XBenchMatch: a Benchmark for XML Schema Matching Tools*

Fabien Duchateau
LIRMM
Université Montpellier 2
34000 Montpellier - France
duchatea@lirmm.fr

Zohra Bellahsène
LIRMM
Université Montpellier 2
34000 Montpellier - France
bella@lirmm.fr

Ela Hunt
Department of Computer Science
ETH Zurich, CH-8092
hunt@inf.ethz.ch

ABSTRACT

We present XBenchMatch, a benchmark which uses as input the result of a schema matching algorithm (set of mappings and/or an integrated schema) and generates statistics about the quality of this input and the performance of the matching tool.

1. INTRODUCTION

Over the years, several approaches to schema matching [1, 3, 5, 7, 8, 10] have been proposed, demonstrating their benefit in different scenarios, and many matching systems have been designed. Most of the papers describing a schema matching tool provide an experimental section. However, in most cases, these experiments reflect a particular scenario, using real-world schemas. We know that a matching tool can provide an acceptable matching quality with good performance in a specific scenario, but it can be unreliable and slow in another case. Thus, it is difficult to compare two schema matching tools, and to identify the one which performs best. XBenchMatch addresses the requirement of finding out if a schema matching tool is appropriate in a given context.

To the best of our knowledge, there is no comprehensive benchmark of schema matching tools. In [2] an evaluation of schema matching tools is presented and the main matching criteria are discussed. A summary of the capabilities of each matching tool is then provided. However, as the authors explain, it is quite difficult to evaluate the matching tools for several reasons. First, the tools are not always available as a demo and it is not possible to test them against specific sets of schemas. Second, some tools require specific resources to be efficient, like an ontology or a thesaurus, which are not always available. Finally, some matching tools take as input specific additional files, for example Rondo. The above evaluation suffers from two drawbacks. First, by evaluating the matching tools with the scenarios provided in the papers, one cannot objectively judge the capabilities of each matching tool. Secondly, some matching tools generate an integrated schema instead of a set of mappings, and the measures used to evaluate a set of mappings are insufficient to evaluate the quality of an integrated schema. Our work extends the criteria provided in [2] by adding new scoring functions which evaluate the quality of integrated schemas and extend the evaluation methodology. Indeed, in XBenchMatch all the matching tools are evaluated against the same scenario, which produces an improved objective comparison.

A new method for the evaluation of schema matching tools was proposed in [10]. This extends [2] by timing tool performance, and uses real-world schemas. However, the input is limited to a set of mappings, while some matchers provide a more interesting output by building an integrated schema. Moreover, the evaluation system is not in the public domain, and cannot be used as a benchmark.

We present XBenchMatch, a benchmark involving a set of criteria for testing and evaluating XML schema matching tools. We focus on the assessment of the matching tools in terms of matching quality and performance. We also provide a testbed involving a large schema corpus that can be used by everyone to quickly benchmark their new schema matching algorithms. Contrary to [10], our work takes into account the integrated schemas generated by matching tools and evaluates them with new scoring measures.

The rest of the paper is organised as follows. In Section 2 we present the properties of our schema matching benchmark. Section 3 describes the evaluation criteria and the scoring functions. Section 4 contains an overview of our prototype and Section 5 contains the demonstration scenario. In Section 6 we conclude.

2. PROPERTIES OF XBENCHMATCH

A benchmark needs to be:

- **extensible**, as it will evolve over time in order to allow future schema matching tools to be benchmarked. New functionalities or improvements can be added without significant reorganisation of the benchmark. For example, new scoring functions can be added to evaluate new quality criteria.
- **portable**, that is OS-independent. This requirement is fulfilled by using Java.
- **simple**, as both end-users and schema matching experts are targeted by the benchmark.

*Supported by ANR Research Grant ANR-05-MMSA-0007 and by an EU Marie-Curie Fellowship to EH

VLDB '07, September 23-28, 2007, Vienna, Austria.

- **scalable** in two ways: the creation of new scenarios should be possible and not require a lot of work; the composition of a benchmark consisting of several scenarios should be possible.

- **generic**, as it should work with most of the available matchers. It should allow one to restrict the matching criteria to include various types of matchers, based on their shared capabilities, i.e. their least common denominator. It should be able to compare schema matching tools producing both an integrated schema and a set of mappings, and those that only produce one of them.

- **complete**, as it should provide the complete set of measures which can be derived in each case.

3. QUALITY MEASURES

The aim of automated schema matching is to avoid a manual, laborious task which often leads to errors. To measure the quality of matching we use a number of scoring functions. Those are complemented by the performance evaluation which reports matching execution time. Our benchmark provides system level statistics, like resource consumption (disk space and memory usage), as well as statistics of the schemas, including the dimensions of the integrated schema (depth, no of children, no of nodes).

