Corpus-based Schema Matching

Jayant Madhavan, Philip Bernstein[1], Kuang Chen, Alon Halevy and Pradeep Shenoy
{jayant, kkchen, alon, pshenoy}@cs.washington.edu, philbe@microsoft.com
University of Washington, [1]Microsoft Research

Abstract

Schema matching is the problem of determining a set of *correspondences* that identify similar elements in two different schemas. In this paper we propose a novel method for matching schemas that leverages previous matching experiences by extracting knowledge from a corpus of known schemas and mappings, and applying this knowledge to match new schemas. We describe the Mapping Knowledge Base that captures the knowledge gleaned from the past; methods to apply the gleaned knowledge to new matching tasks; and interesting tradeoffs in building such a system. Finally, we present preliminary experimental results that demonstrate that the use of past experience can result in an improvement in the generated matches.

1 Introduction

Semantic heterogeneity is a key problem in any data sharing system, be it a federated database [Sheth and Larson, 1990], a data integration system [Wiederhold, 1992], a message passing system [BEA, 2003], a web service, or a peer-data management system [Halevy *et al.*, 2003]. The data sources involved are typically designed independently, and hence use different schemas. To obtain meaningful interoperation, one needs a *semantic mapping* between the schemas, i.e., a set of expressions that specify how the data in one source corresponds to the data in the other.

This paper considers the specific problem of *schema matching* - determining a set of *correspondences* that identify similar elements in different schemas. This is typically the first phase before a more complete set of mapping expressions are constructed. Constructing a match manually is a very labor intensive task that requires complete knowledge of the semantics of the data in the schemas being matched. Given schemas S_1 and S_2, we say that element e_1 in S_1 *matches* element e_2 in S_2, if there exists in the eventual mapping a (possibly complex) expression that relates e_1 to e_2.

Solutions that try to provide some automatic support for schema matching have received steady attention over the years (see [Rahm and Bernstein, 2001] for a recent survey and [Doan *et al.*, 2002; Melnik *et al.*, 2002; Do and Rahm, 2002; Berlin and Motro, 2002] for some subsequent work). Perhaps the key conclusion from this body of research is that an effective schema matching tool requires a principled combination of many techniques, such as linguistic matching of names of schema elements, comparison of their data instances, considering structural similarities between schemas, and using domain knowledge and user feedback.

Our approach is based on the observation that matching tasks are often (at least partially) repetitive. Thus, we are frequently asked to match schemas in the same or overlapping domains where we have matched schemas previously. This suggests that it will be beneficial for a schema matching system to be able to leverage experience from previous matching tasks and other known mappings [Doan *et al.*, 2001; Do and Rahm, 2002]. Specifically, every time we complete a matching task (whether it is partially automated or not) or are provided with a mapping, the match between the schemas provides opportunities for extracting *general purpose knowledge* that can be applied in later tasks. Our solution leverages this intuition by constructing a corpus of schemas, known mappings and results from earlier matching tasks. This is akin to the notion of a document corpus in information retrieval - a tool that has been put to effective use for answering keyword queries over document collections.

This paper proposes a general architecture for a schema matching system that leverages experience from the past. Its key component, called the MKB (*Mapping Knowledge Base*), captures knowledge gleaned from previous matching tasks. At a very coarse level, the MKB learns classifiers for each of the schema elements seen in the past. When the classifier for an element e is applied to a new schema element e_1 (and possibly accompanying data values), it predicts the degree of similarity between e and e_1. To find a match between schemas S_1 and S_2, the classifiers in the MKB are applied on the elements of each. If two schema elements, one from S_1 and one from S_2, are predicted to be similar to the same elements in the MKB, then we predict that they are similar to each other. The classifiers, as we shall see, are trained from knowledge gleaned from previous tasks, and used in future matching tasks. We provide a proof of concept with preliminary experimental results that demonstrate the utility of the MKB in improving matching accuracy.

2 Mapping Knowledge Base

In this section we describe the components of the MKB and the process of extracting knowledge from past experience.

