Research Online Publishing
www.ronpub.com

Open Access

*Open Journal of Databases (OJDB)*
*Issue X, Number X, 20XX*

*http://www.ronpub.com/ojdb*

# Designing a Benchmark for the Assessment of Schema Matching Tools

Fabien Duchateau [A], Zohra Bellahsene [B]

[A] LIRIS, UMR5205, Université Claude Bernard Lyon 1, Lyon, France, fduchate@liris.cnrs.fr
[B] LIRMM, Université Montpellier 2, Montpellier, France, bella@lirmm.fr

## Abstract

*Over the years, many schema matching approaches have been developed to discover correspondences between schemas. Although this task is crucial in data integration, its evaluation, both in terms of matching quality and time performance, is still manually performed. Indeed, there is no common platform which gathers a collection of schema matching datasets to fulfil this goal. Another problem deals with the measuring of the post-match effort, a human cost that schema matching approaches aim at reducing. Consequently, we propose XBenchMatch, a schema matching benchmark with available datasets and new measures to evaluate this manual post-match effort and the quality of integrated schemas. We finally report the results obtained by different approaches, namely COMA++, Similarity Flooding and YAM. We show that such a benchmark is required to understand the advantages and failures of schema matching approaches. Therefore, it could help an end-user to select a schema matching tool which covers his/her needs.*

## Type of Paper and Keywords

Regular research paper: *schema matching, benchmark, data integration, heterogeneous databases, evaluation, integrated schema*

## 1 Introduction

Data integration is a crucial task which requires that the models which represent the data (e.g., schemas, ontologies) have to be matched [5]. Due to the growing availability of information in companies, agencies, or on the Internet, decision makers may need to quickly understand some concepts before acting, for instance in international emergency events. In such context, schema matching is a crucial process since its effectiveness (matching quality) and its efficiency (performance) directly impacts the decisions [46]. In a different scenario, the costs for integrating multiple data sources can reach astronomic amounts. Thus, evaluating these costs enables project planners to assess the feasibility of a project by analysing incomplete integrated schemas or detecting the overlap between the data sources. In addition, XML and XSD have become standard exchange formats for many web sites and applications.

As a consequence, the schema matching community has been very prolific in producing tools during the last decades. Many surveys [43, 48, 24, 45, 41] reflect this interest and their authors propose various classifications of matching approaches according to their features. Each schema matching tool has been designed to satisfy one or more schema matching tasks. For instance, some tools are dedicated to large scale scenarios, i.e., to match a large number of input schemas. Others may be devoted to discover complex correspondences, in which more than two elements from the input schemas are involved. In papers related to a schema matching approach, there is often an *experiment* section which demonstrates the benefit or gain, mainly in terms of matching quality or time performance, of the proposed approach. Sometimes, authors have also compared their work with existing tools, thus showing how their approach may (or not) perform better than others. Yet, this is not a sufficient evaluation for several reasons. First, the schemas against which approaches are evaluated are not clearly defined. In other words, authors do not always detail enough the properties of the schemas used in the experiments and the tasks they may fulfil. Besides, experiments cannot be reproduced with ease. As a result, it is difficult for an end-user to choose a matching tool which covers his/her needs.

For these reasons, we believe that a benchmark for the schema matching community is necessary to evaluate and compare tools in the same environment. In information retrieval community, authors of the Lowell report [28] clearly call for the design of a test collection. Thalia [29], INEX [26] and TREC [4] were therefore proposed. Another closely-related domain is called ontology alignment. Schemas differ from ontologies mainly because they include less semantics, for instance on the relationships. Ontology alignment researchers have designed OAEI (Ontology Alignment Evaluation Initiative) [23, 27]. Every year since 2004, an evaluation campaign of ontology alignment tools is performed. The entity matching task, which consists of discovering correspondences at the instance level, also benefits from various benchmarks [30, 32]. Conversely, there is currently no common evaluation platform for explicit schema matching, although some attempts of evaluation have been proposed [13, 48, 2]. In [13], the authors present an evaluation of schema matching tools. The main drawback deals with the evaluation of the matching tools with the scenarios provided in their respective papers, thus one cannot objectively judge the capabilities of each matching tool. Secondly, some matching tools generate an integrated schema along with a set of correspondences, and the common measures used to assess the quality of a set of correspondences are not appropriate to evaluate the quality of an integrated schema.
Another proposal for evaluating schema matching tools has been done in [48]. It extends [13] by adding time measures and relies on real-world schemas to evaluate the matching tools. However, the evaluation system has not been implemented and it does not automatically compute quality and time performance results. Finally, STBenchmark [2] was proposed to evaluate the relationship of the mappings (i.e., the transformations of source instances into target instances), but it does not deal with schema matching tools. Our work extends the criteria provided in [13], 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 scenarii.

In this paper, we present the foundation of a benchmark for XML schema matching tools. Our evaluation system named XBenchMatch involves a set of criteria for testing and evaluating schema matching tools. It provides 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 fulfil. 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 that can be used by everyone to quickly benchmark schema matching tools. Here we outline the main contributions of our work:

- **Testbed.** We have proposed a classification of schema matching datasets according to schema properties and schema matching task perspectives. This allows to choose the features that the user needs to test.
- **Evaluation metrics.** To improve the evaluation of the matching quality for schema matching, we propose new metrics to measure the post-match effort, the rate of automation performed by using a matching tool and the quality of integrated schemas.
- **Experimental validation.** We have extended our demonstration tool, XBenchMatch [20], to take into account the classification and the new quality measures. Experiments with three matching tools and over ten datasets demonstrate the benefit of our benchmark for the matching community.

The rest of the paper is organized as follows. First, we give some definitions and preliminaries in Section 2. Related work is presented in Section 3. In Section 4, we present an overview of XBenchMatch. Section 5 describes the datasets and their classification. Section 6 covers the new measures that we have designed for evaluating a set of discovered correspondences. We report in Section 7 the results of three schema matching tools by using XBenchMatch. Finally, we conclude and outline future work in Section 8.

## 2 Preliminaries

Here we introduce the notions used in the paper. **Schema matching** is the task which consists of discovering semantic correspondences between schema elements. We consider **schemas** as edge-labelled trees (a simple abstraction that can be used for XML schemas, web interfaces, or other semi-structured or structured data models). A **correspondence** is a semantic link between several schema elements which represent the same real-world concept. Contrary to [2], evaluating the quality of the mapping (i.e., the transformation function between instances of a first schema element and those of another schema element) is out of scope of this paper since we focus on evaluating the quality of correspondences. We also limit correspondences to *1:1* (i.e., one schema element is matched to only one schema element) or to *1:n* (i.e., one schema element is matched to several schema elements). Currently, only a few schema matching tools such as IMAP [12] are able to produce *n:m* correspondences.

A **schema matching dataset** is composed of a **schema matching scenario** (the set of schemas to be matched, without schema instances), a set of expert correspondences (between the schemas of the scenario) and/or the integrated expert schema along with expert correspondences (between the integrated schema and each schema of the scenario) [8]. Such datasets [23], also called ground truth or test collections [4, 29], are used by most evaluation tools as an oracle, against which they can evaluate and compare different approaches or tools.

The quality in schema matching is traditionally computed using well-known metrics [7]. **Precision** calculates the proportion of correct correspondences extracted among the discovered ones. Another typical measure is **recall** which computes the proportion of correct discovered correspondences among all correct ones according to the ground truth. Finally, **F-measure** is a trade-off between precision and recall.

## 3 Related Work

This section presents related work which covers the main topics of this paper. First, we focus on benchmarks for matching tools in ontology and schema domains. Then, we describe schema matching tools that are publicly available for evaluation with our benchmark.

### 3.1 Ontology Benchmarks

In the ontology domain, a major work for evaluating ontology matching is called OAEI, standing for Ontology Alignment Evaluation Initiative [23, 27]. Since 2004, this campaign yearly invites researchers to test their ontology matching tools against different scenarios (datasets). They run and test their tools during several months and they send both the system and the produced alignments to OAEI organizers. Results are then published and the OM workshop enables researchers to share feedback. The OAEI datasets fulfil various criteria. For instance, the *benchmark* dataset gathers many schemas in which a specific type of information has been altered (modifications, deletions, etc.). Consequently, it aims at detecting the weaknesses of a tool according to available information. Other datasets might be very specific like the *FAO*[1] ontologies. In such case, dictionaries are available as external resource for the matching tools. However, only the *benchmark* and *anatomy* datasets are provided with the complete set of expert correspondences. For the remaining datasets, OAEI organizers are in charge of evaluating the results w.r.t. expert correspondences. A tool, AlignAPI [11], can be used to automatically compute precision, recall, F-measure and fall-out values for a given dataset. However, this tool is mainly useful with datasets for which expert correspondences should be provided by users. A recent work finally proposes a method for automatically generating new datasets [25].

Last but not least, the OAEI organizers have pointed out that the recent campaigns do not enable a significant impact on the quality. They conclude that the tools may have reached an upper limit of automation, and they call in 2013 for a new interactive track in which users are involved [27]. In such context, new metrics are needed for evaluating the number of user interactions. The OAEI ”interactive track” used in the 2013 contest the number of both positive and negative interactions. This is crucial since the tools are traditionally in charge of suggesting the next interactions, and counting the negative ones helps to detect whether the tool is able to suggest relevant interactions. In our case, only positive interactions are taken into account since XBenchMatch aims at estimating the worst-case scenario for an expert to reach $100\%$ F-measure from a given set of correspondences. Other metrics have been refined to consider various types of interactions, such as providing the definition of the semantic link of a correspondence [42]. The AUL metric (Area Under Learning curve) is proposed to evaluate interactive matching tools and compare their F-measure given a cost of interaction. Thus, the AUL metric is an estimation because it is calculated during the interactive matching. On the contrary, our metrics are computed during evaluation, with knowledge of the correct set of correspondences.

The maturity achieved by the OAEI campaign is an interesting starting point for designing a schema matching benchmark. Only a few schema matching tools which are able to parse ontologies have participated in OAEI, for instance COMA++ in the 2006 campaign [35] and YAM++ since 2011 [22, 38, 40]. Indeed, most of schema matching tools do not parse ontologies and are not designed to match them, since ontologies are richer than schemas in terms of semantics for instance. On the contrary, our benchmarking tool is specifically designed for the schema matching community. Furthermore, it offers additional features:

[1]Food and Agriculture Organization

- New metrics for evaluating the matching quality of correspondences (e.g., post-match effort [19]) and the quality of an integrated schema (e.g., schema proximity [18]). Such metrics may be useful to the recent OAEI interactive track[2], for which user interaction is allowed, to determine which system needs the smallest amount of interactions.

- Dedicated schema matching datasets. Contrary to ontologies which include semantics (e.g., concepts are linked using specific types of relationships such as "subClassOf"), schemas do not have such semantics. Thus, ontology matching tools and schema matching tools use different similarity measures and do not exploit the same type of information. For those reasons, dedicated schema matching datasets are required.

- Generation of plots of the evaluation results both for a comparison between tools and for the assessment of a single tool.

## 3.2 Schema Matching Benchmarks

To the best of our knowledge, there is no complete benchmark for schema matching.

In [13], the authors mainly discuss the criteria required to evaluating schema matching tools. More precisely, they provide a report about the evaluation results of the matching tools: COMA [14], Cupid [33], LSD [17], and Similarity Flooding [36]. However, they do not propose any evaluation tool. As the authors explain, it is quite difficult to evaluate the matching tools for several reasons, especially when they are not available as demo. Therefore, it is not possible to test them against specific sets of schemas. Finally, schema matching is not a standardized task, thus its inputs and outputs can be totally different from one schema matching tool to another. For instance, a given tool might require specific resources to be effective, like an ontology, a thesauri, or training data, which are not necessarily provided with the tool or with the matching scenario. Consequently, users do not have sufficient information to choose the schema matching tool which suits their needs. Secondly, some matching tools also generate an integrated schema along with a set of correspondences, and the measures provided to evaluate a set of correspondences are not appropriate in that case.

Another proposal for evaluating schema matching tools has been done in [48]. It extends [13] by adding time measures and it relies on real-world schemas to compare matching tools. Three schema matching tools (COMA, Similarity Flooding and Cupid) were evaluated against four scenarios. Two of these scenarios are small (fewer than 20 elements) and the two others have an average size (fewer than 100 elements). Most of these scenarios have labels with low heterogeneity. Besides, their structure is not very nested (1 to 3 depth levels). The results obtained by the matching tools on four scenarios are probably not sufficient to judge on their performance. No evaluation system has been implemented, and results (in term of quality and time values) are not automatically computed, thus making it difficult for extending the work. Finally, the quality of integrated schemas produced by schema matching tools is not evaluated.

