

Designing a Benchmark for the Assessment of XML Schema Matching Tools*

Fabien Duchateau
LIRMM - UMR 5506
Université Montpellier 2
34392 Montpellier Cedex 5 - France
duchatea@lirmm.fr

Zohra Bellahsène
LIRMM - UMR 5506
Université Montpellier 2
34392 Montpellier Cedex 5 - France
bella@lirmm.fr

ABSTRACT

Over the years, many XML schema matching systems have been developed. A benchmark for assessing the capabilities of schema matching systems and providing uniform conditions and the same testbed for all schema matching prototypes, has become indispensable as the matching systems grow in complexity. However, developing a benchmark for the schema matching problem is very challenging, given the wide range of techniques that can be applied to assist in schema matching. In this paper, we present the foundations and desiderata of a benchmark for XML schema matching. Moreover, we have extended the notion of quality of an integrated schema by proposing new scoring functions. Finally, we have designed and implemented XBenchMatch, an application which takes as input: an ideal schema and the result of a matching from a schema matching prototype (i.e. a set of mappings and/or an integrated schema) and generates as output: statistics on the quality of this input. Our proposal is aimed to provide two kinds of evaluations: (i) quality matching evaluation, which is based on the use of the quality measures and (ii) performance of matching schema. The first criteria is very important in automatic schema matching and the second is crucial in large scale when the schema to be matched are very large. In this paper, we present XBenchMatch, a benchmark for testing and assessing schema matching tools and report the experiments results of some matching tools over a large corpus of schemas using our benchmark.

1. INTRODUCTION

Over the years, several approaches of schema matching [6, 9, 14, 18, 22, 25, 28] 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 experiment section. However, these experiments reflect a particular scenario, us-

*Supported by ANR Research Grant ANR-05-MMSA-0007

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 '07, September 23-28, 2007, Vienna, Austria.
Copyright 2007 VLDB Endowment, ACM 978-1-59593-649-3/07/09.

ing real-world schemas. For example, 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 seems difficult to compare two schema matching tools, and to evaluate the one which performs best. And end-users might not know which one is the most appropriate for their task.

To the best of our knowledge, there is no complete benchmark for schema matching tools. In [8], the authors present an evaluation of schema matching tools. This evaluation suffers from two drawbacks. First, by evaluating the matching tools with the scenarios provided in their respective 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 provided to evaluate a set of mappings appear not sufficient to evaluate the quality of an integrated schema.

Another proposal for evaluating schema matching tools has been done in [28]. It extends [8] by adding time measures and relies on real-world schemas to evaluate the matching tools. However, the evaluation system has not been implemented. Our work extends the criteria provided in [8], by adding scoring functions to evaluate the quality of integrated schemas. It goes further on the evaluation aspect. Indeed all the matching tools are evaluated against the same scenario. In this paper, we present the foundation of a benchmark for XML schema matching tools. Our evaluation system involves a set of criteria for testing and evaluating schema matching tools. It is aimed to provide uniform conditions and the same testbed for all schema matching prototypes. Our approach focuses on the evaluation of the matching tools in terms of matching quality and performance. Next, we also aim at giving an overview of a matching tool by analysing its features and deducing some tasks it might fulfill. This should help an end-user to choose among the available matching tools depending on the criteria required to perform his task. Finally, we provide a testbed involving a large schema corpus described in 7 that can be used by everyone to quickly benchmark matching algorithms. Here we outline the main contributions of our work:

- We describe the notion of benchmark for the schema matching application. More precisely we list the different features involved in this process, and we give a methodology on how to evaluate them and to choose the most appropriate for a defined task.
- We have extended the notion of quality for a schema, by proposing new measures like structural overlap.

- We have designed XBenchMatch, an application which takes as input an ideal schema the result of a matching from a schema matching system (i.e. a set of mappings and/or an integrated schema). It generates statistics on the quality of this input, based on the criteria defined above.

The rest of the paper is organised as follows: first we give some definitions and preliminaries in Section 2. In Section 3, the list of criteria is explained. In Section 4, we present the main features of schema matching tools. In Section 5 the scoring functions of quality are described. Section 6 briefly presents our XBenchMatch application and the results of our experiments. Section 9 contains the related work; and in Section 10, we conclude and outline some future work.

2. PRELIMINARIES

In this section, we define the main notions used in this paper.

Definition 1 (Schema) : A schema is labeled unordered tree $S = (V_S, E_S, r_S, label)$ where V_S is a set of nodes; r_S is the root node; $G_S \subseteq V_S \times V_S$ is a set of edges; and $label E_S \rightarrow \Lambda$ where Λ is a countable set of labels.

Definition 2 (Semantic Similarity Measure): Let E_1 be a set of elements of schema 1, and E_2 be a set of elements of schema 2. A semantic similarity measure between two elements $e_1 \in E_1$ and $e_2 \in E_2$, noted as $S_m(e_1, e_2)$, is a metric value based on the likeness of their meaning/ semantic content, given as:

$$S_m : E_1 \times E_2 \rightarrow [0, 1]$$

$(e_1, e_2) \rightarrow S_m(e_1, e_2)$ where a zero means a total dissimilarity and 1 value stands for total similarity.

Definition 3 (Automatic Schema Matching)

Given two schema elements sets E_1 and E_2 and a similarity measure threshold t . We define Automatic Schema Matching, between two elements e_1 and e_2 , noted as $match(e_1, e_2)$, as follows:

For all $(e_1, e_2) \in E_1 \times E_2$

If $S_m(e_1, e_2) < t$ then $match(e_1, e_2) = \text{false}$

Else If $S_m(e_1, e_2) \geq t$ then $match(e_1, e_2) = \text{true}$;

$d = S_m(e_1, e_2)$ where d is the similarity degree

Threshold t may be adjusted by an expert, depending upon the strategy, domain or algorithms used by the match tools.

Example 2.1: If $match(\text{adresse}, \text{address})$ is calculated using edit distance algorithm¹, the value of d is 0.857 and if 3-gram² algorithm is used the result for d is 0.333. For another example $match(\text{dept}, \text{department})$; edit distance value of d is 0 and 3-gram result is 0.111. The examples show that the threshold has to be adjusted by an expert depending upon the properties of strings being compared and the match algorithms being applied.

