AgreementMaker: Efficient Matching for Large Real-World Schemas and Ontologies*

Isabel F. Cruz
ADVIS Lab
Dept. of Computer Science
University of Illinois at Chicago
ifc@cs.uic.edu

Flavio Palandri Antonelli
ADVIS Lab
Dept. of Computer Science
University of Illinois at Chicago
flav@cs.uic.edu

Cosmin Stroe
ADVIS Lab
Dept. of Computer Science
University of Illinois at Chicago
cstroe1@cs.uic.edu

ABSTRACT

We present the AgreementMaker system for matching real-world schemas and ontologies, which may consist of hundreds or even thousands of concepts. The end users of the system are sophisticated domain experts whose needs have driven the design and implementation of the system: they require a responsive, powerful, and extensible framework to perform, evaluate, and compare matching methods. The system comprises a wide range of matching methods addressing different levels of granularity of the components being matched (conceptual vs. structural), the amount of user intervention that they require (manual vs. automatic), their usage (stand-alone vs. composed), and the types of components to consider (schema only or schema and instances). Performance measurements (recall, precision, and runtime) are supported by the system, along with the weighted combination of the results provided by those methods. The AgreementMaker has been used and tested in practical applications and in the Ontology Alignment Evaluation Initiative (OAEI) competition. We report here on some of its most advanced features, including its extensible architecture that facilitates the integration and performance tuning of a variety of matching methods, its capability to evaluate, compare, and combine matching results, and its user interface with a control panel that drives all the matching methods and evaluation strategies.

1. INTRODUCTION

The issue of schema matching in databases [11], which has been investigated since the early 80's, is fundamental to data integration, as is the closely-related issue of ontology alignment or matching [12]. The matching problem consists of defining *mappings* among schema or ontology elements that are semantically related. Such mappings are typically defined between two schemas or two ontologies at a time one being called the *source* and the other being called the *target*.

*This research was supported in part by NSF Awards ITR IIS-0326284, IIS-0513553, and IIS-0812258.

Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the VLDB copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Very Large Data Base Endowment. To copy otherwise, or to republish, to post on servers or to redistribute to lists, requires a fee and/or special permission from the publisher, ACM.
VLDB '09, August 24-28, 2009, Lyon, France
Copyright 2009 VLDB Endowment, ACM 000-0-00000-000-0/00/00.

We have been developing the AgreementMaker[1] matching system, whose name takes after *agreement*, the encoding of a mapping. The capabilities of our system have been driven by the real-world problems of end users who are sophisticated domain experts. We have considered a variety of domains and applications, including: geospatial [2], environmental [4], and biomedical [13]. The conceptual information for these applications is stored in the form of ontologies. However, as demonstrated by others, the same approach can be used for schema matching [1, 10]. To validate our approach, we competed against seven other systems in the biomedical track of the 2007 Ontology Alignment Evaluation Initiative (OAEI), to match ontologies describing the mouse adult anatomy of the Mouse Gene Expression Database Project (2744 classes) and the human anatomy of the National Cancer Institute (3304 classes). We came in third in terms of accuracy (F-measure) [5].

The AgreementMaker, which is currently in its third version, has been evolving to accommodate: (1) user requirements, as expressed by domain experts; (2) a wide range of input (ontology) and output (agreement file) formats; (3) a large choice of matching methods depending on the different granularity of the set of components being matched (local vs. global), on different features considered in the comparison (conceptual vs. structural), on the amount of intervention that they require from users (manual vs. automatic), on usage (stand-alone vs. composed), and on the types of components to consider (schema only or schema and instances); (4) improved performance, that is, accuracy (precision, recall, F-measure) and efficiency (execution time) for the automatic methods; (5) an extensible architecture to incorporate new methods easily and to tune their performance; (6) the capability to evaluate, compare, and combine different strategies and matching results; (7) a comprehensive user interface supporting both advanced visualization techniques and a control panel that drives all the matching methods and evaluation strategies.

In this demo paper, we focus on the most recent developments of the system, which has been almost completely redesigned in the last year. In particular, we describe: (1) the user interface with particular emphasis on the control panel and improved visualization and interaction capabilities; (2) the automatic matching methods and execution capabilities; and (3) the evaluation strategies for determining the efficiency of the matching methods and for performing the combination of results.

