Comparison of Schema Matching Evaluations

Hong-Hai Do, Sergey Melnik, Erhard Rahm

University of Leipzig
Augustusplatz 10-11, 04109, Leipzig, Germany
{hong, melnik, rahm}@informatik.uni-leipzig.de
dbs.uni-leipzig.de

Abstract. Recently, schema matching has found considerable interest in both research and practice. Determining matching components of database or XML schemas is needed in many applications, e.g. for E-business and data integration. Various schema matching systems have been developed to solve the problem semi-automatically. While there have been some evaluations, the overall effectiveness of currently available automatic schema matching systems is largely unclear. This is because the evaluations were conducted in diverse ways making it difficult to assess the effectiveness of each single system, let alone to compare their effectiveness. In this paper we survey recently published schema matching evaluations. For this purpose, we introduce the major criteria that influence the effectiveness of a schema matching approach and use these criteria to compare the various systems. Based on our observations, we discuss the requirements for future match implementations and evaluations.

1. Introduction

Schema matching is the task of finding semantic correspondences between elements of two schemas [11, 14, 17]. This problem needs to be solved in many applications, e.g. for data integration and XML message mapping in E-business. In today's systems, schema matching is manual; a time-consuming and tedious process which becomes increasingly impractical with a higher number of schemas (data sources, XML message formats) to be dealt with. Various systems and approaches have recently been developed to determine schema matches (semi-)automatically, e.g., Autoplex [1], Automatch [2], Clio [22, 16], COMA [7], Cupid [14], Delta [6], DIKE [19], EJX [10][1] , GLUE [9], LSD [8], MOMIS (and ARTEMIS) [3, 5], SemInt [11, 12, 13], SKAT [18], Similarity Flooding (SF) [15], and TranScm [17]. While most of them have emerged from the context of a specific application, a few approaches (Clio, COMA, Cupid, and SF), try to address the schema matching problem in a generic way that is suitable for different applications and schema languages. A taxonomy of automatic match techniques and a comparison of the match approaches followed by the various systems is provided in [20].

For identifying a solution for a particular match problem, it is important to understand which of the proposed techniques performs best, i.e., can reduce the manual work required for the match task at hand most effectively. To show the effectiveness

[1] The authors did not give a name to their system, so we refer to it in this paper using the initials of the authors' names.

of their system, the authors have usually demonstrated its application to some real-world scenarios or conducted a study using a range of schema matching tasks. Unfortunately, the system evaluations were done using diverse methodologies, metrics, and data making it difficult to assess the effectiveness of each single system, not to mention to compare their effectiveness. Furthermore, the systems are usually not publicly available making it virtually impossible to apply them to a common test problem or benchmark in order to obtain a direct quantitative comparison.

To obtain a better overview about the current state of the art in evaluating schema matching approaches, we review the recently published *evaluations* of the schema matching systems in this paper. For this purpose, we introduce and discuss the major criteria influencing the effectiveness of a schema matching approach, e.g., the chosen test problems, the design of the experiments, the metrics used to quantify the match quality and the amount of saved manual effort. We intend our criteria to be useful for future schema matching evaluations so that they can be documented better, their result be more reproducible, and a comparison between different systems and approaches be easier. For our study, we only use the information available from the publications describing the systems and their evaluation.

In Section 2, we present the criteria that we use in our study to contrast the evaluations described in the literature. In Section 3, we review the single evaluations by giving first a short description about the system being evaluated and then discussing the methodology and the result of the actual evaluation. In Section 4, we compare the evaluations by summarizing their strengths and weakness. We then present our observations concerning the current situation of the match systems as well as the challenges that future match implementations and evaluations should address. Section 5 concludes the paper.

2. Comparison criteria

To compare the evaluations of schema matching approaches we consider criteria from four different areas:

- *Input*: What kind of input data has been used (schema information, data instances, dictionaries etc.)? The simpler the test problems are and the more auxiliary information is used, the more likely the systems can achieve better effectiveness. However, the dependence on auxiliary information may also lead to increased preparation effort.
- *Output*: What information has been included in the match result (mappings between attributes or whole tables, nodes or paths etc.)? What is the correct result? The less information the systems provide as output, the lower the probability of making errors but the higher the post-processing effort may be.
- *Quality measures*: What metrics have been chosen to quantify the accuracy and completeness of the match result? Because the evaluations usually use different metrics, it is necessary to understand their behavior, i.e. how optimistic or pessimistic their quality estimation is.
- *Effort*: How much savings of manual effort are obtained and how is this quantified? What kind of manual effort has been measured, for example, pre-match effort (training of learners, dictionary preparation etc.), and post-match effort (correction and improvement of the match output)?

In the subsequent sections we elaborate on the above criteria in more detail.