¹ $match(s_1, s_2) = \max\{0, \frac{\min\{|s_1|, |s_2|\} - EditDistance(s_1, s_2)}{\min\{|s_1|, |s_2|\}}\}$

² $match(s_1, s_2) = \frac{1}{1 + |ngram(s_1)| + |ngram(s_2)| - 2 \times |ngram(s_1) \cap ngram(s_2)|}$

Definition 4 (Best Match selection): There can be the possibility of more than one match for an element $e_1 \in E_1$ in E_2 . In such situation the match with maximum similarity degree has to be selected. This case can be formally defined as:

Given $E_{i2} \subseteq E_2$ of size n , such that $\forall e_{ij}$ corresponding to element e_i , $match(e_i, e_{ij})$ is true; where $1 \leq j \leq n$. Best match for element e_i of E_1 noted as $match_{ib}$ is given as following:

$$match_{ib} = \max_{j=1}^n S_m(e_i, e_{ij})$$

Definition 5 (Schema Mapping): Given E_1 a set of elements of schema 1, E_2 a set of elements of schema 2 and I a set of mappings identifiers. We define a mapping between two elements $e_1 \in E_1$ and $e_2 \in E_2$ by the following function noted as Map:

$$\text{Map: } I \times E_1 \times E_2 \times Fs \rightarrow I \times E_1 \times E_2 \times [0, 1] \times K \\ (id, e_1, e_2, S_m) \rightarrow (id, e_1, e_2, d, k)$$

where Fs is a set of functions performing similarity measure, d is the similarity degree returned by $match(e_1, e_2)$ and K is the set of mapping expressions e.g. equivalence, synonym, inclusion etc., depending upon the data model being represented by schemas 1 and 2.

Schema mapping can be uni-directional i.e., from schema 1 toward schema 2, or bidirectional i.e., the correspondence holds in both directions e.g. if an element e_1 from schema 1 is mapped to an element e_2 of schema 2 then there exists another correspondence for element e_2 of schema 2 toward element e_1 of schema 1 [1].

3. DESIDERATA

The schema matching benchmark needs to have the following properties in order to be complete and efficient. It needs to be:

- **Extensible**, the benchmark is able to evolve according to progress. Thus, future schema matching tools could be benchmarked, as well as new measures can be added to evaluate the matching quality. The benchmark deals with well-formed XML schemas, and a set of mappings can easily be converted into the default set of mappings formats using a wrapper. Thus, the outputs of future matching tools should be handled. For the new measures, we intend to release the benchmark in open-source, allowing everyone to add new measures or functionalities.
- **Portable**. The benchmark should be OS-independent, since the matching tools might run on different OS. This requirement is fulfilled by using Java.
- **Simple** since both end-users and schema matching experts are targeted by this benchmark.
- **Scalable** on two aspects: creating new benchmark scenarii is an easy task. And a benchmark composed of many scenarii should be easy to construct and evaluate.
- **Generic**, it should work with most of the matchers available. Thus, the criteria have been restricted to

the average capabilities of the matchers. For example, some schema matching tools are able to match a large number of schemas at a time, but some others do not. This involves the number of schemas to be limited to 2. Another example is: some schema matching tools may provide as output both an integrated schema and a set of mappings while some others only provide a single output.

All these requirements should be met to provide an acceptable matching benchmark. Next we focus on the criteria dedicated to the schema matching process itself.

4. MATCHING TOOLS FEATURES

Some schema tools have enhanced the match task, namely in automatic schema matching, with pre-match and post-match phases.

This section covers the general features which define the characteristics and the capabilities of the matching tools. This section is organized in four parts describing these features as follows: (i) the pre-match phase, (ii) the matching method, (iii) the output of the schema matching tool and (iv) the post-match phase.

4.1 Pre-Match Phase

This phase normally includes configuration of various parameters e.g. setting weights, thresholds of the matching algorithms etc. It can have three possibilities :

- **External resources.** They make use of some external resources, like ontologies (domain specific), thesauri or dictionaries (for example Wordnet) [13].
- **Tuning.** A matching tool might be flexible by allowing some parameters or thresholds (example 2.1) to be tuned by the user [12]. This step may be optional or compulsory, but these parameters generally affect other criteria. For instance, they can be varied to enable better performance by degrading the quality.
- **Training** Some approaches provide a new set of machine learning based matchers for specific types of complex matchings. For example, LSD [10] uses machine learning algorithms for matching as well as in summing up the match results for each pair of attribute comparison.

This pre-matching step involves more work at the beginning. However, this effort is often rewarded since it positively affects the matching quality. In our benchmark, the pre-match appears as a list of pre-processing tasks of the matching tool, performed at this phase. For example, use of dictionaries, use of ontologies, use of synonyms table, etc.

4.2 Matching Method

Schema matching is a complex problem, which starts by discovering similarities between schema elements' names, mainly by using basic string matching approaches adapted from the information retrieval domain. These algorithms have been dependent on some basic techniques of element level string matching, linguistic similarities or constraints likeliness at element level or higher schema structure level. Similar graph algorithms utilized in schema matching is a special form of constraints matching [25]. The kernel of

schema matching tool is the matcher. It correspond to the match operator defined in [5]. Some tools use composite approach to combine different matchers, for example, LSD [10] and COMA++ [1]. Our benchmark, by means, of the scoring functions described in section 5, allows to test the quality of a matching algorithm or a combination of matching algorithms for a given scenario.

4.3 The Output

There are three main issues regarding with the output:

- **Type of output.** Most matching tools generate either an integrated schema or/and a set of mappings. The interesting aspect is to study how they produce the integrated schema. Our benchmark, by means of dedicated scoring functions, e.g. *structural overlap* would allow to test whether the method is appropriate. For example, is the method for building an integrated schema from scratch, or from a particular input schema, a good method regarding the ideal schema ?
- **Format of the output.** This is an important feature which gathers the possibilities to use this output. Since our benchmark is dealing with XML schema, the output can be queryable with XQuery.
- **Complexity of the mappings.** Several types of mappings need to be handled. All matching tools supports the 1:1 mapping, i.e. one element from one schema is mapped with one element of another schema. Complex mappings, involving several elements considered as 1:n, n:1, and n:m [22] are not supported by all matching tools. The possible relationship between the mapping elements can be specified: for example, some matching tools precise that an element *price* is mapped to the element *amount* with the relationship $price = amount \times VAT$.

Our benchmark is able to deal with all kinds of mappings.

4.4 Post-Match Phase

The post-match phase uses different measures to select the best correspondence for an element from a set of possible matches which show the semantic equivalence aspect for that element. These techniques are termed as match quality measures in the literature [8]. In our benchmark, the post-match is handled by the overall and the *schema proximity* measures.

5. QUALITY MEASURES

The aim of automatic schema matching process is to avoid a manual, labour and error-prone task in large scale scenarii. For this purpose we have designed a set of score functions for evaluating the quality of the integrated schema. They are complemented by the performance aspect, although it just consists in the matching execution time. Our benchmark also provide some statistics like resource consumption (maximum memory needed, disk space storage) and statistics on the collection of schemata used (dimension of the integrated schema = min/max depth and width, number of nodes, etc.)

5.1 Mapping Quality Measures

Precision is an evaluation criterion very appropriate to the schema matching framework. Precision calculates the proportion of relevant mappings among the extracted mappings. A 100% precision means that all the mappings extracted by the system are relevant.

Another typical measurement coming from the machine learning approach is **recall** which computes the proportion of relevant mappings extracted among all the relevant mappings. A 100% recall means that all relevant mappings have been found.

The main objective of schema matching is to avoid a manual process, or at least save time since an expert is still required: the output of the matcher needs to be checked and eventually completed. Hence the **overall** measure [19] has been specifically designed to evaluate the post match effort. That is, the amount of work needed to add the relevant mappings that have not been discovered and to remove those which are not relevant but have been extracted by the matcher. The Overall measure can have negative values. It is often important to determine a compromise between recall and precision. We can use a measurement taking into account these two evaluation criteria by calculating the **F-measure** [27].