To motivate our approach, suppose that we are trying to determine whether an element e_1 in schema S_1 and an element e_2 in schema S_2 match each other. Consider a hypothetical

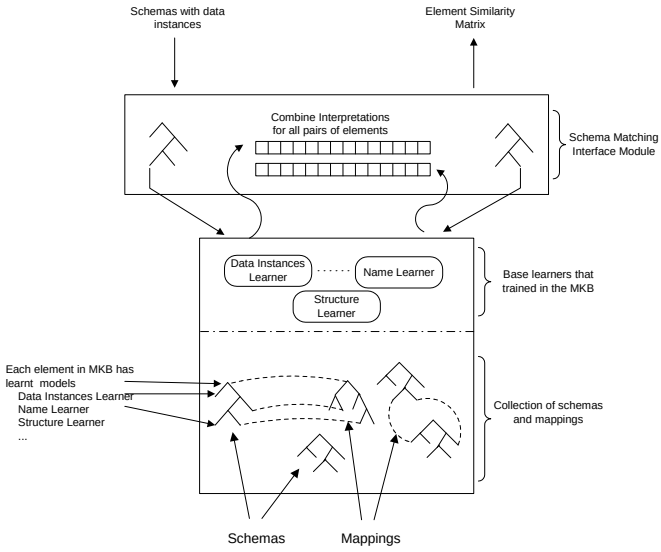


Figure 1: Architecture of the MKB

situation in which we have (1) a *universal schema* for the domain of S_1 and S_2 - one that is aware of every possible element in the domain, and (2) a *perfect* similarity measure or an oracle that returns the similarity between any element, such as e_1 and e_2, and the elements in the universal schema. In this situation, matching is relatively straightforward: if e_1 and e_2 match the same element(s) in the universal schema, then we will deem them to be similar to each other.

However, neither the universal schema nor the perfect similarity measure exist in practice. The key behind our approach is to build a structure, called the *Mapping Knowledge Base* (MKB), that captures our knowledge about the domain of S_1 and S_2, and a set of techniques that try to approximate the perfect similarity measure. Our MKB is *not* a universal schema in the sense of being a well designed schema that captures all the aspects of a domain. Instead, it evolves automatically as it receives more schemas in the domain and witnesses more matches between these schemas.

Our goal is to prove that such a MKB can in fact be built and used effectively for schema matching.

2.1 Architecture of the MKB

The MKB is organized in two parts (see Figure 1).

Schemas and Mappings: Schemas and mappings are continually added to the MKB. The information maintained about each schema includes the name, relationships, data types, and any available data instances and textual descriptions of each element in the schema. Each mapping is maintained as a set of matches (i.e., correspondences) between pairs of elements of two schemas.

Learned Knowledge: We extract and maintain knowledge about *every unique element* that is present in some schema seen by the MKB. This extracted knowledge the MKB captures for each element is in the form of *classifier models* that are trained to recognize that element. The learned model for an element can be used to make a probabilistic prediction of whether a different element is similar to it. There are different cues that can be used in matching schema elements and there different classifiers that can used to exploit each type of cue. Hence we train an ensemble of classifiers with a separate classifier (called a *base learner*) to exploit each of the available cues. As in [Doan *et al.*, 2001], we employ a *multi-strategy learning* approach to combine, for each element in the MKB, the predictions of the different classifiers trained for that element. Described below are some of the different base learners that we train. Our design is rather general, and hence any schema matching technique that can evolve from past experience can be adapted as a base learner.

Base Learners: The different base learners we currently use are described below. We later describe a general technique to extract training data for each of these.

