Volume : 2 | Issue : 4 | April 2013 • ISSN No 2277 - 8160

Research Paper | Computer Science

A Survey of Schema Matching Using Interattribute Dependencies

D. John Aravindhar — Research Scholar, School of Computing Sciences, Hindustan University, Chennai-603103

Dr. E. R. Naganathan — Professor and Head, School of computing sciences, Hindustan University, Chennai-603103

ABSTRACT

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

KEYWORDS: Schema matching, attribute dependency, graph matching

1. Introduction

Finding correspondences between elements of data schemas or data instances is required in many application scenarios. This task is often referred to as matching. Consider a comparison shopping website that aggregates product offers from multiple independent online stores. The comparison site developers need to match the product catalogs of each store against their combined catalog. For in-stance, the 'product code' field in one catalog may match the 'product ID' and 'store ID' fields in the combined catalog. Or, think of a merger between two corporations, both of which need to consolidate their relational databases deployed by different departments [1].

To clarify our aims and provide some context, consider a classical schema mapping problem where two employee tables are integrated. How should we determine which attributes in one table should be mapped to which attributes in the other table? 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 given in [3], such an approach is an example of schema-based matching [5], [4]. However, for many columns, schema-based matching will not be effective because different institutions may use different names for semantically identical attributes, or use similar names for unrelated attributes.

When schema-based matching fails, the next logical approach is to look at the data values stored in the schemas. Again referring to the classification from [3], this approach is called instance-based matching [6], [7], [8]. Instance-based matching also will work in many cases. For example, if we are deciding whether to match Deptin one schema to either DeptName or DeptIDin the other, by looking at the column instances, one may easily find the mapping because DeptName and DeptIDare likely to be drawn from different domains. Unfortunately, however, instance-based matching is also not always successful.

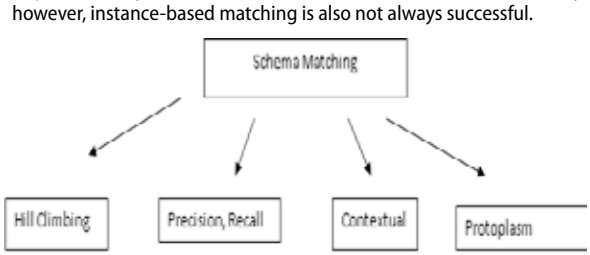


Figure 1: Types of Schema Matching

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 and 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.

2. SCHEMA MATCHING

2.1 HILL CLIMBIMG APPROACH

Jaewoo Kang and Jeffrey F. Naughton [2] proposed a two-step schema-matching technique that worked even in the presence of opaque column names and data values. In the first step, they measured the pair wise attribute correlations in the tables to be matched and construct a dependency graph using mutual information as a measure of the dependency between attributes. In the second stage, they have find matching node pairs across the dependency graphs by running a graph-matching algorithm. They 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 inter attribute correlations. In proposed work, they also investigated approximation algorithms for the matching problem and showed that an efficient implementation can be possible for our approach. Among the algorithms they 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.

2.2 PRECISION AND RECALL

Zohra Bellahsene and Fabien Duchateau [9] provided an overview about what has been done for tuning schema matchers. At first, schema matchers enabled users to configure some of their low-level parameters (e.g., thresholds). They mainly allowed filtering or selecting the output (the set of mappings). They added more flexibility and the set of discovered mappings depends on the configuration of these parameters. More recently, some works went up one level further by selecting the appropriate matcher for a given matching task. These tools lessen the burden of the user by automatically tuning most of the low-level parameters. In the meanwhile, much effort has also been spent to integrate user preferences or input data parameters. Most of them based on machine learning techniques so that schema instances or expert feedback can be used in the process. The integration of such parameters was often an extra means for improving matching quality.