2.1. Input: test problems and auxiliary information

To document the complexity of the test problems, we consider the following information about the test schemas:

- *Schema language* (relational, XML schemas, etc.): Different schema languages can exhibit different facets to be exploited by match algorithms. However, relying on language-specific facets will cause the algorithms to be confined to the particular schema type. In current evaluations, we have observed only homogeneous match tasks, i.e. matching between schemas of the same type.
- *Number of schemas and match tasks*: With a high number of different match tasks, it is more likely to achieve a realistic match behavior. Furthermore, the way the match tasks are defined can also influence the problem complexity, e.g. matching many independent schemas with each other vs. matching source schemas to a single global schema.
- *Schema information*: Most important is the number of the schema elements for which match candidates are to be determined. The bigger the input schemas are, the greater the search space for match candidates will be, which often leads to lower match quality. Furthermore, matchers exploiting specific facets will perform better and possibly outperform other matchers when such information is present or given in better quality and quantity.
- *Schema similarity*: Intuitively, a match task with schemas of the same size becomes “harder” if the similarity between them drops. Here we refer to schema similarity simply as the ratio between the number of matching elements (identified in the manually constructed match result) and the number of all elements from both input schemas [7].
- *Auxiliary information used*: Examples are dictionaries or thesauri, or the constraints that apply to certain match tasks (e.g., each source element must match at least one target element). Availability of such information can greatly improve the result quality.

2.2. Output: match result

The output of a match system is a mapping indicating which elements of the input schemas correspond to each other, i.e. match. To assess and to compare the output quality of different match systems, we need a uniform representation of the correspondences. Currently, all match prototypes determine correspondences between schema elements (element-level matches [20]) and use similarity values between 0 (strong dissimilarity) and 1 (strong similarity) to indicate the plausibility of the correspondences. However, the quality and quantity of the correspondences in a match result still depend on several orthogonal aspects:

- *Element representation*: Schema matching systems typically use a graph model for the internal representation of schemas. Hence, schema elements may either be represented by nodes or paths in the schema graphs which also impacts the representation of schema matches. Figure 1 shows a simple match problem with two small (purchase order) schemas in directed graph representation; a sample match between nodes would be *Contact*↔*ContactPers*. However, shared elements, such as *ContactPers* in *PO2*, exhibit different contexts, i.e. *DeliverTo* and *BillTo*, which should be considered independently. Thus, some systems return matches between node paths, e.g., *PO1.Contact*↔*PO2.DeliverTo.ContactPers*. Considering paths possibly leads to more elements, for which match candidates can be individually determined, and thus, possibly to more correspondences. Furthermore, the paths implicitly include valuable join information that can be utilized for generating the mapping expressions.

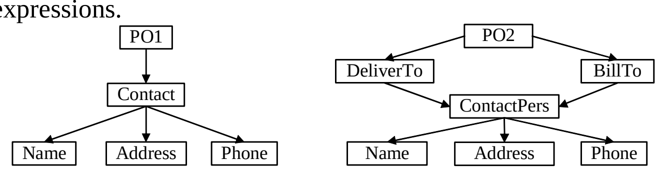


Figure 1. Schema examples for a simple match task

- *Cardinality*: An element from one schema can participate in zero, one or several match correspondences from the second input schema (*global cardinality* of 1:1, 1:n/n:1, or n:m). Moreover, within a correspondence one or more elements of the first schema may be matched with one or more elements of the second schema (*local cardinality* of 1:1, 1:n/n:1, n:m) [20]. For example, in Figure 1, *PO1.Contact* may be matched to both *PO2.DeliverTo.ContactPers* and *PO2.BillTo.ContactPers*. Grouping these two match relationships within a single correspondence, we have 1:n local cardinality. Representing them as two separate correspondences leads to 1:n global and 1:1 local cardinality. Most automatic match approaches are restricted to 1:1 local cardinality by selecting for a schema element the most similar one from the other schema as the match candidate.

2.3. Match quality measures

To provide a basis for evaluating the quality of automatic match strategies, the match task first has to be manually solved. The obtained real match result can be used as the "gold standard" to assess the quality of the result automatically determined by the match system. Comparing the automatically derived matches with the real matches results in the sets shown in Figure 2 that can be used to define quality measures for schema matching. In particular, the set of derived matches is comprised of B, the *true positives*, and C, the *false positives*. *False negatives* (A) are matches needed but not automatically identified, while false positives are matches falsely proposed by the automatic match operation. *True negatives*, D, are false matches, which have also been correctly discarded by the automatic match operation. Intuitively, both false negatives and false positives reduce the match quality.

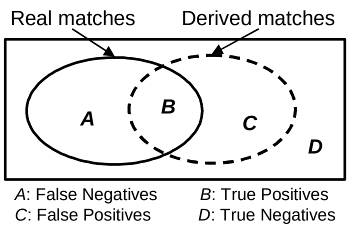


Figure 2. Comparing real matches and automatically derived matches

Based on the cardinality of these sets, two common measures, *Precision* and *Recall*, which actually originate from the information retrieval field, can be computed:

- $Precision = \frac{|B|}{|B| + |C|}$ reflects the share of real correspondences among all found ones
- $Recall = \frac{|B|}{|A| + |B|}$ specifies the share of real correspondences that is found

In the ideal case, when no false negatives and false positives are returned, we have *Precision*=*Recall*=1. However, neither *Precision* nor *Recall* alone can accurately assess the match quality. In particular, *Recall* can easily be maximized at the expense of a poor *Precision* by returning all possible correspondences, i.e. the cross product of two input schemas. On the other side, a high *Precision* can be achieved at the expense of a poor *Recall* by returning only few (correct) correspondences.