- **Name Learner:** that exploits the names of the elements. The learning technique we use relies on parsing the name of any element into its corresponding *n-grams*[1], and is based on the TF/IDF (term frequency / inverse document frequency) measure [Salton, 1971]. We found this to work well in the presence of short forms, incomplete names and spelling errors that are common in schema names.
- **Description Learner:** that exploits text descriptions that often are available with the schemas. These descriptions typically bring out the semantics of the element more than the element names. We use a naive Bayes classifier [Domingos and Pazzani, 1997] for this purpose. Naive Bayes has been successfully applied in the past to the text classification problem.
- **Instance Learner:** that exploits features in data instances of elements. Instances of similar elements might share features – such as the number of words or digits or characters, the presence of special symbols ($\$, \%, \#$), etc., even though the exact same instances do not occur in different databases. For example, instances of monetary elements can be expected to have a $ symbol. We extract feature vectors from each instance and train a naive Bayes classifier on the examples.
- **Data type Learner**: that exploits the data types of elements. We learn (1) a single global type compatibility table that estimates the probability that an element of one data type gets matched to an element of a particular different data type, and (2) for each element a probability distribution of the data types of elements that have been matched to it.
- **Structure Learner:** that attempts to identify other elements that frequently co-occur with an element. We define the *neighborhood* of an element to include its parent, its children and its siblings (*i.e.* when the schema is viewed as a graph). Consider that element s_1 in schema S matches element t_1 in schema T, and s_2, another element in the neighborhood of s_1, matches t_2 in schema T. If t_2 is also in the neighborhood of t_1, we can say

[1] *e.g.* the 3-grams of *itemnum* are *(ite tem emn mnu num)*.

that there is evidence that s_2 co-occurs with s_1 across schemas. Known mappings are a great source for inferring such co-occurrences. We again use a naive Bayes classifier to learn these neighborhood patterns, but the details are beyond the scope of this paper. We consider our structure learner to be an interesting contribution, since the neighborhood patterns being learned for each element are specific to that element and are unlike generic heuristics that state that elements are similar if there are other similar elements in their neighborhood.

Training Data: For each element, each of the different cues can be extracted from the schema it belongs to and used as training data for different classifiers. Further, if two elements in different schemas have been declared to be similar to each other in some schema match, then the data used to train a classifier for one of these elements can be used to train a classifier for the other element. Very briefly, consider the example of the Name Learner that exploits the names of the elements. For an element s_i of schema S, its own name is a positive example and the names of all other elements in the schema are negative examples. Further, given a mapping from S to a schema T, if s_i is mapped to element t_j, then the name of t_j is a positive example for s_i (and vice-versa) and the names of other elements in T are negative examples. This scheme can be used in general for the other base learners also. Thus we are able to extract training data from previously performed mappings, learn better classifiers and are hence able to better recognize new elements that are similar to a given element.

Meta Learner: For each element, we use a meta learner to estimate the relevance of each of the base learners. The meta learner prediction for an element is based on the predictions of the base learners. We use the sigmoid or the logistic function (as opposed to a simple linear combination in [Doan *et al.*, 2001]) for combining predications. In practice, we found that this gives us greater flexibility in choosing base learners. The training data for the meta learner is collected in much the same way as for the base learners. The parameters for the combination are estimated by hill-climbing.

3 Schema Matching using the MKB

In this section, we describe how the learned knowledge in the MKB can be used to match schemas.

Given two schemas S_1 and S_2 that have to be matched, each element in S_1 and S_2 is compared against the elements in the MKB – by applying the different classifiers for the MKB element and combining the results using the meta-learner. For each element e_i in the schema S_1 (or S_2), the result is an *interpretation vector* $\bar{P}_i = \langle p_{ik} \rangle$, where p_{ik} is the belief that e_i corresponds to element c_k in the MKB.

The elements e_i and e_j, of schemas S_1 and S_2 respectively, can be deemed to be similar if they cannot be distinguished using their interpretations. Hence, we compare interpretation vectors $\bar{P}_i$ and $\bar{P}_j$ of elements e_i and e_j and compute a similarity value between them. This similarity value should capture the degree of agreement between the two interpretations. Different possible measures are conceivable for this purpose. Two simple measures are the magnitude of the vector difference of the interpretation vectors, and the vector dot product. However we found the Average Weighted Difference(**AWD**) and its variation the Average Significant Weighted Difference (**ASWD**) to be the most useful.

The AWD measure tries to bias the similarity value such that predictions of the elements in the MKB that are estimated to be similar to either of e_i or e_j are given greater weight. For each element c_k in the MKB, we weight the difference of the predictions for e_i and e_j ($|p_{ik} - p_{jk}|$) by the maximum of p_{ik} and p_{jk}.

$$AWD(\bar{P}_i, \bar{P}_j) = \sum_{k=1}^{n} |p_{ik} - p_{jk}| \times \max(p_{ik}, p_{jk}) \qquad (1)$$

The ASWD is a variation where we consider only elements in the MKB such that $\max(p_{ik}, p_{jk})$ is greater than some threshold. Both these measures eliminate the case where e_i and e_j are deemed similar because neither was similar to any element in the MKB.

