On semi-automated matching and integration of database schemas

Ünal Karakaş, Ö.

Publication date
2010

Chapter 6

Empirical validation of SASMINT

In order to measure the quality and performance of our approach to schema matching and integration, we have performed a number of experiments. These experiments consider and make use of schemas that include different types of schema heterogeneities, as addressed in Chapter 3. This chapter describes these experiments and their results. In this respect, in Section 6.1 we first address a number of related experiments performed by other main research efforts. A number of specific quality measures used for assessing the results of our schema matching and schema integration components of SASMINT are described next in Section 6.2. The main characteristics of test schemas are addressed in Section 6.3. The setup and details related to the performed experimental evaluations are given in Section 6.4. Our evaluation results are addressed in Sections 6.5 to 6.7. Finally, Section 6.8 concludes the chapter with a summary of evaluation results.

This chapter contains some research results, which were previously published in the Journal of Knowledge and Information Systems (Unal & Afsarmanesh, 2010).

6.1 Schema Matching Evaluations in Related Research

The evaluation performed in most existing schema matching research does not use any benchmark; rather they each use their own test schemas in evaluating specific aspects of their proposed system.

A comparison of different evaluations introduced by different research for schema matching systems is provided in (Do et al., 2002). It specifies four different types of criteria to compare existing evaluations, including the evaluation of COMA (Do & Rahm, 2002), Cupid (Madhavan et al., 2001), Similarity Flooding (Melnik et al., 2002), SEMINT (Li & Clifton, 2000), and GLUE (Doan et al., 2002). These criteria include:

1) *Input:* Types of input data used, such as dictionaries used and schema specification.
2) *Output:* Information included in the match result, such as the mappings between different schema elements.
3) *Quality measures:* Measures used to assess the accuracy of the match result.
4) *Effort:* Types of needed manual effort measured in evaluations, such as pre-match and post-match efforts.

As (Do et al., 2002) concludes, it is difficult to compare results of different schema matching evaluations with each other, as these evaluations have been carried out in different ways and aimed at specific features. Authors further point at the requirement for a schema matching benchmark to make the comparison of results of different research evaluations possible. Such a generic benchmark has however not yet been defined and/or considered in any research work. So far, only a benchmark for evaluating the systems, which match XML Schemas is proposed in (Duchateau et al., 2007). This benchmark, called XBenchMatch, consists of quality measures for both schema matching and schema integration. It also provides some evaluation of the matching performance. In XBenchMatch it is assumed that for evaluating the quality of schema matching, mappings must be given as XML path correspondences (e.g. person.person_name - person.lastName). Furthermore, for evaluating the quality of "integrated schema", a number of measures are introduced as a part of XBenchMatch. However, these measures assume that also the correct (ideal) integrated schema is provided to the XBenchMatch. As such, the integrated schema which is generated as the output of the schema integration tool is compared against the ideal integrated schema. Both of these schemas need to be in the XML Schema format. For the purpose of evaluating the quality of "schema matching", XBenchMatch applies the four measures of *Precision, Recall, F-measure, and Overall,* as most other evaluation approaches also apply some of these methods. Detailed description of these measures is provided in the next section.

In summary, most evaluation approaches consider only the quality of schema matching. Although the XBenchMatch prototype measures the quality of both the schema matching and schema integration, it can only support the XML Schema formats. Furthermore, there are some assumptions of XBenchMatch (e.g. the availability of the ideal integrated schema) as explained in the previous paragraph, which makes the general use of this benchmark difficult. Since SASMINT works with relational schemas and due to other reasons addressed above, we could not apply XBenchMatch for the evaluation of SASMINT. Nevertheless, as addressed in Section 6.2 below in details, nearly all measures introduced in other competitive research are considered and applied for validation of SASMINT.

6.2 Quality Measures Used for Evaluating SASMINT

The main goal of SASMINT is to automate the schema matching and integration processes to the extent possible. In other words, our main concern for SASMINT is its effectiveness, in how accurately the system can identify the matching pairs and generate the integrated schema automatically. For this reason, we consider only the quality and accuracy measures in our evaluations of the SASMINT system, and do not take into account the time performance related measures and assessment. Performance measures depend on the underlying environment and the technologies used, and thus it is challenging to obtain neutral objective evaluations. Furthermore, for schema matching and integration, when performance is considered, it is not only related to how fast the system works but also how much time the user spends correcting the results manually. Therefore, when the system produces more accurate results, the user needs to spend less manual time and the overall performance increases. Therefore, the accuracy aim of SASMINT also improves the performance of its schema matching and schema integration.

We apply two types of quality measures in our experiments: 1) quality measures for schema matching, and 2) quality measures for schema integration. Details of these measures are provided in the next sub-section.

6.2.1 Quality Measures for Schema Matching

Similar to most other schema matching evaluations, we used the concepts of precision and recall from the information retrieval field (Cleverdon & Keen, 1966) for measuring the quality of schema matching. Precision (P) and Recall (R) are computed as follows:

$$P = \frac{x}{x+z} \text{ and } R = \frac{x}{x+y}$$

where x is the number of correctly identified similar strings (i.e. true positives), z is the number of strings found as similar, while actually they were not (i.e. false positives), and y is the number of those similar strings, which the system missed to identify (i.e. false negatives). As such the higher the precision value is and the higher the recall value is, the better is the system.

Although precision and recall measures are widely used for a variety of evaluation purposes, neither of them alone can accurately assess the match quality. For instance, recall can be increased by returning all pairs as similar, but increasing the number of false positives and thus decreasing the precision. Therefore, a measure combining precision and recall is better suited for accuracy evaluation. F-measure (Rijsbergen, 1979) is one such measure, combining recall and precision using the following formula. As such the higher the f-measure value is, the better is the system.

$$F = \frac{2}{\frac{1}{P} + \frac{1}{R}}$$

Another such measure, called Overall, is proposed by (Melnik et al., 2002). It is different from f-measure in that overall takes into account the amount of work needed to correct the results, namely to add the relevant needed matches that have not been discovered (false negatives) and to remove those matchers, which are incorrect but have been extracted by the matcher (false positives). Overall is always lower than f-measure, and if the precision is lower than 0.5, the result for overall becomes negative (Melnik et al., 2002) (Do et al., 2002). Overall, represented by O, and also called as accuracy, is defined by the following formula. As such the higher the overall value is, the better is the system.