As explained in [8], the F-measure is more optimistic than overall.

5.2 Integrated Schema Quality Measures

A matching tool may provide three types of output: a set of mappings, or an integrated schema, or both. Our benchmark can evaluate the integrity of the integrated schema. In that case, our benchmark is able to evaluate the semantic integrity of the integrated schema. The previous score functions are not appropriate since they do not deal with the structure of the schema. We have designed the following measures to reach this goal.

The first measure takes into account the **backbone** of the tree. More formally, it shows if both trees share a large common subtree, seen as a backbone. This measure returns a value between 0 (no common subtree) and 1 (both trees are the same) is given by the following formula:

Given an input schema tree S_i and another integrated tree noted S_g , then

$$Backbone = \frac{|LSub(S_i \cap S_g)|}{|S_i|} \quad (1)$$

Where $LSub(S_i \cap S_g)$ represents the largest common subtree between trees S_i and S_g , and $|S_i|$ is the number of elements of the tree S_i . This measure reflects the structural similarity of the largest shared component of two trees. Note that this backbone measure is mainly efficient with similar trees.

In the following, a subtree is defined as 'an extract' of a tree which is composed of at least two nodes and has its own root. All the nodes in this subtree must be descendants of this and only this subtree root.

Considering an ideal (model or expert) schema tree S_i and another tree noted S_g which is evaluated against the ideal tree, we define Sub as the set of all disjoint subtrees which are common to S_i and S_g . $|S_i|$ stands for the number of elements in tree S_i , and k for the total number of elements of all subtrees in Sub .

Based on these assumptions, the **structural overlap** is a measure representing the number of elements which are shared by both trees and are included in a common subtree. A 0 value represents a lack of common subtrees while a value closer to 1 shows that most of the elements are included in a common subtree. The following formula processes this structural overlap measure.

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

Another interesting measure we have designed is the **structural proximity**. This measure extends the structural overlap by adding several metrics seen as differences. Indeed, the structural overlap only measures the percentage of elements in the common subtrees, and this needs to be enhanced to evaluate a structural proximity between the two trees. Thus, we have added the number of common subtrees. If S_i and S_g are similar, they have only one common subtree, which is the whole tree. And the more common subtrees, the less similar the trees are. Another difference is 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 its nodes which are missing in the common subtrees affect the structural proximity between the two trees. First we define o the number of elements in S_i that are not included in any common subtree. Thus, $o = |S_i| - k$. And the tree proximity is obtained by the following formula:

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

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

Finally, the last measure, denoted **schema proximity**, computes the similarity between two trees. It takes into account both the structural aspect and the dissimilarity between the tree elements. This dissimilarity gathers the extra elements, namely those that appear in S_g but not in S_i , and the missing elements, which are in S_i but not in S_g . We define this dissimilarity $d = (|S_i| - |Com|) + (|S_g| - |Com|)$ where Com stands for the set of common elements between S_i and S_g trees. The schema proximity formula is then given by:

$$SchemaProximity = \frac{1}{|Sub|} \times \frac{k - d}{|S_i|} \quad (4)$$

The value computed by the schema proximity measure stands between 1 for a complete similarity and $-\infty$ for a total dissimilarity between the two trees.

6. XBENCHMARK: XML SCHEMA MATCHING BENCHMARK

To evaluate and compare XML schema matching tools, we have implemented XBenchMatch. The main goal of this application is to provide two kinds of evaluations: (i) quality matching evaluation, which is based on the use of the measures described in section 5 and (ii) performance of matching schema. The first criteria is very important in automatic schema matching and the second is crucial in large scale and when the schema to be matched are very large. Finally, our tool should also help an end-user to choose the

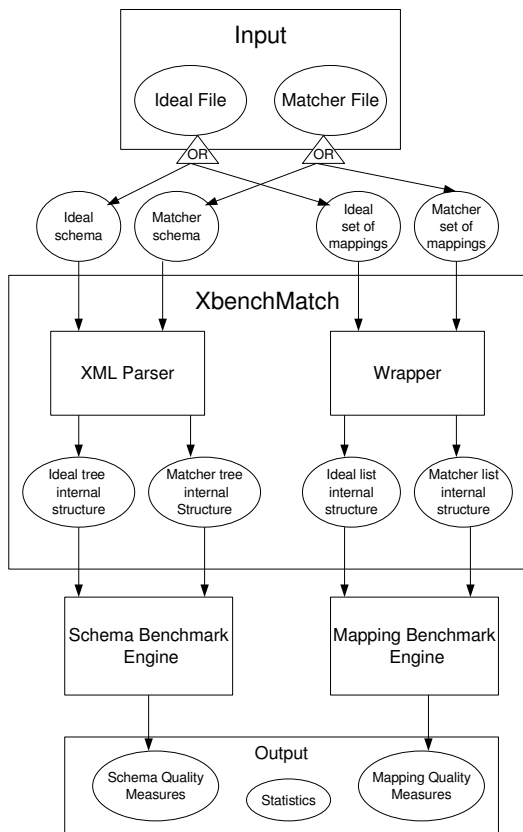


Figure 1: Architecture of XBenchMatch Prototype

more appropriate among schema matching tools according to his requirements. This section gives an overview of our benchmark.

Figure 1 describes the architecture of our prototype. The input files may be of two types, either a well-formed 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 respectively. However, the file generated by the matching tool must be of the same type as that of the expert one. Creating new wrappers will ensure the extensibility by supporting new sets of mappings format. Next, the benchmark engines are able to compute different measures between the ideal file and the matcher’s file. XBenchMatch finally outputs various statistics (performance, size and depth of input schemas, ...) and the quality measures explained in 5. Schema matching systems can also be compared on one or more scenarii, especially by comparing their f-measure and structural proximity measures. Note that the user may also choose the schema corpus that has been matched by the matcher. This only enables to generate statistics on the corpus, for example the average number of nodes, the maximum depth, etc. The static information, i.e the features of the matching tool, is displayed to help the user to understand the results of the matching. The analysis of these results is given by the dynamic criteria, or the measures. Our tool also generate plots for precision, recall, F-measure and overall. If the type of the input files are integrated schemas, then some more measures like structural overlap and structural proximity are also computed.

7. EXPERIMENTS PROTOCOL

All experiments were run on a 3.0 Ghz laptop with 2G RAM under Windows XP. A demo version of the prototype is available at www.lirmm.fr/~duchatea/XBenchMatch. To obtain comparable results, our benchmark provides uniform conditions and use the same test schemas for all matching prototypes. In this section, we present the capabilities of XBenchMatch using four real-world scenarii: the first one describes a person, the second is related to a business order, the third one on university courses and the last one comes from the biology domain. All the ideal integrated schemas have been done manually by an expert and are provided with our benchmark application. Before using XBenchMatch, the user has to generate an integrated schema for each scenario with the matching tools he would like to evaluate.