In 2008, STBenchmark [2] was proposed to evaluate mapping systems, namely the transformation function from source instances into target instances. This benchmark aims at evaluating both the quality of these transformations and their execution time. However, the discovery of mappings is performed manually using visual interfaces which generate XSLT scripts. Benchmark scenarios, against which mapping systems are evaluated, are gathered according to common transformations (e.g., copying, flattening). To enrich this corpus, STBenchmark also includes scenarios and instances generators. Both generators can be tuned thanks to configuration parameters (e.g., kind of joins, nesting levels). Finally, a simple usability model enables users to quantify their number of actions (in terms of keyboard inputs and mouse clicks) that they have to perform to design the produced mappings. A cost value is finally returned by computing a weighted sum of all actions. Four mapping systems (whose names have not been provided in the paper) have been evaluated both in terms of quality (based on the usability model) and time performance. Contrary to STBenchmark, our solution is dedicated to the discovery of correspondences rather than mappings.

## 3.3 Schema Matching Tools

Many approaches have been devoted to schema matching [6]. In various surveys, researchers have proposed a classification for matching tools [43, 24], which has been later refined in [45]. However, this section only focuses on schema matching tools which are publicly available for evaluation with our benchmark.

### 3.3.1 Similarity Flooding/Rondo

Similarity Flooding (SF) and its successor Rondo [36, 37] can be used with Relational, RDF and XML schemas. These input data sources are initially converted into labelled graphs and SF approach uses fix-point computation to determine correspondences between graph nodes. The algorithm has been implemented as a hybrid matcher, in combination with a terminological similar-

[2] `http://web.informatik.uni-mannheim.de/oaei-interactive/2013/`

ity measure. First, the prototype does an initial element-level terminological matching, and then feeds the computed correspondences to the structural similarity measure for the propagation process. This structural measure states that two nodes from different schemas are considered similar if their adjacent neighbours are similar. When similar elements are discovered, their similarity increases and it impacts adjacent elements by propagation. This process runs until there is no longer similarity increasing. Like most schema matchers, SF generates correspondences for pairs of elements having a similarity value above a certain threshold. The generation of an integrated schema is performed using Rondo's *merge* operator. Given two schemas and their correspondences, SF converts the schemas into graphs and it renames elements involved in a correspondence according to the priorities provided by the users.

#### 3.3.2 COMA/COMA++

As described in [14, 3], COMA++ is a hybrid matching tool that incorporates many independent similarity measures. It can process Relational, XML, RDF schemas as well as ontologies. Different strategies, e.g., reuse-oriented matching or fragment-based matching, can be included, offering different results. When loading a schema, COMA++ transforms it into a rooted directed acyclic graph. Specifically, the two schemas are loaded from the repository and the user selects required similarity measures from a library. For linguistic matching, it also utilizes user-defined synonym and abbreviation tables. For each measure, each element from the source schema is attributed a similarity value between $0$ (no similarity) and $1$ (total similarity) with each element of the target schema, resulting in a cube of similarity values. The final step involves combining the similarity values given by each similarity measure by means of aggregation operators like $max$, $min$, $average$, etc. Finally, COMA++ displays all correspondences possibilities whose similarity value is above a given threshold and the user checks and validates their accuracy. COMA++ supports a number of other features like merging the schemas into an integrated schema which can be saved in an ASCII tree format.

#### 3.3.3 YAM

YAM (Yet Another Matcher) [21, 22] is not (yet) another schema matching system as it enables the generation of *a la carte* schema matchers. It considers the schema matching task as a classification problem. To produce a schema matcher, it automatically combines and tunes similarity measures based on various machine learning classifiers. Then it selects the most appropriate schema matcher for a given schema matching scenario. In addition, optional requirements can be provided such as a preference for recall or precision, similar[3] schemas that have already been matched and expert correspondences between the schemas to be matched. YAM uses a knowledge base that includes a (possibly large) set of similarity measures and classifiers. YAM++ is an extension for matching ontologies [39].

## 4 Overview of XBenchMatch

In this section, we first describe the desiderata for a schema matching benchmark. Then, we present the architecture of our XBenchMatch tool.

### 4.1 Desiderata

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

- **Extensible**, the benchmark is able to evolve according to research progress. This extensibility gathers three points : (i) future *schema matching tools* could be benchmarked, hence XBenchMatch deals with well-formed XML schemas; (ii) new *evaluation metrics* could be added to measure the matching quality or time performance; and (iii) users should easily add new *schema matching datasets*.
- **Portable**. The benchmark should be OS-independent, since the matching tools might run on different OS. This requirement is fulfilled by using Java programming language.
- **Ease of use**, since end-users and schema matching experts are both targeted by this benchmark. Besides, from the experiment results computed by the benchmark, the users should be able to decide between several matching tools the most suitable for a given scenario. Indeed, one of the challenge is to provide sufficient information between conflicting constraints, such as time performance and F-measure.
- **Generic**, it should work with most of the available matching tools. Thus, we have divided the benchmark into datasets, each of them reflecting one or several specific schema matching issues. For instance, tools which are able to match a large number of schemas can be tested against a *large scale scenario*. Dividing the benchmark into datasets enables us to facilitate the understanding of the results. Besides, it does not constrain the benchmark to the common capabilities of the tools. Indeed, if

[3]Similar in terms of heterogeneity degree, structure or domain.

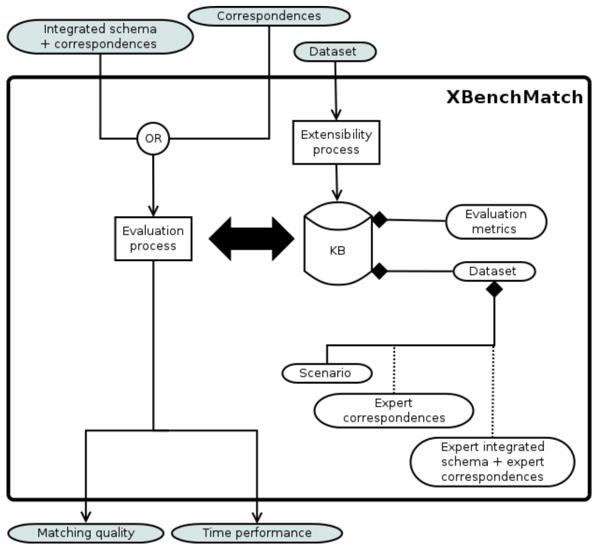


**Figure 1: Architecture of XBenchMatch**

some matching tools can only match two schemas at the same time, this does not prevent other tools to be tested against a datasets with a large number of schemas.

All these requirements should be met to provide an acceptable schema matching benchmark. From these desiderata, we have designed the XBenchMatch architecture.

## 4.2 XBenchMatch Architecture

To evaluate and compare schema matching tools, we have implemented XBenchMatch. Its architecture is depicted by Figure 1 and it relies on two main components: **extensibility process** and **evaluation process**.

The former deals with the **extensibility** of the tool. It takes a dataset as input, and the extension process applies some checking (i.e., the schemas are well-formed, or the expert correspondences have elements which exist in the schemas, etc.). If the dataset is validated by the extension process, then it is added into the knowledge base (KB). This KB stores all information about datasets and evaluation metrics. Consequently, it interacts with the two main components.

The latter process, namely **evaluation**, takes as input the results of a schema matching tool. Indeed, we assume that the schema matching tool to be evaluated had performed matching against a schema matching scenario from our benchmark. Thus, the evaluated matching tool produces a set of correspondences between the schemas of the scenario, and/or an integrated schema along with its associated correspondences. This input is used by the evaluation process, which compares them to the expert set of correspondences and/or expert integrated schema included in the scenario. XBenchMatch outputs quality and time performance results of the matching tool for the given scenario.

## 4.3 Methodology

Before using XBenchMatch, a user has to generate an integrated schema and/or a set of correspondences for each dataset with the matching tool(s) she would like to evaluate. Recall that each dataset contains a schema matching scenario, the expert integrated schema and/or the expert sets of correspondences. Thus, the idea is compare the output produced by a matching tool against those expert ones.

Let us describe a scenario. A user would like to know if her matching tool performs well, in terms of quality, when matching large schemas. She chooses in our benchmark a dataset with large schemas (see Section 5). Then, she runs her matching tool against the schemas of the chosen dataset, which produces a set of correspondences. As she wants to evaluate the quality of this set of correspondences, she uses it as input for XBenchMatch. Our benchmark compares the set of correspondences produced by the matching tool against the expert set of correspondences provided with the dataset. Quality metrics such as post-match effort are computed to assess the quality of the set of correspondences (see Section 6). The user can finally interpret these results visually and discover if her matching tool is suitable for matching large schemas (see Section 7).

## 5 DATASETS CLASSIFICATION

To evaluate schema matching tools, our benchmark includes various schema matching datasets. Indeed, the discovery of correspondences is an initial step before integration or mediation, and the nature of the datasets clearly impacts this matching quality. First, we describe our datasets. For all of them, the expert set of correspondences has been manually expertised. Then, we propose a classification of these datasets based on the datasets properties and the schema matching features.

Here are the available datasets in our benchmark, which are available online[4]

- **Betting** and **finance** datasets. Each of them contains tens of web forms, extracted from various websites [34]. As explained by authors of [46], schema matching is often a process which evaluates the costs (in terms of resources and money) of

[4]`http://liris.cnrs.fr/~fduchate/research/tools/xbenchmatch/`

a project, thus indicating its feasibility. Our *betting* and *finance* datasets can be a basis for project planning, i.e., to help users decide if integrating their data sources is worth or not.

- **Biology dataset**. The two large schemas originate from different biology collections which describe proteins, namely UniProt[5] and GeneCards[6]. The UniProt schema contains 896 elements and the GeneCards schema has more than 2530 elements. This is an interesting dataset for deriving a common specific vocabulary from different data sources which have been designed by human experts. A third source such as ProSite[7] could be used as external resource for facilitating the matching.
- **Currency** and **sms** datasets are popular web services[8]. Matching the schemas extracted from web services is a recent challenge to build new applications such as mashups or to automatically compose web services.
- **Order dataset** deals with business. The first schema is drawn from the XCBL collection[9], and it includes about 850 elements. The second schema (from OAGI collection[10]) also describes an order but it is smaller with only 20 elements. This dataset reflects a real-case scenario in which a repository of schemas exist (similar to our large schema) and the users would like to know whether a new schema (the small one in our case) is still necessary. It can also be used in a context where users need to determine the overlap between schemas, i.e., whether a schema or subset of a schema can be reused from the repository.
- **Person dataset** contains two small-sized schemas describing a person with low heterogeneity. On the Web, many applications only need small schemas to store the data of specific concepts (e.g., customers, products).
- **Travel dataset** includes 20 schemas that have been extracted from airfare web forms [1]. In data sharing systems, partners have to choose a schema or a subset of schema that will be used as a basis for exchanging information. This *travel* dataset clearly reflects this need, since schema matching enables data sharing partners to identify similar concepts that they are willing to share.
- **University courses dataset**. These 40 schemas have been taken from Thalia collection presented in [29]. Each schema has about 20 elements and they describe the courses offered by some worldwide universities. As explained in [46], this dataset could refer to a scenario where users need to generate an exchange schema between various data sources.
- **University department dataset** describes university departments and it has been widely used in the literature [16]. These two small schemas have very heterogeneous labels.

According to their descriptions, it is clear that these datasets either have different criteria or fulfil various schema matching tasks. These features are summed up in Table 1 and have been classified into five categories represented by the gray columns. The first four categories deals with the properties of datasets. The column *Label heterogeneity* is computed thanks to terminological similarity measures applied to the expert set of correspondences. If these measures are able to discover most of the correspondences, this means that the labels have a very low heterogeneity. Conversely, if the terminological measures only discover a few correspondences, then the labels are strongly heterogeneous. The second column *Domain specific* means that the vocabulary is uncommon and it cannot be found in general dictionaries such as Wordnet [47]. The *size* column indicates the average number of schema elements in a dataset while the *structure* column checks how deep the schema elements are nested. Unfortunately, we were not able to obtain very nested schemas (depth $> 7$). The latter category provides information about the number of schemas in the dataset. This is an important feature because of the growing information stored on the Internet and in distributed organizations [44]. Other schema matching features could have been added, for instance *use of external resources* (e.g., domain ontology), *complex correspondences*, *use of instances*, *evolution of schemas*, etc. But they would require the corresponding schema matching datasets, which involves a demanding effort to create the ground truth. However, we intend to add more datasets that reflect such interesting features by semi-automatically discovering the correspondences with a schema matching tool. Using this classification of the datasets in our benchmark enables a better understanding of the matching tools' successes and failures.