User preferences such as the promotion of precision or recall let users choose how they intend to manage post-match effort. These options were also interesting in contexts where high dynamicity leads to a quick evolution of data sources, thus implying that a high precision is preferred. On the contrary, recall can be promoted when data sources are going to be fully integrated and manually checked. Although a default configuration should still be proposed with a matcher, they believe that they were heading towards a specific configuration of a schema matcher for a given matching task. Namely, various properties of the matching scenario can be computed by the tool. The latter can then deduce, based on previous experiments or properties values, the best configuration.

2.3 CONTEXTUAL SCHEMA MATCHING

Philip Bohannon el al. [10] introduced contextual schema matching, in which the schema matching system infers certain views of source and/or tar-get tables as part of the schema matching process. They have investigated a class of simple and disjunctive contextual conditions that cover many practical examples in which matches were meaningful only under these conditions, and developed novel algorithms for inferring views to characterize these conditions. In response to the increased ambiguity introduced by inferred views. They have presented the results of an extensive experimental study on the performance and quality of our techniques, concluding that contextual matching supports the further automation of schema mapping in the face of (a) horizontal partitioning of schemas and (b) promotions of data values to attributes. Significant future work was suggested by prposed approach on both the theoretical and practical fronts. In particular, views on the target schema should be handled and more complex view conditions may be necessary, for example, if both horizontal partitioning and attribute promotion were encountered by the same match. It would lead to more involved views and added even more challenge to the automated inference of such views. Another issued the need for a new form of constraints to accommodate complex contextual conditions and attribute promotion and associated semantic-association rules. A third issue concerned the interaction between constraints on the target and contextual matches, which called for a systematic method to assure that contextual schema mapping, did not violate the target constraints.

3. PROTOPLASM

Philip A. Bernstein el al. [11] proposed a technique in which most systems-integration work required creating mappings between models, such as database schemas, message formats, interface definitions, and user-interface forms. As a design activity, schema mapping was similar to database design in that it required digging deeply into the semantics of schemas. Schema matching identified elements of two given schemas that correspond to each other. Although there were many algorithms for schema matching, little has been written about building a system that can be used in practice. They described their initial experience building such a system, a customizable schema matcher called Protoplasm. Since schema matching algorithms were inherently fragile, a customizable schema matcher needs to include multiple easily-customizable scripts. One challenge was to define a few such scripts that cover the space of useful matchers and offer a small set of simple customization choices. An important capability of a schema matcher was reuse of previously-developed mappings. Often, a reusable mapping was between sub-schemas of the schema matcher's inputs. The combinatory of finding reusable mappings and applying them to a given mapping problem can be daunting.

4. SCHEMA MANIPULATION OPERATION

Jayant Madhavan el al. [12] proposed a technique in which match was a schema manipulation operation that taken two schemas as input and returns a mapping that identifies corresponding elements in the two schemas. Schema matching was a critical step in many applications, such as XML message mapping, data warehouse loading, and schema integration. In proposed paper, they investigated algorithms for generic schema matching, outside of any particular data model or application. They first presented a taxonomy for past solutions, showing that a rich range of techniques is available. They then proposed a new algorithm, Cupid, that discovered map-pings between schema elements based on their names, data types, constraints, and schema structure, using a broader set of techniques than past approaches. Some of their innovations were the integrated use of linguistic and structural matching, context-dependent matching of shared types, and a bias toward leaf structure where much of the schema content resides.