Each of these scenarii is described with more details:

- **Scenario 1. General schemas** are small-sized schemas describing a person. The ideal set of mappings and the ideal integrated schema have also been expertized manually.
- **Scenario 2. Business schemas** dealing with an order. The first schema is drawn from the XCBL collection³, and has about 160 elements. The second schema also describes an order but it is smaller with only 12 elements. This scenario reflects the possibility to matching a large schema with a smaller one. A human expert has manually generated the set of mappings between these schemas.
- **Scenario 3. University schemas** have been taken from Thalia collection presented in [15]. Each schema has about 20 nodes and the set of mappings contains 15 mappings. An expert has manually mapped the two schemas produced both output matching files, the set of mappings and the integrated schema.
- **Scenario 4. Biology schemas.** The two schemas come from different collections which are protein domain oriented, namely Uniprot⁴ and GeneCards⁵. Both are quite large, with GeneCards around 400 XML paths, and 57 paths in UniProt. A domain expert has manually mapped both schemas, and produced 57 mappings.

	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: Details about the evaluation scenarii.

Table 1 summarizes the characteristics of the scenarii which are used in the benchmark.

The user can run the default benchmark, which involves four scenarii described above against the matcher’s integrated schema for all the schema (person, order, university

³www.xcbl.org

⁴<http://www.ebi.uniprot.org/support/docs/uniprot.xsd>

⁵<http://www.geneontology.org/GO.downloads.ontology.shtml>

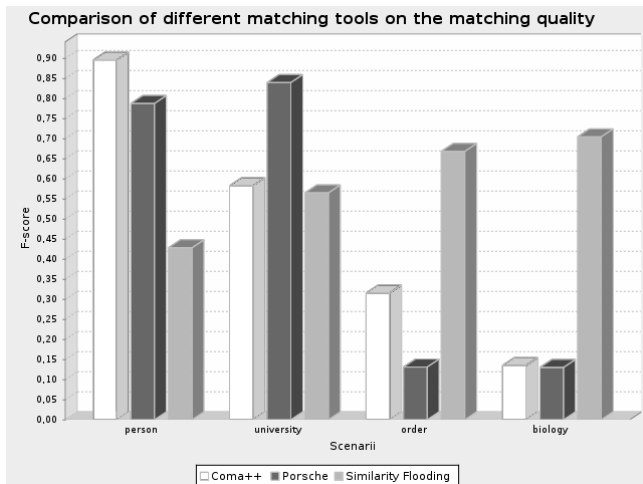


Figure 2: Example of evaluation tools

and biology). XBMATCH is able to calculate the matching quality of these matchers' integrated schema against the ideal integrated schemas. It outputs the following measures: precision, recall, f-measure, overall, structural overlap, structural proximity. A plot is automatically drawn to show the quality according to the number of common elements between the two trees. Another plot focuses on the schema structure by comparing the structural overlap and proximity to the number of elements in the common subtrees.

As XBMATCH is meant to be generic and extensible, it is also possible to run the benchmark using other scenarii. It provides the GUI for this option. The process is identical to the default benchmark, except that the user needs to choose, for a specific scenario, both the ideal integrated schema and the matcher's generated integrated schema. Then the measures showing the quality of the matcher's integrated schema are displayed in the main window.

Finally, XBMATCH enables one to compare the quality of different matching tools on one or several scenarii. For example, figure 2 shows the comparison of three Matchers: COMA++, PORSCHE [24] and Similarity Flooding [19].

8. EXPERIMENT RESULTS

In this section, we present the evaluation results of the following matching tools: COMA++, PORSCHE, Similarity Flooding and BTreeMatch. However, our benchmark application is easily extended to other matchers. We notice, it is hard to find available matchers to test. COMA++ and Similarity Flooding matchers are considered by the schema matching community to provide good matching quality. PORSCHE [23] is a recent tool developed in our team, and it is performance-oriented. BTreeMatch [11] is another recent prototype from our team, which is aimed to provide good performance and quality as well.

8.1 Quality of COMA++

COMA++ generates an integrated schema in ASCII tree format. Thus we developed a wrapper to convert it into an XML schema, which is the normal format of our benchmark. The quality of the integrated schema are given in

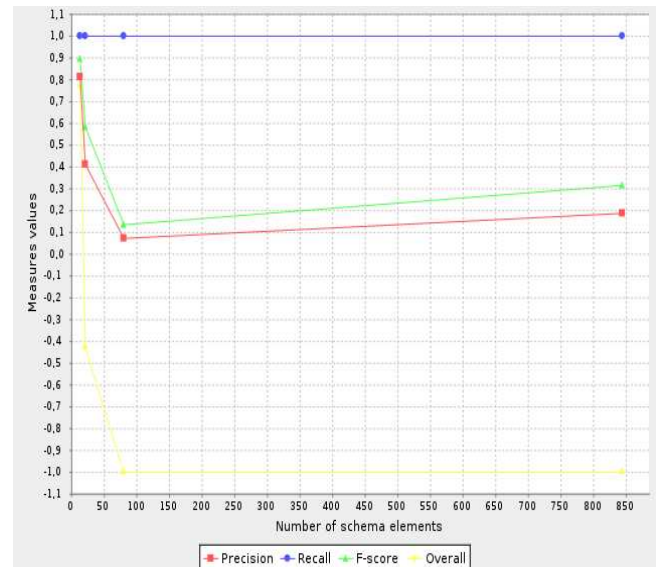


Figure 3: COMA++ quality of the integrated schema

figure 3 and figure 4. The first remark is that COMA++ is able to keep most of the relevant elements, since the recall is equal to 1 on each scenario. However the precision shows that COMA++ becomes less accurate when the size of the schema increases, namely most of the discovered elements should not be in the integrated schema. Except on the first scenario dealing with *person* description, COMA++ needs much post-match effort to add non-discovered elements and to remove the non-relevant ones. This is illustrated by a negative overall value in three scenarii. However, this matching tool uses a list of synonyms, and none has been provided in these experiments. And the domain-specific scenario on *biology* is particularly difficult for such matching tool which mainly uses a combination of terminological measures. As for the quality of structure, the results follow the same direction: the two small scenarii provide an acceptable quality in terms of schema structure, but this quality decreases with bigger schemas.

To improve the understanding of the graph, the overall value has been limited to -1 instead of $-\infty$. One should consider a negative overall value as not significant as it was explained in [19].

COMA++ also produces a set of mappings. The quality on the set of mappings generated by COMA++ is shown in figure 5. COMA++ results are difficult to interpret. Indeed, it discovers most of the relevant mappings in two scenarii (f-measure is above 0.6) but it does not perform as well in two other scenarii (f-measure is less than 0.1). Although the set of mappings does not enable to discover most of the information compared to the integrated schema, the quality is better with the set of mappings than with the integrated schema. Therefore, the post-match effort is reduced.