$$O = R * (2 - \frac{1}{P})$$

As an example, assume that an automatic schema matching system correctly identifies 10 matches out of 25 real matches that can be identified manually by the user, and incorrectly identifies 4 other matches. In this case, the number of true positives (x) is 10, false negatives (y) is 25-10 = 15, and false positives (z) is 4. As a result, the system has the following precision, recall, f-measure, and overall values:

$$P = \frac{10}{10+4} = 0.71 \quad R = \frac{10}{10+15} = 0.40$$

$$F = \frac{2}{1/0.71 + 1/0.40} = 0.51 \qquad O = 0.40 * (2 - (1/0.71)) = 0.24$$

6.2.2 Quality Measures for Schema Integration

Quality measures used for the assessment of schema integration in SASMINT benefit from the ideas presented in (Batini et al., 1986). Schema merging and restructuring processes described in (Batini et al., 1986) aim at improving the resulting schema with respect to the following three qualities:

1) *Completeness:* Merged or integrated schema must cover concepts of all participating schemas.
2) *Minimality:* If the same concept is represented in more than one participating schemas, then the integrated schema must contain only a single representation of this concept. In other words, redundancies must be eliminated.
3) *Understandability:* Resulting integrated schema must be easily understandable by the user.

In evaluation of SASMINT, we are interested in quantitative objective measures. For this reason, we only consider measuring the *completeness* and *minimality* which will produce objective results. The *understandability* of SASMINT, while not measured rigorously, was satisfactory for the empirical tests we performed in the lab. The two measures of completeness and minimality applied to SASMINT are inspired by (Batini et al., 1986). However, within each of these measures, we have introduced two other measures for *key completeness* and *key minimality* to validate the generated primary and foreign keys when measuring the quality of SASMINT's schema integration approach. We belive that these added measures, which are missing from Batini's approach, are required for proper validation of schema integration. These measures are explained below:

- **Completeness Measure:** In the resulting integrated schema, all concepts (i.e. tables and columns in the relational schema) of both the donor and recipient schemas must be covered. Completeness measure determines how much this goal has been achieved.

  Therefore, $\forall\ c_i \in \{c_1, c_2, .. c_k\}$, where c_i is a concept in the donor or recipient schema and k is the total number of concepts in that donor or recipient schemas, $\exists c_j \in \{c_1, c_2, .. c_l\}$ where c_j is a concept of the integrated schema and $c_j \supseteq c_i$ and l is the number of concepts in the integrated schema. Taking this definition as the base, completeness of an integrated schema in SASMINT is measured using the following formula:

  $$m_{completeness} = \frac{n_{complete}}{n_{total}},$$

  where $n_{complete}$ is the number of concepts of recipient and donor schemas that are covered in the integrated schema and n_{total} (also l above) is the total number of concepts involved in donor and recipient schemas.

  Schema integration in SASMINT also handles primary and foreign keys, which will be referred to as "keys" from this point onward. Therefore, another completeness measure, called *key completeness*, is also defined for SASMINT to measure how many of the keys of the recipient and donor schemas are covered in the integrated schema. Given that $n_{completeKey}$ is the number of keys of recipient and donor schemas that are covered in the integrated schema and $n_{totalKey}$ is the total number of keys involved in donor and recipient schemas, the following formula measures the key completeness, $m_{completenessKey}$, of an integrated schema in SASMINT:

$$m_{completenessKey} = \frac{n_{completeKey}}{n_{totalKey}}$$

- **Minimality Measure:** The amount of redundancy in the resulting integrated schema must be minimal to the extent possible. Each joint and/or related concept of the donor and recipient schemas shall appear only once in the integrated schema. Namely, if the donor and recipient schemas have common concepts, only one of them must be represented in the integrated schema. Minimality measure identifies how many redundant concepts exist in the integrated schema.
Suppose that $\exists c_i \in \{c_1, c_2, ..c_k\}$, where c_i is a concept of the donor schema and k its total number of concepts, and $\exists c_j \in \{c_1, c_2, ..c_l\}$, where c_j is a concept of the recipient schema and l its total number of concepts. If $\exists c_x, c_y \in \{c_1, c_2, ..c_m\}$, where c_x and c_y are concepts of the integrated schema and m its total number of concepts, such that $c_i \equiv c_j \equiv c_x \equiv c_y$, then either c_x or c_y is redundant. Following formula is used to calculate the amount of redundancy in an integrated schema:

$$m_{redundancy} = \frac{n_{redundant}}{n_{total}},$$

where $n_{redundant}$ is the number of redundant concepts in the integrated schema and n_{total} is the total number of concepts introduced in the donor and recipient schemas.
Based on this formula, we derive the following formula to measure the minimality of the SASMINT integrated schema.

$$m_{minimality} = 1 - \frac{n_{redundant}}{n_{total}}$$

Similar to the case of completeness measure, another minimality measure, called *key minimality*, is also defined for SASMINT to determine if the resulting integrated schema is minimal considering its primary and foreign keys. Key minimality, $m_{minimalityKey}$, is measured using the following formula:

$$m_{minimalityKey} = 1 - \frac{n_{redundantKey}}{n_{totalKey}}$$

where the $n_{redundantKey}$ is the number of redundant primary and foreign keys in the integrated schema and the $n_{totalKey}$ is the total number of such keys introduced in the donor and recipient schemas.

6.3 Test Schemas

We have carried out the experimental evaluation of SASMINT using six pairs (donor and recipient) of "test schemas", characteristics of each of which are shown in Table 6.1 and the six pairs of schemas are represented in Appendix D. As for the evaluation of **schema matching**, each pair was matched by the SASMINT, and then the results were compared against the correct matches shown in Table 6.2. We carried out the same tests for schema matching in COMA++ (a leading competitor) and compared its results with the results of SASMINT. On the other hand, for evaluation of **schema integration**, three pairs of schemas all from the university domain (in Table 6.1) were integrated. Moreover, in order to evaluate the Sampler component, first five schema pairs were used in the Sampler tests. Details of these tests are provided in the next sections.