Having computed the similarity value for all pairs of elements in the two schemas S_1 and S_2, the result we obtain is a similarity matrix relating their elements. This matrix can be used directly to predict matches, to boot-strap graph matching algorithms such as [Madhavan *et al.*, 2001; Melnik *et al.*, 2002], or to be combined with results from other matchers in an extensible framework of systems such as [Doan *et al.*, 2001; 2002; Do and Rahm, 2002]. The key observation is that unlike other sources of information that might be used in these systems, here we are not comparing the information in the two schemas to be matched, but are relying on accumulated knowledge from previous matches that have been validated by experts.

4 Preliminary Results

We compare the performance of the MKB in isolation, and in combination, with that of a schema matching technique that does not exploit past matches.

Test Data: We use relational schemas from an INVENTORY domain. The schemas were constructed *independently*, by the students in a senior-level database course. Each participant was given the same high-level domain description (see [Schemas, 2003]). The schemas were moderately diverse: the number of tables in a schema varied from 3 to 7, the number of columns in a schema varied from 31 to 47. We obtained schemas over two other domains – the results were very similar to those reported below.

Basicmatcher: We implemented a BASICMATCHER that uses the learning techniques described earlier, but is trained only on the schemas being matched (similar to [Doan *et al.*, 2002]). Consider an element s in schema S and t in T. The classifier models for s are trained only on the information in schema S (and similarly for t). Similarities between element s in schema S and t in schema T are obtained by applying the learners trained on s to the element t, and vice-versa, and taking the average. Observe that BASICMATCHER is rather sophisticated and can weigh the predictions of the different

base learners (using the meta-learner) automatically and is hence a very powerful *strawman*.

Combination: We implemented COMBINATION that uses the average of the similarity computed for each element by the BASICMATCHER and the MKB.

Performance measure: In order to compute the match for any element, we simply report the element in the other schema that has the highest similarity value. We report the recall of each of the techniques, *i.e.* the fraction of the correct matches that were obtained using the particular technique. The correct matches were generated by us manually, and were often not easy to compute.

Matching performance of the MKB: The MKB was seeded with 8 schemas and 6 mappings between randomly chosen pairs of schemas in the corpus. So a schema participates in 1.5 mappings on the average. We compare the performance of the MKB and BASICMATCHER on schemas that are in the same domain but not in the MKB. Figure 2a compares the MKB and BASICMATCHER on four different schema-pairs in the INVENTORY domain. The recall values are averaged over 3 trials where the schemas in the MKB were kept constant but the mappings were chosen randomly. For the reported results, the interpretation vectors were combined using the ASWD measure that was found to work best in general.

The graph shows that the performance of the MKB is comparable to BASICMATCHER. This is already encouraging, since they are being trained on disjoint training data. Figure 2b presents a closer look – the average number of matches that the MKB found, but BASICMATCHER did not, and vice-versa. This shows that each of these techniques discovers matches that were not discovered by the other. This result is encouraging and indicates the utility of the MKB. The performance of COMBINATION (Figure 2a) illustrates that the two techniques are combined easily to increase the resulting recall. Hence, this present a case for using the MKB as an *additional* source of information for schema matching. Observe that the results of the MKB can be output in the form of a similarity matrix to facilitate its integration into a more sophisticated combinational scheme, such as [Doan *et al.*, 2001; Do and Rahm, 2002] to further improve matching performance.

Evolution of the MKB: Figure 2c shows the results of an experiment in which the MKB was seeded with 8 schemas, as in the basic experiment, but the mappings were added to the MKB one by one. At each step, the MKB's recall on matching the two input schemas was measured. The figure brings out two features clearly: First, even having only schemas in the MKB, in the absence of mappings between them, provides a fair amount of information for mapping two schemas with the MKB. Second, the performance of the MKB improves steadily as more mappings are added, due to the additional training data being provided to the learners. This gain can be as high as 30%. This experiment validates our intuition that mapping history is an important source of information for the matching process.