Hence it is necessary to consider both measures or a combined measure. Several combined measures have been proposed so far, in particular:

- $F\text{–}Measure(\alpha) = \frac{|B|}{(1-\alpha)*|A| + |B| + \alpha*|C|} = \frac{Precision*Recall}{(1-\alpha)*Precision + \alpha*Recall}$, which also stems from the information retrieval field [21]. The intuition behind this parametrized measure ($0 \le \alpha \le 1$) is to allow different relative importance to be attached to *Precision* and *Recall*. In particular, *F-Measure*(α)→*Precision*, when α→1, i.e. no importance is attached to *Recall*; and *F-Measure*(α)→*Recall*, when α→0, i.e. no importance is attached to *Precision*. When *Precision* and *Recall* are considered equally important, i.e. α=0.5, we have the following combined measure:
- $F\text{–}Measure = \frac{2*|B|}{(|A|+|B|)+(|B|+|C|)} = 2*\frac{Precision*Recall}{Precision+Recall}$, which represents the harmonic mean of *Precision* and *Recall* and is the most common variant of *F-Measure*(α) in information retrieval. Currently, it is used in [2] for estimating match quality.
- $Overall = 1 - \frac{|A|+|C|}{|A|+|B|} = \frac{|B|-|C|}{|A|+|B|} = Recall*\left(2-\frac{1}{Precision}\right)$, which has been introduced in [15][2] and is also used in [7]. Unlike *F-Measure*(α), *Overall* was developed specifically in the schema matching context and embodies the idea to quantify the post-match effort needed for adding false negatives and removing false positives.

To compare the behavior of *F-Measure* and *Overall*, Figure 3 shows them as functions of *Precision* and *Recall*, respectively. Apparently, *F-Measure* is much more optimistic than *Overall*. For the same *Precision* and *Recall* values, *F-Measure* is still much higher than *Overall*. Unlike the other measures, *Overall* can have negative values, if the number of the false positives exceeds the number of the true positives, i.e. *Precision*<0.5. Both combined measures reach their highest value (1.0) with *Precision*=*Recall*=1.0. In all other cases, while the value of *F-Measure* is within the range determined by *Precision* and *Recall*, *Overall* is smaller than both *Precision* and *Recall*.

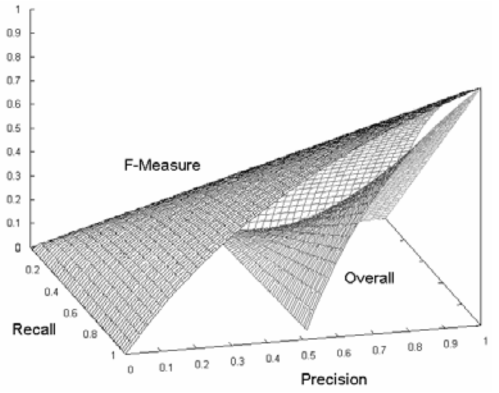


Figure 3. *F-Measure* and *Overall* as functions of *Precision* and *Recall*

[2] Here it is called *Accuracy*

2.4. Test methodology: what effort is measured and how

Given that the main purpose of automatic schema matching is to reduce the amount of manual work quantifying the user effort still required is a major requirement. However this is difficult because of many subjective aspects involved and thus a largely unsolved problem. To assess the manual effort one should consider both the *pre-match effort* required before an automatic matcher can run as well as the *post-match effort* to add the false negatives to and to remove the false positives from the final match result.

Pre-match effort includes:

- Training of the machine learning-based matchers
- Configuration of the various parameters of the match algorithms, e.g., setting different threshold and weight values
- Specification of auxiliary information, such as, domain synonyms and constraints

In fact, extensive pre-match effort may wipe out a large fraction of the labor savings obtained through the automatic matcher and therefore needs to be specified precisely. In all evaluations so far the pre-match effort has not been taken into account for determining the quality of a match system or approach.

The simple measures *Recall* and *Precision* only partially consider the post-match effort. In particular, while 1–*Recall* gives an estimate for the effort to add false negatives, 1–*Precision* can be regarded as an estimate for the effort to remove false positives. In contrast, the combined measures *F-Measure*(α) and *Overall* take both kinds of effort into account. *Overall* assumes equal effort to remove false positives and to identify false negatives although the latter may require manual searching in the input schemas. On the other hand, the parameterization of *F-Measure*(α) already allows to apply individual cost weighting schemes. However, determining that a match is correct requires extra work not considered in both *Overall* and *F-Measure*(α).

Unfortunately, the effort associated with such manual pre-match and post-match operations varies heavily with the background knowledge and cognitive abilities of users, their familiarity with tools, the usability of tools (e.g. available GUI features such as zooming, highlighting the most likely matches by thick lines, graying out the unlikely ones etc.) making it difficult to capture the cost in a general way.

Finally, the specification of the real match result depends on the individual user perception about correct and false correspondences as well as on the application context. Hence, the match quality can differ from user to user and from application to application given the same input schemas. This effect can be limited to some extent by consulting different users to obtain multiple subjective real match results [15].

3. Studies

In the following, we review the evaluations of eight different match prototypes, Autoplex, Automatch, COMA, Cupid, LSD, GLUE, SemInt, and SF. We have encountered a number of systems, which either have not been evaluated, such as Clio, DIKE, MOMIS, SKAT, and TranScm, or their evaluations have not been described with sufficient detail, such as Delta, and EJX. Those systems are not considered in our study. For each system, we shortly describe its match approach and then discuss the details of the actual evaluation. According to the taxonomy presented in [20], we briefly characterize the approaches implemented in each system by capturing

- The type of the matchers implemented (schema vs. instance level, element vs. structure level, language vs. constraint based etc.)
- The type of information exploited (e.g., schema properties, instance characteristics, and external information)
- The mechanism to combine the matchers (e.g., hybrid or composite [20, 7]).