[1] http://www.AgreementMaker.org

2. RELATED WORK

There are several notable systems related to ours, including Clio [6], COMA++ [1], Falcon-AO [7], and RiMOM [14] (just to mention a few). Clio stands apart because of its single focus on database-specific constraints and operators (e.g., foreign keys, joins) to infer the mappings whereas constraints in ontologies (as implemented in the other three systems and in AgreementMaker) are of a different nature [12]. This different emphasis also permeates the remaining components of the various systems, as those that also support ontology matching implement a rich tool box of string-similarity and structural-based techniques and focus on performance. Consequently, some of these systems do not focus on user interaction: for example, Falcon-AO and RiMOM provide simple interfaces that offer limited user interaction (e.g., no manual manipulation of the ontologies). However, what separates AgreementMaker from these other systems (including from COMA++, which has a more sophisticated user interface than the other two) is the degree to which it integrates the evaluation of the quality of the obtained mappings with the graphical user interface and therefore with the iterative matching process. This tight integration emerged from our work with domain experts, who required that the evaluation be an integral part of the matching process, not an "add on" capability.

3. ARCHITECTURE

The AgreementMaker supports a wide variety of methods or *matchers*. Our architecture (see Figure 1) allows for serial and parallel composition where, respectively, the output of one or more methods can be used as input to another one, or several methods can be used on the same input and then combined. A set of mappings may therefore be the result of a sequence of steps, called *layers*.

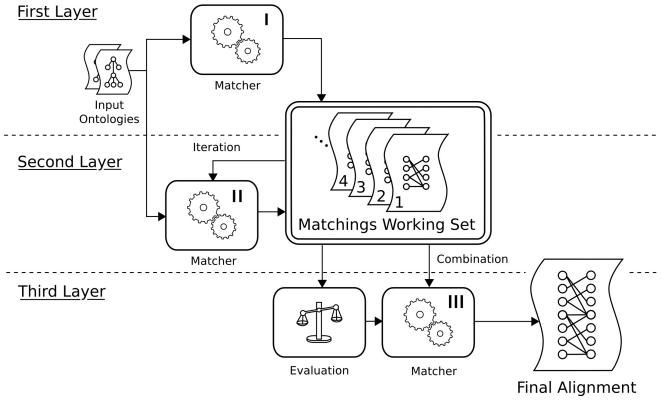

Figure 1: System architecture.

The matching process of a generic matcher (see Figure 2), can be divided into two main modules: (1) *similarity computation* in which each concept of the source ontology is compared with all the concepts of the target ontology, thus producing two similarity matrices (one for classes and the other one for properties), which contain a value for each pair of concepts; (2) *mappings selection* in which the matrix is scanned to select only the best mappings according to a given threshold and to the cardinality of the correspondences, for example, 1-1, 1-N, N-1, M-N.

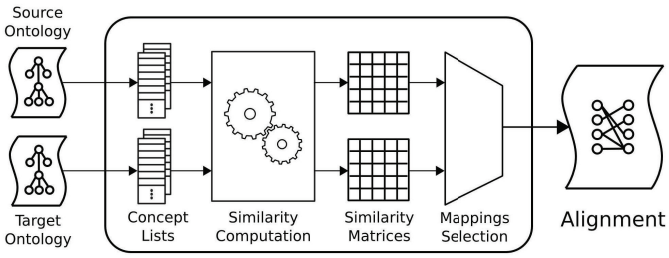

Figure 2: Structure of a generic matcher.

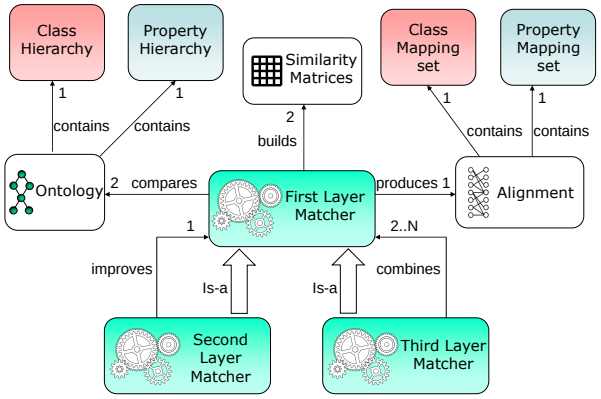

Figure 3: Matchers' schema.

