Measuring the Relative Performance of Schema Matchers

Shlomo Berkovsky, Yaniv Eytani
University of Haifa
{slavax,ieytani}@cs.haifa.ac.il

Avigdor Gal
Technion – Israel Institute of Technology
avigal@ie.technion.ac.il

Abstract

Schema matching is a complex process focusing on matching between concepts describing the data in heterogeneous data sources. There is a shift from manual schema matching, done by human experts, to automatic matching, using various heuristics (schema matchers). In this work, we consider the problem of linearly combining the results of a set of schema matchers. We propose the use of machine learning algorithms to learn the optimal weight assignments, given a set of schema matchers. We also suggest the use of genetic algorithms to improve the process efficiency.

1. Introduction

Schema matching is the task of matching between concepts describing the meaning of data in heterogeneous, distributed data sources. As such, it is recognized to be one of the basic operations required by the process of data integration [3]. Due to its cognitive complexity [4], traditionally schema matching has been performed by human experts [11]. As the process of data integration has become more automated, the ambiguity inherent in concept interpretation has become one of the main obstacles to this process. For obvious reasons, manual concept reconciliation in dynamic environments is inefficient and at times close to impossible. Introduction of the Semantic Web vision [2] and shifts toward machine-understandable Web resources have made even clearer the vital need for automatic schema matching.

In attempt to address these practical needs, several heuristics for automatic schema matching (*schema matchers* hereafter) have been proposed and evaluated. While in many domains these heuristics succeed in finding the right matching, empirical analysis shows that there is no single heuristic that is guaranteed to be effective in all possible domains and applications [8]. To overcome this problem, several tools allow combining schema matchers [5], [7], [12]. Like many before us, we hypothesize that a combination of different schema matchers can improve the matching result over mappings obtained independently by each matcher.

In this work, we consider the problem of linearly combining the results of a set of schema matchers. Each matcher is assigned a different weight, yielding a vector of relative weights. The optimal vector is not known a priori, and may change from one matching problem to another. We propose the use of machine learning algorithms to learn the optimal weight assignments, given a set of schema matchers. Our aim is twofold. First, we attempt at identifying general rules in assigning weights to various schema matchers. Secondly, we analyze the relative performance of various schema matchers, aiming at identifying matchers' dominance.

For this purpose we performed a close to exhaustive search and checked the correlation between algorithm weights and mapping precision. Our preliminary analysis show that no single matcher dominates as was predicted.

The rest of the paper is organized as follows. Section 2 provides an overview of the research field of schema matching. We next formalize schema matching as a search problem in Section 3 and propose the use of a genetic search to reduce the search process complexity. Section 5 provides preliminary empirical results. We provide concluding remarks and our proposed future research directions in Section 6.