3.1. Autoplex and Automatch

System description: Autoplex [1] and its enhancement Automatch [2] represent single-strategy schema matching approaches based on machine learning. In particular, a Naive Bayesian learner exploits instance characteristics to match attributes from a relational source schema to a previously constructed global schema. For each source attribute, both match and mismatch probability with respect to every global attribute are determined. These probabilities are normalized to sum to 1 and the match probability is returned as the similarity between the source and global attribute. The correspondences are filtered to maximize the sum of their similarity under the condition that no correspondences share a common element. The match result consists of attribute correspondences of 1:1 local and global cardinality.
Evaluation: In both Autoplex and Automatch evaluation, the global schemas were rather small, containing 15 and 9 attributes, respectively. No information about the characteristics of the involved source schemas was given. First the source schemas were matched manually to the global schema, resulting in 21 and 22 mappings in the Autoplex and Automatch evaluation, respectively. These mappings were divided into three portions of approximately equal content. The test was then carried out in three runs, each using two portions for learning and the remaining portion for matching.

The Autoplex evaluation used the quality measures *Precision* and *Recall*,[3] while for Automatch, *F-Measure* was employed. However, the measures were not determined for single experiments but for the entire evaluation: the false/true negatives and positives were counted over all match tasks. For Autoplex, they were reported separately for table and column matches. We re-compute the measures to consider all matches and obtain a *Precision* of 0.84 and *Recall* of 0.82, corresponding to an *F-Measure* of 0.82 and *Overall* of 0.66. Furthermore, the numbers of the false/true negatives and positives were rather small despite counting over multiple tasks, leading to the conclusion that the source schemas must be very small. For Automatch, the impact of different methods for sampling instance data on match quality was studied. The highest *F-Measure* reported was 0.72, so that the corresponding *Overall* must be worse.

3.2. COMA

System description: COMA [7] follows a composite approach, which provides an extensible library of different matchers and supports various ways for combining match results. Currently, the matchers exploit schema information, such as element and structural properties. Furthermore, a special matcher is provided to reuse the results from previous match operations. The combination strategies address different aspects of match processing, such as, aggregation of matcher-specific results and match candidate selection. Schemas are transformed to rooted directed acyclic graphs, on which all match algorithms operate. Each schema element is uniquely identified by

[3] Here they are called *Soundness* and *Completeness*, respectively

its complete path from the root of the schema graph to the corresponding node. COMA produces element-level matches of 1:1 local and m:n global cardinality.

Evaluation: The COMA evaluation used 5 XML schemas for purchase orders taken from www.biztalk.org. The size of the schemas ranged from 40 to 145 unique elements, i.e. paths. Ten match tasks were defined, each matching two different schemas. The similarity between the schemas was mostly only around 0.5, showing that the schemas are much different even though they are from the same domain. Some pre-match effort was needed to specify domain synonyms and abbreviations.

A comprehensive evaluation was performed with COMA to investigate the impact of different combination strategies on match quality and to compare the effectiveness of different matchers, i.e. single matchers vs. matcher combinations, with and without reuse. The entire evaluation consisted of over 12,000 test series, in each of which a different choice of matchers and combination strategies was applied. Each series in turn consisted of 10 experiments dealing with the (10) predefined match tasks. The quality measures *Precision*, *Recall*, and *Overall* were first determined for single experiments and then averaged over 10 experiments in each series (average *Precision*, etc.). Based on their quality behavior across the series, the best combination strategies were determined for the default match operation.

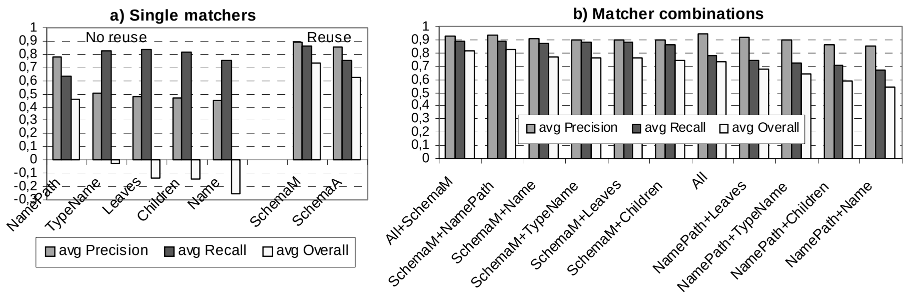


Figure 4. Match quality of COMA [7]

Figure 4a shows the quality of the single matchers, distinguished between the no-reuse and reuse-oriented ones. The reuse matchers yielded significantly better quality then the no-reuse ones. Figure 4b shows the quality of the best matcher combinations. In general, the combinations achieved much better quality than the single matchers. Furthermore, a superiority of the reuse combinations over the no-reuse ones was again observed. While the best no-reuse matcher, *All*, combining all the single no-reuse matchers, achieved average *Overall* of 0.73 (average *Precision* 0.95, average *Recall* 0.78), the best reuse combination, *All+SchemaM*, reached the best average *Overall* in the entire evaluation, 0.82 (average *Precision* 0.93, average *Recall* 0.89). These combinations also yielded the best quality for most match tasks, i.e. high stability across different match tasks. However, while optimal or close to optimal *Overall* was achieved for the smaller match tasks, *Overall* was limited to about 0.6-0.7 in larger problems. This was apparently also influenced by the moderate degree of schema similarity.