To enable extensibility, we adopted the object-oriented template pattern by defining the skeleton of the matching process in a generic matcher, which defers only a few operations to the concrete matcher extensions (see Figure 3). This abstraction minimizes development effort by completely decoupling the structure of a single method from the architecture of the whole system, thus allowing reuse or any possible composition of matching modules.

A first layer matcher produces the similarity matrices, while the second and third layer matchers extend the first layer matchers. In particular, a second layer matcher improves on the results of a first layer matcher using conceptual or structural information, depending on whether it considers one concept alone or a concept and its neighbors. Finally, a third layer matcher combines the results of two or more matchers from the previous layers, in order to obtain a final *matching* or *alignment*, that is, a set of mappings.

4. USER INTERFACE

The source and target ontologies (in XML, RDFS, OWL, or N3) are visualized side by side using the familiar outline tree paradigm (see Figure 4). Agreements can be exported in different formats (e.g., XML, Excel). Because all the matching operations and their results are managed by this interface, we gave special consideration to its design [4]. We describe next two new features of the interface: the control panel and the visualization of non-hierarchical ontologies (e.g., due to multiple inheritance in OWL). The latter feature allows for specific subtrees to be visually duplicated. Because we adopt the Model-View-Control pattern, this duplication does not affect the underlying data structures.

The control panel (see Figure 5) allows users to run and manage matching methods and their results. Users can se-

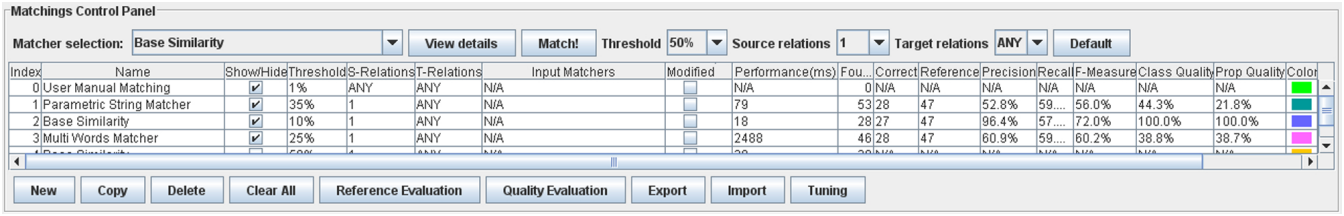

Figure 5: Control panel.

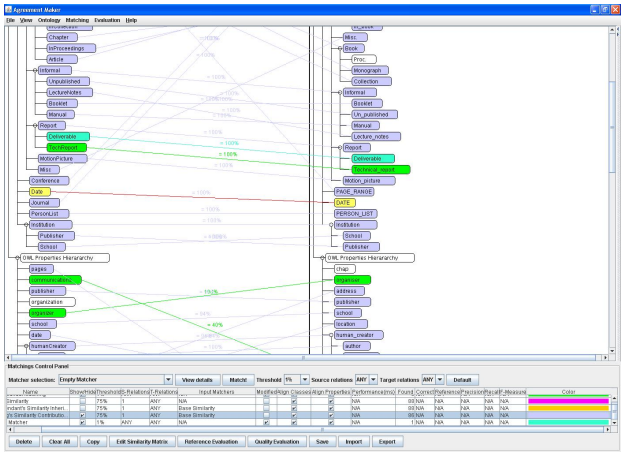

Figure 4: Graphical User Interface.

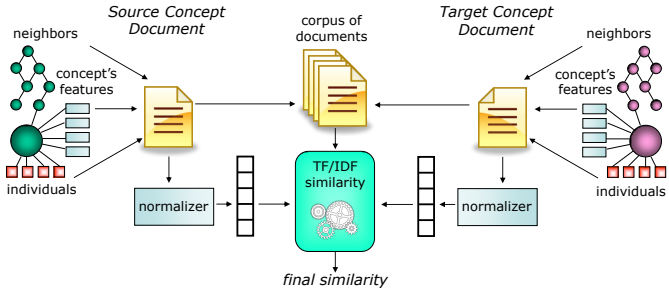

Figure 6: Concept document-based matcher.