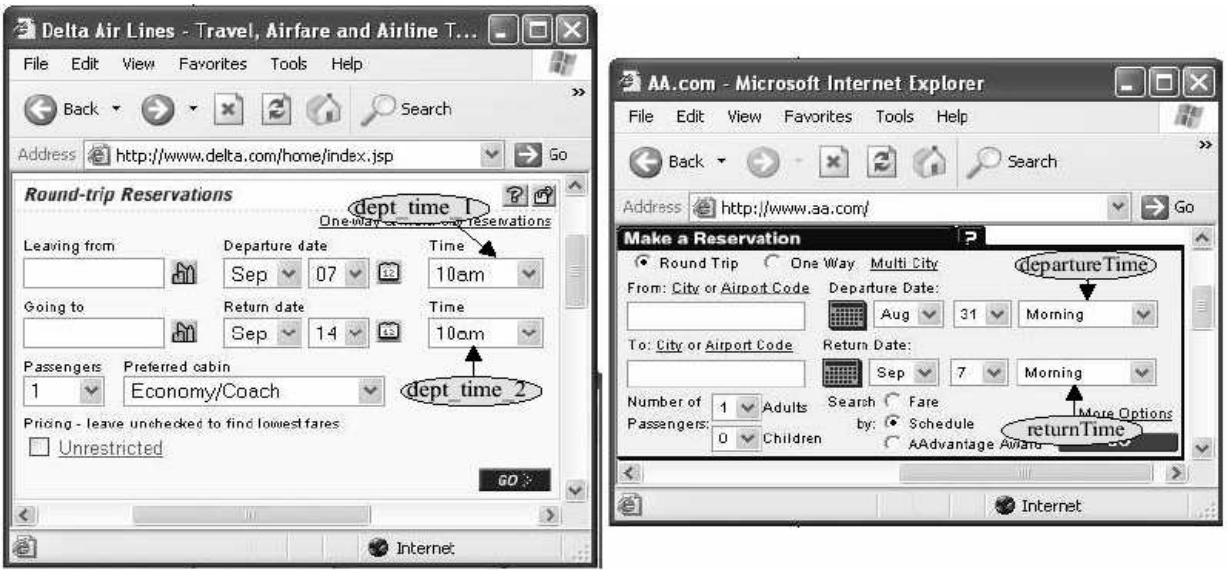

Figure 1: Delta Airlines (left) versus American Airlines (right) Reservation Sites

2. Schema Matching: Overview

Various models for schema matching have been proposed in the literature. For simplicity sake, we provide all basic concepts in terms of one such model, namely this proposed in COMA [5].

Given two sets of concepts S and S' (henceforth referred to as *schemata*), a real-valued degree of similarity $\mu(a,b)$ is automatically computed for all possible pairs of concepts (a,b) from SxS'. This similarity information is used to quantify the quality of different mappings from the concepts in S to the concepts in S'. Typically, a single mapping from S to S' is then chosen as the *best mapping*. [1] The selection process aims at maximizing some aggregation function (e.g., average) of the degree of similarity of the individual concept pair mappings.

Various schema matchers differ mainly in the measures of similarity that they employ, yielding different similarity degrees. These measures can be arbitrarily complex, and may use various techniques such as name matching, domain matching, structure matching, etc.

For illustration purposes, we present next the set of schema matchers, as employed in OntoBuilder [8], specializing in extracting ontologies from Web forms (a feature we have used in our experiments). OntoBuilder accepts two ontologies as input, a candidate ontology and a target ontology. It attempts to match each attribute in the target ontology with an attribute in the candidate ontology. OntoBuilder supports an array of matching and filtering algorithms and can be used as a framework for developing new schema matchers which can be plugged-in and used via GUI or as an API. OntoBuilder uses the following four matchers (detailed description of which can be found in [8]):

Term: A term is a combination of a label and a name. Term matching compares labels and names to identify syntactically similar terms. To achieve better performance, terms are preprocessed using several techniques originating in IR research. Term matching is based on either complete word or string comparison.

Value: Value matching utilizes domain constraints (e.g., drop lists, check boxes, and radio buttons) to compute similarity measure among terms. The availability of constrained value-sets becomes valuable when comparing two terms that do not exactly match through their labels.

Composition: A composite term is composed of other terms (either atomic or composite). Composition can be translated into a hierarchy. This schema matcher assigns similarity to terms, based on the similarity of their neighbors.

Precedence: The precedence relationship is unique to OntoBuilder and therefore worth of a lengthier discussion. In any interactive process, the order in which data are provided may be important. In

[1] We differentiate the process of *matching* from its output (denoted *mapping*).

particular, data given at an earlier stage may restrict the availability of options for a later entry. For example, a car rental site may determine which car groups are available for a given session, using the information given regarding the pick-up location and time. Therefore, once those entries are filled in, the information is sent back to the server and the next form is brought up. Such precedence relationships can usually be identified by the activation of a script, such as (but not limited to) the one associated with a SUBMIT button. As with composition, precedence can be translated into a precedence graph, and the matching of two terms is determined by their neighboring terms.

To illustrate the different capabilities of schema matchers, consider the Delta Airlines and American Airlines online reservation systems (Figure *1*). Due to wrong design (or designer's error), the departure time entry is named *'dept_time_1'*, and return time is named *'dept_time_2'* (Figure *1* left). Thus, a schema matcher based on term matching will not be able to map *'dept_time_2'* to the correct field *'returnTime'* (Figure *1* right). However, a schema matcher based on structure similarity has a higher chance of identifying the mapping, based on the location of the field within the form.