3.3. Cupid

System description: Cupid [14] represents a sophisticated hybrid match approach combining a name matcher with a structural match algorithm, which derives the similarity of elements based on the similarity of their components hereby emphasizing the name and data type similarities present at the finest level of granularity (leaf level). To address the problem of shared elements, the schema graph is converted to a tree, in which additional nodes are added to resolve the multiple relationships between a shared node and its parent nodes. Cupid returns element-level correspondences of 1:1 local and n:1 global cardinality.

Evaluation: In their evaluation, the authors compared the quality of Cupid with 2 previous systems, DIKE and MOMIS, which had not been evaluated so far. For Cupid, some pre-match effort was needed to specify domain synonyms and abbreviations. First, the systems were tested with some canonical match tasks considering very small schema fragments. Second, the systems were tested with 2 real-world XML schemas for purchase order, which is also the smallest match task in the COMA evaluation [7]. The authors then compared the systems by looking for the correspondences which could or could not be identified by a particular system. Cupid was able to identify all necessary correspondences for this match task, and thus showed a better quality than the other systems. In the entire evaluation, no quality measures were computed.

3.4. LSD and GLUE

System description: LSD [8] and its extension GLUE [9] use a composite approach to combining different matchers. While LSD matches new data sources to a previously determined global schema, GLUE performs matching directly between the data sources. Both use machine-learning techniques for individual matchers and an automatic combination of match results. In addition to a name matcher, they use several instance-level matchers, which discover during the learning phase different characteristic instance patterns and matching rules for single elements of the target schema. The predictions of individual matchers are combined by a so-called meta-learner, which weights the predictions from a matcher according to its accuracy shown during the training phase. The match result consists of element-level correspondences with 1:1 local and n:1 global cardinality.

Evaluation: LSD was tested on 4 domains, in each of which 5 data sources were matched to a manually constructed global schema, resulting in 20 match tasks altogether. To match a particular source, 3 other sources from the same domain were used for training. The source schemas were rather small (14-48 elements), while the largest global schema had 66 attributes. GLUE was evaluated for 3 domains, in each of which two website taxonomies were matched in two different directions, i.e. A→B and B→A. The taxonomies were relatively large, containing up to 300 elements. Both systems rely on pre-match effort

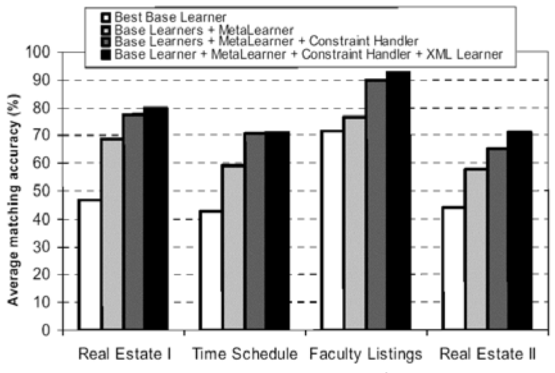


Figure 5. Match quality of LSD [8]

on the one side to train the learners, and on the other side, to specify domain synonyms and constraints.

For both LSD and GLUE, different learner combinations were evaluated. For LSD, the impact of the amount of available instance data on match quality was also studied. Match quality was estimated using a single measure, called *match accuracy,* defined as the percentage of the matchable source attributes that are matched correctly. It corresponds to *Recall* in our definition due to one single correspondence returned for each source element. Furthermore, we observe that at most a *Precision* equal to the presented *Recall* can be achieved for single match tasks; that is, if all source elements are matchable. Based on this conclusion, we can derive the highest possible *F-Measure* (=*Recall*) and *Overall* (=2**Recall*-1) for both LSD and GLUE. Figure 5 shows the quality of different learner combinations in LSD. The best quality was usually achieved when all learners were involved. In the biggest match tasks, LSD and GLUE achieved *Recall* of around 0.7, i.e. *Overall* of at most 0.4. In the case of GLUE, this quality is quite impressive considering the schema sizes involved (333 and 115 elements [9]). On average (over all domains), LSD and GLUE achieved a *Recall* of ~0.8, respectively. This corresponds to an *Overall* of at most 0.6.

3.5. Similarity Flooding (SF)

System description: SF [15] converts schemas (SQL DDL, RDF, XML) into labeled graphs and uses fix-point computation to determine correspondences of 1:1 local and m:n global cardinality between corresponding nodes of the graphs. The algorithm has been employed in a hybrid combination with a simple name matcher, which suggests an initial element-level mapping to be fed to the structural SF matcher. Unlike other schema-based match approaches, SF does not exploit terminological relationships in an external dictionary, but entirely relies on string similarity between element names. In the last step, various filters can be specified to select relevant subsets of match results produced by the structural matcher.