8.2 Quality of PORSCHE

PORSCHE produces an integrated schema. The produced set of mappings includes those between the input schema and the integrated schema. While in the other tested schema matching the mappings are those between the input schema

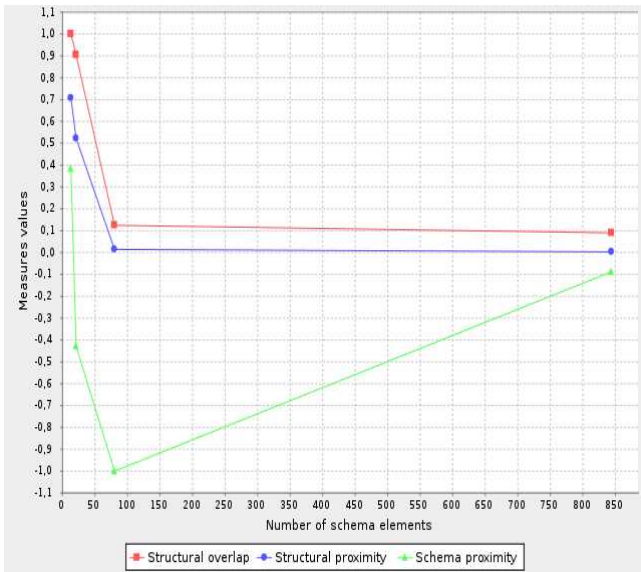


Figure 4: COMA++ quality of the integrated schema

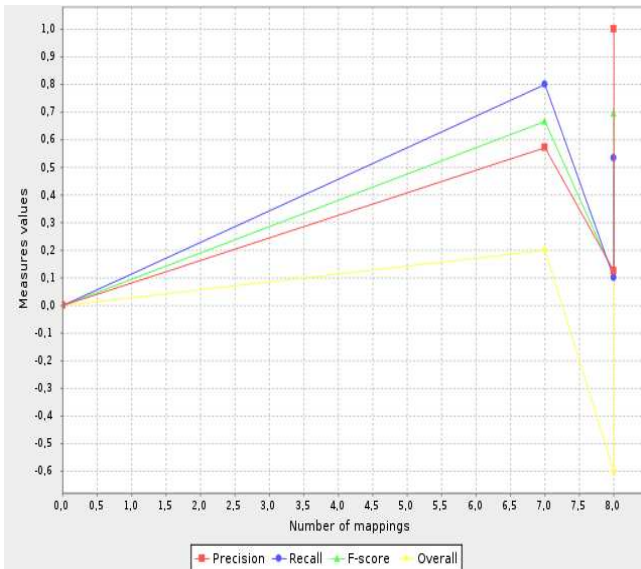


Figure 5: COMA++ quality of the mappings

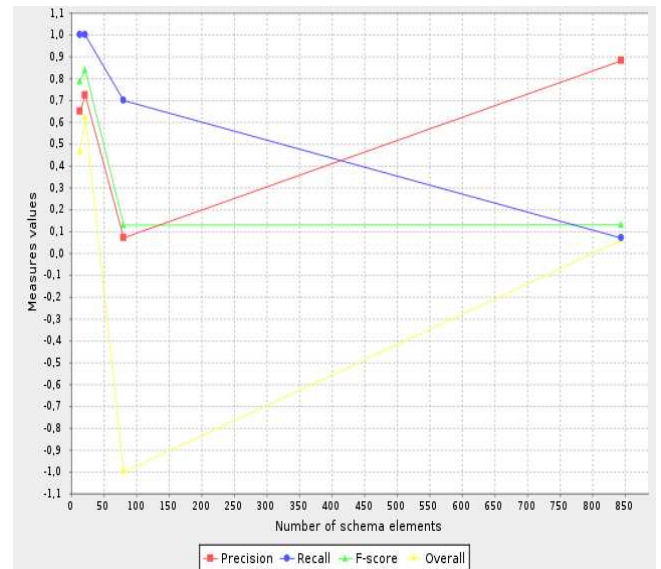


Figure 6: PORSCHE quality of the integrated schema

Therefore, we decide to measure only the quality of the integrated schema. The results of experiments over PORSCHE are depicted in figure 6 and figure 7 on the four scenarii. Both the structure and quality measures on the first small scenarii are acceptable, with a F-score around 0.8 and a structural proximity above 0.4. Note that the post-match effort is minimized in these cases. However, when the number of elements increases, the quality tends to decrease: PORSCHE either discovers many elements with only a few relevant, or it discovers a few common elements among which most of them are relevant. The structural quality values are quite low. Thus, with large schemas, the integrated schemas are not similar with the ones provided by the experts. Like COMA++, PORSCHE normally uses a list of synonyms, and this can explain the average results on the *order* scenario. Besides, one can notice the importance of the precision for the overall measure: a good precision enables to avoid a negative overall value, even with a low recall, as is shown in figure 6.

8.3 Quality of Similarity Flooding

Next experiments carries on Similarity Flooding (SF), implemented in Rondo matching tool. The quality of the integrated schema is given in the two graphs of figure 3 and figure 9. In contrast to the previous matching tools, SF has a better quality with large schemas. Although the precision value stands around 0.5, the structural proximity and the recall are equal to 1 when the number of elements is higher than 75. As this matching tool propagates the benefit of discovering a match to the neighbour nodes, it seems normal that it provides better results with large schemas. The quality on smaller schemas is also acceptable, with values above 0.4. However, the structural quality on the small schemas is low. We can also notice that even in a specific scenario like *biology*, where other matchers may require auxiliary information (e.g. list of synonyms), in SF the quality of integrated schema does not decrease.