## 6 Quality of Matching

The schema matching community evaluates the results produced by its tools using common metrics, namely precision, recall and F-measure [6]. However, the aim

[5]`http://www.uniprot.org/docs/uniprot.xsd`
[6]`http://www.geneontology.org`
[7]`http://prosite.expasy.org/`
[8]`http://free-web-services.com/`
[9]`http://www.xcbl.org`
[10]`http://www.oagi.org`

| | Label heterogeneity | | | Domain | Average size | | | Structure | | Number |
|---|---|---|---|---|---|---|---|---|---|---|
| | Low or normalized | Average | High | Specific | Small (<10) | Average (10-100) | Large (>100) | Flat | Nested (3<depth<7) | of schemas |
| **Betting** | | x | | | | x | | x | | 12 |
| **Biology** | | x | | x | | | x | | x | 2 |
| **Currency** | | x | | | | x | | | x | 2 |
| **Finance** | | x | | x | | x | | x | | 8 |
| **Order** | x | | | | | | x | | x | 2 |
| **Person** | x | | | | x | | | | x | 2 |
| **Sms** | | x | | | | x | | | x | 2 |
| **Travel** | | x | | | x | | | x | | 20 |
| **Univ. courses** | | x | | | | x | | x | | 40 |
| **Univ. dept** | | | x | | x | | | x | | 2 |

**Table 1: Datasets classification according to their properties**

of schema matching is to avoid a manual, fastidious and error-prone process by automating the discovery of correspondences. Therefore, the post-match effort, which consists of checking these discovered correspondences, should be reduced at most. Yet, the overall is the only metric which computes this effort, but it is not sufficiently realistic to reflect it [36]. Indeed, this metric implies that validating a discovered correspondence consumes as much resources as searching for a missing correspondence. Furthermore, schema matching tools may produce an integrated schema (using the set of correspondences). To the best of our knowledge, there are only a few metrics in charge of assessing the **quality of an integrated schema**, which do not take into account the structure of the produced schemas [10]. Consequently, we present new metrics to complete the evaluation of the quality for a set of correspondences and an integrated schema.

## 6.1 Quality of Matching

As the matching process mainly aims at helping users saving both time and resources, it is interesting to measure the gain of using a matching tool. Thus, we present an alternative solution to the overall metric for computing the **post-match effort**, i.e., a maximum estimation of the amount of work that the user must provide to check the correspondences that have been discovered by the tool and to complete them [19]. Furthermore, inverting the post-match effort enables us to measure the **human-spared resources**, i.e., the rate of automation that has been gained by using a matching tool.

When a set of correspondences is generated by a tool, users first have to check each correspondence from the set, either to validate it or to remove it. Then, they have to browse the schemas and discover the missing correspondences. Thus, we propose to evaluate this user post-match effort by **estimating the number of user interactions** to reach a 100% F-measure, i.e., to correct the two previously mentioned issues. A user interaction is an (in)validation of one pair of schema elements (either from the set of discovered correspondences or between the schemas).

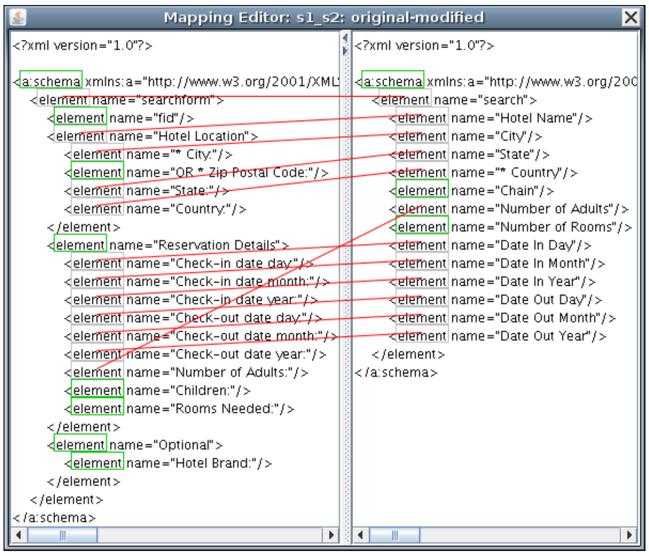
Here are our assumptions which underlie our metric:

- All pairs of schema elements, which have not already been matched, must be (in)validated. Besides, the last pair to be validated would be a correspondence (worst-case scenario).
- Missing correspondences are discovered with the same frequency, to enable a comparison of the quality at different times (uniformity).
- Only **correspondences 1:1** are taken into account. The metric can be applied to *1:n correspondences* (represented by several 1:1 correspondences), but we do not consider more complex correspondences (namely *n:m*).

Now, let us introduce an example. Figure 2 depicts a set of correspondences discovered by the matching tool Rondo between two hotel booking schemas. The expert set of correspondences is shown in Figure 3. By comparing the correspondences in Figures 2 and 3, we notice that one discovered correspondence is incorrect (*¡Hotel Location, Hotel Name¿*). Consequently, it has to be invalidated. Besides, the matching tool missed two correspondences, namely *¡Hotel Brand:, Chain¿* and *¡Rooms Needed:, Number of Rooms¿*. These two correspondences have to be searched among the 23 pairs that have not been validated ($8 \times 3$ possible remaining pairs minus 1 incorrect pair discovered by the tool).

## 6.2 Computing the Number of User Interactions (NUI)

In the following, we present the different parameters that are involved in the post-match effort. Given two schemas $S_\ell$ and $S_L$ of respective sizes $|S_\ell|$ and $|S_L|$, with $|S_\ell| \leq |S_L|$ (i.e., $S_L$ is a larger schema than $S_\ell$), their expert set of correspondences $E$ contains $|E|$ correspondences. A matching tool applied against these schemas has discovered a set of correspondences $M$,


Mapping Editor: s1_s2: original-modified

```
<?xml version="1.0"?>

<a:schema xmlns:a="http://www.w3.org/2001/XML
  <element name="searchform">
    <element name="fid"/>
    <element name="Hotel Location">
      <element name="* City:"/>
      <element name="OR * Zip Postal Code:"/>
      <element name="State:"/>
      <element name="Country:"/>
    </element>
    <element name="Reservation Details">
      <element name="Check-in date day:"/>
      <element name="Check-in date month:"/>
      <element name="Check-in date year:"/>
      <element name="Check-out date day:"/>
      <element name="Check-out date month:"/>
      <element name="Check-out date year:"/>
      <element name="Number of Adults:"/>
      <element name="Children:"/>
      <element name="Rooms Needed:"/>
    </element>
    <element name="Optional">
      <element name="Hotel Brand:"/>
    </element>
  </element>
</a:schema>
```

```
<?xml version="1.0"?>

<a:schema xmlns:a="http://www.w3.org/200
  <element name="search">
    <element name="Hotel Name"/>
    <element name="City"/>
    <element name="State"/>
    <element name="* Country"/>
    <element name="Chain"/>
    <element name="Number of Adults"/>
    <element name="Number of Rooms"/>
    <element name="Date In Day"/>
    <element name="Date In Month"/>
    <element name="Date In Year"/>
    <element name="Date Out Day"/>
    <element name="Date Out Month"/>
    <element name="Date Out Year"/>
  </element>
</a:schema>
```

**Figure 2: Correspondences discovered by a schema matcher (Rondo)**

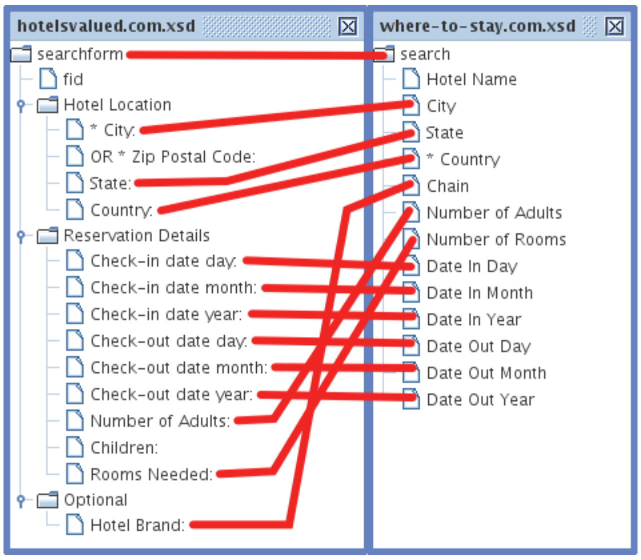


**Figure 3: Expert correspondences between the two hotel booking schemas**

which contains $|M|$ correspondences. Among these discovered correspondences, $|R|$ of them are correct, with $0 \leq |R| \leq |M|$.

$|S_\ell|, |S_L|, |E|, |M| and |R|$ are the five inputs required to compute the number of user interactions. In our hotel booking example, we have the following values:

- $|S_\ell| = 14$, the number of elements in the smallest schema[11].
- $|S_L| = 19$, the number of elements in the largest schema[11].
- $|E| = 13$, the number of given expert correspondences, shown in Figure 3.
- $|M| = 12$, the number of correspondences discovered by the matching tool, shown in Figure 2.
- $|R| = 11$, the number of correct correspondences discovered by the matching tool.

The post-match evaluation process can be divided into three phases.

**Phase 1: checking of all discovered correspondences.** This step is very easy to compute. A user has to check each correspondence from the set of discovered correspondences, and (in)validate it. Thus, this requires a number of interactions equal to the number of discovered correspondences in the set, $|M|$ in our case. We call this metric *effort*$_{prec}$ since it is directly impacted by precision. Indeed, a high precision reduces the number of user interactions since there are fewer incorrect correspondences which have been discovered. Note that at the end of this step, precision is equal to 100%.

$$\text{effort}_{prec} = |M| \tag{1}$$

In our example, there are 12 discovered correspondences, thus effort$_{prec}$ = 12. It means that the number of user interactions during this step is equal to 12, among which 11 validations and 1 invalidation for the incorrect correspondence.

**Phase 2: manual discovery of missed correspondences.** The second step deals with the manual discovery of all missing correspondences. At the end of this step, recall reaches 100%, and as a consequence F-measure does too. We assume that all pairs which have not been invalidated yet must be analysed by the user. As we consider only 1:1 and 1:n correspondences, elements that have already been matched (during phase 1) are not checked anymore. The main idea is to check every unmatched element from the smallest schema against all unmatched elements from the largest schema.

Due to the uniformity assumption, we manually discover a missing correspondence with the same frequency. This frequency is computed by dividing the number of unmatched elements in the smallest schema by the number of missing correspondences, as shown by Formula 2. Due to 1:1 correspondences assumption, the number of correct correspondences $|R|$ is equal to the number of correctly matched elements in each schema.

$$\text{freq} = \frac{|S_\ell| - |R|}{|E| - |R|} \tag{2}$$

[11]We do not count the root element tagged with <*a:schema*>.

Back to our example, freq $= \frac{14-11}{13-11} = \frac{3}{2}$ means that the user will manually find a missing correspondence for every three-halves unmatched elements from the smallest schema.

Since we now know the frequency, we can compute the number of interactions using a sum function. We call this metric effort$_{rec}$ since it is affected by recall. The higher recall you achieved, the fewer interactions you require during this step. $|S_L| - |R|$ denotes the number of unmatched elements from the largest schema. With $i$ standing for the analysis of the $i^{th}$ unmatched element from $S_\ell$, $\frac{i}{\text{freq}}$ represents the discovery of a missing correspondence (when it reaches 1). Finally, we also uniformly remove the pairs which may have been already invalidated during step 1, by computing $\frac{|M|-|R|}{|S_\ell|-|R|}$. Thus, we obtain this Formula 3:

$$\text{effort}_{rec} = \sum_{i=1}^{|S_\ell|-|R|} (|S_L| - |R| - \frac{i}{\text{freq}} - \frac{|M| - |R|}{|S_\ell| - |R|}) \quad (3)$$

We now detail for our example the successive iterations of this sum function, which vary from 1 to 3.

