a national medical patient index. *Commun. ACM 44*, 8 (2001), 83 - 88.

2. Doan, A. and Domingos, P. Learning to match the schemas of data sources: a multistrategy approach. *Machine Learning* 50, 3 (2003), 279-301.

3. Fan, W., Lu, H., Madnick, S., and Cheung, D. DIRECT: a system for mining data value conversion rules from disparate sources. *Decision Support Systems 34*, 1 (2002), 19-39.

4. Fellegi, P. and Sunter, A. A theory of record linkage. *Journal of the American Statistical Association 64*, 328 (1969), 1183-1210.

5. Hansen, M., Madnick, S., and Siegel, M. Data integration using Web services. In *Proceedings of International Workshop on Data Integration over the Web*, 2002.

6. Hernández, M. and Stolfo, S. Real-world data is dirty: data cleansing and the merge/purge problem. *Data Mining and Knowledge Discovery 2*, 1 (1998), 9-37.

7. Li, W. and Clifton, C. SEMINT: a tool for identifying attribute correspondences in heterogeneous databases using neural networks. *Data & Knowledge Engineering 33*, 1 (2000), 49-84.

8. Ram, S. and Zhao, H. Detecting both schema-level and instance-level correspondences for the integration of E-catalogs. In *Proceedings of Workshop on Information Technology and Systems*, 2001, 187-192.

9. Tejada, S., Knoblock, C., and Minton, S. Learning object identification rules for information integration. *Information Systems 26*, 8 (2001), 607-633.

10. Verykios, V., Elmagarmid, A., and Houstis, E. Automating the approximate record-matching process. *Information Sciences 126*, 1-4 (2000), 83-98.

11. Zhao, H. and Ram, S. Clustering schema elements for semantic integration of heterogeneous data sources. *Journal of Database Management 15*, 4 (2004), 88-106.

12. Zhao, H. and Ram, S. Entity identification for heterogeneous database integration - a multiple classifier system approach and empirical evaluation. *Information Systems 30*, 2 (2005), 119-132.

FirstNm	LastNm	Gender	Hair	Eyes	Height	Weight
Andrew	Keafer	Male	Black	Black	5’8”	160
Lillian	Lee	Female	Black	Black	5’2”	130
Carole	Smith	Female	Blond	Blue	6’3”	310

Table 1. Sample entries in table Suspect of database A.

FName	LName	Sex	HairColor	EyeColor	Height	Weight
Andy	Keefer	1	BLK	BLK	173	73
Lillian	Li	2	BLK	BLK	157	58
Carol	Smith	2	BLD	BLU	190	140

Table 2. Sample entries in table Criminal of database B.

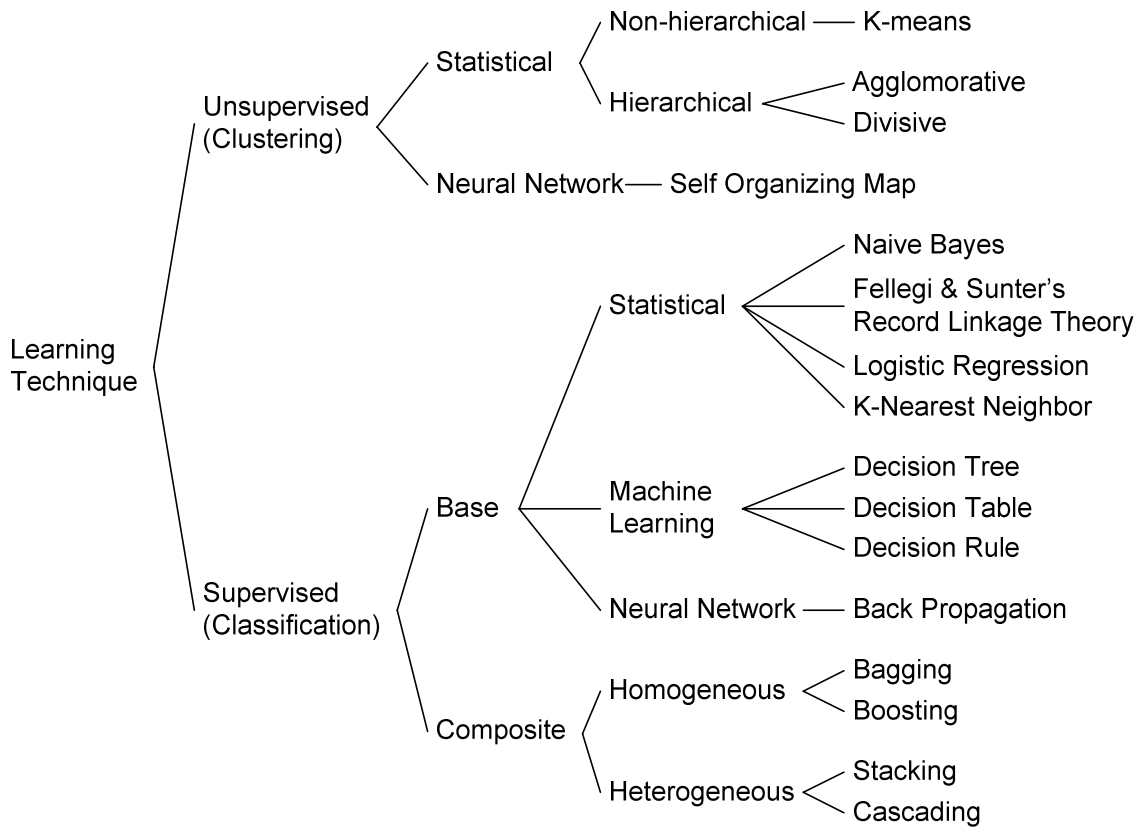

Figure 1. Some widely-used learning methods.

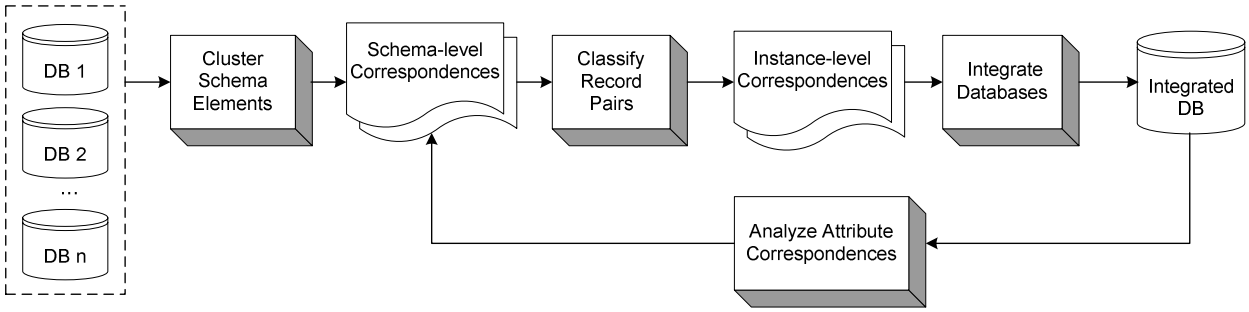

Figure 2. A general procedure for semantic matching across heterogeneous data sources.