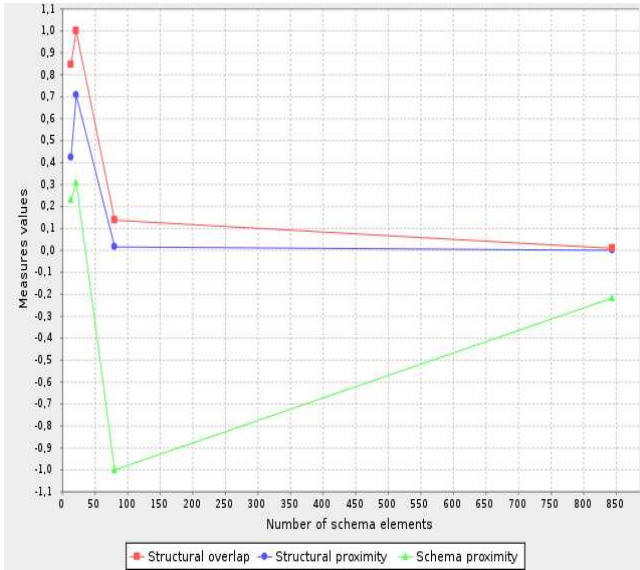


Figure 7: PORSCHE quality of the integrated schema

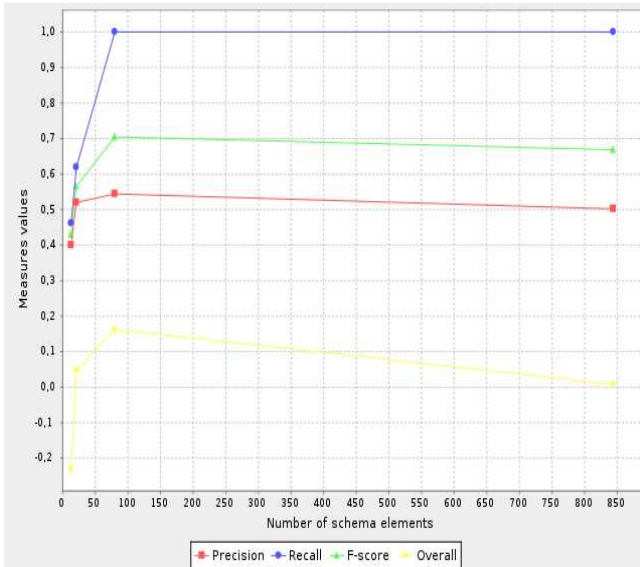


Figure 8: SF Results Evaluation

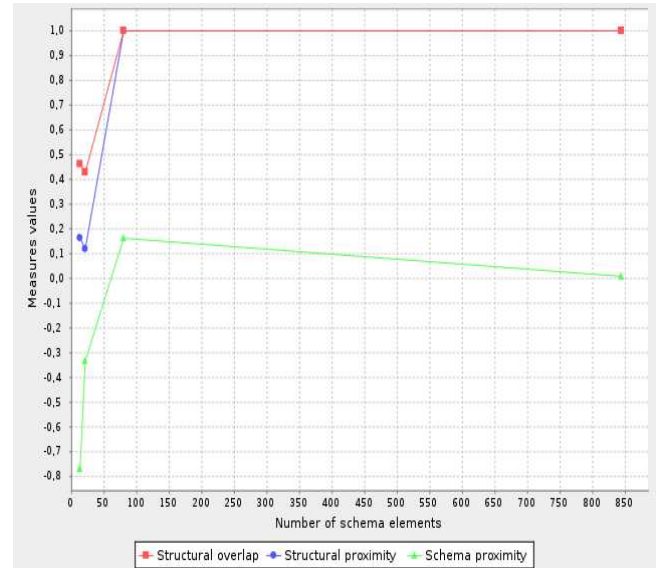


Figure 9: SF quality of the integrated schema

8.4 Quality of BTreeMatch

Figure 10 depicts the quality of the mappings that have been produced by BtreeMatch. We remark that with small schemas, the quality is very low, since the F-score is less than 0.2. However, this measure reaches 0.6 on larger schemas. This behaviour can be explained by the matching algorithms used by BtreeMatch. Indeed, it is based on both terminological and structural techniques, like Similarity Flooding. Thus, it seems that the structural algorithms are able to match large schemas while ensuring an acceptable quality.

8.5 Performance evaluation

	Person		University		Order		Biology	
	S1	S2	S1	S2	S1	S2	S1	S2
	11	10	18	18	20	844	719	80
COMA++	≤ 1 s		≤ 1 s		3 s		4 s	
PORSCHE	≤ 1 s		≤ 1 s		≤ 1 s		≤ 1 s	
SF	≤ 1 s		≤ 1 s		2 s		4 s	
BtreeMatch	≤ 1 s		≤ 1 s		≤ 1 s		2 s	

Table 2: Matching performance on the different scenarii.

Table 2 depicts the matching performances of each matching tool on the evaluation scenarii. All matchers are able to match the small schemas in less than one second. However, when one schema from the scenario is large, COMA++ and Similarity Flooding are less efficient. Similarity Flooding propagates until it reaches a fixpoint computation, involving this process to take more time. On the other hand, PORSCHE, which has been designed to match many large schemas, do not have decreasing performances with schemas up to 800 nodes.

8.6 Discussion.

These experiments show that some matchers are best suited for some scenarii. For example, COMA++ and PORSCHE

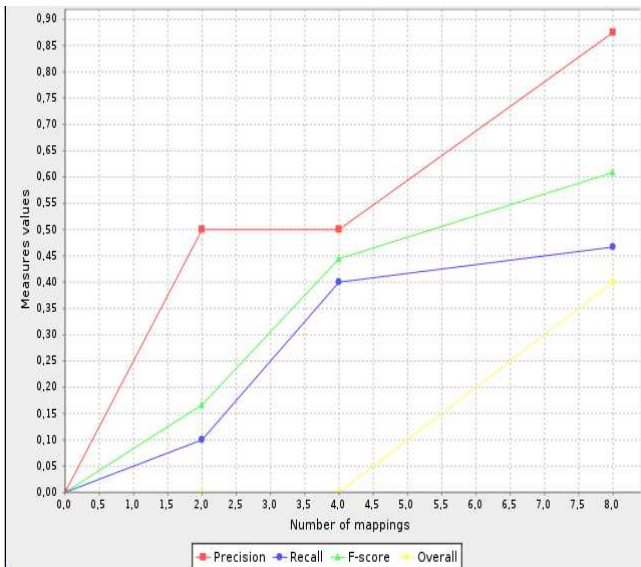


Figure 10: BTreeMatch quality of the mappings

generate integrated schemas with an acceptable quality on small schemas. Similarity Flooding seems to be more quality oriented when the external similarity oracle is not available and the match decision is more structure oriented. Based on the versatility of the state of the art of schema matching tools, we require more experimentations with our benchmark tool. This will enable us to classify the current tools for different domains and matching activities (matching, integration, ...). Thus converting our benchmark into a handy tool for both naive and domain expert users.

9. RELATED WORK

9.1 Tentative for Benchmarking Schema Matching Tools

To the best of our knowledge, there is no complete benchmark for schema matching tools. In [8], the authors present an evaluation of schema matching tools. The main criteria required to reach this goal are discussed. A summary of the capabilities of each matching tool is finally provided. However, as the authors explained, it is quite difficult to evaluate the matching tools for several reasons: they are not always available as a demo. Therefore, it is not possible to test them against specific sets of schemas. Some require specific resources to be efficient, like an ontology or a thesauri, which are not always available. Finally some matching tools take as input specific files, for example Rondo. This evaluation suffers from two drawbacks, not mentioning the fact it was published 5 years ago: by evaluating the matching tools with the scenarios provided in their respective papers, one cannot judge efficiently on the capabilities of each matching tool. Secondly, some matching tools generate an integrated schema instead of a set of mappings, and the measures provided to evaluate a set of mappings are not sufficient to evaluate the quality of an integrated schema.

A proposal for evaluating on schema matching has been done in [28]. It extends [8] by adding time measures and

relies on real-world schemas to evaluate the matching tools. 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 has not been implemented. In contrast to our work, this system is not available and is not extensible.

Our work extends the criteria list provided in [8], by adding some measures to evaluate the quality of integrated schemas. It goes further on the evaluation aspect. Indeed all the matching tools are evaluated against the same scenarios, thus involving a better and more thorough comparison.

9.2 Schema Matching Tools

In this section we review works classified under the schema matching. The surveys [22, 25, 28] incorporate solutions from schema level (metadata), as well as instance level (data) research, including both Database and Artificial Intelligence domains. Most of the methods discussed in these surveys compare two schemas (with or without their data instances) and work out quality matching for the elements of schema 1 to the schema 2. Some of the tools also suggest the merging process of the schemas based on the matching found in first step. Here we present the main schema matching namely the one we have tested with our benchmark.