Table 6.1. Characteristics of Test Schemas

Test Schema Pair #	Short Name	Domain	Donor/ Recipient	Number of Tables	Number of columns
1	PO	Purchase Order	Recipient	5	27
			Donor	5	25
2	Hotel	Hotel	Recipient	6	21
			Donor	5	14
3	SDB	Biology	Recipient	9	21
			Donor	9	22
4	Univ1	University	Recipient	9	30
			Donor	5	22
5	Univ2	University	Recipient	9	38
			Donor	7	27
6	Univ3	University	Recipient	5	17
			Donor	3	10

We used schemas from four different domains: Schema Pair#1 contains two purchase order schemas that we generated ourselves. Schema Pair#2 consists of two hotel schemas. We modified the hotel schemas used for MAPONTO (An et al., 2006) evaluation tests. Similarly, in Schema Pair#3, we used a modified version of MAPONTO SDB schemas from the biology domain. In Schema Pair#4, we used MAPONTO schemas from the university domain, again after modifying them. Schema Pair#5 consists of university schemas that we generated. As Schema Pair#6, we modified the test schemas of Similarity Flooding (Melnik et al., 2002) from the university domain. We intentionally selected three pairs from the university domain in order to also use them for the schema integration evaluation. Therefore, the schema integration tests integrated six schemas from the university domain.

The correct matches represented in Table 6.2 are matches that are generated manually by ourselves. These constitute the source for verification of correctness of automatic matchings.

Table 6.2. Correct Matches between Schema Pairs

Schema Pair#	Type	Correct Matches
1	**table-table**	purchase_order=po, customer=buyer, product=item
	Colum-column	purchase_order:custNo=po:buyer_no, purchase_order:deliverDate=po:deliver_date, purchase_order:deliverCity=po:deliver_city, purchase_order:deliverStreet=po:deliver_street, purchase_order:deliverZip=po:deliver_zip, purchase_order:purchaseOrderNo=po:po_no, customer:street=buyer:buyer_street, customer:name=buyer:f_name, customer:telephone=buyer:phone, customer:city=buyer:buyer_city, customer:custNo=buyer:buyer_No, customer:name=buyer:l_name, customer:zip=buyer:buyer_zip, product:stock=item:stock, product:price=item:cost, product:productNo=item:item:item_no, product:productName=item:item:item_name
2	**table-table**	one_room=room, suite=room, town_house=room, num_beds=num_beds_attribute, on_floor=on_floor_attribute, smoking_preference=smoking_attribute
	column-column	one_room:roomNum=room:roomNum, one_room:hasNumBedsAttribID=room:numBedsAttribID, one_room:hasOnFloorAttribID=room:onFloorAttribID, one_room:hasSmokingPreferenceAttribID=room:smokingOrNoAttribID, one_room:oneRoomID=room:roomID, suite:suiteID=room:roomID, suite:roomNum= room:roomNum, suite:hasNumBedsAttribID=room:numBedsAttribID, suite:hasOnFloorAttribID=room.onFloorAttribID, suite:hasSmokingPreferenceAttribID=room.smokingOrNoAttribID, town_house:townHouseID=room.roomID, townhouse:roomNum=room.roomNum, town_house:hasNumBedsAttribID=room:numBedsAttribID, town_house:hasOnFloorAttribID=room:onFloorAttribID, town_house:hasSmokingPreferenceAttribID=room:smokingOrNoAttribID, num_beds:numBedsID=num_beds_attribute:numBedsAttributeID, num_beds:numBedsAttrib=num_beds_attribute:numBedsAttrib, on_floor:onFloorID=on_floor_attribute:onFloorAttributeID, on_floor:onFloorAttrib=on_floor_attribute:onFloorAttrib, smoking_preference:smokingPreferenceID=smoking_attribute:smokingAttributeI D, smoking_preference:smokingPreferenceAttrib=smoking_attribute:smokingAttrib
3	**table-table**	diagnoses=diagnoses, donor=donor, sample=sample, family_history=family_history, life_style_factors=life_style_factors, lab_test=lab_test, medications=medications, animal_donor=donor, human_donor=donor
	column-column	animal_donor:strain=donor:strain, diagnoses:diagID=diagnoses:diagID, donor:gender=donor:gender, donor:id=donor:id, donor:species=donor:species, family_history:histID=family_history:histID, human_donor:dob=donor:dob, lab_test:testID=lab_test:testID, life_style_factors:factID=life_style_factors:factID, medications:medicID=medications:medicID, sample:name=sample:name, animal_donor:species=donor:species, animal_donor:animalID=donor:id, human_donor:gender=donor:gender, human_donor:humanID=donor:id
4	**table-table**	course=course, student=student, faculty_member=academic_staff

Schema Pair#	Type	Correct Matches
	column-table	faculty_member:researchInterest=areasOfInterest
	Column-column	faculty_member:email=academic_staff:email, faculty_member:faculty_member_id=academic_staff:academic_staff_id, faculty_member:personName=academic_staff:name, course:number=course:courseNumber, course:courseTitle=course:courseTitle, course:instructor=course:instructor, course:prerequisites=course:prerequisite, course:description=course:description, student:studentName=student:student_name, student:email=student:email, student:advisor=student:supervisor, student:student_id=student:student_id,
5	**table-table**	university=acedemic_institution, program=program, academic_programme=academic_programme, department=department, course=academic_course, academic_staff_member=university_academic_instructor
	column-column	university:university_ID= academic_institution:academic_institution_ID, university:UNIVERSITY_NAME= academic_institution:ACADEMIC_INSTITUTION_NAME, university:UNIVERSITY_WEBSITE=academic_institution:ACADEMIC_INSTI TUTION_WEBSITE, program:program_ID= program:program_ID, program:PROGRAM_NAME=program:PROGRAM_NAME, program:PROGRAM_DESC=program:PROGRAM_DESC, academic_programme:academic_programme_ID=academic_programme: academic_programme_ID, academic_programme:ACADEMIC_YEAR=academic_programme:YEAR, academic_programme:ACADEMIC_SEMESTER=academic_programme:SEME STER, academic_programme:PROGRAM_REF= academic_programme:PROGRAM_REF, department:department_ID=department:department_ID, department:DEPT_NAME=department:DEPT_NAME, course:course_ID=academic_course:academic_course_ID, course:COURSE_NAME=academic_course:ACADEMIC_COURSE_NAME, course:COURSE_CREDITS = academic_course:ACADEMIC_COURSE_CREDITS, course:COURSE_PROVIDER=academic_course:ACADEMIC_COURSE_PRO VIDER, course:COURSE_INSTRUCTOR= academic_course:ACADEMIC_COURSE_INSTRUCTOR, academic_staff_member:academic_staff_member_ID = university_academic_instructor:university_academic_instructor_ID, academic_staff_member:STAFF_NAME=university_academic_instructor:NAME , academic_staff_member:STAFF_EMAIL = university_academic_instructor:ELECTRONIC_MAIL, academic_staff_member:STAFF_PHONE=university_academic_instructor.TELE PHONE
6	**table-table**	professor=professor, student=student, workson=workson
	table-column	address=professor:address
	column-column	professor:Id=professor:Id, professor:Name=professor:Name, professor:Sal=professor:Salary, student:Name=student:Name, student:GPA=student:GradePointAverage, student:Yr=student:Year, workson:Name=workson:StudentName, workson:Proj=workson:Project