- $\textit{effort}_{rec}(i = 1)$, $19 - 11 - \frac{1}{1.5} - \frac{1}{3} = 7$
- $\textit{effort}_{rec}(i = 2)$, $19 - 11 - \frac{2}{1.5} - \frac{1}{3} = 6\frac{1}{3}$
- $\textit{effort}_{rec}(i = 3)$, $19 - 11 - \frac{3}{1.5} - \frac{1}{3} = 5\frac{2}{3}$

Thus, the second step to discover all missing correspondences requires $\textit{effort}_{rec} = 7 + 6\frac{1}{3} + 5\frac{2}{3} = 19$ user interactions.

**Phase 3: computing NUI.** Finally, to compute the number of user interactions between two schemas $S_\ell$ and $S_L$, noted *nui*, we need to sum the values of the two previous steps by applying Formula 4. Note that if the set of correspondences $|M|$ is empty, then using a matching tool was useless and the number of user interactions is equal to the total number of pairs between the schemas.

$$\text{nui}(S_\ell, S_L) = \begin{cases} |S_\ell| \times |S_L| & if |M| = 0 \\ \text{effort}_{prec} + \text{effort}_{rec} & otherwise \end{cases} \quad (4)$$

## 6.3 Computing the Post-match Effort

The number of user interactions is not sufficient to measure the benefit of using a matching tool because the size of the schemas should be taken into account. Thus, the *post-match effort* is a normalization of the number of user interactions. In addition, we derive the number of user interactions into a *human spared resources* measure that calculates the rate of automation by a schema matching tool.

**Post-match effort (PME).** From the number of user interactions, we can normalize the post-match effort value into [0,1]. It is given by Formula 5. Indeed, we know the number of possible pairs ($|S_\ell| \times |S_L|$). Checking all these pairs means that the user performs a manual matching, i.e., nui $= |S_\ell| \times |S_L|$ and pme $= 100\%$.

$$\text{pme}(S_\ell, S_L) = \frac{\text{nui}(S_\ell, S_L)}{|S_\ell| \times |S_L|} \quad (5)$$

We notice that the post-match effort cannot be equal to 0%, although F-measure is 100%. Indeed, the user must at least (in)validate all discovered correspondences, thus requiring a number of interactions. Then, (s)he also has to check if no correspondence has been forgotten by analysing every unmatched element from input schemas. This is realistic since we never know in advance the number of correct correspondences.

**Human spared resources (HSR).** We can also compute the percentage of automation of the matching process thanks to a matching tool. This measure, noted *hsr*, for human spared resources, is given by Formula 6. If a matching tool achieves a 20% post-match effort, this means that the user has to perform a 20% manual matching for removing and adding correspondences, w.r.t. a complete (100%) manual matching. Consequently, we can deduce that the matching tool managed to automate 80% of the matching process.

$$\text{hsr}(S_\ell, S_L) = 1 - \frac{\text{nui}(S_\ell, S_L)}{|S_\ell| \times |S_L|} = 1 - \text{pme}(S_\ell, S_L) \quad (6)$$

In our dating example, the post-match effort is equal to pme $= \frac{31}{14\times19} \simeq 12\%$ and human spared resources is equal to hsr $= 1 - 0.12 \simeq 88\%$. The matching tool has spared 88% resources of the user, who still has to manually perform 12% of the matching process.

## 6.4 Discussion

The **overall** measure [36] was specifically designed to compute the post-match effort using the formula: overall $=$ recall $\times (2 - \frac{1}{\text{precision}})$. However, it entails a major drawback since it assumes that validating discovered correspondences requires as much effort as searching for missed ones. Another drawback explained by the authors deals with a precision below 50%: it implies more effort from the user to remove extra correspondences and add missing ones than to manually perform the matching, thus resulting in a negative overall value which is often disregarded. On the contrary, our measure returns values in the range $[0, 1]$ and it does not assume that a low precision involves much effort during post-match because of the number of user interaction estimation. Finally, the overall measure does not consider the size of the schemas. Yet, even with the same number of expert correspondences, the manual task for checking the discovered correspondences and finding the missed correspondences in two large schemas requires a larger effort than in smaller ones. An extensive comparison between overall and post-match effort can be read in [19].

We now discuss several points about the post-match metric. Our metric does not take into account the fact that some schema matching tools [36, 3] returns the top-K correspondences for a given element. By proposing several correspondences, the discovery of missing correspondences is made easier when the correct correspondence appears in the top-K. Note that without the 1:1 correspondence assumption (i.e, in case of n:m correspondences), formula of the number of user interactions is reduced to $\text{nui}(S_1, S_2) = |S_1| \times |S_2|$. Indeed, as complex correspondences can be found, all elements from one schema have to be compared with all elements from the other schema. However, many schema matching tools do not produce such complex correspondences [43]. At best, they represent 1:n complex correspondences using two or more 1:1 correspondences. During the second step of the post-match effort, F-measure has the same value distribution for all matching tools (since precision equals $100\%$ and only recall can be improved). This facilitates a comparison between matching tools for a given dataset. We claim that recall is more important than precision. Indeed, it enables a better reducing of the post-match effort, since (in)validating discovered correspondences requires less user interactions than searching missed correspondences.

### 6.5 Quality of Integrated Schemas

As matching tools can also generate an integrated schema, we describe several metrics to assess their quality. In our context, we have an integrated schema produced by a matching tool and a reference integrated schema. This reference integrated schema can be: (i) provided by an expert, thus indicating that the tool integrated schema should be at most similar to this reference integrated schema; (ii) a repository of schemas, that many organizations maintain, in which case we can compare the overlap between the tool integrated schema and this repository; or (iii) one of the input schemas, which means that we would like to know how similar the tool integrated schema is related to one of the input schemas. This reference integrated schema does not necessarily take into account the domain application, user requirements and other possible constraints, so several reference integrated schemas could be considered as valid [8].

In [10], authors define two measures for integrated schema w.r.t. data sources. First, **completeness** represents the proportion of elements in the tool integrated schema which are common with the reference integrated schema. Secondly, **minimality** is the percentage of extra elements in the tool integrated schema w.r.t. the reference integrated schema. Both metrics are in the range $[0, 1]$, with a $1$ value meaning that the tool integrated schema is totally complete (respectively minimal) w.r.t. to the reference integrated schema.

As stated by Kesh [31], these metrics are crucial to produce a more efficient schema, i.e. that reduces query execution time. However, they do not measure the quality of the structure of the produced integrated schema. We believe that the structure of an integrated schema produced by a schema matching tool may also decrease schema efficiency if it is badly built. Besides, an integrated schema that mostly keeps the semantics of the source schemas has a better understanding for an end-user. Thus, we have completed the two previous metrics by another one which evaluates the **structurality** of an integrated schema [18]. We mainly check that each element of the tool integrated schema shares the same structure than its corresponding element in the reference integrated schema. By same structure, we mean that an element in both integrated schemas shares the maximum number of common ancestors, and that no extra ancestor have been added in the tool integrated schema.

These three metrics, namely completeness, minimality and structurality, are finally averaged to evaluate the **schema proximity** between two integrated schemas, which returns values in the range $[0, 1]$ [18].

As all these metrics have been implemented in our benchmark, we have run different experiments with three schema matching tools to understand their results.

## 7 Experiments Report

In this section, we present the evaluation results of the following matching tools: COMA++ [14, 3], Similarity Flooding (SF) [36, 37] and YAM [21, 22]. These tools are described in Section 3.3. The default configuration for SF was used in the experiments. We have tested the three pre-configured strategies of COMA++ (*AllContext*, *FilteredContext* and *NoContext* in the version *2005b*) and we kept the best score among the three. YAM includes a machine learning process in charge of tuning its parameters, but each score is an average of 200 runs to reduce the impact of the randomly-chosen training data. When dealing with a dataset with more than two schemas, all schemas are matched two by two. Consequently, the quality value is global, i.e., equal to the average of all individual quality scores obtained for all possible pairs of schemas. The three schema matching tools were evaluated using the datasets[12] presented in Section 5. All experiments were run on a 3.0 Ghz laptop with 2G RAM. We first discuss the matching quality of these tools for each dataset from our benchmark. Then, we evaluate their time performance.

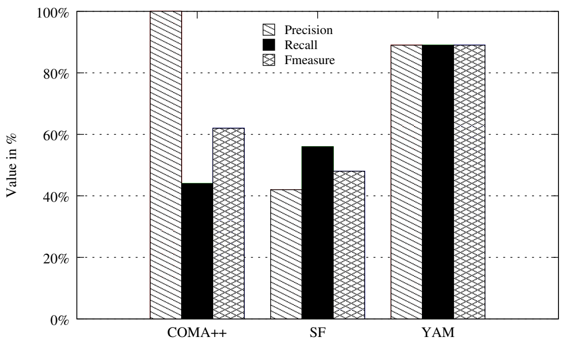
## 7.1 Matching Quality Evaluation

We report the results achieved by each schema matching tool over 10 datasets, with three plots per dataset. The first plot depicts the common metrics: precision, recall and F-measure (see Section 2). The second plot enables a comparison of our *human spared resources* measure (HSR, the inverse of the post-match effort, see Section 6) with F-measure and **overall**[13]. Finally, the last plot shows the quality of the integrated schema[14] with computed values for completeness, minimality, structurality and schema proximity. We conclude this section by a general discussion about the experimental results.

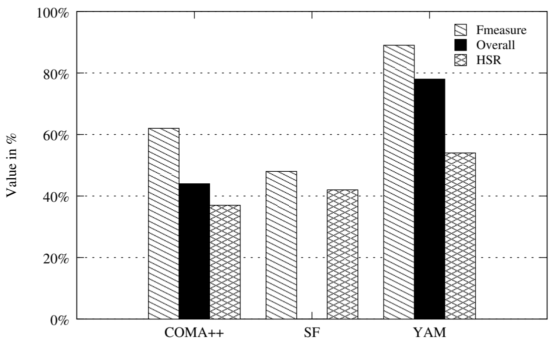
### 7.1.1 Betting dataset

Figures 4(a) and 4(b) depict the matching quality for the *betting* dataset, which features flat schemas from the web. COMA++ obtains the highest precision but it discovers fewer correct correspondences than SF. According to their HSR values, both tools manage to spare around $40\%$ of user resources. We also note that SF's overall is negative because of its precision below $50\%$. Yet, it was able to discover half of the correct correspondences, meaning that the human post-match effort is reduced. YAM achieves a $87\%$ F-measure. Besides, YAM's overall is quite high ($78\%$) and its HSR is lower ($54\%$). The reason is that half of the elements in the schemas do not have a correspondence. Yet, the overall metric does not take into account these unmatched elements while our HSR metric considers that a human expert has to check them, thus increasing the post-match effort and reducing HSR.

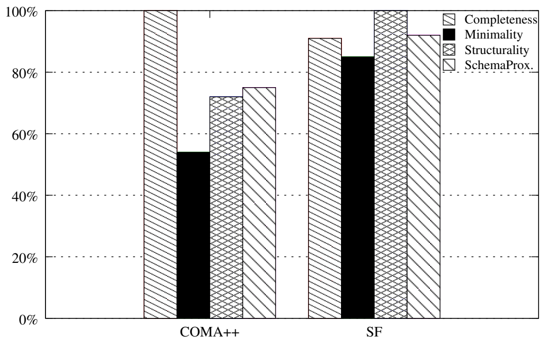
For the quality of integrated schemas, shown by Figure 4(c), COMA++ successfully encompasses all concepts ($100\%$ completeness) while SF produces the same structure than the expert ($100\%$ structurality). Both tools did not achieve a minimal integrated schema, i.e., without redundancies. SF generates the most similar integrated schema w.r.t. the expert one (schema proximity equal to $92\%$).



