Journal of Computer Applications (JCA)
ISSN: 0974-1925, Volume IV, Issue 4, 2011

Effective Integration by Inter-Attribute Dependency graphs of schema matching

S.Thiruvenkatasamy [a,*], C.Sathyapriya [b,1]

Abstract—**Schema-matching problem is the most basic level refers to the problem of mapping schema elements in one information repository to corresponding elements in a second repository. Schema matching is one of the key challenges in information integration. In fact, significant improvements will be observed. The technique that is in the existing paper is an instance-based technique. I emphasize that our claims is that this technique, is not the best of techniques to apply as a useful addition to a suite of automated schema mapping tools. I propose a new usage-based schema matching technique. The proposed technique exploits the usage information of the attributes in the query logs to find matches, in contrast to relying on the schema information or the data instances. The existing methods for weighted graph matching Algorithms for schema matching is compared with the new proposed methods of 1) Direct Tree Search 2) Ullman 3) Clique 4) Hash coding. The performance and accuracy of the schema matching of the various techniques are compared and analyzed.**

***Index Terms*—Attribute dependency, graph matching, quadratic assignment problem (QAP), Schema matching, WGMP.**

I. INTRODUCTION

The schema-matching problem at the most basic level refers to the problem of mapping schema elements (for example, columns in a relational database schema) in one information repository to corresponding elements in a second repository. While schema matching has always been a problematic and interesting aspect of information integration, the problem is exacerbated as the number of information sources to be integrated, and hence, the number of integration problems that must be solved, grows. Such schema-matching problems arise both in "classical" scenarios such as company mergers and in "new" scenarios such as the integration of diverse sets of queriable information sources over the web. Purely manual solutions to the schema-matching problem are too labor intensive to be scalable; as a result, there has been a great deal of research into automated techniques that can speed this process by either automatically discovering good mappings, or by proposing likely matches that are then verified by some human expert.

In this paper, we present such an automated technique that is designed. Assistance in the particularly difficult cases in which the column names and data values are "opaque," and/or cases in which the column names are opaque and the data values in multiple columns are drawn from the same domain. Our approach works by computing the "mutual information" between pairs of columns within each schema, and then using this statistical characterization of pairs of columns in one schema to propose matching pairs of columns in the other schema. To clarify our aims and provide some context, consider a classical schema mapping problem where two employee tables are integrated. First, one logical approach is to compare attribute names across the tables. Some of the attribute names will be clear candidates for matching, due to common names or common parts of names. Using the classification approach is on schema-based matching, the technique we propose in this paper is also an instance-based technique. However, it applies to the cases where previously proposed techniques do not apply because

1. It does not rely on any interpretation of data values
2. It considers correlations among the columns in each table.

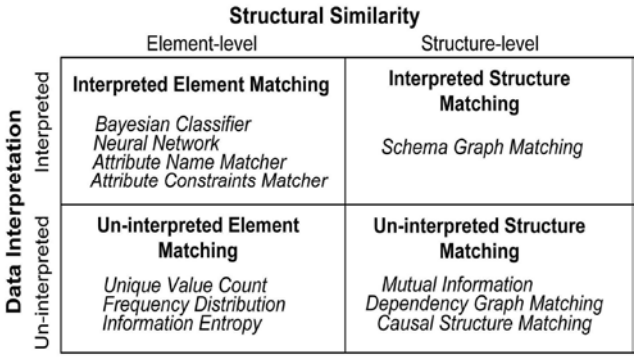
We emphasize that our claim is not that our technique dominates previously proposed techniques (it does not); rather, since it applies where previous techniques do not apply; it is a useful addition to a suite of automated schema mapping tools. To gain insight into our approach, consider the example tables in Table 1. Suppose these tables are from two automobile plants in different companies. Imagine that the column names of the second table and data instances in columns B and C are some incomprehensible values to the schema-matching tools. Conventional instance-based matchers may find correspondence between the columns Model and A due to their syntactic similarity. However, no further matches are likely to be found because the two columns B and C cannot be interpreted, and they share exactly same statistical characteristics; that is, they have the same number of unique values, similar distributions, and so forth.

Table 1
Two Tables from Car Part Databases

Model	Color	Tire	A	B	C
XLE	White	P2R6	GL3.5	b1	c1
XG2.5	Silver	XR5	XGL	b2	c2
LE	Red	GM6	XE	b3	c3

To make progress in such a difficult situation, our technique exploits dependency relationships between the attributes in each table.