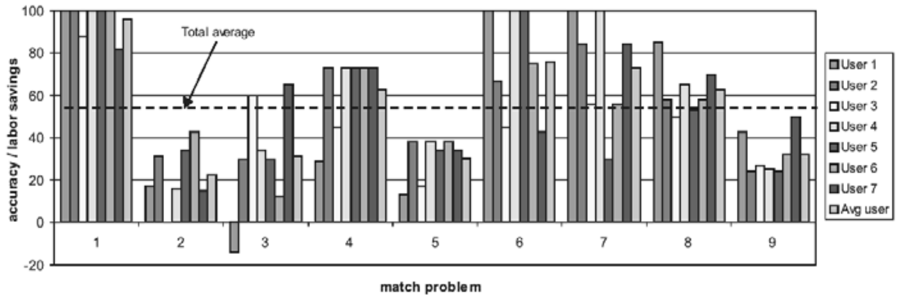
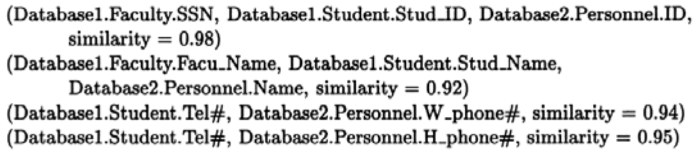


Figure 6. Match quality of Similarity Flooding Algorithm [15]

Evaluation: The SF evaluation used 9 match tasks defined from 18 schemas (XML and SQL DDL) taken from different application domains. The schemas were small with the number of elements ranging from 5 to 22, while showing a relatively high similarity to each other (0.75 on average). Seven users were asked to perform the manual match process in order to obtain subjective match results. For each match tasks, the results returned by the system were compared against all subjective results to estimate the automatic match quality, for which the *Overall* measure was used. Other experiments were also conducted to compare the effectiveness of different filters and formulas for fix-point computation, and to measure the impact of randomizing the similarities in the initial mapping on match accuracy. The best configuration was identified and used in SF. Figure 6 shows the *Overall* values achieved in the single match tasks according to the match results suggested by the single users. The average *Overall* quality over all match tasks and all users is around 0.6.

3.6. SemInt

System description: SemInt [11, 12] represents a hybrid approach exploiting both schema and instance information to identify corresponding attributes between relational schemas. The schema-level constraints, such as data type and key constraints, are derived from the DBMS catalog. Instance data are exploited to obtain further information, such as actual value distributions, numerical averages, etc. For each attribute, SemInt determines a signature consisting of values in the interval [0,1] for all involved matching criteria. The signatures are used first to cluster similar attributes from the first schema and then to find the best matching cluster for attributes from the second schema. The clustering and classification process is performed using neural networks with an automatic training, hereby limiting pre-match effort. The match result consists of clusters of similar attributes from both input schemas, leading to m:n local and global match cardinality. Figure 7 shows a sample output of SemInt. Note that each cluster may contain multiple 1:1 correspondences, which are not always correct, such as in the first two clusters.



```
(Database1.Faculty.SSN, Database1.Student.Stud_ID, Database2.Personnel.ID,
     similarity = 0.98)
(Database1.Faculty.Facu_Name, Database1.Student.Stud_Name,
     Database2.Personnel.Name, similarity = 0.92)
(Database1.Student.Tel#, Database2.Personnel.W_phone#, similarity = 0.94)
(Database1.Student.Tel#, Database2.Personnel.H_phone#, similarity = 0.95)
```

Figure 7. SemInt output: match result [12]

Evaluation: A preliminary test consisting of 3 experiments is presented in [11]. The test schemas were small with mostly less than 10 attributes. However, the quality measures for these experiments were only presented later in [12, 13]. In these small match tasks, SemInt performed very well and achieved very high *Precision* (0.9, 1.0, 1.0) and *Recall* (1.0). In [12, 13], SemInt was evaluated with two further match tasks. In the bigger match task with schemas with up to 260 attributes, SemInt surprisingly performed very well (*Precision* ~0.8, *Recall* ~0.9). But in the smaller task with schemas containing only around 40 elements, the quality dropped drastically (*Precision* 0.20, *Recall* 0.38).

On average over 5 experiments, SemInt achieved a *Precision* of 0.78 and *Recall* of 0.86. Using the *Precision* and *Recall* values presented for each experiment, we can also compute the average *F-Measure*, 0.81, and *Overall*, 0.48. On the other hand, it is necessary to take into consideration that this match quality was determined from match results of attribute clusters, each of which possibly contains multiple 1:1 correspondences. In addition to the match tasks, further tests were performed to measure the sensitivity of the single match criteria employed by SemInt [12]. The results allowed to identify a minimal subset of match criteria, which could still retain the overall effectiveness.

4. Discussion and conclusions

We first summarize the strengths and weaknesses of the single evaluations and then present our conclusions concerning future evaluations.

4.1. Comparative discussion

Table 1 gives a summary about the discussed evaluations. The test problems came from very different domains of different complexity. While a few evaluations used simple match tasks with small schemas and few correspondences to be identified (Autoplex, Automatch, SF), the remaining systems also showed high match quality for more complex real-world schemas (COMA, LSD, GLUE, SemInt). Some evaluations, such as Autoplex and Automatch, completely lack the description of their test schemas. The Cupid evaluation represents the only effort so far that managed to evaluate multiple systems on uniform test problems. Unlike other systems, Autoplex, Automatch and LSD perform matching against a previously constructed global schema.