5. PROBLEMS IN SCHEMA MATCHING

Jaewoo Kang and Jeffrey F. Naughton [13] demonstrate 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. They investigated schema matching techniques that worked in the presence of opaque column names and data values. They proposed a two-step technique that did not rely on the interpretation of data elements or schema elements. To their knowledge their paper was the first to introduce an un-interpreted matching technique utilizing inter-attribute dependency relations. They have shown that while a single column un-interpreted matching such as entropy-only matching can be somewhat effective alone, further improvement were possible by exploiting inter-attribute correlations. The improvement obtained was in the range of 9% to 31% depending on the cardinality constraints for the mapping problem and the data sets used in the experiments. good deal of room for future work exists. In their work, they have only tested two simple distance metrics, Euclidean and normal. It was possible that more sophisticated distance metrics could produce better results. An interesting and important direction would be to search for metrics that specialize in the task of finding correct subset of attributes from target schemas. Another potentially interesting direction would be to find an accurate yet computationally efficient approximation algorithm for the instances of the graph matching problem generated by our approach to the schema mapping problem. It would also be interesting to evaluate other dependency models using different un-interpreted methods. In proposed work, they focused only on matching flat tables. Extending the technique to nested structures (for example, XML or object-oriented schemas) would be another interesting direction for future work.

6. FUTURE DIRECTIONS

In the previous paper [2], various graph matching algorithms have been given. But these algorithms were some drawback, which are given below:

1 Did not provide taxonomy for graph matching.
2 They did not derive a new approximation of the original problem that can improve state-of-the-art algorithms in graph matching.3Analysis of obtaining result was also an important issue.

In this proposed method, a new factorized graph matching (FGM), a novel framework for interpreting and optimizing graph matching problems. It has two-step schema-matching technique that can work even in the presence of opaque column names and data values. Step 1: Measure the pair wise property correlations in the tables to be matched and construct the dependency graph using mutual information as a measure of the dependency between attributes. Step 2: Find matching node pairs across the dependency graphs by applying a graph-matching algorithm. Graph matching is a challenging problem in today's era. These challenging problems mainly due to two reasons: (1) The objective function is non-convex and prone to local minima; (2) The constraints that the solution has to satisfy are combinatorial.

7. CONCLUSION

Factorized graph matching algorithm is an efficient algorithm which will provide optimize matching. This algorithm will reduce many drawback of previous paper. Some of advantages of factorized graph matching is given below: It allows for a more efficient implementation, especially for large graphs. This allows understanding commonalities and differences among many pair-wise graph matching problems.

REFERENCES

[1] Sergey Melnik, Hector Garcia-Molina, Erhard Rahm, "Similarity Flooding: A Versatile Graph Matching Algorithm and its Application to Schema Matching", 2002. | [2]Jaewoo Kang and Jeffrey F. Naughton, "Schema Matching Using Interattribute Dependencies", Ieee Transactions On Knowledge And Data Engineering, Vol. 20, No. 10, October 2008. | [3]E. Rahm and P.A. Bernstein, "On Matching Schemas Automatically,"The VLDB J.,vol. 10, no. 4, Dec. 2001. | [4]T. Milo and S. Zohar, "Using Schema Matching to Simplify Heterogeneous Data Translation,"Proc. 24th Int'l Conf. Very Large Data Base (VLDB),1998. | [5]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. | [6]A. Doan, P. Domingos, and A.Y. Halevy, "Reconciling Schemas of Disparate Data Sources: A Machine-Learning Approach,"Proc. ACM SIGMOD,2001. | [7]K.-J.R.H. Mannila, "Dependency Inference,"Proc. 13th Int'l Conf.Very Large Data Base (VLDB '87),pp. 155-158, 1987. | [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. 2000. | [9]Zohra Bellahsene and Fabien Duchateau, "Tuning for Schema Matching", 2008. | [10]Philip Bohannon, Eiman Elnahrawy, Wenfei Fan, Michael Flaster, "Putting Context into Schema Matching", VLDB Endowment, 2006. | [11]Philip A. Bernstein, Sergey Melnik, Michalis Petropoulos, Christoph Quix, "Industrial-Strength Schema Matching", SIGMOD Record, Vol. 33, No. 4, December 2004. | [12]Jayant Madhavan, Philip A. Bernstein, Erhard Rahm, "Generic Schema Matching with Cupid", Microsoft Research, 2002. | [13]Jaewoo Kang and Jeffrey F. Naughton, "On Schema Matching with Opaque Column Names and Data Values", ACM, 2003.