For instance, in the first table in Table 1, there will exit some degree of dependency between Model and Tire if model partially determines the kinds of tires a car can use. On the other hand, perhaps Model and Color are likely to have very little interdependency. If we can measure the dependency between columns A and B and columns A and C, and compare them with the dependency measured from the first table, it may be possible to find the remaining correspondences.



Manuscript received, 14-Dec-2011.
S.Thiruvenkatasamy [a,*], Assistant Professor,
Department of Computer Science and Engineering,
Shree Venkateshwara Hi-Tech Engineering College, Gobi
E-mail: samysdotcom@yahoo.com
C.Sathyapriya [b, 1] , Assistant Professor,
Department of Information Technology,
Shree Venkateshwara Hi-Tech Engineering College, Gobi
E-mail: c.sathyapriya@gmail.com

As we can see, an advantage of using dependency relations in schema matching is that this approach does not require data interpretation; that is, even if the data sets in the schemas to be matched use different encodings, considers correlations among the columns in each table we can still measure the dependency relations. As a result, our proposed matching technique can be applied to multiple unrelated domains without retraining or customization. We refer to matching techniques that are not dependent of data interpretation as uninterrupted matching.

II. UNINTERPRETED MATCHING

In this section, we describe in detail our uninterpreted structure-matching technique[13]. The algorithm[3] takes two table instances as input and produces a set of matching node pairs. Our approach works in two main steps as shown below:

1. G1ß **Table2DepGraph** (S1);
   G2ß **Table2DepGraph** (S2) and
2. (Gi(a),G2(b)) ß **GraphMatch**(G1,G2)

Where Si = input table, Gi = dependency graph,
(G1(a), G2(b)) = matching node pair.

The function Table2DepGraph() in the first step transforms an input table like the one shown in Fig. 2a into a dependency graph shown in Fig. 2c. The function GraphMatch() in the second step takes as input the two dependency graphs generated in the first step and produces a mapping between corresponding nodes in the two graphs. The two steps are described in detail later in this section.

A. Modeling Dependency Relation

Consider the example illustrated in Fig. 2. Figs. 2a and 2b show two four-column input tables, and Figs. 2c and 2d show the corresponding dependency graphs. The Table2DepGraph () function produces such dependency graphs by calculating the pair wise mutual information over all pairs of attributes in a table and structuring them in an undirected labeled graph. For instance, each edge in the dependency graph G1 (Fig. 2c) has a label indicating mutual information between the two adjacent nodes; for example, the mutual information between nodes A and B is 1.5, and so on. The label on a node represents the entropy of the attribute, which is equivalent to its mutual information with itself or self-information. Hence, we can model our dependency graph in a simple symmetric square matrix of mutual information, which is defined as follows:

B. Dependency graph

Let S be a schema instance with n attributes and aj(l < i < n) be its ith attribute. We define dependency graph of schema S using square matrix M by M = (rriij), where nrnj = MI(cii; cij), 1 < i, j < n. The intuition behind using mutual information as a dependency measure is twofold:

1) it is value independent; hence, it can be used in uninterpreted matching
2) it captures complex correlations between two probability distributions in single number, which simplifies the matching task in the second stage of our algorithm.

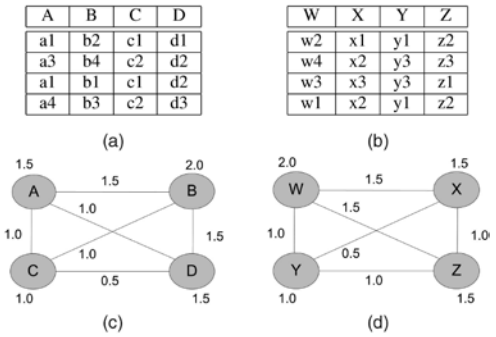


Figure 2.Two input table examples and their dependency graphs.

A weight on an edge represents mutual information between the two adjacent attributes and a weight on a node represents entropy of the attribute (or equivalently, self-information MI (A; A)). (a) Example table S1. (b) Example table S2. (c) Dependency graph G1 of S1. (d) Dependency graph G2 of S2.

III. MATCHING STRATEGIES

In this section, we investigate efficient approximation algorithms for the graph-matching problem in the second step of our approach. We focus our discussion particularly on the bijective mapping problem for two reasons.