6.4 Setup for the Experimental Evaluation

We compared the "schema matching" component of SASMINT against one of the state of the art system, COMA++ (Aumueller et al., 2005). We selected COMA++ research prototype, because it is the most complete schema matching tool so far developed, consisting of a library of variety of matching algorithms and a sophisticated GUI. SASMINT and COMA++ are comparable, since they both support matching of relational schemas and aim at providing similar functionalities. Of course not all algorithms or metrics that these two systems apply are the same. Furthermore, how they combine the results of different algorithms is not the same either. Output of the schema matching is given in the range [0-1] in both systems. However, it is not clear in COMA++, in what format the results of schema matching are stored internally. In other words, COMA+++ has an internal repository where the results are stored, but how the results are represented there is not clear.

Before starting the evaluation tasks, we inserted a number of abbreviations and their long forms into the abbreviation lists of both systems. One important difference between SASMINT and COMA++ is that SASMINT uses WordNet for semantic matching, whereas COMA++ requires the user to add all needed synonyms in the schema domains manually. Since WordNet might not contain all semantic relationships among the concepts of schemas, in order to make a fair comparison, we did not make any addition to COMA++'s default synonyms list, to make a fair comparison. Furthermore, COMA++ uses only the synonymy relationship; on the other hand, SASMINT also makes use of the IS-A relationships as well as gloss overlaps, which are available in the WordNet dictionary.

Representation of schemas through the GUI is also different for the two systems. COMA++ does not explicitly show foreign keys. Instead of showing the foreign key column, it displays the table that is pointed by the foreign key. However, in some cases this functionality of COMA++ does not work as expected.

Several different metrics or algorithms are considered and combined in both systems, in the manner that is explained below:

- **For SASMINT:** We selected the default strategy of SASMINT for combining the algorithms, which is the weighted sum of them with equal weights applied to each algorithm in each group of syntactic, semantic, and structure matching. Although not the default approach, rather assigning appropriate weights for each match task would give better results, we decided to use SASMINT's default strategy in order to make a fair comparison with COMA++. In other words, in real practice, the results of SASMINT would be better than what they are in these tests. Sampler could help the user to identify appropriate weights for the linguistic matching algorithms. The reported evaluation results in Section 6.5 and Appendix E are without applying the Sampler component of SASMINT. Results of experiments showing how Sampler can accomplish this improvement of results accuracy, i.e. how these weights affect the match results, are addressed in Section 6.7.

- **For COMA++:** We used the default matching strategy of COMA++, which is called COMA. The COMA matcher combines the *name*, *path*, *leaves*, *parents*, and *siblings* matchers, by averaging them. In their tests, this combination was the winner and that is why we selected it.

We used the default threshold, which is 0.5, in the experiments. As for the selection of match results, we used two different approaches that we call as "select all above threshold" and "select max above threshold", as detailed below. Please note that while the results of "select all above threshold" are presented in Section 6.5, the results of "select max above threshold" are presented in Appendix E.

1) **Select All above Threshold:** Selecting all matched pairs that have the similarity above a certain threshold value.

2) **Select Max above Threshold:** Selecting the pairs with the maximum similarity. In other words, whenever there is more than one concept matching a single concept in a schema, the one with the highest similarity is selected as the matching candidate. SASMINT and COMA++ use different strategies for selecting the maximal similar pairs. SASMINT's approach is explained in Chapter 4. COMA++'s default strategy works as follows: When there is more than one match to the same concept, the one with the highest similarity is selected if the difference between the similarity values is more than 0.0080.

We also carried out tests in order to validate the Sampler component that helps to identify appropriate weights for each linguistic matching algorithm in the schema matching process. The results of these tests are presented in Section 6.6. The first five schema pairs (Schema Pairs #1, 2, 3, 4, and 5) were selected as the test schemas when evaluating Sampler.

Although we compared SASMINT with COMA++ for the purpose of schema matching, we could not carry out functionality comparison for schema integration between them. COMA++ provides a simple schema merging functionality, but it is limited and not comparable to SASMINT's schema integration. To the best of our knowledge, there is no other system supporting both schema matching and schema integration. Therefore, we evaluated the integration component of SASMINT alone. For this purpose, we used the six schemas from the university domain, introduced as Schema Pair#4, 5, and 6. Since the aim of schema integration is integrating two schemas at a time, based on the correspondences between them, we corrected the wrong or missing matches after the schema matching step and then continued with the integration process.

6.5 Evaluation of Schema Matching – For "select all above threshold" strategy

In the first experiment that we performed to evaluate SASMINT and compare it with COMA++, we used the "select all above threshold" strategy. We present the results of this experiment in Figures 6.1 through 6.8. Correspondingly, we provide detailed explanations about the four comparison results of precision, recall, f-measure, and overall in the following paragraphs 6.5.1 to 6.5.4. Although the results gained from applying this strategy are worse than the "select max above threshold" strategy, this strategy is important when there is a need for suggesting multiple candidates for each schema element and leaving it to the user to identify the correct match among the alternatives. Namely, instead of proposing only one matched candidate for each schema element, which could be incorrect, the system suggests all possible match candidates, which makes it easier for the user to determine the final match result.