(a) Correspondence (precision, recall, F-measure)



(b) Correspondence (F-measure, overall, HSR)



(c) Integrated schema

**Figure 4: Quality obtained for the *betting* dataset**

### 7.1.2 Biology dataset

With this large scale and domain-specific dataset, we notice on Figures 5(a) and 5(b) that the three schema matching tools have poorly performed for discovering correspondences (less than $10\%$ F-measure). Thus, their overall values are all negatives and therefore considered insignificant. But using these schema matching tools enables users to spare a few resources (HSR around $5\%$). These poor results might be explained by the fact that no external resource such as a domain ontology was provided. An external resource contains knowledge

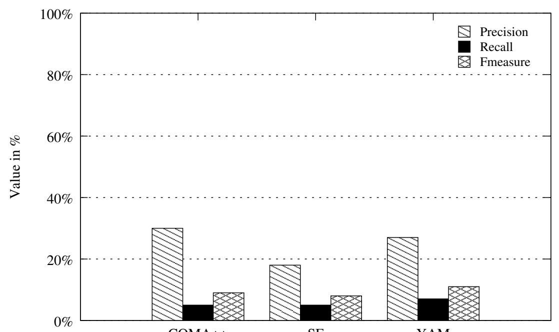
[12]`http://liris.cnrs.fr/~fduchate/research/tools/xbenchmatch/`

[13]Overall values have been limited to 0 instead of $-\infty$. One should consider a negative overall value as not significant as explained in [36].

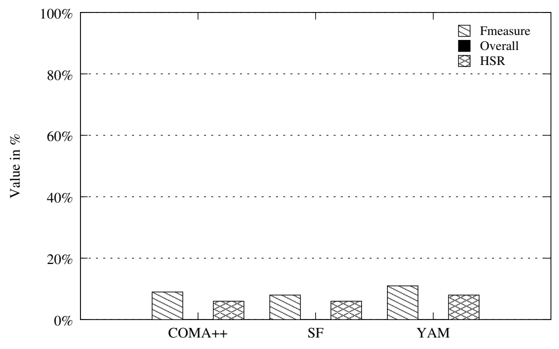
[14]Only for COMA++ and Similarity Flooding, since YAM does not build an integrated schema.

which can be extracted and exploited to facilitate schema matching.

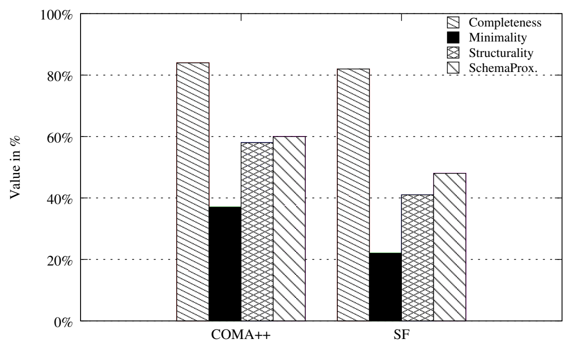
However, as shown by Figure 5(c), the tools were able to build integrated schemas with acceptable completeness (superior to 80%) but many redundancies (minimality inferior to 40%) and different structures (58% and 41% structurality values). These scores can be explained by the failure for discovering correct correspondences. As a consequence, many schema elements have been added into the integrated schemas, including redundant elements. For structurality, we believe that for unmatched elements, the schema matching tools have copied the same structure than the one of the input schemas.



(a) Correspondence (precision, recall, F-measure)



(b) Correspondence (F-measure, overall, HSR)



(c) Integrated schema

**Figure 5: Quality obtained for the *biology* dataset**

### 7.1.3 Currency dataset

Figures 6(a) and 6(b) depict the quality obtained for *currency*, a nested average-sized dataset. COMA++ discovered more correct correspondences than SF and YAM (respective F-measures of 60%, 33% and 52%). However, YAM achieves the highest recall. Dealing with the post-match effort, SF and YAM have a negative overall value, which is not realistic. Indeed, our HSR measure shows that the matching tools spared between 16% and 34% human resources.

On Figure 6(c), we can observe the quality of the integrated schemas built by COMA++ and SF. This last tool manages to build a more similar integrated schema (83% schema proximity against 62% for COMA++). Although both tools have a 100% completeness, COMA++ avoids more redundancies (due to a better recall) while SF respects more the schema structure.

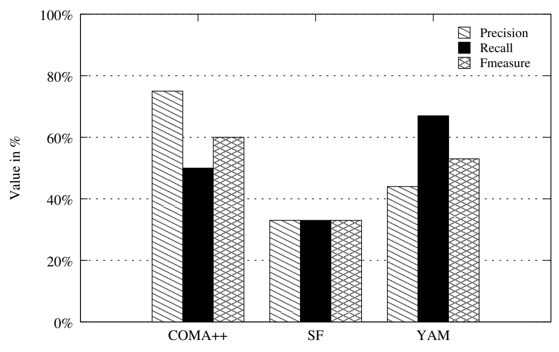
### 7.1.4 Finance dataset

Based on Figures 7(a) and 7(b), we analyse the quality of the correspondences for the *finance* domain. For this dataset with a specific vocabulary, we notice that COMA++ and SF only discovers a few correct correspondences (respectively 18% and 45%), probably because of the specific vocabulary of *finance*. But COMA++ achieves a 100% precision. YAM slightly obtains better results with a F-measure equal to 56%. The interesting point with this dataset deals with the results of SF and YAM. YAM obtains a far better precision than SF, while this last manages to discover one more correct correspondence than YAM. Although YAM's F-measure is 15% better than SF's, their HSR is the same (41%). With SF, the user has to process around 40% more invalidations, but these extra invalidations are compensated by the discovery of one more correct correspondence (that the YAM user has to manually find thanks to (in)validations). In other words, the 40% extra precision obtained by YAM over SF are equivalent, for this dataset, to the 4% extra recall obtained by SF over YAM, in terms of user (in)validations. This clearly shows that promoting recall instead of precision enables a better reducing of the post-match effort in most cases.

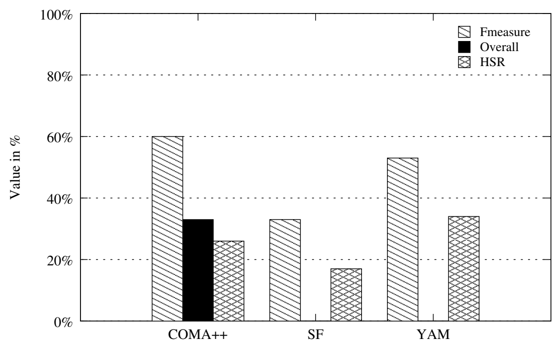
Figure 7(c) depicts the quality of the integrated schema for the finance dataset. The experimental results indicate that both tools produced an integrated schema which is 80% similar with the expert one. They mainly achieve a high completeness.

### 7.1.5 Order dataset

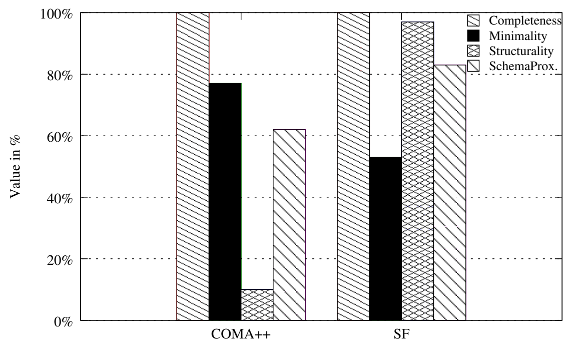
This experiment deals with large schemas whose labels are normalized. Similarly to the other large scale scenario, schema matching tools do not perform well for this



(a) Correspondence (precision, recall, F-measure)

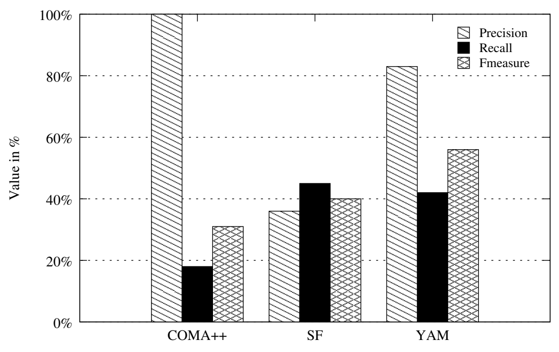


(b) Correspondence (F-measure, overall, HSR)

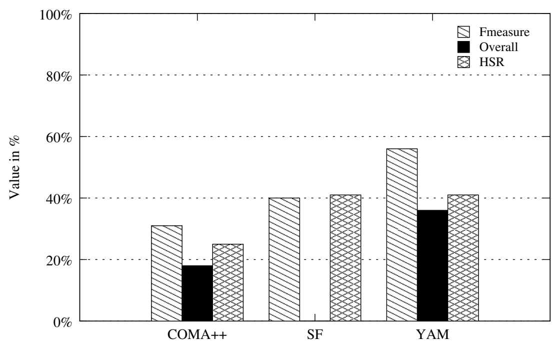


(c) Integrated schema

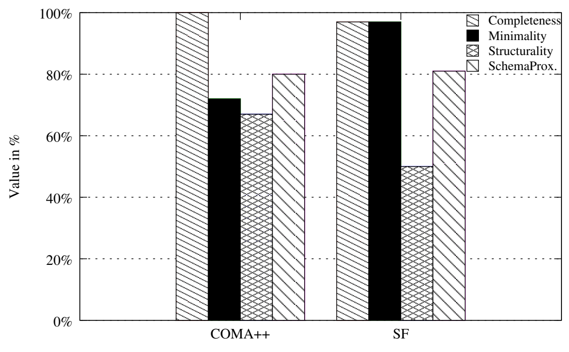
**Figure 6: Quality obtained for the *currency* dataset**



(a) Correspondence (precision, recall, F-measure)



(b) Correspondence (F-measure, overall, HSR)



(c) Integrated schema

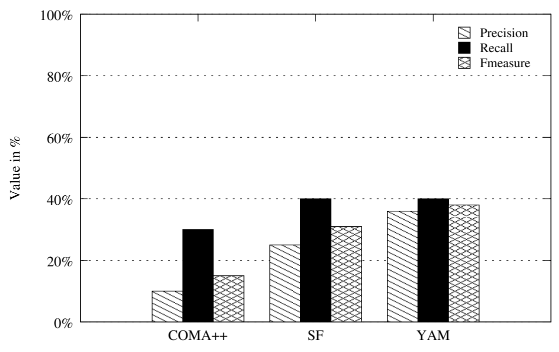
**Figure 7: Quality obtained for the *finance* dataset**

*order* dataset (F-measures less than 40%), as depicted by Figures 8(a) and 8(b). Contrary to most datasets, the tools obtain higher recall values than precision values. The normalized labels of the schema elements might explain why their precisions are so low. Although they discover 30% to 40% of the correct correspondences, their overall values are negative. However, our HSR measure indicates that using COMA++, SF or YAM respectively enables the sparing of 11%, 16% and 21% human resources.

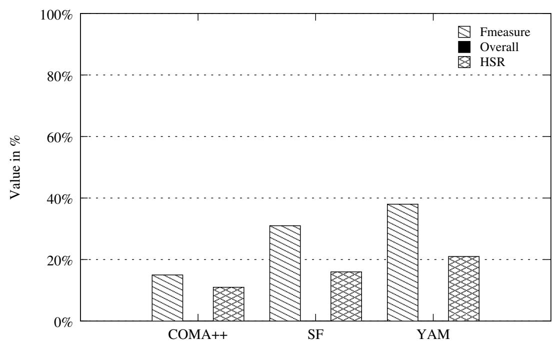
As for quality of the integrated schema, given by Figure 8(c), both tools achieve a schema proximity above 70%, mainly due to a high completeness. The reason for such high completeness value is the low quality results for correspondence discovery. In that case, users may be more interested by the results about structurality and minimality.