1) The solution to this problem can be used as an integral part of the general solutions for the other two problems because the other problems can be formulated with multiple bijective mappings. For example, an injective mapping between graphs S (m nodes) and T (n nodes, where m > n) can be solved by finding an n node sub graph of S that minimizes the bijective[5] mapping distance to T. The partial mapping problems can be formulated similarly. Of course, this may not be an ideal solution for them. Evaluating this approach versus approaches specifically tailored to the injective and partial mapping problems is an interesting area for future work.
2) The problem can be formulated in a clean mathematical optimization framework and because of that a large number of approximation algorithms have been developed. We will investigate a spectrum of the solutions covering a wide range of optimization techniques that can work for our problem context.

In what follows, we introduce five weighted graph-matching algorithms:

1. Umeyama's eigen-decomposition (ED) approach,
2. Linear programming (LP),
3. Convex quadratic programming (QP),
4. Hill climbing (HC), and, finally,
5. Branch and bound.

A. Eigen-Decomposition Approach

Umeyama introduced a polynomial time approximate algorithm for WGMP in the context of a vision problem. The proposed algorithm relaxes the original problem of finding the permutation matrix P to the problem of finding an orthogonal matrix X that minimizes the metric. The algorithm then finds an approximate solution for the original problem by manipulating the solution obtained from the relaxed problem. The relaxed problem can be written a

$$minp||As - PTATP||F > minx||AS - XTATR||F$$

B. Linear Programming Approach

A lm ohamad and Duffuaa introduced an LP approach for the WGMP. Whereas Umeyama [9] relaxed the permutation matrix to an orthogonal matrix, their algorithm relaxes the original problem of finding a permutation matrix P to the problem of finding a doubly stochastic matrix X that minimizes the distance between the graphs. A doubly stochastic matrix X = (x^) has a linear constraint as follows:

Xij0,xij=1., xij=1, forall I and j

Obviously, a permutation matrix is a special case of a doubly stochastic matrix. Another interesting aspect of this approach is that it optimizes the L1 distance metric and not the L2 distance metric. This essentially makes it possible to formulate WGMP as an LP optimization problem.

C.Convex Quadratic Programming Approach

Anstreicher and Brixius introduced a new bound for the quadratic assignment problem (QAP) [4]. The new bound is obtained by relaxing QAP to a convex QP optimization problem. Schellewald et al. showed that WGMP can be reduced to QAP and solved QAP by minimizing the new bound much the same way; we can relax WGMP directly to a convex QP problem. Unlike the LP

relaxation, we do not need to choose an alternative metric. The relaxation process is given below.

D. Hill-Climbing Approach

So far, we have considered three deterministic approximation algorithms for WGMP[11]. All of them are based on the relaxation of the original problem to an algebraic optimization framework. We now introduce a simple nondeter-ministic, iterative improvement algorithm. The HC algorithm is simply a loop that moves, in each state transition, to a state where the most improvement can be achieved. A state represents a permutation that corresponds to a mapping between the two graphs. We limit the set of all states reachable from one state in a state transition, to a set of all permutations obtained by one swapping of any two nodes in the permutation corresponding to the current state. The algorithm stops when there is no next state available that is better than the current state. As we can see, it is nondeterministic; depending on where it starts, even for the same problem, the final states may differ. To avoid being stuck in a local minimum after an unfortunate run, the usual practice is to perform some number of random restarts.

E. Branch and Bound Approach

Due to the combinatorial nature of the problem, an exact search algorithm would hardly be practical, but we present here one based on the branch andbound method for the purpose of comparison. As we will see in the experiments, this approach cannot handle problems as large as those handled by we will see in the experiments, this approach cannot handle problems as large as those handled by problems large as those handled by the approximate algorithms. Our implementation of the branch and bound is to approach for WGMP algorithm.

The branch and bound[7] in Algorithm generates an initial permutation for the mapping using a fast approximate algorithm. It then constructs a permutation tree using the initial mapping as the seed. It traverses the tree in depth first order while improving the distance bound. If the current prefix produces a distance worse than the current bound, it branches to the next sibling without exploring the sub tree. When it reaches to the leaf it computes the distance with the current permutation. If it is better than the current bound, it updates the bound and backtracks to the next available permutation. So far, we have investigated algorithms for WGMP[2]. These algorithms take as input two dependency graphs generated in the first step and find the mapping between them. Among the three mathematical optimization algorithms, the ED approach is the fastest. It runs asymptotically in the order of n3, where n is the number of nodes in an input graph. The other two algorithms, LP and convex QP approaches, run asymptotically in the order of n6 for the same n. The branch and bound is obviously the slowest as it performs the exact search. Lastly, the HC algorithm is a heuristic interactive improvement algorithm, and its running time largely depends on the number of restarts, seed selection.