6.5.1 Evaluation of Schema Matching Using Precision

Precision shows how correct the system works. Precision values for COMA++ and SASMINT are shown in Figure 6.1 and Figure 6.2 respectively. Since in the “select all above threshold” strategy, all match pairs with similarity above the threshold are selected, the number of false positives was high for some schema pairs. Especially for schemas that consisted element names with more than one token, precision was low. In our test cases, these schemas are the purchase order schemas (Schema Pair#1), university schemas of Maponto (Schema Pair#4), and the university schema that we generated ourselves (Schema Pair#5). In these cases, the low precision was due to the fact that for element names containing similar tokens, although the whole names were different, the final similarity result was usually above the threshold. Furthermore, the systems interpreted and treated all tokens equally, while some tokens had none or little effect in the meaning. For example, “deliverDate” and “deliver_zip” were identified as similar because both names contained the token “deliver”. However, the first one is the name of the column that contains the date of delivery, whereas, the second one is the name of the column that contains the zip code information. In such situations, SASMINT and COMA++ both found similarity values around 0.5. These cases could have been prevented by raising the threshold value, but then some correct matches could have been also missed. When precision was considered, SASMINT achieved almost 9 times better than COMA++ for the Hotel schemas test case. For other schemas, except for Schema Pair#6 from the university domain, for which COMA++ achieved just a little bit better (around 1.05 times), SASMINT achieved on average 2 times better than COMA++. Precision of SASMINT was on the average 0.58, whereas that of COMA++ was 0.26. This result was because of the high number of false positives identified by COMA++. In other words, COMA++ identified high number of irrelevant matches, which can be a bigger problem when schemas being compared are large.

6.5.2 Evaluation of Schema Matching Using Recall

Recall shows how well the system finds all true matches and thus it indicates the completeness of the applied system. The average recall for COMA++ was 0.92, whereas for SASMINT it was 0.85. Figures 6.3 and 6.4 show the recall values for COMA++ and SASMINT

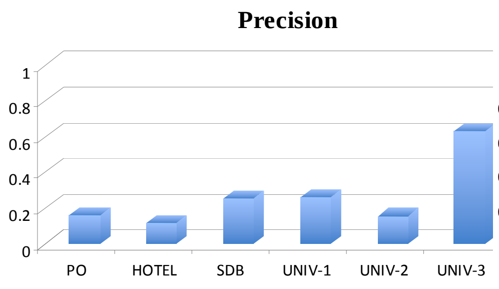

Fig. 6.1. Precision values for **COMA++** - select all above threshold strategy

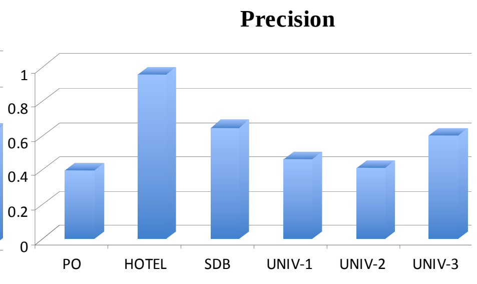

Fig. 6.2. Precision values for **SASMINT** - select all above threshold strategy

respectively. For UNIV-3 schemas, they both had the recall value of 1.0 and for SDB schemas, SASMINT was 1.14 times better than COMA++. For the remaining schemas, which were purchase order, hotel, UNIV-1, and UNIV-2, COMA++ achieved a bit (on the average 1.17 times) higher than SASMINT. However, it should be noted that this happened at the expense of very low precision values for COMA++. That means, in order to achieve just a bit higher recall values, COMA++ sacrificed the precision, resulting in very low precision values for these test cases, as indicated in Figures 6.1 and 6.2. This is due to the fact that there is an inverse relationship between precision and recall. Since COMA++ tries to find all possible matches, it also identifies a large number of false positive matches, which decrease the precision. SASMINT missed some of the correct matches, mostly due to low semantic similarity values that it could compute for some name pairs, such as (product, item) and (suite, room). Especially the gloss-based measure was not as successful as expected. Since the last version of WordNet (3.0) is not available yet for the Windows operating system, we had to use the previous version (2.0) of WordNet. We think that when the new version is ready, WordNet will provide more types of semantic relationships, and therefore the semantic similarity values for both path-based and gloss-based measures of SASMINT will be much more enhanced.

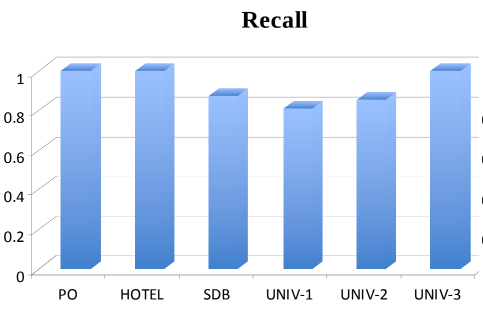

Fig. 6.3. Recall values for **COMA++** - select all above threshold strategy

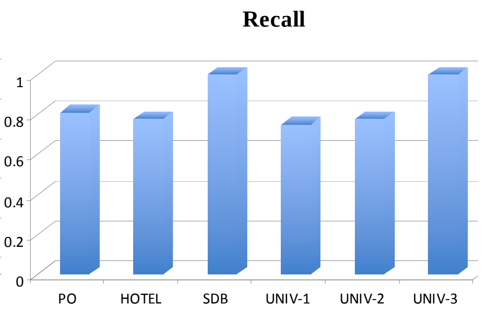

Fig. 6.4. Recall values for **SASMINT** - select all above threshold strategy

6.5.3 Evaluation of Schema Matching Using F-Measure

As stated before, f-measure is used to combine the results of precision and recall. In other words, the higher the f-measure value, the better is the quality of the system. Most evaluation experiments in fact use f-measure as the measure to compare the systems, and not the individual precision and recall values. When f-measure is considered, the difference between SASMINT and COMA++ becomes clearer. This is due to the fact that f-measure considers both the precision and recall, and although recall values for COMA++ were a bit higher than those for SASMINT, precision of SASMINT was much better than that of COMA++, which results in higher f-measure values for SASMINT. As it is clear from the Figures 6.5 and 6.6, f-measure values for SASMINT were on average 2.2 times higher than those for COMA++ for all schema pairs, except the last schema pair (UNIV-3), for which they almost achieved the same. What can be inferred from these results is that the quality of results achieved by the SASMINT system is much higher than COMA++, considering the f-measure evaluation.