The true test of a schema matcher is whether the best mapping matches well an *exact mapping*, a mapping as could have been determined by a human expert. A matcher that manages to mimic well the decisions made by a human expert is more trustworthy when automating the process of schema matching. Quantifying the performance of a schema matcher can be done in various ways. For example, a strict QoS approach can determine a best mapping to be successful only if it is equal to the exact mapping.

Another, less strict approach, measures the precision and recall of the best mapping [13] with respect to the exact mapping, as follows. Let A be the set of individual concept pair mappings in the exact mapping and let B be the set of individual concept pair mappings in the best mapping. Precision (P) and recall (R), are measured as:

$$P = \frac{|A \cap B|}{|A|} \qquad R = \frac{|A \cap B|}{|B|}$$

Precision and recall both reach the maximum value of *1* whenever $A=B$. Low precision is an indication of many false negatives and low recall is an indication of many false positives.

3. Schema Matching as a Search Problem

Given two schemata S and S' and m schema matchers, the degree of similarity between concepts $a \in S$ and $b \in S'$ is computed as

$$sim(a,b) = \sum_{i=1}^{m} \omega_i \mu_i(a,b)$$

where $\mu_i(a,b)$ represents the degree of similarity assigned by matcher i and $\{\omega_i\}$ is a set of weights that sum to unity, i.e., $\Sigma_i\{\omega_i\}=1$.

This work is concerned with the problem of inference with classifiers in which several local classifiers (the schema matchers) are utilized to achieve a global task (best mapping). This problem can be conceptualized at two levels. First, each schema matcher is used to produce a best mapping according to its capabilities (heuristics) and world knowledge. Then a consensus is reached, based on the outcomes of the classifiers. In the context of schema matching, the population to be investigated is the vector of weights.

More formally, given schemata S and S', their a priori known exact mapping M, and a set $A=\{a_1, a_2, \ldots, a_m\}$ of available schema matchers, we are interested in finding a vector of weights $\omega=\{\omega_1, \omega_2, \ldots, \omega_m\}$ such that applying those of the m schema matchers with positive weights and assigning each matcher a_i with a non-negative relative weight ω_i will produce a mapping M' that is sufficiently similar to M.

For the schema matching task to be effective, two decisions need to be taken. First, deciding on which schema matchers will be utilized and secondly, to determine the weights vector values. For small search space it is possible to perform a close to exhaustive search to explore all weight combinations. However in the general case, the size of the search space is exponential in the size of the schemata. Thus, we propose to employ the learning mechanism of *Genetic Algorithms* [9] (GA hereafter) to both tasks.

4. Genetic Search

A typical matching scenario involves choosing a subset of schema matchers for generating the best mapping. Evaluating a mapping is determined, as discussed in Section 2, using metrics as recall, and precision or a combination of such metrics. Clearly, the size of the search space is exponential in schema size. To accommodate this problem we suggest the use of genetic algorithms.

Genetic algorithms are nature-inspired class of algorithms that mimic an evolutionary rule of “the

fittest survives". Initially, a population of individuals is created, each represented by a chromosome (a set of genes). The value of each gene is assigned randomly (within known bounds). The population is then evaluated to determine how well each individual fits the required task.

The new generation of individuals is chosen from the parent and the offspring generation in accordance with a survival strategy that favors fit individual, but does not preclude the survival of the less fit. An offspring is created by selecting two parents at random and combining parts of their chromosomes. Random changes to the genes are made to mimic the natural role of mutation. This process is repeated until a required performance level is achieved (or no further improvement seems possible).

Genetic algorithms are typically applied when searching optimal or near optimal solutions to a problem within a large multi-modal search space. Typically, it is infeasible to apply a precise analytic algorithm to accurately solve such problems. Schema matching fits nicely with this definition, given the set of competing constraints that have to be balanced in reaching a (close to) optimal solution using multi objective functions (recall, precision and others).