3.1 The Quality of Mappings

We assume that mappings are given as XML path correspondences, for instance *(Schema1.path1, Schema2.path2)*. To express $1:n$ and $n:1$ mappings, one concatenates the paths, for instance *(Schema1.path1, Schema2.path2;path4)*. Perfect mappings are provided by an expert. Let mappings provided by the expert be T_{ex} and mappings provided by a matcher be T_{map}.

Precision expresses the proportion of correct mappings among the mappings produced by the mapping tool [9].

$$Precision = \frac{|T_{map} \cap T_{ex}|}{|T_{map}|}$$

A 100% precision means that all the mappings extracted by the system are correct.

Recall shows the proportion of correct mappings extracted by the system, as a fraction of the expert mappings. A 100% recall means that all relevant mappings have been found.

$$Recall = \frac{|T_{map} \cap T_{ex}|}{|T_{ex}|}$$

A compromise between recall and precision is provided by the F-measure.

$$F{-}measure = \frac{2 \cdot precision \cdot recall}{precision + recall}$$

The **overall accuracy** measure [6] evaluates the post match effort, that is the amount of work needed to add the relevant mappings that have not been discovered (false negatives) and to remove those which are incorrect but have been extracted by the matcher (false positives). The measure can have negative values [2], and the F-measure is more optimistic than *overall accuracy*.

3.2 The Quality of the Integrated Schema

A matching tool may produce two outputs: a set of mappings and an integrated schema. For an integrated schema S_i, and any of the input schemas S_g, our benchmark evaluates the *structural integrity* of the integrated schema. We implemented a number of measures of structural integrity.

Backbone measure, BM corresponds to the size of the largest common subtree of S_g and S_i, seen against the background of the integrated schema S_i. We count tree nodes in the shared subtree $LCSub$ and in the merged schema S_i. BM returns a value between 0 (no common subtree) and 1 (both trees are the same).

$$BM = \frac{\mid LCSub(S_i, S_g) \mid}{\mid S_i \mid} \quad (1)$$

We define Sub as the set of all disjoint subtrees (each containing a minimum of two nodes) which are common to S_i and S_g. $kSub$ is the total number of elements of all subtrees in Sub.

Structural overlap corresponds to the number of nodes shared by both S_i and S_g and included in a common subtree. A 0 stands for no common subtrees while a value closer to 1 shows that most of the elements are included in a common subtree. The formula for structural overlap is:

$$StructuralOverlap = \frac{kSub}{|S_i|} \quad (2)$$

Another measure we propose is **structural proximity**. This extends the structural overlap. While the structural overlap only measures the percentage of elements in the common subtrees, structural proximity also considers the number of common subtrees. If S_i and S_g are highly similar, they have only one large common subtree. The higher the number of common subtrees, the less similar the trees are. Further, we consider the number of missing elements, i.e the elements in S_i that are not in one of the common subtrees. As S_i is the ideal schema, all of its nodes which are missing in the common subtrees affect the structural proximity between the two trees. First, we define o as the number of elements in S_i that are not included in any common subtree. $o = \mid S_i \mid - kSub$. Tree proximity is then obtained by:

$$StructuralProximity = \frac{kSub}{|S_i| \times \sqrt{|Sub| + o}} \quad (3)$$

This formula generates a value between 0 and 1, 0 meaning the trees are totally different, and 1 when the trees are identical.

4. IMPLEMENTATION

XBenchMatch was implemented in Java. It delivers the measures described in 3 and should help one choose among the available schema matching tools the one that best satisfies the needs. XBenchMatch takes as input two files:

- the output generated by a matching tool that needs to be benchmarked.

- an *oracle*, that is a set of perfect matchings for the same corpus.