### 7.1.6 Person dataset

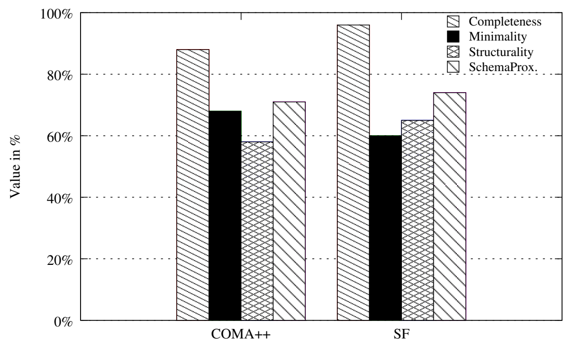
Figures 9(a) and 9(b) depict quality for the *person* dataset. With these small schemas featuring low heterogeneity in their labels, all matching tools achieve acceptable results (F-measure above 75%). With such good results, overall and HSR values are roughly identical (less than 5% difference). COMA++ achieves 86% precision



(a) Correspondence (precision, recall, F-measure)

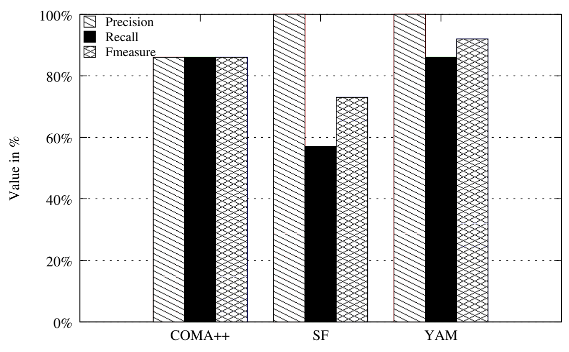


(b) Correspondence (F-measure, overall, HSR)

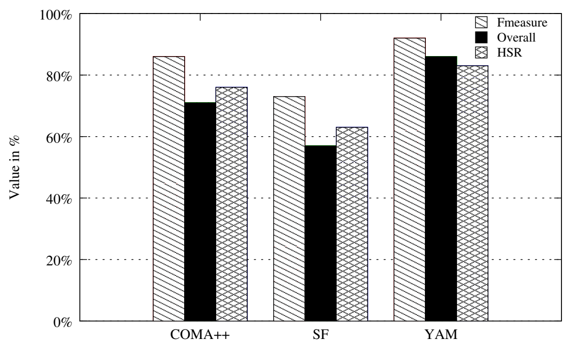


(c) Integrated schema

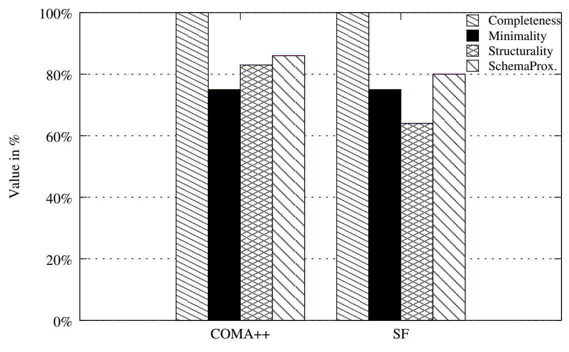
**Figure 8: Quality obtained for the *order* dataset**



(a) Correspondence (precision, recall, F-measure)



(b) Correspondence (F-measure, overall, HSR)



(c) Integrated schema

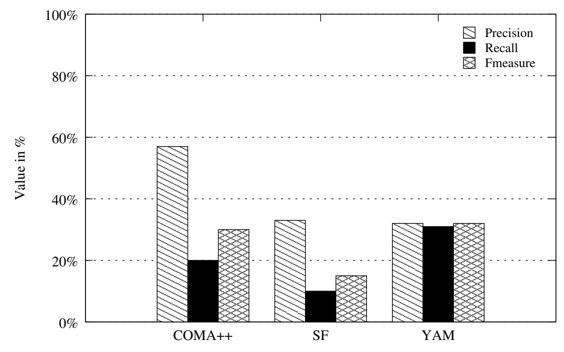
**Figure 9: Quality obtained for the *person* dataset**

and recall. SF only discovers correct correspondences (100% precision), but it finds roughly half of them (57% recall). YAM also obtains 100% precision, but with the same recall value as COMA++ (86%). Given these high results and the size of the input schemas, the post-match effort, either measured by overall or HSR, follows the same trend as F-measure.

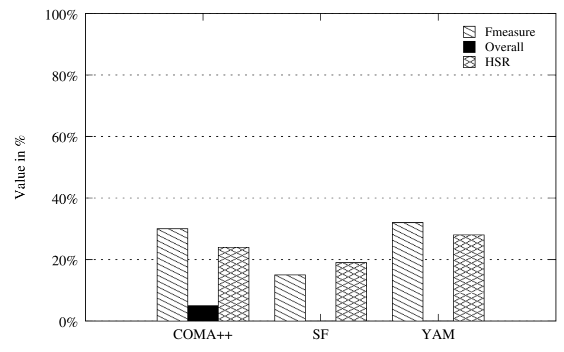
On the integrated schema plot (Figure 9(c)), we notice that both generated schemas are complete and they achieve the same minimality (76%). However, for this dataset containing nested schemas, COMA++ is able to respect a closer structurality than SF. The tools achieve a 80% schema proximity, mainly due to the good precision and recall that they achieve.

### 7.1.7 Sms dataset

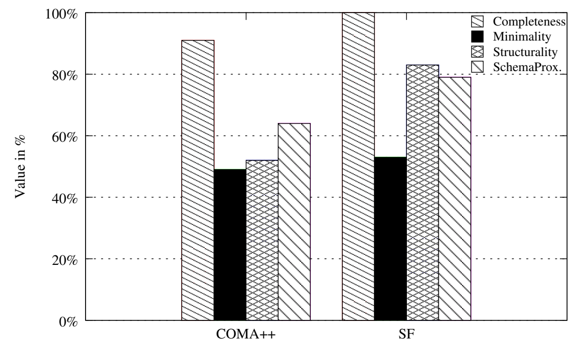
The *sms* dataset does not feature any specific criteria, but it is composed of web services. Figure 10(a) depicts the precision, recall and F-measure achieved by the tools. The experimental results indicate a low quality for discovering correspondences (all F-measures below 30%). This is probably due to the numerous similar tokens shared by the labels of the schema elements. Once again, only COMA++, which has a precision above 50%,



(a) Correspondence (precision, recall, F-measure)



(b) Correspondence (F-measure, overall, HSR)



(c) Integrated schema

**Figure 10: Quality obtained for the *sms* dataset**

achieves a positive overall value (see Figure 10(b)). But our HSR metric shows that SF and YAM have spared as many human resources as COMA++. We also note that HSR values can be superior to F-measure (SF's results). As a trade-off between precision and recall, F-measure does not reflect the post-match effort since it does not consider the size of the schemas.

As they missed many correct correspondences, the integrated schemas produced by the tools have a minimality around 50%, as shown on figure 10(c). SF obtains better completeness and structurality than COMA++.

### 7.1.8 Travel dataset

Figures 11(a) and 11(b) depict the matching quality for *travel*, a dataset whose schemas come from web forms. A first comment deals with COMA++, which does not discover any correspondence (and consequently, it does not generate an integrated schema). Maybe the threshold applied to computed similarity values is too high, since the labels in this dataset are rather heterogeneous. This assumption is supported by the average precision obtained by the other tools. SF and YAM both discover the same number of relevant correspondences (recall value of 60%), but SF did not discover as many irrelevant ones as YAM (75% against 42%). They respectively achieve 67% and 48% F-measures, thus sparing roughly half of human resources (HSR equal to 52% and 43%).

As for the integrated schema, we notice on Figure 11(c) that SF produces an integrated schema quite similar to the expert one (schema proximity equal to 87%).

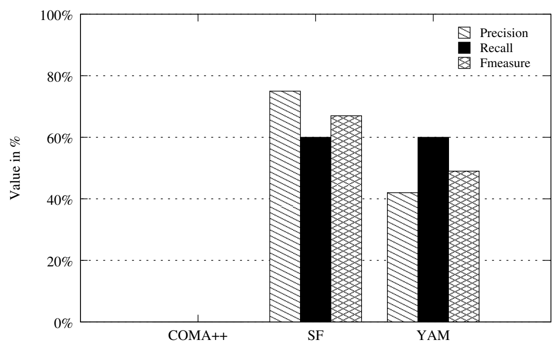
### 7.1.9 Univ-courses dataset

Figures 12(a) and 12(b) depict the matching quality of the *univ-courses* dataset, which contains flat and average-sized schemas. Both COMA++ and YAM obtain a high precision (100%) with an average recall (50% to 60%), thus resulting in F-measures around 70%. On the contrary, SF achieves the same precision and recall values (60%). Although the difference between their F-measures is not significant (10%), there is a large gap between overall values of YAM and COMA++ (50% to 60%) and the overall value for SF (20%). On the contrary, our HSR measure indicates that SF enables the sparing of as many resources as COMA++ and YAM (all HSR around 70%).

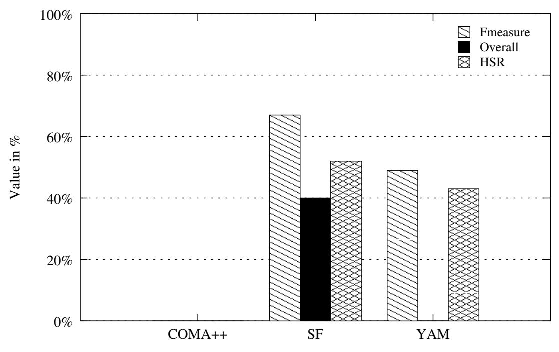
On Figure 12(c), the quality of COMA++ and SF's integrated schemas are evaluated. It appears that both tools produces an acceptable integrated schema w.r.t. the expert one (schema proximity equal to 94% for COMA++ and 83% for SF). Notably, COMA++ achieves a 100% completeness and 100% structurality.

### 7.1.10 Univ-dept dataset

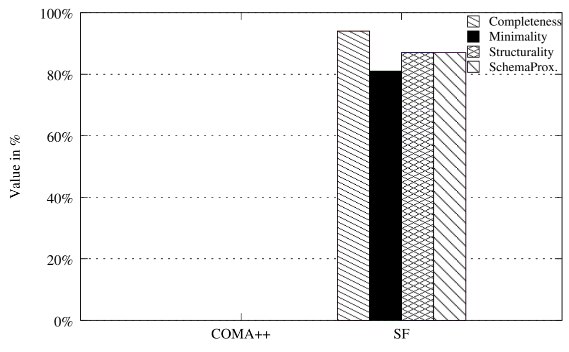
The last dataset, *univ-dept*, has been widely used in the literature. It provides small schemas with high heterogeneity for which a dictionary or a list of synonyms might be useful. The results of the three schema matching tools are shown on Figures 13(a) and 13(b). COMA++ obtains the highest F-measure, namely with a 100% precision. SF and YAM have the same score for F-measure (60%). But YAM achieves the lowest overall (10%) due to a precision value just above 50%. Yet, this



(a) Correspondence (precision, recall, F-measure)

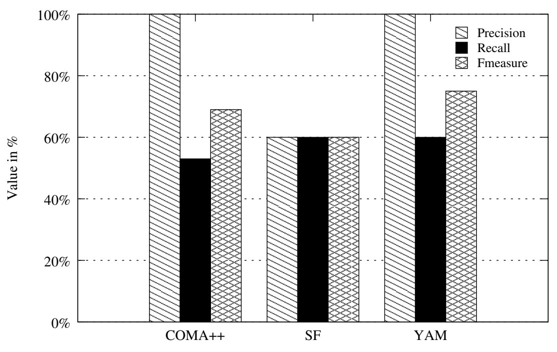


(b) Correspondence (F-measure, overall, HSR)

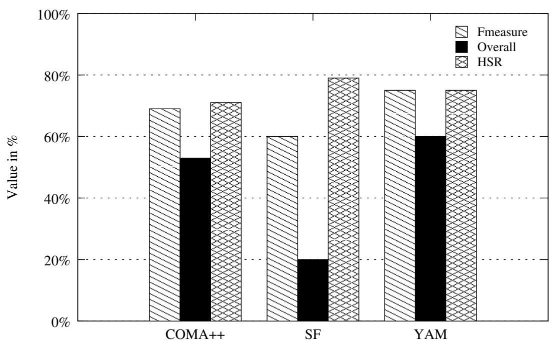


(c) Integrated schema

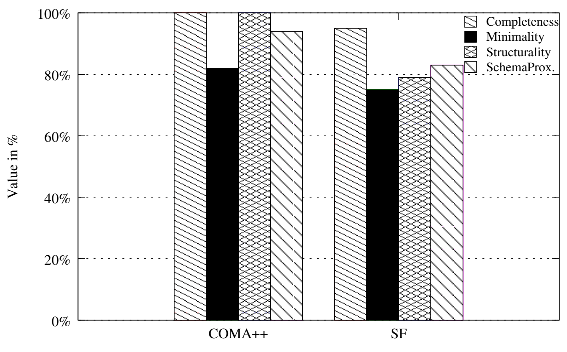
**Figure 11: Quality obtained for the *travel* dataset**



(a) Correspondence (precision, recall, F-measure)



(b) Correspondence (F-measure, overall, HSR)



(c) Integrated schema

**Figure 12: Quality obtained for the *univ-courses* dataset**

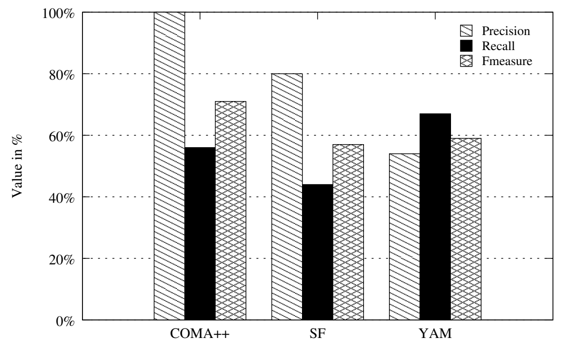
tool is the one that spares the most human resources according to HSR values (at least 20% more than the other tools) because it achieved the highest recall value.