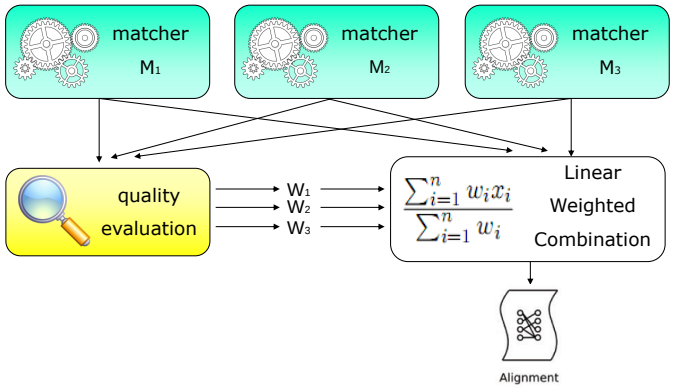

Figure 7: Linear Weighted Combination matcher.

lect parameters common to all methods (such as threshold and cardinality) and method-specific parameters. When a method has run, a new row is dynamically added to the table that is part of the control panel at the same time that lines depicting the mappings between the concepts are added (see Figure 4). Each row is color coded and allows for its selection so that the corresponding mappings (of the same color) can be compared visually. Each row also displays the performance values for the associated methods, thus allowing for the comparison with those of other rows. In addition, users can modify at runtime the method parameters by changing directly their values in the table or by selecting previously calculated matchings as input to the methods to be applied next. Multiple matchings can also be combined manually or with an automatic combination matcher.

5. MATCHING METHODS

First layer matchers compare concept features (e.g., label, comments, annotations, and instances) and use a variety of methods including syntactic and lexical comparison algorithms as well as the use of a lexicon like WordNet. Of those methods some were proposed by others (e.g., edit distance, Jaro-Winkler) and some devised by us, including a substring-based comparison that favors the length of the common substrings and a concept document-based comparison containing a wide range of features. Those features are represented as TF-IDF vectors and use a cosine similarity metric (see Figure 6).

Second layer matchers use structural properties of the ontologies. Our own methods include the Descendant's Similarity Inheritance (DSI) and the Sibling's Similarity Contribution (SSC) matchers [3].

Finally, third layer matchers combine the results of two or more matchers so as to obtain a unique final matching in two steps. In the first step, a similarity matrix is built for each pair of concepts, using our Linear Weighted Combination (LWC) matcher, which processes the weighted average for the different similarity results (see Figure 7). Weights can be assigned manually or automatically, the latter assignment being determined using our evaluation methods. The second step uses that similarity matrix and takes into account a threshold value and the desired cardinality. When the cardinality is 1-1, we adopt the Shortest Augmenting Path algorithm [9] to find the optimal solution for this optimization problem (namely the assignment problem reduced to the maximum weight matching in a bipartite graph) in polynomial time.

6. EVALUATION

The design of optimal methods to find correct and complete mappings between real-world ontologies is a hard task for several reasons. First of all, an algorithm may be effective for a given scenario, but not for others. Even within the same scenario, the use of different parameters can change significantly the outcome. Moreover, in interviewing domain

experts in the geospatial domain, we discovered that they do not trust automatic methods unless quality metrics are associated with the matching results. These observations have motivated a variety of evaluation techniques, that determine runtime and accuracy (precision, recall, and F-measure).

The most effective evaluation technique compares the mappings found by the system between the two ontologies with a reference matching or "gold standard," which is a set of correct and complete mappings as built by domain experts. When a reference matching is available, the AgreementMaker can determine the quality of the found matching analytically or visually. A reference matching can also be used to tune algorithms by using a feedback mechanism provided by a succession of runs.

When a gold standard is not available, "inherent" quality measures need to be considered. Quality measures can be defined at two levels as associated with the two main modules of a matcher (see Figure 2): *similarity* or *selection* level. We can consider *local* quality as associated with a correspondence at the *similarity* level (or mapping at the *selection* level) or *global* quality as associated with all the correspondences at the *similarity* level (or with all possible mappings at the *selection* level). We have incorporated in our system a *global-selection* quality measure proposed by others [8] and a *local-similarity* quality measure that we have devised. Experiments have shown that our quality measure is usually effective in defining weights for the LWC matcher.

7. DEMONSTRATION

Our demo focuses on the matching methods and evaluation strategies for determining the efficiency of ontology matching methods. Due to the tight integration of the evaluation strategies with the graphical user interface, a unique feature of our system, all the steps will be performed through the interface. Users will start by uploading their own ontologies, load our own, or download ontologies from the web, thus taking advantage of the several standard formats supported. Users can then explore the interface freely or follow a walk-through, consisting of browsing the ontologies, expanding and contracting nodes, and customizing the display. They have access to the information associated with each concept to be aligned, including descriptions, annotations, and (context) relations, and they can use them to visually detect mappings.