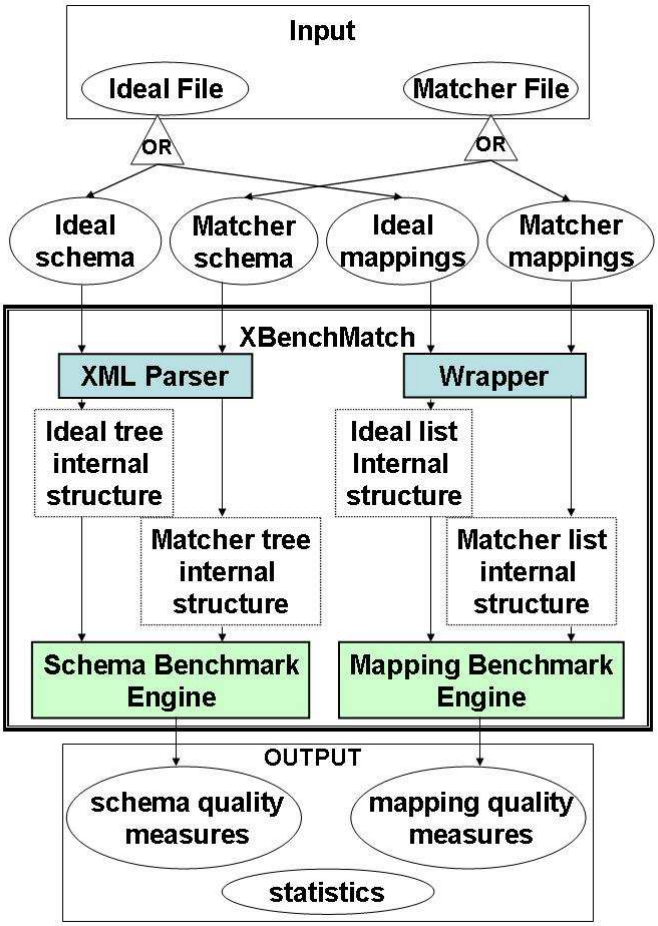

Figure 1: Architecture of XBenchMatch

Those files may be of two types, either an integrated schema, or a set of mappings, and the file generated by the matching tool must be of the same type as the oracle. The user may also generate statistics for the schema corpus that has been used by the matcher, such as the average number of nodes, or the maximum depth. The static information, i.e. the features of the matching tool, helps the user to understand the results of the matching. The analysis of the matching results is given by the dynamic criteria, or the measures: *precision*, *recall*, *F-measure* and *overall accuracy*, which generate plots. If input consists of integrated schemas, more measures like *structural overlap* and *structural proximity* are also computed.

Figure 1 shows the architecture. As the input file is either an integrated schema or a set of mappings, two modules are in charge of converting them into an internal structure, the `XML Parser` and the `Wrapper`. Creating new wrappers ensures extensibility by supporting new sets of mappings formats. Then the benchmark engines compute different measures comparing two sets of mappings or two integrated schemas. XBenchMatch then outputs statistics (performance, schema summaries) and the quality measures described in 3. Schema matching systems can be compared in one or more scenarios, especially by comparing their F-measure and structural proximity.

5. DEMONSTRATION SCENARIO

We demonstrate four real-world XML matching tasks. The first set of schemas describes a person. Person schemas are small and strongly heterogeneous. The second set concerns purchase orders and demonstrates matching of a large schema to a smaller one. Schemas come from the XCBL[1]. The third scenario covers university courses from Thalia [4]. Finally, the last demo is from biology. First schema represents Uniprot[2], while the second one, GeneCards [3], selects data from over 100 databases. These scenarios support the comparison of matching tools when facing specific domains. Detailed features of each scenario are shown in Table 1. All the perfectly integrated schemas have been built and verified by an expert and are provided with our benchmark. Before using XBenchMatch, the user generates an integrated schema and a set of mappings for each scenario with the matching tools she would like to evaluate.

	Person	University	Order	Biology
NB nodes (S_1/S_2)	11/10	18/18	20/844	719/80
Avg NB of nodes	11	18	432	400
Max depth (S_1/S_2)	4/4	5/3	3/3	7/3
NB of Mappings	5	15	10	57

Table 1: Summary of four evaluation scenarios.

The following schema matching tools were tested: Porsche, COMA++, and Similarity Flooding. Our benchmark could be easily extended to include other matchers.

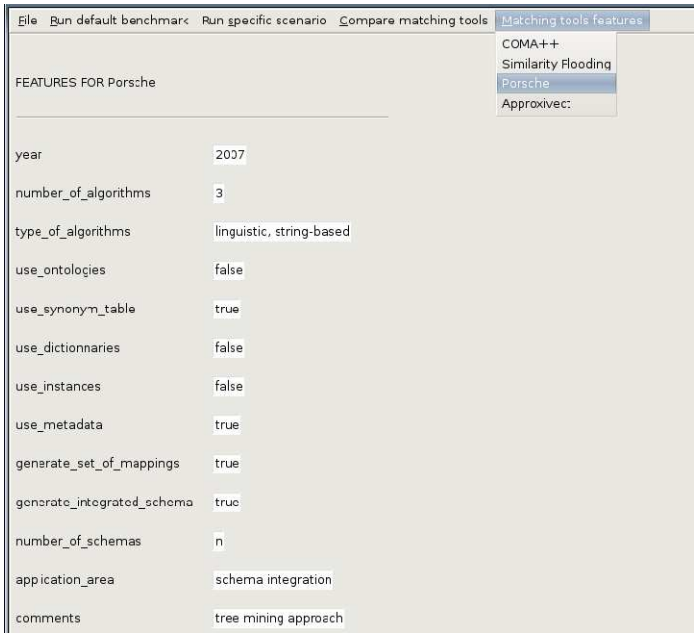

Figure 2: Main Menu of XBenchMatch

Figure 2 shows the interface. First, the user can run the **default benchmark** which is composed of the four scenarios described above. The integrated schema for the person