TRANSCM [20] objective is to transform instances of source schema into target schema. It can have input schemas as DTD or OODB. Internally the schemas are converted into labeled trees and the match process is performed node by node in the top-down manner. TRANSCM presumes a high degree of similarity between the two schemas. TRANSCM supports a number of matchers (rules), to find correspondences between schema nodes. Each rule may in turn combine multiple match criteria, e.g. name similarity and the number of descendants. The rules are assigned distinct priorities and applied in a fixed order. If more than one target elements are found as possible match, user interaction is required to select the match. And in case no match is found user is allowed to apply a new rule to find a match.

DIKE [21] prototype implements a hybrid approach to automatically find synonymy, hypernymy and homonym correspondences between elements of Entity-Relationship (ER) schemas. User specific set of synonyms, hypernym and homonym are utilized, constructed by some expert or using some thesauri. Other than the linguistic and syntactic comparison, the main algorithm is a structural matcher, which performs a pair-wise comparison of elements from the input schemas. The weight of similarity between two elements is increased, if the algorithm finds some similarity between the related elements of the pair of elements.

CUPID [18] is a generic, hybrid schema matching prototype, consisting of a name matcher and a structural one. It has been used for XML and relational schemas. Internally, schemas are converted into trees, in which additional nodes are added to resolve the multiple/ recursive relationships between a shared node and its parent nodes. First, linguistic similarity of pair of nodes is calculated using external oracles of synonyms and abbreviations. Then the structural matcher is applied on the tree structures in post order manner. This technique gives similarity possibilities for non-leaf nodes, depending upon the similarity of their leaves. For

each pair of nodes, their linguistic and structural similarity are aggregated to a weighted similarity using a weighted sum. If the weighted similarity exceeds a threshold, the structural similarity of the leaf pairs is increased. Otherwise, it is decreased. For each source element, CUPID selects the target element with the highest weighted similarity exceeding a given threshold as the match candidate.

Similarity Flooding [19] have been used with Relational, RDF and XML schemas. These schemas are initially converted into labeled graphs and SF approach uses fix-point computation to determine correspondences of 1:1 local and m:n global cardinality between corresponding nodes of the graphs. The algorithm has been implemented as a hybrid matcher, in combination with a name matcher based on string comparisons. First, the prototype does an initial element-level name mapping, and then feeds these mappings to the structural SF matcher. The weight of similarity between two elements is increased, if the algorithm finds some similarity between the related elements of the pair of elements. In a modular architecture, the components of rondo, such as schema converters, the name and structural matchers, and filters, are available as high-level operators and can be flexibly combined within a script for a tailored match operation.

PROTOPLASM [6] target is to provide a flexible and a customizable framework for combining different match algorithms. Present CUPID and Similarity flooding are being used as the base matchers it. SQL and XML schemas, converted into graphs internally, have been successfully matched. PROTOPLASM supports various operators for computing, aggregating, and filtering similarity matrices. Using a script language, it allows flexibly defining and customizing the work flow of the match operators.

COMA/COMA++ [1, 9] is a generic, composite matcher with very effective match results. It uses the same architecture like that of Protoplasm but its range of match algorithms is more complete. It can process the relational, XML, RDF schemas as well as ontologies. Internally it converts the input schemas as trees for structural matching. For linguistic matching it utilizes a user defined synonym and abbreviation tables like CUPID, along with n-gram name matchers. Similarity of pairs of elements is calculated into a similarity matrix. At present it uses 17 element level matchers. For each source element, elements with similarity higher than than threshold are displayed to the user for final selection. The COMA++ supports a number of other features like merging, saving and aggregating match results of two schemas.

S-MATCH/S-MATCH++ [2, 14] takes two directed acyclic graphs like structures e.g. XML schemas or ontologies and returns equivalence, subsumption type correspondences between pairs of elements. It uses external oracle Wordnet to evaluate the linguistic matching along with its structural matcher to return a subsumption type match. It is also heavily dependent on SAT solvers, which decreases its time efficiency. At present it uses 13 element-level matchers and 3 structural level matchers.

Smiljanic et al. work, [26] shows how personal schema for

querying, can be efficiently matched and mapped to a large repository of related XML schemas. The method identifies fragments with in each schema of the repository, which will best match to the input personal schema, thus minimizing the target search space. The prototype implementation, called bellflower, uses k-means data mining algorithm as the clustering algorithm. The authors also demonstrate that this work can be implemented as an intermediate phase with in the framework of existing matching systems. The technique does produce efficient system but with some reduction in effectiveness.

Porsche [23] utilizes tree mining technique to cluster and holistically match and merge large number of schemas (represented as trees). It gives approximate matchings and generates an integrated schema with mappings from source schemas to this integrated schema. It has been devised to cater the quality as well as the performance element for large scale scenarios using domain specific linguistic matching (domain specific synonym and abbreviation oracles). It works in three steps. First, in the pre-mapping part, schema trees are input to the system as a stream of XML and calculate the scope and node number for each of the nodes in the input schema trees. Other statistics like each schema size, maximum depth and node parent are also calculated. A listing of nodes and a list of distinct labels for each tree is constructed. Next, a linguistic matcher identifies semantically distinct node labels in the labels list. The user can set the level of similarity of labels as A) Label String Equivalence, B) Label Token Set Equivalence (abbreviation table) and C) Label Synonym Token Set Equivalence (synonym table). Then Porsche derives the meaning for each individual token and combines these meanings to form a label concept. Finally, similar labels are clustered together. Since each input node remains attached to the its label object, this intuitively forms **similar label nodes clusters** within a certain schema.

BtreeMatch [11] approach uses the B-tree as the main structure to locate matches and create mappings between XML tree structures. The advantage of searching for mappings using the B-tree approach is that B-tree have indexes that significantly accelerate this process. For example, let us consider two schemas $S1$ and $S1$ with respectively 8 and 9 elements. Matching these schemas will entail 2 matching possibilities with an algorithm that tries all combinations. By indexing in a B-tree, we are able to reduce this number of matching possibilities, thus involving better performance. BtreeMatch does not use a matrix to compute the similarity of each couple of elements. Instead, a B-tree, whose indexes represent tokens, is built and enriched as we parse new schemas, and the discovered mappings are also stored in this structure. The tokens reference all labels which contains it. For each input XML schema, the same algorithm is applied: the schema is parsed element by element by pre-order traversal. This enables to compute the context vector of each element. The label is split into tokens. We then fetch each of those tokens in the B-tree, resulting in two possibilities:

- no token is found, so we just add it in the B-tree with a reference to the label.
- or the token already exists in the B-tree, in which case

we try to find semantic similarities between the current label and the ones referenced by the existing token. We assume that in most cases, similar labels have a common token (and if not, they may be discovered with the context similarity).

9.3 Data Instance Based Schema Matching

In this section we consider some recent prototypes, which use schema instance data and machine learning techniques to find possible matches between two schemas. These matchers compute all possible match or mismatch possibilities among the attributes of the two source schemas to come up with best results.