All systems return correspondences at the element level with similarity values in the range of [0,1]. Those confined to instance-level matching, such as Autoplex, Automatch, and SemInt, can only deliver correspondences at the finest level of granularity (attributes). In all systems, except for SemInt, correspondences are of 1:1 local cardinality, providing a common basis for determining match quality.

Only the SF evaluation took into account the subjectivity of the user perception about required match correspondences. Unlike other approaches, SemInt and SF do not require any manual pre-match effort. In several evaluations, e.g. COMA, LSD, GLUE, SemInt and SF, different system configurations were tested by varying match parameters on the same match tasks in order to measure the impact of the parameters on match quality. Those results have provided valuable insights for improving and developing new match algorithms.

Usually, the quality measures were computed for single match experiments. Exceptions are Cupid with no quality measure computed, and Autoplex, Automatch with quality measures mixing the match results of several experiments in a way that does not allow us to assess the quality for individual match tasks. Whenever possible, we tried to translate the quality measures considered in an evaluation to others not considered so that one can get an impression about the actual meaning of the measures. Still, the computed quality measures cannot be used to directly compare the effectiveness of the systems because of the great heterogeneity in other evaluation criteria. Only exploiting schema information, COMA seems quite successful, while the LSD/GLUE approach is promising for utilizing instance data.

4.2. Conclusions

The evaluations have been conducted in so different ways that it is impossible to directly compare their results. While the considered match problems were mostly simple, many techniques have proved to be quite powerful such as exploiting element and structure properties (Cupid, SF, COMA), and utilizing instance data, e.g., by Bayesian and Whirl learners (LSD/GLUE) or neural networks (SemInt). Moreover, the combined use of several approaches within composite match systems proved to be very successful (COMA, LSD/GLUE). On the other side, there are still unexploited opportunities, e.g. in the use of large-scale dictionaries and standard taxonomies and increased reuse of

Table 1. Summary of the evaluations

	Autoplex & -match	**COMA**	**Cupid**	**LSD & GLUE**	**SemInt**	**SF**
References	[1] and [2]	[7]	[14]	[8] and [9]	[11, 12, 13]	[15]
Test problems						
Tested schema types	relational	XML	XML	XML	relational	XML, relational
#Schemas / #Match tasks	15/21 & 15/22	5/10	2/1	24/20 & 3/6	10/5	18/9
Min/Max/Avg schema size	-	40/145/77	40/54//47	14/66/- & 34/333/143	6/260/57	5/22/12
Min/Max/Avg schema similarity	-	0.43/0.8/0.58	-	-	-	0.46/0.94/0.75
Match result representation						
Matches	element-level correspondences with similarity value in range [0,1]					
Element repr.	node (attr.)	path	path	node	node (attr.)	node
Local/global cardinality	1:1/1:1	1:1/m:n	1:1/n:1	1:1/n:1	m:n/m:n (attr. cluster)	1:1/m:n
Quality measures and test methodology						
Employed quality measures	Precision, Recall & F-Measure	Precision, Recall, Overall	none	Recall	Precision, Recall	Overall
Subjectivity	1 user					7 users
Studied impact on match quality	Automatch: methods for sampling instance data	matchers, combination, reuse, schema characteristics	none	learner combinations, LSD: amount of data listings	constraints (discriminators)	filters, fix-point formulas, randomizing initial sim
Pre-match effort	training	specifying domain synonyms	specifying domain synonyms	training, specifying domain synonyms, constraints	none	none
Best average match quality						
Prec./Recall	0.84/0.82	0.93/0.89	-	~0.8/0.8	0.78/0.86	-
F-Measure	0.82 & 0.72	0.90	-	~0.8	0.81	-
Overall	0.66	0.82	-	~0.6	0.48	~0.6
Evaluation highlights						
		Big schemas, Systematic evaluation	Comparative evaluation of 3 systems	Big schemas	Big schemas, No pre-match effort	User subjectivity, No pre-match effort

previous match results (COMA). Future match systems should integrate those techniques within a composite framework to achieve maximal flexibility.

Future evaluations should address the following issues:

- *Better conception for reproducibility*: To allow an objective interpretation and easy comparison of match quality between different systems and approaches, future evaluations should be conceived and documented more carefully, if possible, including the criteria that we identified in this paper.
- *Input factors – test schemas and system parameters*: All evaluations have shown that match quality degrades with bigger schemas. Hence, future systems should be evaluated with schemas of more realistic size, e.g. several hundreds of elements.
  Besides the characteristics of the test schemas, the various input parameters of each system can also influence the match quality in different ways. However, their impact has rarely been investigated in a comprehensive way, thus potentially missing opportunities for improvement and tuning. Similarly, previous evaluations

typically reported only some peak values w.r.t. some quality measure so that the overall match quality for a wider range of configurations remained open.

- *Output factors – match results and quality measures*: Instead of determining only one match candidate per schema element, future systems could suggest multiple, i.e. *top-K*, match candidates for each schema element. This can make it easier for the user to determine the final match result in cases where the first candidate is not correct. In this sense, a top-K match prediction may already be counted as correct if the required match candidate is among the proposed choices.
  Previous studies used a variety of different quality measures with limited expressiveness thus preventing a qualitative comparison between systems. To improve the situation and to consider precision, recall and the degree of post-match effort we recommend the use of combined measures such as *Overall* in future evaluations. However, further user studies are required to quantify the different effort needed for finding missing matches, removing false positives, and verifying the correct results. Another limitation of current quality measures is that they do not consider the pre-match effort and the hardness of match problems.