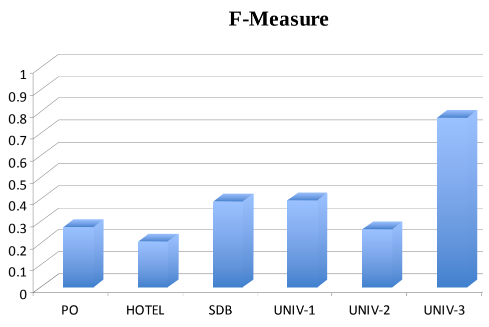

Fig. 6.5. F-measure values for **COMA++** - select all above threshold strategy

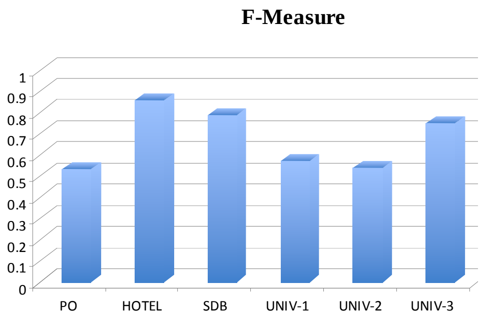

Fig. 6.6. F-measure values for **SASMINT** - select all above threshold strategy

6.5.4 Evaluation of Schema Matching Using Overall

Similar to f-measure, overall also represents a combination of precision and recall. Its value is smaller than both Precision and Recall and it can even have negative values, if the number of the false positives is more than the number of true positives. The overall indicator measures the overall accuracy of the system. It aims to identify how much manual effort is required in order to identify all correct matches. Overall values for SASMINT were consistently much higher than those for COMA++. In some cases, for example the hotel schemas, SASMINT achieved overall value around 0.7. In the case of UNIV-2 and purchase order schemas, on the other hand, it did not do very well because of the high number of false positive matches that SASMINT identified for these schemas. Since the number of false positive matches for COMA++ was very high, it had very low overall values, which means a lot of manual intervention by user is required in order to remove these wrongly identified matches. This result is very clear especially for the first five schema pairs (purchase order, hotel, SDB, and the UNIV-1, and UNIV-2 schemas). Evaluation results for COMA++ and SASMINT, based on the overall values are shown in Figures 6.7 and 6.8 respectively. Since the aim of such systems is to achieve the schema matching as automatically as possible, the amount of required human intervention is an important measure for comparing these systems. The lesser manual effort is required, the better the system is.

6.6 Evaluation of Schema Matching with Sampler

In order to evaluate the Sampler component, we carried out tests using the first five schema pairs introduced in Table 6.1. As explained before, Sampler is used to compute the weights only for linguistic matching algorithms. In test cases where the element names from two schemas were highly similar, we set the threshold to a value higher than 0.5. In other cases, we used the default threshold value, which was 0.5.

After setting the threshold value, we performed the tests using both equal weights for the linguistic matching algorithms and the weights suggested by Sampler for these algorithms. We

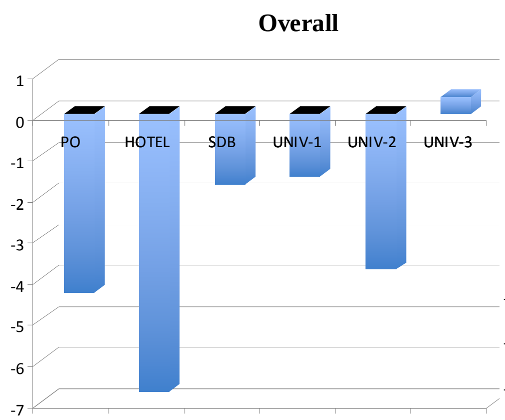

Fig. 6.7. Overall values for **COMA++** - select all above threshold strategy

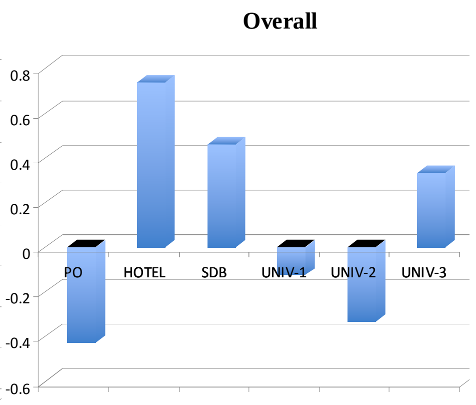

Fig. 6.8. Overall values for **SASMINT** - select all above threshold strategy

used the "select max above threshold" strategy for the Sampler tests. Furthermore, we did not use the last schema pair (schema pair#6) in the tests, because for this pair, precision, recall, f-measure, and overall values were already identified as 1 in the tests using the "select max above threshold" strategy, when equal weights were used. Details of tests with the Sampler component are explained below.

6.6.1 Test with Purchase Order Schemas-PO (Schema Pair#1)

In this test, we used the default threshold value, which was 0.5. We provided the similar pairs shown in Table 6.3 to Sampler, which computed the weights for semantic similarity algorithms shown in the same table. Results for precision, recall, f-measure, and overall were already high before the Sampler component was used. With the use of Sampler, (product, item) pair was correctly identified as similar, which was false negative before. As the result, Sampler helped to increase the values of recall, f-measure, and overall, as shown in Figure 6.9. Precision was 1 both before and after the use of the Sampler component.

Table 6.3. Similar Pairs and Computed weights for Schema Pair#1

Similar Pairs
Semantically Similar Pairs: customer - buyer product – item
Computed Weights
Wu and Palmer: 1.0 Gloss: 0.0

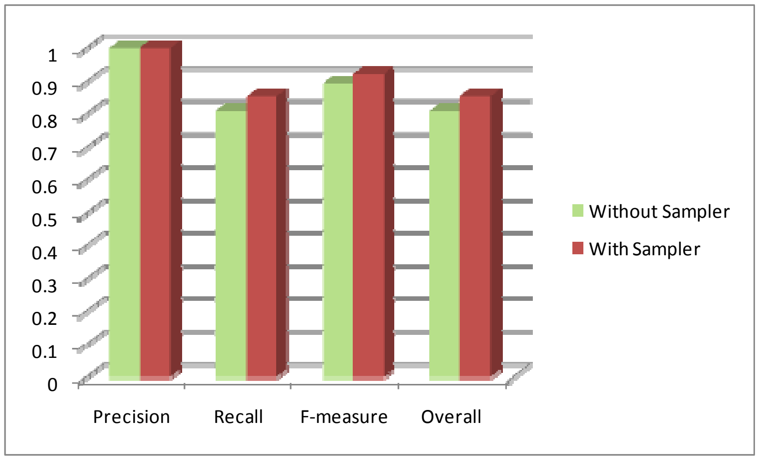

Fig. 6.9. Results of the Test with Schema Pair#1

6.6.2 Test with Hotel Schemas-Hotel (Schema Pair#2)

In this test, we set the threshold value as 0.7. We provided the similar pairs shown in Table 6.4 to Sampler, which computed the weights for syntactic similarity algorithms shown again in Table 6.4. When SASMINT used these weights for matching the hotel schemas, results for recall, f-measure, and overall were on the average 1.75 times (57%) better than the case without the use of Sampler. This result can be seen in Figure 6.10.