5 Related Work

As noted earlier, most previous matching algorithms have used linguistic similarities among names in the schema as well as structural similarities (see survey [Rahm and Bernstein, 2001]). However, these cues have only been taken from the two schemas being matched. In contrast, the key to our approach is in harvesting these types of information from collections of example schemas and matches, and applying the past experience to new matching tasks.

The LSD system [Doan *et al.*, 2001] exploited previous matching tasks in a very limited context – mapping multiple data sources to a *single* mediated schema. Classifiers were trained to recognize the different (but fixed set) of elements in this single schema, and then applied to new schema elements that had to again be mapped to the same mediated schema. Our goal is to use previous mappings for a match between *any pair* of schemas, and therefore we need to extract general purpose knowledge, not only knowledge w.r.t. a particular schema.

The COMA system also makes use of stored mappings [Do and Rahm, 2002]. Given two schemas S_1 and S_2 that are to be matched, COMA's reuse component looks for a schema S in its reuse library for which it has stored matches between S and S_1, and between S and S_2. These stored results are combined to produce a new match. Their technique has a rather restricted application, since S_1, S_2,and their mappings to S have to already exist (which is not the common case).

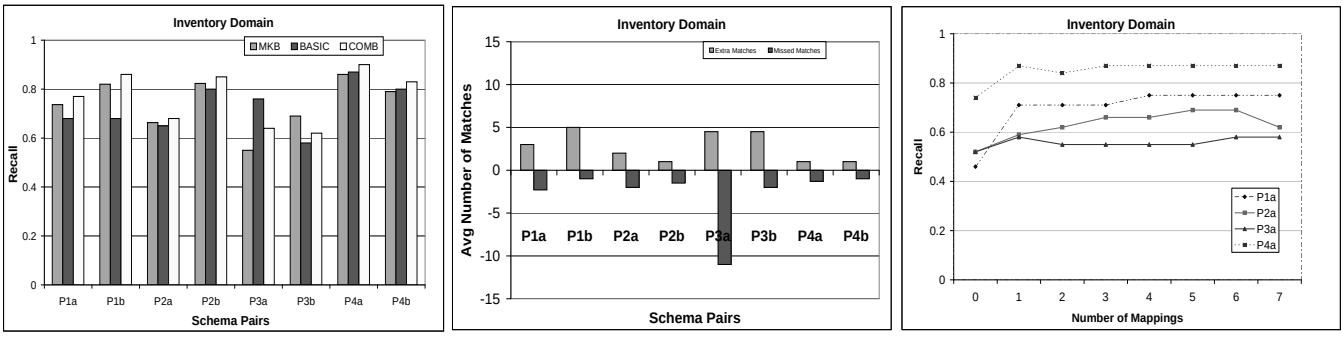
In [Berlin and Motro, 2002] a technique in similar spirit to the MKB was proposed – an *element dictionary* was incrementally maintained. However the system was much more simplistic, *e.g.* the only cues exploited were data examples, and the different tradeoffs in constructing an entire system were not considered.

6 Conclusions and Future Work

Our preliminary results suggest that the MKB built by gleaning knowledge from a corpus of known schemas and and previous mappings has the potential for being a very powerful resource, especially for schema matching.

We see a number of promising avenues for future research. Foremost among these are the issues identified below.

- **Learning Techniques and Measures**: We plan to explore the use of many more matching techniques, such as using a thesaurus for name matching, and their adaptation into our framework. We also plan to investigate sophisticated techniques for combining interpretation vectors. This is a special case of multi-dimensional clustering – a known hard problem.
- **Granularity of the MKB**: Currently, the MKB learns an ensemble of models for every element of every schema present in the corpus. Hence, there is an obvious potential for scalability problems. As the number of schemas increase, a large number of elements will either be irrelevant or redundant. Two promising directions seem to be merging and pruning elements in the MKB based on their predictions made for different matching tasks – only the



(a) Recall: MKB *vs* BASICMATCHER *vs* COMBINATION

(b) Mismatch: MKB−BASICMATCHER

(c) Evolution of the MKB

Figure 2: Performance of the MKB

elements that contribute in a non-redundant way to identifying elements to be similar or different (in new matching tasks) need to be maintained. Redundant elements can be collapsed and irrelevant ones pruned away. The exact mechanism for collapsing elements may be quite subtle, as is deciding what knowledge to learn about the resulting collapsed elements.