Ultimately, a *schema matching benchmark* seems very helpful to better compare the effectiveness of different match systems by clearly defining all input and output factors for a uniform evaluation. In addition to the test schemas, the benchmark should also specify the use of all auxiliary information in a precise way since otherwise any hard-to-detect correspondences could be built into a synonym table to facilitate matching. Because of the extreme degree of heterogeneity of real-world applications, the benchmark should not strive for general applicability but focus on a specific application domain, e.g., a certain type of E-business. Alternatively, a benchmark can focus on determining the effectiveness of match systems with respect to specific match capabilities, such as name, structural, instance-based and reuse-oriented matching. Currently we are investigating how such benchmarks could be generated.

5. Summary

Schema matching is a basic problem in many database and data integration applications. We observe a substantial research and development effort in order to provide semi-automatic solutions aiding the user in this time-consuming task. So far, many systems have been developed and several of them evaluated to show their effectiveness. However, the way the systems have been tested varies to a great extent from evaluation to evaluation. Thus it is difficult to interpret and compare the match quality presented for each system.

We proposed a catalog of criteria for documenting the evaluations of schema matching systems. In particular, we discussed various aspects that contribute to the match quality obtained as the result of an evaluation. We then used our criteria and the information available in the literature to review several previous evaluations. Based on the observed strengths and weaknesses, we discussed the problems that future system implementations and evaluations should address. We hope that the criteria that we identified provide a useful framework for conducting and describing future evaluations.

Acknowledgements

We thank Phil Bernstein and AnHai Doan for many useful comments. This work is supported by DFG grant BIZ 6/1-1.

References

1. Berlin, J., A. Motro: Autoplex: Automated Discovery of Content for Virtual Databases. CoopIS 2001, 108–122
2. Berlin, J., A. Motro: Database Schema Matching Using Machine Learning with Feature Selection. CAiSE 2002
3. Bergamaschi, S., S. Castano, M. Vincini, D. Beneventano: Semantic integration of heterogeneous information sources. Data & Knowledge Engineering 36: 3, 215–249, 2001
4. Bright, M. W., A.R. Hurson, S. Pakzad: Automated Resolution of Semantic Heterogeneity in Multidatabase. ACM Trans. Database Systems 19: 2, 212–253, 1994
5. Castano, S., V. De Antonellis: A Schema Analysis and Reconciliation Tool Environment. IDEAS 1999, 53–62
6. Clifton, C., E. Housman, E., A. Rosenthal: Experience with a Combined Approach to Attribute-Matching Across Heterogeneous Databases. IFIP 2.6 Working Conf. Database Semantics 1996
7. Do, H.H., E. Rahm: COMA – A System for Flexible Combination of Schema Matching Approach. VLDB 2002
8. Doan, A.H., P. Domingos, A. Halevy: Reconciling Schemas of Disparate Data Sources: A Machine-Learning Approach. SIGMOD 2001
9. Doan, A.H., J. Madhavan, P. Domingos, A. Halevy: Learning to Map between Ontologies on the Semantic Web. WWW 2002
10. Embley, D.W., D. Jackman, L. Xu: Multifaceted Exploitation of Metadata for Attribute Match Discovery in Information Integration. WIIW 2001
11. Li, W.S., C. Clifton: Semantic Integration in Heterogeneous Databases Using Neural Networks. VLDB 1994
12. Li, W.S., C. Clifton: SemInt: A Tool for Identifying Attribute Correspondences in Heterogeneous Databases Using Neural Network. Data and Knowledge Engineering 33: 1, 49–84, 2000
13. Li, W.S., C. Clifton, S.Y. Liu: Database Integration Using Neural Networks: Implementation and Experiences. Knowledge and Information Systems 2: 1, 2000
14. Madhavan, J., P.A. Bernstein, E. Rahm: Generic Schema Matching with Cupid. VLDB 2001
15. Melnik, S., H. Garcia-Molina, E. Rahm: Similarity Flooding: A Versatile Graph Matching Algo-rithm. ICDE 2002
16. Miller, R.J. et al. The Clio Project: Managing Heterogeneity. SIGMOD Record 30:1: 78–83, 2001
17. Milo, T., S. Zohar: Using Schema Matching to Simplify Heterogeneous Data Translation. VLDB 1998, 122–133
18. Mitra, P., G. Wiederhold, J. Jannink: Semi-automatic Integration of Knowledge Sources. Fusion 1999
19. Palopoli, L., G. Terracina, D. Ursino: The System DIKE: Towards the Semi-Automatic Synthesis of Cooperative Information Systems and Data Warehouses. ADBIS-DASFAA 2000, 108–117
20. Rahm, E., P.A. Bernstein: A Survey of Approaches to Automatic Schema Matching. VLDB Journal 10: 4, 2001
21. Van Rijsbergen, C. J.: Information Retrieval. 2nd edition, 1979, London, Butterworths.
22. Yan, L.L., R.J. Miller, L.M. Haas, R. Fagin. Data-Driven Understanding and Refinement of Schema Mappings. SIGMOD, 2001