Table 6.4. Similar Pairs and Computed weights for Schema Pair#2

Similar Pairs
Syntactically Similar Pairs: smoking_Preference_Attrib - smoking_Attrib smoking_Preference_ID - smoking_Attribute_ID hasSmokingPreferenceAttribID - smokingOrNoAttribID on_floor - on_Floor_attribute numBedsID – numBedsAttributeID
Computed Weights
Levenshtein: 0.0 Jaccard: 0.11 LCS: 0.20 Monge-Elkan: 0.22 Jaro: 0.22 TF*IDF: 0.25

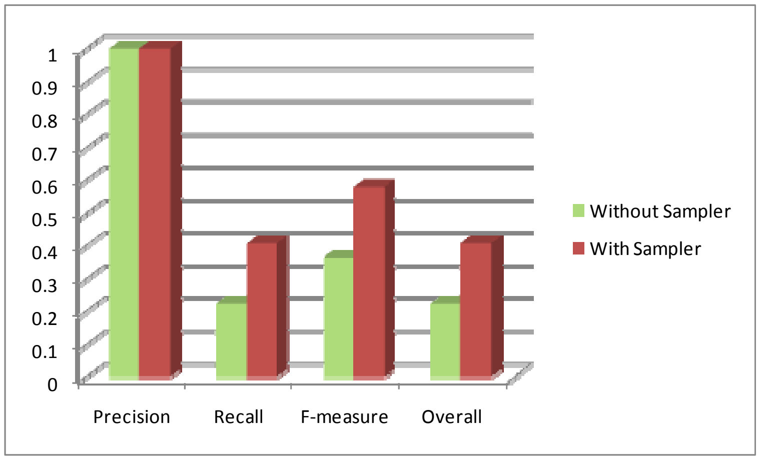

Fig. 6.10. Results of the Test with Schema Pair#2

6.6.3 Test with Biology Schemas-SDB (Schema Pair#3)

Two schemas (donor and recipient) in Schema Pair#3 use the same names for most of their schema elements. We set the threshold value to 0.9 and provided the two similar pairs of (animal_donor-donor) and (human_donor-donor). Sampler computed 1.0 for the weight of Monge-Elkan distance metric and 0.0 for other syntactic similarity metrics, as shown in Table 6.5. When we ran SASMINT with these weights, the results were as shown in Figure 6.11. There was a slight decrease in Precision when Sampler was used. This was due the two false positives (donor, donor_visit) and (donorID, donorVisitID). However, recall, f-measure, and overall were all improved.

Table 6.5. Similar Pairs and Computed weights for Schema Pair#3

Similar Pairs
<u>Syntactically Similar Pairs:</u> animal_donor - donor human_donor – donor
Computed Weights
Levenshtein: 0.0 Jaccard: 0.0 LCS: 0.0 Jaro: 0.0 TF*IDF: 0.0 Monge-Elkan: 1.0

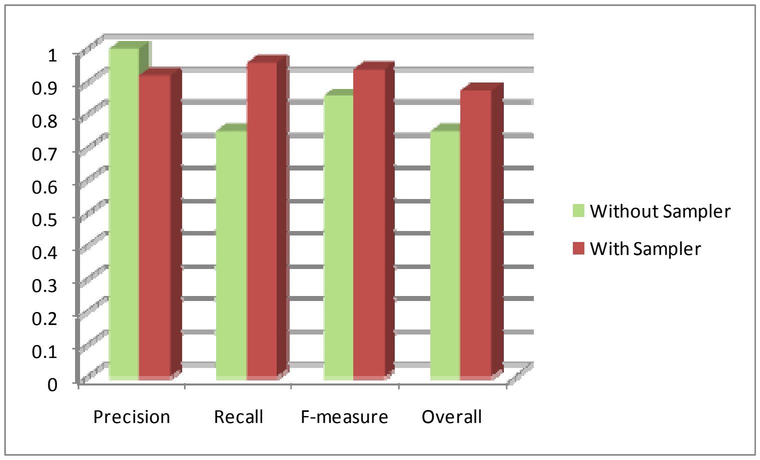

Fig. 6.11. Results of the Test with Schema Pair#3

6.6.4 Test with University Schemas-UNIV1 (Schema Pair#4)

For the test with these schema pairs, we set the threshold value as 0.7. We provided syntactically similar pairs, shown in Table 6.6. As shown in Figure 6.12, precision was slightly better before, whereas recall, f-measure, and overall values were higher with the use of Sampler. Since we provided Sampler (personName, name) as the syntactically similar pair, the *personName* column of the *faculty_member* table was successfully matched to the *name* column of the *academic_staff* table. However, at the same time, it incorrectly matched the *personName* column of the *faculty_member* and the *name* column of the *admin_staff* table. This in turn, increased the number of false positives, and thus slightly decreased the precision. However, since Sampler helped to identify more number of similar pairs, recall was much better than the case without the Sampler. As the result, f-measure and overall were better with the use of Sampler, as shown in Figure 6.12.