Genetic algorithms success in finding an optimum solution depends on the choice of a fitness function that directs the search along promising pathways. The fitness function is a weighted combination of one or more objective functions and characterizes what is considered to be a good solution. Given two possible solutions, it determines which of them supplies a better set of desired properties. In our experiments we evaluate the fitness of a solution using the precision metric. In the future we plan to use a multi-objective fitness function.

5. Preliminary Results

We extracted a set of *25* pairs of schemata from Web site forms from various domains. For each pair, a human expert has manually constructed the exact mapping. For the purpose of evaluating the combination of different matchers we have performed a close to exhaustive search, exploring many weight combinations. Thus, we could verify the results of GA and assess how the changes in the weight of each matcher affect the overall performance. To evaluate performance, we correlated the changes with the precision obtained by four available matchers (respectively denoted by a_1, a_2, a_3, and a_4), exploiting *terms*, *values*, *composition*, and *precedence* algorithms [6], respectively. Given an algorithm a_i and a schemata S and S', w_i denotes a weight assigned to a_i, while p_i denotes the average level of precision for all the mappings for which a_i is assigned with w_i. Correlation is measured as:

$$cor = \frac{COV(w_i, p_i)}{SD(w_i) \cdot SD(p_i)}$$

where $COV(w_i,p_i)$ is the covariance of w_i and p_i, and $SD(w_i)$ and $SD(w_i)$ are the standard deviations. Partial experimental results are presented in Table *1*. Positive values indicate that there is positive correlation between the weight of an algorithm and the precision. Negative value indicates that the correlation is negative, while values close to *0* show no correlation.

pair num.	a_1	a_2	a_3	a_4
2	0.66	0.01	0.05	-0.69
5	0.63	-0.01	0.07	-0.66
16	0.70	-0.01	-0.16	-0.50
6	0.33	0.25	0.15	-0.70
13	0.49	0.26	0.05	-0.77
21	0.47	0.26	0.07	-0.77
15	-0.32	0.30	-0.19	0.19
17	-0.38	0.48	0.06	-0.15

Table 1: Correlation between algorithm weights and precision: four algorithms

Analyzing the averages of the weight correlations in between the four matchers allows us to conclude the following:

- a_1 has average positive correlation of about *0.3* with the precision and it is a highest correlation in the case we combine *4* matchers.
- a_2 has low average positive correlation of about *0.05* with the precision when *4* matchers are combined.
- a_3 has very low average negative (or NO) correlation with the precision
- a_4 has high negative average correlation of about *-0.25* with the precision.

These results provide a preliminary indication regarding the relative performance of different matchers. However, as was expected, no single matcher performs perfectly over all schema pairs. In this regard, we hypothesize that the different matchers perform differently relating to various ontological structures. For example, we can hypothesize that factors such as the size of schema, application domain, and the types of schema attributes (free text, selection of predefined values, yes/no mark etc.) might determine the suitability of a particular matcher.

Note that the values, composition, and precedence matchers base on the terms matcher (which seems to be the dominating one). Thus, we repeated the

experiment while neutralizing the influence of a_1. Partial results are presented in table *2*.

pair num.	a_2	a_3	a_4
2	0.30	0.54	-0.80
5	0.31	0.54	-0.81
16	0.38	0.44	-0.77
6	0.56	0.37	-0.87
13	0.48	0.33	-0.76
21	0.64	0.32	-0.90
15	0.67	-0.05	-0.57
17	0.74	0.03	-0.72

Table 2: Correlation between the algorithm weights and precision: three algorithms

The experiment shows that as a result of the neutralization of a_1, the values matching algorithm a_2 has a higher positive correlation of about *0.3* with the precision.

The observation regarding the strength of a_2 in the absence of a_1 is an interesting one. This could be an example of a situation where not all schema matchers should be used. Consider a situation where matchers are provided as 'paid' services. Given that a_2 can work well without a_1, means that a user needs to spend less on the matching process, by using three, instead of four, matchers.

6. Conclusions and Future Research

Schema matching is the task of matching between concepts describing the data in heterogeneous data sources. As such, it is recognized to be one of the basic operations in data integration domain. Due to its cognitive complexity, schema matching has been traditionally performed by human experts. However, manual schema matching in dynamic environments is inefficient and at times close to impossible.