- **Computation efficiency**: The computational efficiency of the learning part of our system is less of a concern, since it is done offline. However, since the schema matching is done interactively, as the MKB gets larger, we will need to quickly prune significant parts of the MKB from consideration in order to provide interactive performance to a designer.
- **Domain-dependence of the** MKB: We are interested in studying the extent of domain reliance on the matching performance. More specifically, we are interested in whether a MKB trained on a particular domain can be used directly or easily adapted to match schemas in a related or even completely different domain.

The MKB demonstrates the potential for schema and mapping corpuses in matching schemas; however, there are clearly several design issues yet to be explored in order to harness this potential effectively. At a higher level, our approach points towards the feasibility of building adaptive applications that leverage off large collections of past knowledge, and we hope in future work to carry this insight and experience over to other problem domains.

References

[BEA, 2003] http://edocs.bea.com/tuxedo/msgq/mqscgde/2develop.htm, 2003.

[Berlin and Motro, 2002] Jacob Berlin and Amihai Motro. Database Schema Matching Using Machine Learning with Feature Selection. In *Proceedings of the 14th International Conference on Advanced Information Systems Engineering (CAiSE)*, 2002.

[Do and Rahm, 2002] Hong-Hai Do and Erharm Rahm. COMA - A System for Flexible Combination of Schema Matching Approaches. In *Proceedings of the 28th International Conference on Very Large Databases (VLDB)*, 2002.

[Doan *et al.*, 2001] AnHai Doan, Pedro Domingos, and Alon Y. Halevy. Reconciling Schemas of Disparate Data Sources: A Machine Learning Approach. In *Proceedings of the ACM SIGMOD Conference*, 2001.

[Doan *et al.*, 2002] Anhai Doan, Jayant Madhavan, Pedro Domingos, and Alon Y. Halevy. Learning to Map between Ontologies on the Semantic Web. In *Proceedings of the 11th International World Wide Web Conference (WWW)*, 2002.

[Domingos and Pazzani, 1997] Pedro Domingos and Micheal Pazzani. On the Optimality of the Simple Bayesian Classifi er under Zero-One Loss. *Machine Learning*, 29:103–130, 1997.

[Halevy *et al.*, 2003] Alon Y. Halevy, Zachary G. Ives, Dan Suciu, and Igor Tatarinov. Schema Mediation in Peer Data management Systems. In *Proceedings of the International Conference on Data Engineering (ICDE)*, 2003.

[Madhavan *et al.*, 2001] Jayant Madhavan, Philip A. Bernstein, and Erhard Rahm. Generic Schema Matching with Cupid. In *Proceedings of the 27th International Conference on Very Large Databases (VLDB)*, 2001.

[Melnik *et al.*, 2002] Sergey Melnik, Hector Garcia-Molina, and Erhard Rahm. Similarity Flooding: A Versatile Graph Matching Algorithm. In *Proceedings of the 18th International Conference on Data Engineering (ICDE)*, 2002.

[Rahm and Bernstein, 2001] Erhard Rahm and Philip A. Bernstein. A survey on approaches to automatic schema matching. *VLDB Journal*, 10(4), 2001.

[Salton, 1971] Gerard Salton, editor. *The SMART Retrieval System—Experiments in Automatic Document Retrieval*. Prentice Hall Inc., Englewood Cliffs, NJ, 1971.

[Schemas, 2003] http://www.cs.washington.edu/education/courses/444/03wi/project/proj1.htm, 2003.

[Sheth and Larson, 1990] Amit P. Sheth and James A. Larson. Federated database systems for managing, distributed, heterogenous, and autonomous databases. *ACM Computing Surveys*, 22(3):183–236, September 1990.

[Wiederhold, 1992] Gio Wiederhold. Mediators in the architecture of future information systems. *IEEE Computer*, pages 38–49, March 1992.