IV. EXPERIMENTS

In this section, we present the results of schema-matching experiments using our proposed approach.

The validation is performed in two steps. In the first step, we attempt to address the first question (in Section II) asking if the proposed schema-matching framework works given the assumption that we have a perfect graph-matching algorithm.

A. Injective Mapping

In this experiment, we kept the target schema size constant at 22 attributes while increasing the source schema size from 2 to 20 attributes. As was the case in the bijective mapping experiments, the census data match result is slightly better than that of the lab exam data. For example, when the schema size reached 20, census data yielded 91 percent precision while lab exam data turned out only 80 percent.

In both data sets, mutual information matching outperformed entropy-only matching. The precision of lab exam data matching was improved approximately 31 percent (from 61 percent in entropy only to 80 percent with mutual information), while precision in census data improved 12 percent (from 81 percent in entropy only to 91 percent for mutual information). We see that mutual information was more helpful for the lab exam data than it was for the census data. This is because in the lab exam data, more attributes had similar entropy, so that entropy-only mapping was more likely to get "confused." Turning now to compare our two metrics, euclidean and normal, the euclidean distance[9] metric yielded better results overall in both data sets. To summarize the situation up to this point, we have considered the performance of two matching methods and two distance metrics, and the results have been consistent with those in the bijective mapping case. However, there is a notable difference: the precision of matching in the injective case improves as the size of source schema increases, which is the opposite of what we saw in the bijective mapping.

B. Data Set

We used real-world data sets from two different data domains: medical data[6] and census data[8]. The medical data set we used in our experiments contains patients' lab exam results for diagnosing thrombosis shows the measured entropies of 30 randomly chosen attributes of the thrombosis lab exam data, and it also shows a fragment of the first 10 (out of the 30) attributes' data values. The original table contains 12 years worth of patient exam records, which is approximately 50,000 tuples, and each tuple consists of 44 attributes representing test types. The column data types are mostly numeric, and a significant portion of the table is left blank. Our basic experimental technique with the medical data set was to range partition the original table into two sub tables based on exam dates and to use these two sub tables for experiments. Obviously, we "knew" the correct answer for the mapping, but the mapping algorithm did not. For our second data set, we used census data. Attribute entropies and a table fragment from the census data set, respectively. We used two state census data files, CA and NY, in our experiments, each table consists of 240 attributes. We ran the experiments over a randomly chosen set of 30 attributes.

C.Approximate Matching Algorithms for Schema Matching

In this section, we try to address the second question asking if there is an efficient algorithm for matching that works for our problem context. We present the experimental results for valuating the algorithms introduced in Section 3. We examined the five algorithms:

1. Umeyama's ED approach,
2. the LP approach,
3. the convex QP approach,
4. the HC approach, and finally,
5. the branch and bound algorithm.

For HC, we used five iterations, each from a randomly chosen starting point, and chose the best result from the five trials. The computational complexity of an algorithm is another important factor to consider when we choose an algorithm. An exact search algorithm such as branch and bound would obviously be the best in terms of the accuracy but it could be too slow for some of the large problems.

V. CONCLUSION AND FUTURE ENHANCEMENTS

We have proposed a two-step schema-matching technique that works even in the presence of opaque column names and data values. In the first step, we measure the pairwise attribute correlations in the tables to be matched and construct dependency graph using mutual information as a measure of the dependency between attributes. In the second stage, we find matching node pairs across the dependency graphs byrunninga graph-matching algorithm[10]. To our knowledge, our work is the first to introduce an uninterrupted matching technique utilizing interattri-bute dependency relations. We have shown that while a single column uninterrupted matching such as entropy-only matching can be

somewhat effective alone, further improvement was possible by exploiting interattribute correlations. In this work, we also investigated approximation algorithms for the matching problem and showed that an efficient implementation can be possible for our approach. Among the algorithms we evaluated, the HC approach showed the most promising results. It found close to optimal solutions very quickly, suggesting that the graph-matching problems arising in our schema-matching domain are amenable to HC[12].

A good deal of room for future work exists. In our work, we have only tested two simple distance metrics— Euclidean and normal. It is possible that more sophisticated distance metrics could produce better results. It would also be interesting to evaluate other dependency models using different uninterrupted methods.