For the integrated schemas, shown on Figure 13(c), both matching tools achieve acceptable completeness and structurality (all above 90%), but they have more difficulties to respect the minimality constraint, merely due to their average recall.

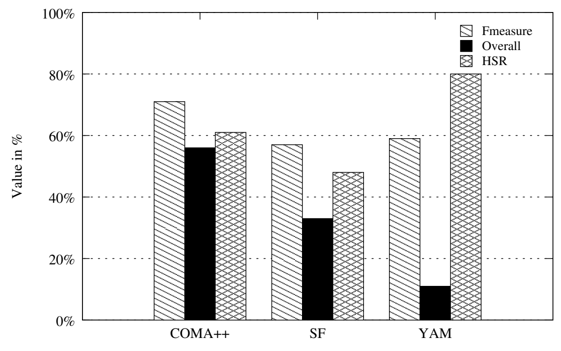
### 7.1.11 Discussion about the quality

To perform a general comparison, we have averaged the quality results of the three tools. Let us first discuss several points about the matching quality:

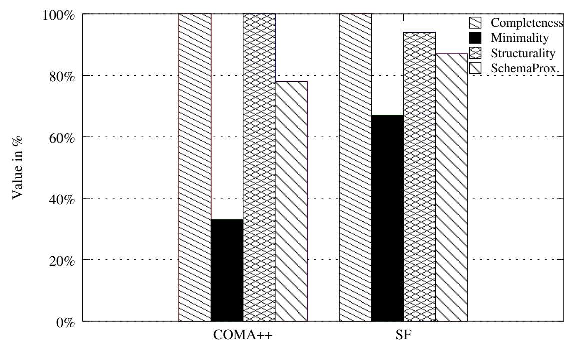
- COMA++ strongly favours precision (66% in average) to the detriment of recall (36%). SF obtains more balanced results between these measures (50% precision and 41% recall). On average, YAM achieves a precision equal to 61% and a recall equal to 55%. Average HSR is roughly similar for all



(a) Correspondence (precision, recall, F-measure)



(b) Correspondence (F-measure, overall, HSR)



(c) Integrated schema

**Figure 13: Quality obtained for the *univ-dept* dataset**

matching tools (35% for COMA++, 39% for SF and 47% for YAM).

- We notice that all matching tools only discover a few correct correspondences for large scale datasets (*order* and *biology*). Thus, the large scale challenge is far from being solved. However, we did not use any extension dedicated to large scale matching such as the fragment-oriented module for COMA++ [44, 15].
- Most of the evaluated tools mainly favour precision to the detriment of recall. Besides, this choice strongly impacts post-match effort. Indeed, given a schema matching scenario, it could be smarter to promote recall, for instance with large schemas.
- Our HSR measure is more optimistic than overall. When precision is below 50%, overall values are negatives. Yet, it does not mean that using the tool was a lack of time. HSR is somehow more correlated to recall than to precision. With a high precision and an average recall, HSR does not reach high values. This is due to the fact that a low recall implies more costly post-match effort from the user than precision does. Thus, HSR is a more balanced metric than overall, and probably more realistic as well [19].
- The obtained results should be carefully considered, because of the pre-match effort. Indeed, one of the tool (SF) was not tuned, the second one (COMA++) was run using three pre-configured strategies while the last tool (YAM) has an automatic tuning process. Therefore, it should be necessary to evaluate the impact of this pre-match effort on the quality results. In addition, these schema matching tools may not be only used by the database community, and the expertise for tuning a tool should be minimized. We also believe that a campaign such as OAEI for the ontology alignment domain is useful since the designers of a tool are best able to correctly tune its parameters to obtain the best results. Consequently, a similar campaign dedicated to schema matching could be challenging for our community.

Next, we draw some conclusions about the quality of integrated schemas:

- Average completeness (for all tools and all datasets) is equal to 91%. On the contrary, average minimality is 58% and average structurality reaches 68%. Indeed, schema matching tools mainly promote precision, thus they avoid the discovery of incorrect correspondences and they do not miss too many schema elements when building the integrated schema. On the contrary, a lower recall means that many similar schema elements are added in the integrated schema, thus reducing minimality.
- We also notice that it is possible to obtain a high minimality with a low recall, when precision is low too. Indeed, the low recall means that we have missed many correct correspondences, thus two similar elements could be added twice in the integrated schema. But with a low precision, there are many incorrect discovered correspondences, and only one of their elements would be added in the integrated schema. As an example, let us imagine that a correct correspondence between elements *A*

and *A'* is not discovered. Both *A* and *A'* are added in the integrated schema, unless one of them has been incorrectly matched to another element. This explains the high minimality achieved with some datasets, despite of a low recall.

- If a correct correspondence is missed by a matching tool, then both elements of this missed correspondence are added in the integrated schema. Structurality only takes into account one of these elements (the one which is in the expert integrated schema). The other is ignored, but it also penalizes minimality. This explains why structurality and completeness have high values even when correspondence quality measures return low values.
- Similarity Flooding provides a better quality for the integrated schema that it builds (79% average schema proximity against 67% for COMA++). Thus, schema proximity mainly computes high values, simply because it averages completeness and structurality values which are already high. For instance, when a few correct correspondences are discovered (*order* or *biology* datasets), many elements are added into integrated schema, thus ensuring a high completeness but a low minimality. Due to the missed correspondences, lots of elements have to be added into the integrated schema, and the easiest way is to keep the same structure that can be found in the source schemas.

## 7.2 Performance Evaluation

This section covers the time performance of the matching tools for discovering correspondences and presenting them to the user. Note that we did not include the time for generating the integrated schema. For COMA++, we measured the time spent by the tool to convert schemas and store them in the database. Indeed, other schema matching tools also have to do this process.

Table 2 presents the time performance of the three schema matching tools for all datasets. Here, we underline the fact that time performance during matching is nowadays not much significant, except in some specific environments, for instance large scale and highly dynamic. Indeed, matching time does not exceed one minute to match any of these datasets, whatever tool is used. Conversely, **the time during post-match effort is crucial**. When dealing with a dataset with more than two schemas, the performance is the average time or number of interactions for processing two schemas. Based on the estimated Number of User Interactions *NUI* (required to compute HSR or PME, see Section 6), we have converted the post-match effort into time. Indeed, we assume that the user needs 30 seconds to (in)validate a pair of elements. This assumption seems realistic by utilizing state-of-the-art GUI [9]. Except for some small datasets (e.g., *travel* or *univ-courses*), the manual post-match effort usually requires several hours in the worst case. Thus, measuring this effort (e.g., with PME) is necessary to improve it.

## 7.3 Concluding the Experiments Report

We finally discuss the results of these experiments by providing some advice about the studied schema matching tools according to the aim of the matching task.

If the goal of the matching process is to discover a subset of the relevant correspondences, then **COMA++** is the ideal tool because it achieves a high precision, i.e., it does not discover too many irrelevant correspondences. This feature is very important when one needs a quick estimation of the matching task or if a few relevant correspondences are required as training data. COMA++ is also more appropriate with Web interfaces datasets (except for *travel*). However, this tool should not be used with datasets containing normalized labels, since most labels share similar tokens that could mislead COMA++ if not tuned to handle them. If the use of the matching tool aims at generating an integrated schema, COMA++ will produce a complete schema with most datasets.

Due to its propagation mechanism, **Similarity Flooding** is able to discover relevant correspondences which are more complicated to detect for other tools. Since SF requires initial correspondences before running the propagation process, it could be used in combination to refine and improve the initial correspondences discovered by another tool such as COMA++. We notice that SF is the best choice when the datasets have normalized labels (*order* and *person*). However, our experiments indicate that SF is not suitable when dealing with flat schemas (lower results than the other tools), namely because it heavily relies on structural rules which cannot be applied in such case. For generating an integrated schema, SF promotes a very similar structure and it achieves a good schema proximity with the reference integrated schema.

Although **YAM** depends on the training data, it achieves the best matching quality. In addition, when domain schemas that have already been matched and verified are available, YAM can use them as training data to improve the quality results. A strong point of YAM is that it achieves the highest recall with regards to the other tools, which means that it discovers more of the correct correspondences than the other tools. However, the tool may require to be run several times to reduce the impact of random training data. The time performance may also decrease if the user selects all the classifiers from the library.

| | COMA++ | | | Similarity Flooding | | | YAM | | |
|---|---|---|---|---|---|---|---|---|---|
| | match | NUI | post-match | match | NUI | post-match | match | NUI | post-match |
| **Betting** | 1 sec | 362 | 3 hours | 1 s | 333 | 2.5 hours | 1 s | 264 | 2 hours |
| **Biology** | 44 sec | 36234 | 302 hours | 4 s | 36241 | 302 hours | 14 s | 35741 | 298 hours |
| **Currency** | 5 sec | 247 | 2 hours | 1 s | 272 | 2 hours | 1 s | 235 | 2 hours |
| **Finance** | 1 sec | 439 | 3.5 hours | 1 s | 360 | 3 hours | 1 s | 352 | 3 hours |
| **Order** | 43 sec | 11366 | 94.5 hours | 2 s | 10934 | 91 hours | 12 s | 10941 | 91 hours |
| **Person** | 1 s | 26 | 13 min | 1 s | 39 | 20 min | 1 s | 26 | 13 min |
| **Sms** | 19 s | 2074 | 17 hours | 2 s | 2233 | 19 hours | 3 s | 1960 | 16 hours |
| **Travel** | 1 sec | 78 | 39 min | 1 s | 47 | 23 min | 1 s | 47 | 23 min |
| **Univ. courses** | 1 sec | 67 | 33 min | 1 s | 56 | 28 min | 1 s | 37 | 18 min |
| **Univ. dept** | 1 s | 14 | 7 min | 1 s | 18 | 9 min | 1 s | 11 | 6 min |

**Table 2: Time performance on the different datasets for the three matching tools.**

Although the classification of the datasets enables us to draw some conclusions about the schema matching tools, it is still difficult to interpret these results because of the complexity of the schema matching task. Yet, it appears that a tool may be suitable for a given dataset, but totally useless for another one. Thus, flexibility for combining similarity measures is a crucial feature for the matchers since it could allow to significantly derive new correspondences, and thus improve the matching quality. The large datasets are still a challenging task for any tool: the matching process has been greatly reduced (below one minute for all tools), but the matching quality needs to be improved since the user effort is intensive during the post-match step (*biology* and *order*). The minimality criterion for integrated schemas is difficult to evaluate since it strongly depends on the recall value obtained during the matching.

## 8 Conclusion

In this paper, we have presented XBenchMatch, which enables the evaluation of schema matching tools. This benchmark integrates new measures for evaluating the post-match effort (NUI, PME and HSR) as well as the quality of an integrated schema (completeness, minimality, structurality and schema proximity). A collection of datasets, which tackle one or more specific schema matching issues, enables the schema matching community and end-users to test schema matching tools according to their requirements. Thus, XBenchMatch can produce an improved objective comparison about the quality and performance of a tool. The experiments demonstrate that XBenchMatch clearly helps users to select a matching tool which suits their needs. But it also shows that a new generation of schema matching tools is required, providing both flexibility and automation, namely for matching very large schemas. Furthermore, the post-match effort performed by the user is still time-consuming, so that new matching tools could focus on this challenge.