The matching methods offer a brief description to help users in their selection. Users can manually set parameters like threshold or use default values. Once the mappings are defined, they will be able to interact with the results obtained, for example, to reduce the number of lines depicting the mappings. They can also edit the mappings that are automatically produced by the system, by adding, deleting, and updating mappings. Several matchings can be viewed and managed in parallel to directly detect overlaps and differences visually. Tuning the performance of the algorithms becomes an easy and quick task: for example, mappings can be iteratively improved by running second layer matchers, such as DSI after one of the base similarity methods. In the end, a final set of mappings can be obtained using basic combination operations or an automatic quality composition of previous matchings. Users will also be able to store and inspect the mappings thus defined. The system also provides a powerful framework for developers. Each matching method is implemented as a standard component, which allows reuse and composition. A standard extensible API is defined to let developers plug and test new matching methods thus minimizing effort and maximizing reuse.

8. REFERENCES

[1] D. Aumueller, H. H. Do, S. Massmann, and E. Rahm. Schema and Ontology Matching with COMA++. In *ACM SIGMOD International Conference on Management of Data*, pages 906–908, 2005.

[2] I. F. Cruz, A. Rajendran, W. Sunna, and N. Wiegand. Handling Semantic Heterogeneities using Declarative Agreements. In *International ACM GIS Symposium*, pages 168–174, 2002.

[3] I. F. Cruz and W. Sunna. Structural Alignment Methods with Applications to Geospatial Ontologies. *Transactions in GIS, Special Issue on Semantic Similarity Measurement and Geospatial Applications*, 12(6):683–711, December 2008.

[4] I. F. Cruz, W. Sunna, N. Makar, and S. Bathala. A Visual Tool for Ontology Alignment to Enable Geospatial Interoperability. *Journal of Visual Languages and Computing*, 18(3):230–254, 2007.

[5] J. Euzenat, A. Isaac, C. Meilicke, P. Shvaiko, H. Stuckenschmidt, O. Šváb, V. Svátek, W. R. van Hage, and M. Yatskevich. Results of the Ontology Evaluation Initiative 2007. In *ISWC International Workshop on Ontology Matching*, volume 304, pages 96–132. CEUR-WS, 2007.

[6] M. A. Hernández, R. J. Miller, and L. M. Haas. Clio: A Semi-Automatic Tool For Schema Mapping (demo). In *ACM SIGMOD International Conference on Management of Data*, page 607, 2001.

[7] N. Jian, W. Hu, G. Cheng, and Y. Qu. Falcon-AO: Aligning Ontologies with Falcon. In *K-CAP 2005 Workshop on Integrating Ontologies*. CEUR Workshop Proceedings 156, 2005.

[8] C. Joslyn, A. Donaldson, and P. Paulson. Evaluating the Structural Quality of Semantic Hierarchy Alignments. In *International Semantic Web Conference (Posters & Demos)*, 2008.

[9] R. M. Karp. An Algorithm to Solve the $m \times n$ Assignment Problem in Expected Time $O(mn \log n)$. *Networks*, 10(2):143–152, 1980.

[10] S. Melnik, H. Garcia-Molina, and E. Rahm. Similarity Flooding: A Versatile Graph Matching Algorithm and its Application to Schema Matching. In *IEEE International Conference on Data Engineering (ICDE)*, pages 117–128, 2002.

[11] E. Rahm and P. A. Bernstein. A Survey of Approaches to Automatic Schema Matching. *VLDB Journal*, 10(4):334–350, 2001.

[12] P. Shvaiko and J. Euzenat. A Survey of Schema-Based Matching Approaches. In *Journal on Data Semantics IV*, volume 3730 of *Lecture Notes in Computer Science*, pages 146–171. Springer, 2005.

[13] W. Sunna and I. F. Cruz. Using the AgreementMaker to Align Ontologies for the OAEI Campaign 2007. In *ISWC International Workshop on Ontology Matching*, volume 304, pages 133–138. CEUR-WS, 2007.

[14] J. Tang, J. Li, B. Liang, X. Huang, Y. Li, and K. Wang. Using Bayesian Decision for Ontology Mapping. *Journal of Web Semantics*, 4(4):243–262, 2006.