In this work, we consider the problem of linearly combining the results of multiple schema matchers. Each matcher is assigned a different weight, yielding a vector of relative weights. The optimal vector is not known a priori, and may change from one matching problem to another. We attempt at analyzing the relative performance of various schema matchers, aiming at identifying matchers' dominance.

For the purpose of evaluating the combination of different matchers we had performed a close to exhaustive search and checked the correlation between algorithm weights and mapping precision. Our preliminary analysis show that no single matcher dominates as was predicted.

GA is typically applied when searching optimal or near optimal solutions to a problem within a large multi-modal search space. Matching schemata fit nicely with this definition, given the set of competing constraints that have to be balanced in reaching a (close to) optimal solution using multi-objective functions (recall and precision and others). We could verify that GA handles the matching search problems well as it allowed us to decrease the computation times and converged correctly.

In future work we aim to learn the optimal weight assignments, given a set of schema matchers. We aim to identifying general rules in assigning weights to various schema matchers. In addition we are creating a large repository of human annotated correct matches of ontology pairs. We would like to further use machine other learning algorithms such as collaborative filtering [10] combined with genetic algorithms (see [1] for an example) to provide accurate prediction of optimal or near optimal weight assignments.

We believe that different matchers perform differently relating to different ontological structures and in different domains [6]. In future work we intend to focus on extending our experiments to discover such correlations and incorporate them into our system.

7. Acknowledgement

The work of Gal was partially supported by two European Commission 6th Framework IST projects, QUALEG and TerreGov, and the Fund for the Promotion of Research at the Technion. We thank Pavel Feldman for his assistance in performing the experiments and Sadek Jbara for the genetic algorithm implementation.

8. References

[1] S. Berkovsky, Y. Eytani, E. Furman, *"Developing a Framework for Insurance Underwriting Expert System"*, in proceedings of the International Conference on Informatics, Turkey, 2004.

[2] T.Berners-Lee, J.Hendler, O.Lassila, "*The Semantic Web*", in Scientific American, 2001.

[3] P.A.Bernstein, S.Melnik, "*Meta Data Management*", in Proceedings of the International Conference on Data Engineering, MA, 2004.

[4] B.Convent, "*Unsolvable Problems Related to the View Integration Approach*", in Proceedings of the International Conference on Database Theory, Italy, 1986.

[5] H.H.Do, E.Rahm, "*COMA - a System for Flexible Combination of Schema Matching Approaches*", in Proceedings of the International Conference on Very Large Data Bases, Hong-Kong, 2002.

[6] A. Doan, J. Madhavan, R. Dhamankar, P. Domingos, A. Y. Halevy. *"Learning to match ontologies on the Semantic Web",* VLDB Journal, 12(4): 303-319 (2003).

[7] A.Gal, A.Anaby-Tavor, A.Trombetta, D. Montesi, "*A Framework for Modeling and Evaluating Automatic Semantic Reconciliation*", in VLDB Journal, 2005, to appear.

[8] A.Gal, G.Modica, H.M.Jamil, A.Eyal, "*Automatic Ontology Matching Using Application Semantics*", in AI Magazine, 26(1), 2005.

[9] D.E.Goldberg, R.Burch, "*Genetic Algorithms in Search, Optimization, and Machine Learning*", Addison Wesley Publishers, 1989.

[10] J.L.Herlocker, J.A.Konstan, A.Borchers, J.Riedl, "*An Algorithmic Framework for Performing Collaborative Filtering*", in proceedings of the 22nd International ACM SIGIR Conference on Research and Development in Information Retrieval, Berkeley, CA, 1999.

[11] R.Hull, "*Managing Semantic Heterogeneity in Databases: a Theoretical Perspective*", in Proceedings of Symposium on Principles of Database Systems, AR, 1997.

[12] J.Madhavan, P.A.Bernstein, E.Rahm, "*Generic Schema Matching with Cupid*", in Proceedings of the International conference on Very Large Data Bases, Italy, 2001.

[13] I.H.Witten, A.Moffat, T.C.Bell, *"Managing Gigabytes: Compressing and Indexing Documents and Images*", Morgan Kaufmann publishers, 1999.