REFERENCES

[1] Baatz.S, Frank.M, Kuhl.C, Martini.P, and Scholz.C,“ Conjunctive Query Equivalence of Keyed Relational Schemas,**”,** in Proceedings of the IEEE INFOCOM, the Annual Joint Conference of the IEEE Computer and Communications Societies, June 2007, pp. 782–790.

[2] R. Fagin, P.G. Kolaitis, L. Popa, and W.C. Tan, “Quasi-Inverses of Schema Mappings,” Proc. 26th ACM SIGACT-SIGMOD-SIGART Symp. principles of Database Systems (PODS ’07), pp. 123-132, 2007.

[3] R. Fagin, P.G. Kolaitis, L. Popa, and W.C. Tan, “Quasi-Inverses of Schema Mappings,” Proc. 26th ACM SIGACT-SIGMOD-SIGART Symp. Principles of Database Systems (PODS ’07), pp. 123-132, 2007.

[4] B. He and K.C.-C. Chang, “Making Holistic Schema Matching Robust: An Ensemble Approach,” Proc. 11th ACM SIGKDD Int’’ Conf. Knowledge Discovery in Data Mining (KDD ’09), pp. 429-438,2009.

[5] S. Melnik, A. Adya, and P.A. Bernstein, “Compiling Mappings to Bridge Applications and Databases,” Proc. ACM SIGMOD ’09,pp. 461-472, 2009.

[6] C. Domshlak, A. Gal, and H. Roitman, “Rank Aggregation for Automatic Schema Matching,” IEEE Trans. Knowledge and Data Eng., vol. 19, no. 4, pp. 538-553, Apr. 2007.

[7] S. Castano, V. Antonellis, and S. Vimercati, “Global Viewing of Heterogeneous Data Sources,” IEEE Trans. Knowledge and Data Eng., vol. 13, no. 2, pp. 277-297, Mar./Apr. 2001.

[8] W.-S. Li and C. Clifton, “SEMINT: A Tool for Identifying Attribute Correspondences in Heterogeneous Databases Using Neural Networks,” J. Data and Knowledge Eng., vol. 33, no. 1, Dec. 2009.

[9] R. Fagin, P.G. Kolaitis, L. Popa, and W.C. Tan, “Quasi-Inverses of Schema Mappings,” Proc. 26th ACM SIGACT-SIGMOD-SIGART Symp. Principles of Database Systems (PODS ’09), pp. 123-132, 2009.

[10] R. Fagin, P.G. Kolaitis, L. Popa, and W.C. Tan, “Composing Schema Mappings: Second-Order Dependencies to the Rescue,” ACM Trans. Database Systems, vol. 30, no. 4, pp. 994-1055, 2007.

[11] P. Shvaiko and J. Euzenat, A Survey of Schema-Based Matching Approaches, pp. 146-171, 2005.

[12] P.A. Bernstein, T.J. Green, S. Melnik, and A. Nash, “Implementing Mapping Composition,” Proc. 32nd Int’l Conf. Very Large Data Base (VLDB ’06), pp. 55-66, 2006.

[13] C. Domshlak, A. Gal, and H. Roitman, “Rank Aggregation for Automatic Schema Matching,” IEEE Trans. Knowledge and Data Eng., vol. 19, no. 4, pp. 538-553, Apr. 2007.

BIOGRAPHY

S.Thiruvenkatasamy, received his Post Graduate Degree in Master of Engineering in Computer Science, from Karpagam University, Coimbatore, Tamil Nadu, India. Currently he is working as Assistant Professor in the Department of Computer Science and Engineering in Shree Venkateshwara Hi-Tech Engineering College, Gobichettipalayam, Tamil Nadu, India. He published a book “An Excellent Guide for Visual C#.Net” and had presented 6 Papers in Various National Conferences. His interest includes Data mining, computer networks and Network security.

C.Sathyapriya, received her Post Graduate Degree in Master of Engineering in Computer Science, from Velalar College of Engineering and Technology, Erode, Tamil Nadu, India. Currently she is working as Assistant Professor in the Department of Information Technology in Shree Venkateshwara Hi-Tech Engineering College, Gobichettipalayam, Tamil Nadu, India. She had presented 9 papers in Various National Conference and also she presented 1 International Conference on “A Test Generation Method to Find Errors in HTML Language” in VIT University, Vellore, Tamil Nadu, India on 21st April 2011. Her interest includes Software Engineering, Computer Networks and Data Mining.