[1]www.xcbl.org
[2]http://www.ebi.uniprot.org/support/docs/uniprot.xsd
[3]http://www.genecards.org/GeneCardByFunction.xsd

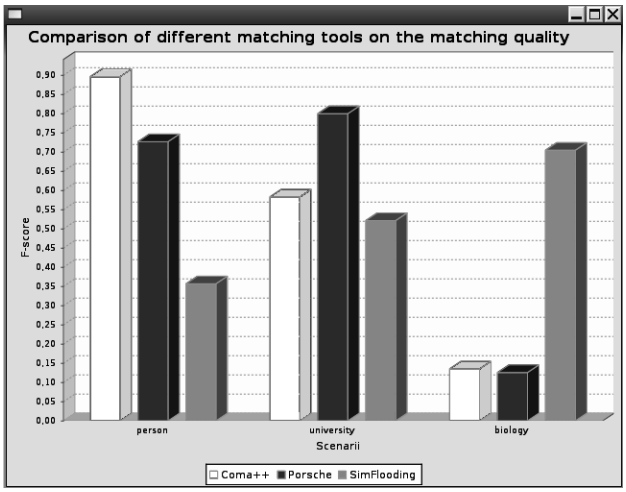

Figure 3: A comparison of three scenarios and matchers.

scenario must be chosen using a file dialog box, and this process is then repeated for the business, university and biology scenarios. XBenchMatch compares the matching quality of these integrated schemas to the ideal integrated schemas. It outputs the following measures: precision, recall, F-score, overall accuracy, structural overlap, and structural proximity. A plot is automatically drawn to show the quality according to the number of common elements in the two trees. Another plot shows structural overlap and proximity measures. Matching performance and schema summaries are also displayed.

XBenchMatch is generic and extensible, and to run it with other scenarios, one uses the **Run Specific Scenario** option. The process is identical to the default benchmark, except that the user needs to choose, for a specific scenario, both the oracle schema and the integrated schema that is to be tested. Then the measures showing the quality comparison of two schemas are displayed.

Finally, XBenchMatch enables one to compare the quality of several matching tools, on one or more scenarios, by using the **Compare Matching Tools** option. The user enters a scenario name, and chooses the ideal integrated schema. Next, the user decides which matchers to compare. It is possible to add as many matchers as desired, by entering their name and choosing the integrated schema. When all matchers have been chosen, the user can add another scenario. This process of adding schemas and scenarios ends when the user has selected all matchers for the different scenarios she wanted. Then, XBenchMatch outputs two bar charts: the first one shows the F-measure for each matcher for the selected scenarios. The second plot outputs the structural proximity of the the integrated schema produced by the matcher and the ideal one. Figure 3 shows a comparison of the F-measure obtained by COMA++, PORSCHE and Similarity Flooding for 3 scenarios.

6. CONCLUSIONS

We designed and implemented a new benchmark for schema matching. XBenchMatch produces an improved objective comparison, since the matching tools are evaluated against the same scenarios. Our benchmark provides two kinds of evaluation: (i) matching quality, based on the use of quality measures, comparing mappings or integrated schemas, and (ii) matching performance. The first criterion is very important in automatic schema matching and the second is crucial in large scale scenarios. A demo version of the prototype is available at *http://www.lirmm.fr/~duchatea/XBenchMatch*.

7. ACKNOWLEDGMENTS

The authors would like to thank the researchers who made their schema matching tools available.

8. REFERENCES

[1] P. A. Bernstein et al. Industrial-strength schema matching. *SIGMOD Record*, 33(4):38–43, 2004.

[2] H. H. Do et al. Comparison of schema matching evaluations. In *Revised Papers from NODe 2002*, pages 221–237. Springer, 2003.

[3] H. H. Do and E. Rahm. Coma - a system for flexible combination of schema matching approaches. In *VLDB*, 2002.

[4] J. Hammer et al. Thalia: Test harness for the assessment of legacy information integration approaches. In *ICDE*, 2005.

[5] J. Madhavan et al. Generic Schema Matching with Cupid. In *VLDB*, pages 49–58, 2001.

[6] S. Melnik et al. Similarity flooding: A versatile graph matching algorithm and its application to schema matching. In *ICDE*, 2002.

[7] E. Rahm and P. A. Bernstein. A survey of approaches to automatic schema matching. *VLDB J.*, 10(4):334–350, 2001.

[8] P. Shvaiko and J. Euzenat. A survey of schema-based matching approaches. *J. Data Semantics*, 4:146–171, 2005.

[9] C. Van-Risbergen. *Information Retrieval*. 2nd edition, London, Butterworths, 1979.

[10] M. Yatskevich. Preliminary evaluation of schema matching systems. Technical Report DIT-03-028, Informatica e Telecomunicazioni, University of Trento, 2003.