As future work, we should add more datasets to our benchmark, which fulfils other criteria such as complex correspondences, evolution or based on instances. With new datasets, we could also refine the classification of schema matching tools according to the dataset criteria for which they perform best. We should also let opportunity to use common external resources, for instance by providing a domain ontology for datasets like *biology*. More experiments should be performed to measure the impact of different selection strategies on the number of user interactions. In particular, the top-K strategy enables the sparing of interactions when the correct correspondence is among the K suggestions. Besides, we intend to quantify pre-match effort. Indeed, some tools may require the tuning of parameters or the selection of various options while other approaches are self-tuning. Measuring the pre-match effort would enable users to tune the tools and study the impact on the matching quality, the time performance and the automation rate of the tool according to the tuning of its inputs. Finally, we strongly advocate in favour of a campaign such as OAEI to identify schema matching issues and share experiences. The recent ”Interactive Track” in OAEI would be an interesting start as it requires new evaluation metrics to measure the number of interactions.

## Acknowledgements

We would like to thank Renée J. Miller for her valuable comments to improve this paper.

## References

[1] “The UIUC web integration repository,” Computer Science Department, University of Illinois at Urbana-Champaign. http://metaquerier.cs.uiuc.edu/repository, 2003.

[2] B. Alexe, W. C. Tan, and Y. Velegrakis, “STBenchmark: towards a Benchmark for Mapping Systems,” *Proceedings of the VLDB*, vol. 1, no. 1, pp. 230–244, 2008.

[3] D. Aumueller, H. H. Do, S. Massmann, and E. Rahm, “Schema and ontology matching with COMA++,” in *ACM SIGMOD*, 2005, pp. 906–908.

[4] P. Bailey, D. Hawking, and A. Krumpholz, “Toward meaningful test collections for information integration benchmarking,” in *Proceedings of II-Web 2006 (WWW Workshop)*, 2006. [Online]. Available: http://es.csiro.au/pubs/bailey_iiweb.pdf

[5] C. Batini, M. Lenzerini, and S. B. Navathe, “A Comparitive Analysis of Methodologies for Database Schema Integration.” *ACM Computing Surveys*, vol. 18, no. 4, pp. 323–364, 1986.

[6] Z. Bellahsene, A. Bonifati, and E. Rahm, *Schema Matching and Mapping*. Heidelberg: Springer-Verlag, 2011. [Online]. Available: http://www.springer.com/computer/book/978-3-642-16517-7

[7] A. Bonifati, Z. Bellahsene, F. Duchateau, and Y. Velegrakis, *On Evaluating Schema Matching and Mapping*. Data-Centric Systems and Applications, Springer, 2011. [Online]. Available: http://www.springer.com/computer/book/978-3-642-16517-7

[8] L. Chiticariu, P. G. Kolaitis, and L. Popa, “Interactive generation of integrated schemas,” in *SIGMOD ’08: Proceedings of the 2008 ACM SIGMOD international conference on Management of data*. New York, NY, USA: ACM, 2008, pp. 833–846.

[9] I. F. Cruz, W. Sunna, N. Makar, and S. Bathala, “A visual tool for ontology alignment to enable geospatial interoperability,” *J. Vis. Lang. Comput.*, vol. 18, no. 3, pp. 230–254, 2007.

[10] M. da Conceição Moraes Batista and A. C. Salgado, “Information quality measurement in data integration schemas,” in *QDB*, 2007, pp. 61–72.

[11] J. David, J. Euzenat, F. Scharffe, and C. T. dos Santos, “The Alignment API 4.0,” *Semantic Web*, vol. 2, no. 1, pp. 3–10, 2011.

[12] R. Dhamankar, Y. Lee, A. Doan, A. Halevy, and P. Domingos, “iMAP: Discovering Complex Semantic Matches between Database Schemas.” in *ACM SIGMOD*, 2004, pp. 383–394.

[13] H. H. Do, S. Melnik, and E. Rahm, “Comparison of schema matching evaluations,” in *Web, Web-Services, and Database Systems Workshop*, 2002.

[14] H. H. Do and E. Rahm, “COMA - A System for Flexible Combination of Schema Matching Approaches,” in *VLDB*, 2002, pp. 610–621.

[15] ——, “Matching large schemas: Approaches and evaluation,” *Inf. Syst.*, vol. 32, no. 6, pp. 857–885, 2007.

[16] A. Doan, J. Madhavan, P. Domingos, and A. Halevy, “Ontology matching: A machine learning approach,” in *Handbook on Ontologies in Information Systems*, 2004.

[17] A. Doan, P. Domingos, and A. Y. Halevy, “Reconciling Schemas of Disparate Data Sources - A Machine Learning Approach,” in *ACM SIGMOD*, 2001.

[18] F. Duchateau and Z. Bellahsene, “Measuring the Quality of an Integrated Schema,” in *ER - International Conference on Conceptual Modeling*, 2010.

[19] F. Duchateau, Z. Bellahsene, and R. Coletta, “Matching and alignment: What is the cost of user post-match effort?” in *OTM Conferences (CooPIS)*, 2011.

[20] F. Duchateau, Z. Bellahsene, and E. Hunt, “XBenchMatch: a Benchmark for XML Schema Matching Tools,” in *VLDB*, 2007, pp. 1318–1321.

[21] F. Duchateau, R. Coletta, Z. Bellahsene, and R. J. Miller, “(Not) Yet Another Matcher,” in *CIKM*, 2009, pp. 1537–1540.

[22] ——, “YAM: a Schema Matcher Factory,” in *CIKM*, 2009, pp. 2079–2080.

[23] J. Euzenat *et al.*, “Ontology alignment evaluation initiative (http://oaei.ontologymatching.org), part of the SEALS project (http://www.seals-project.eu).” [Online]. Available: http://oaei.ontologymatching.org

[24] ——, “State of the art on ontology matching,” Knowledge Web, Tech. Rep. KWEB/2004/D2.2.3/v1.2, 2004.

[25] J. Euzenat, M.-E. Rosoiu, and C. Trojahn, “Ontology matching benchmarks: generation, stability, and discriminability,” *Web Semantics: Science, Services and Agents on the World Wide Web*, vol. 21, no. 0, 2013. [Online]. Available: http://www.websemanticsjournal.org/index.php/ps/article/view/338

[26] N. Fuhr, J. Kamps, M. Lalmas, and A. Trotman, Eds., *Focused Access to XML Documents, 6th International Workshop of the Initiative for the Evaluation of XML Retrieval*, ser. Lecture Notes in Computer Science, vol. 4862. Springer, 2008.

[27] B. C. Grau, Z. Dragisic, K. Eckert, J. Euzenat, A. Ferrara, R. Granada, V. Ivanova, E. Jiménez-Ruiz, A. O. Kempf, P. Lambrix *et al.*, “Results of the ontology alignment evaluation initiative 2013,” in *Proc. 8th ISWC workshop on ontology matching (OM)*, 2013, pp. 61–100.

[28] J. Gray, H. Schek, M. Stonebraker, and J. Ullman, "The Lowell report," in *Proceedings of SIGMOD'03*. New York, NY, USA: ACM, 2003, pp. 680–680.

[29] J. Hammer, M. Stonebraker, , and O. Topsakal, "Thalia: Test harness for the assessment of legacy information integration approaches," in *Proceedings of ICDE*, 2005, pp. 485–486.

[30] E. Ioannou, N. Rassadko, and Y. Velegrakis, "On Generating Benchmark Data for Entity Matching," *Journal on Data Semantics*, vol. 2, no. 1, pp. 37–56, 2013. [Online]. Available: http://dx.doi.org/10.1007/s13740-012-0015-8

[31] S. Kesh, "Evaluating the quality of entity relationship models," in *Information and Software Technology*, vol. 37, 1995, pp. 681–689.

[32] H. Köpcke, A. Thor, and E. Rahm, "Evaluation of entity resolution approaches on real-world match problems," *PVLDB*, vol. 3, no. 1, pp. 484–493, 2010.

[33] J. Madhavan, P. A. Bernstein, and E. Rahm, "Generic Schema Matching with Cupid," in *VLDB*, 2001, pp. 49–58.

[34] A. Marie and A. Gal, "Boosting schema matchers," in *OTM '08: Proceedings of the OTM 2008 (CoopIS)*. Berlin, Heidelberg: Springer-Verlag, 2008, pp. 283–300.

[35] S. Massmann, D. Engmann, and E. Rahm, "COMA++: Results for the Ontology Alignment Contest OAEI 2006," in *Ontology Matching*, 2006.

[36] S. Melnik, H. Garcia-Molina, and E. Rahm, "Similarity Flooding: A versatile graph matching algorithm and its application to schema matching," in *ICDE*, 2002, pp. 117–128.

[37] S. Melnik, E. Rahm, and P. A. Bernstein, "Developing metadata-intensive applications with Rondo," *J. of Web Semantics*, vol. I, pp. 47–74, 2003.

[38] D. H. Ngo, Z. Bellahsene, and R. Coletta, "YAM++ results for OAEI 2011," in *ISWC'11: The 6th International Workshop on Ontology Matching*, ser. CEUR Workshop Proceedings, P. Shvaiko, J. Euzenat, T. Heath, C. Quix, M. Mao, and I. Cruz, Eds., vol. 814, Oct. 2011, pp. 228–235. [Online]. Available: http://hal.inria.fr/lirmm-00649320/en

[39] D. Ngo and Z. Bellahsene, "YAM++: A Multi-strategy Based Approach for Ontology Matching Task," in *Proceedings of the 18th International Conference on Knowledge Engineering and Knowledge Management*, ser. EKAW'12. Berlin, Heidelberg: Springer-Verlag, 2012, pp. 421–425. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-33876-2_38

[40] ——, "YAM++ results for OAEI 2013," in *OM*, 2013, pp. 211–218.

[41] N. F. Noy, A. Doan, and A. Y. Halevy, "Semantic integration," *AI Magazine*, vol. 26, no. 1, pp. 7–10, 2005.

[42] H. Paulheim, S. Hertling, and D. Ritze, "Towards evaluating interactive ontology matching tools," in *The Semantic Web: Semantics and Big Data*. Springer, 2013, pp. 31–45.

[43] E. Rahm and P. A. Bernstein, "A survey of approaches to automatic schema matching," *VLDB Journal*, vol. 10, no. 4, pp. 334–350, 2001. [Online]. Available: citeseer.ist.psu.edu/rahm01survey.html

[44] E. Rahm, H. H. Do, and S. Massmann, "Matching Large XML Schemas," *SIGMOD Record*, vol. 33, no. 4, pp. 26–31, 2004.

[45] P. Shvaiko and J. Euzenat, "A survey of schema-based matching approaches," *Journal of Data Semantics IV*, pp. 146–171, 2005.

[46] K. Smith, M. Morse, P. Mork, M. Li, A. Rosenthal, D. Allen, and L. Seligman, "The role of schema matching in large enterprises," in *CIDR*, 2009.

[47] Wordnet, "http://wordnet.princeton.edu," 2007.

[48] M. Yatskevich, "Preliminary evaluation of schema matching systems," University of Trento, Tech. Rep. DIT-03-028, Informatica e Telecomunicazioni, 2003.

## AUTHOR BIOGRAPHIES

Fabien Duchateau is an Associate Professor at Université Claude Bernard Lyon 1 and at the LIRIS laboratory since 2011. In 2010, he obtained a 18-months post-doctoral fellowship sponsored by ERCIM. He spent the first part of the fellowship at CWI, The Netherlands. He joined the digital libraries team at NTNU, Norway, for the second part of the post-doctoral fellowship. He obtained his Ph.D. in computer science at the LIRMM laboratory, Université of Montpellier 2, in November 2009. His main research interests are data integration, schema matching, entity matching, semantic web, digital libraries, geographical information systems and machine learning.
`http://liris.cnrs.fr/~fduchate/`

Zohra Bellahsene is a professor of CS at University Montpellier 2 and a senior researcher at LIRMM. She received her Ph.D. in CS from University of Paris 6 in 1982 and her HDR in CS from University Montpellier 2 in 2000. She has a long experience in database research, recently focusing on various aspects of data integration, in particular, schema matching, view management and schema and ontology matching. She has organized or chaired several international conferences and workshops, including being the PC co-chair of CoopIS2013, the PC chair of CAiSE'08, the co-chair of the XML Database Symposium (2006, 2009), and the local chair of OTM06. She was the editor of the special issue of the DKE Journal on Data Integration over the Web in 2003. She was a co-editor of Schema and Matching and Mapping book, published in 2011 by Springer. She has been serving as PC member of major international conferences including SIGMOD, EDBT, ICDE, CAiSE, CIKM, ESWC, ISWC, ER, etc.
`http://www.lirmm.fr/~bella/`