AUTOMATCH [4] is the predecessor of AUTOPLEX [3]. It uses single strategy, machine learning match technique. It explicitly uses Naive Bayesian algorithm to analyse the input instances of relational schemas fields against previously built global schema. The match result consists of 1:1 correspondences and global cardinality.

CLIO [16] has been developed at IBM. It has comprehensive GUI interface and provides matching for XML and SQL schemas. It uses a hybrid approach, combining approximate string matcher for element names and Naive Bayes-learning algorithm for exploiting instance data. It also facilitates in producing transformation queries (SQL, XQuery, or XSLT) from source to target schemas, depending upon the computed mappings.

LSD [10] is a composite matcher. It requires an already developed global schema, against which newer schemas and their data instances are matched. LSD uses machine learning algorithms for in matching as well as in summing up the match results for each pair of attribute comparisons. LSD has been further utilized in Corpus-based Matching [17], which creates a CORPUS of existing schema and their matches. In this work, input schemas are first compared to schemas in the corpus before they are compared to each other. Another extension based on LSD is IMAP [7]. Here the authors utilize LSD to find 1:1 and n:m mapping among relational schemas. It provides a new set of machine-learning based matchers for specific types of complex matchings e.g. name is a concatenation of firstname and lastname. It also provides the information about the prediction criteria for a match or mismatch.

10. CONCLUSION

In this paper, we present a benchmark for XML schema matching tools. Our approach is focusing on the evaluation of the matching tools in terms of matching quality and performance. Our work extends the criteria provided in [8] by adding new scoring functions which evaluate the quality of integrated schemas and extends the evaluation methodology. Indeed, in XBenchMatch all the matching tools are evaluated against the same scenario, and produce an improved objective comparison. Next, we also aim at giving an overview of a matching tool by analysing its features and deducing some criteria it might fulfill. This should help an end-user to choose among the available matching tools depending on his requirements. Finally, we Furthermore, we provide a testbed involving a large schema corpus that can be used by everyone to quickly benchmark new matching algorithms.

We are planing to extend our experiments to CUPID prototype and other matching tools if they are available. We also plan to include evaluation about the scalability. This does not require any extension of our benchmark. We only require to manually generate an expert schema for a large number of input schemas to be matched.

11. ACKNOWLEDGMENTS

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

12. REFERENCES

- [1] D. Aumüller, H. H. Do, S. Massmann, and E. Rahm. Schema and ontology matching with coma++. In *ACM SIGMOD Conference, DEMO paper*, pages 906–908, 2005.
- [2] P. Avesani, F. Giunchiglia, and M. Yatskevich. A large scale taxonomy mapping evaluation. In *ISWC Conference*, pages 67–81, 2005.
- [3] J. Berlin and A. Motro. Automated discovery of contents for virtual databases. In *CoopIS Conference*, pages 108–122, 2001.
- [4] J. Berlin and A. Motro. Database schema matching using machine learning with feature selection. In *CAiSE Conference*, 2002.
- [5] P. A. Bernstein. Applying model management to classical meta data problems. In *CIDR Conference*, 2003.
- [6] P. A. Bernstein, S. Melnik, M. Petropoulos, and C. Quix. Industrial-strength schema matching. *SIGMOD Record*, 33(4):38–43, 2004.
- [7] R. Dhamankar, Y. Lee, A. Doan, A. Halevy, and P. Domingos. imap: Discovering complex semantic matches between database schemas. In *ACM SIGMOD Conference*, 2004.
- [8] H. H. Do, S. Melnik, and E. Rahm. Comparison of schema matching evaluations. In *IWWDC Conference*, 2003.
- [9] H. H. Do and E. Rahm. Coma - a system for flexible combination of schema matching approaches. In *VLDB Conference*, pages 610–621, 2002.
- [10] A. Doan, P. Domingos, and A. Y. Halevy. Reconciling schemas of disparate data sources - a machine learning approach. In *IACM SIGMOD Conference*, 2001.
- [11] F. Duchateau, Z. Bellahsene, M. Roantree, and M. Roche. An indexing structure for automatic schema matching. In *SMDB Workshop ICDE*, 2007.
- [12] F. Duchateau, Z. Bellahsene, and M. Roche. A context-based measure for discovering approximate semantic matching between schema elements. In *IEEE RCIS Conference*, 2007.
- [13] A. Gangemi, N. Guarino, C. Masolo, and A. Oltramari. Sweetening wordnet with dolce. *AI Magazine*, 24(3):13–24, 2003.
- [14] F. Giunchiglia, P. Shvaiko, and M. Yatskevich. S-match: an algorithm and an implementation of semantic matching. In *ESWS Conference*, 2004.
- [15] J. Hammer, M. Stonebraker, and O. Topsakal. Thalia: Test harness for the assessment of legacy information

- integration approaches. In *ICDE Conference*, pages 485–486, 2005.
- [16] M. A. Hernandez, R. J. Miller, and L. M. Haas. Clio: A semi-automatic tool for schema mapping (software demonstration). In *ACM SIGMOD Conference*, 2002.
- [17] J. Madhavan, P. A. Bernstein, A. Doan, and A. Y. Halevy. Corpus-based schema matching. In *ICDE Conference*, pages 57–68, 2005.
- [18] J. Madhavan, P. A. Bernstein, and E. Rahm. Generic schema matching with cupid. In *VLDB Conference*, pages 49–58, 2001.
- [19] S. Melnik, H. G. Molina, and E. Rahm. Similarity flooding: A versatile graph matching algorithm and its application to schema matching. In *ICDE Conference*, 2002.
- [20] T. Milo and S. Zohar. Using schema matching to simplify heterogeneous data translation. In *VLDB Conference*, pages 122–133, 1998.
- [21] L. Palopoli, G. Terracina, and D. Ursino. The system dike: Towards the semi-automatic synthesis of cooperative information systems and data warehouses. In *ADBIS-DASFAA Symposium*, pages 108–117, 2000.
- [22] E. Rahm and P. A. Bernstein. A survey of approaches to automatic schema matching. *VLDB J.*, 10(4):334–350, 2001.
- [23] K. Saleem and Z. Bellahsène. Scalable large scale schema matching. In *DECOR Workshop on Semantic Matching, EGC'07*, 2007.
- [24] K. Saleem, Z. Bellahsene, and E. Hunt. Porsche: Performance oriented schema matching. Technical Report RR-06055, Laboratoire d'Informatique, de Robotique et de Microelectronique de Montpellier (LIRMM), 2006.
- [25] P. Shvaiko and J. Euzenat. A survey of schema-based matching approaches. *J. Data Semantics IV*, pages 146–171, 2005.
- [26] M. Smiljanic, M. van Keulen, and W. Jonker. Using element clustering to increase the efficiency of xml schema matching. In *Workshop ICDE*, 2006.
- [27] C. Van-Risbergen. *Information Retrieval*. 2nd edition, London, Butterworths, 1979.
- [28] M. Yatskevich. Preliminary evaluation of schema matching systems. Technical Report DIT-03-028, Informatica e Telecomunicazioni, University of Trento, 2003.