Table 6.6. Similar Pairs and Computed weights for Schema Pair#4

Similar Pairs
Syntactically Similar Pairs: number - courseNumber personName - name researchInterest – areasOfInterest
Computed Weights
Levenshtein: 0.0 Jaccard: 0.0 Jaro: 0.15 LCS: 0.15 TF*IDF: 0.3 Monge-Elkan: 0.4

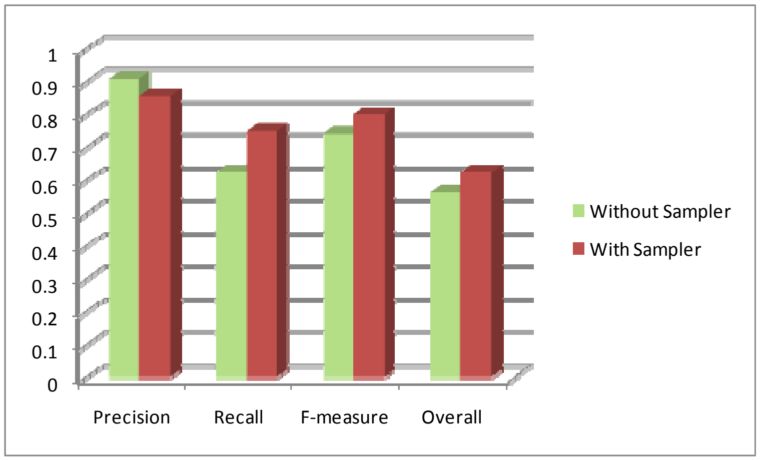

Fig. 6.12. Results of the Test with Schema Pair#4

6.6.5 Test with University Schemas-UNIV2 (Schema Pair#5)

In this test, we set the threshold value to 0.7 and provided the pairs shown in Table 6.7 to Sampler. Weights computed by Sampler for syntactic similarity algorithms are presented in Table 6.7. Similar to the case addressed in Section 6.7.4, with the use of Sampler the precision decreased because some new false positive pairs were introduced. For example, the *university_name* column of the *university* table and the *name* column of the *university_student* table were identified as similar, which was incorrect. However, since the value of recall was much higher when Sampler was used, f-measure and overall increased, as presented in Figure 6.13 also.

Table 6.7. Similar Pairs and Computed weights for Schema Pair#5

Similar Pairs
Syntactically Similar Pairs: academic_semester - semester course_id - academic_course_id course_instructor - academic_course_instructor staff_name - name course - academic_course
Computed Weights
Levenshtein: 0.0 Jaccard: 0.0 Jaro: 0.0 LCS: 0.18 Monge-Elkan: 0.35 TF*IDF: 0.47

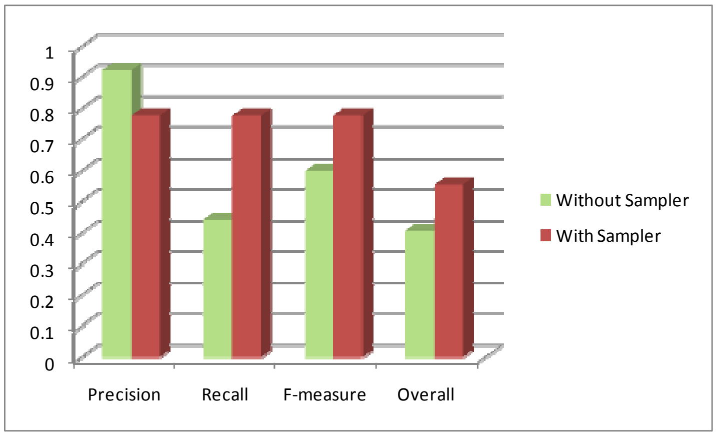

Fig. 6.13. Results of the Test with Schema Pair#5

6.7 Evaluation of Schema Integration Performance

In order to evaluate the schema integration component of SASMINT, we used schema pairs from the university domain. The three university schema pairs introduced in Table 6.1 which are Schema Pairs#4, 5, and 6 are used for this purpose. As addressed further below, please note that the Appendix F provides details of the steps of evaluation.

Figures 6.14 through 6.16 show the elements of these pairs. SASMINT integrates two schemas at a time, therefore, incrementally generating the final integrated schema. The steps we followed for integrating these six schemas are explained below. We have selected to start with larger schemas first, namely Schema Pair#5.

First Schema

academic_programme {academic_programme_ID, ACADEMIC_YEAR, ACADEMIC_SEMESTER,PROGRAM_REF}

academic_staff_member {academic_staff_member_ID, STAFF_NAME, STAFF_EMAIL, STAFF_PHONE, STAFF_FAX, STAFF_IDENTIFICATION_NUM, STAFF_BIRTHDATE}

campus {campus_ID, CAMPUS_NAME, CAMPUS_LOCATION, UNVCAMPUS}

course {course_ID, COURSE_NAME, COURSE_CREDITS, COURSE_PROVIDER, COURSE_INSTRUCTOR}

department {department_ID, DEPT_NAME, FACULTY_REF}

faculty {faculty_ID, FACULTY_NAME, DEAN_REF, UNIVERSITY_REF}

program {program_ID, PROGRAM_NAME, PROGRAM_DESC}

registration {registration_ID, REGISTRATION_ACADEMICSTAFFMEMBER_REF, REGISTRATION_COURSE_REF, REGISTRATION_ACADEMICPROGRAMME_REF}

university {university_ID, UNIVERSITY_NAME, UNIVERSITY_WEBSITE, UNIVERSITY_ESTABLISHMENT_DATE }

Second Schema

academic_course {academic_course_ID, ACADEMIC_COURSE_NAME, ACADEMIC_COURSE_CREDITS, ACADEMIC_COURSE_PROVIDER, ACADEMIC_COURSE_INSTRUCTOR}

academic_institution {academic_institution_ID, ACADEMIC_INSTITUTION_NAME, ACADEMIC_INSTITUTION_WEBSITE}

academic_programme {academic_programme_ID, YEAR, SEMESTER, PROGRAM_REF }

department {department_ID, DEPT_NAME, UNIVERSITY_REF}

program {program_ID, PROGRAM_NAME, PROGRAM_DESC}

university_academic_instructor {university_academic_instructor_ID, NAME, ELECTRONIC_MAIL, OFFICE_ADDRESS, TELEPHONE}

university_student {university_student_ID, NAME, ELECTRONIC_MAIL, TELEPHONE}

Fig. 6.14. Schema Pair#5 (UNIV-2)

First Schema

address {Id, Street, City, PostalCode}
payrate {Rank, HrRate}
professor {Id, Name, Sal, addr}
student {Name, GPA, Yr}
workson {Name, Proj, Hrs, ProjRank}

Second Schema

professor {Id, Name, Salary, Address}
student {Name, GradePointAverage, Year}
workson {StudentName, Project, Expenses}

Fig. 6.15. Schema Pair#6 (UNIV-3)

First Schema

course {number, courseTitle, description, prerequisites, instructor}
faculty_member {faculty_member_id, personName, personTitle, homepage, researchInterest, email}
paper {paperTitle, description, publicationYear}
paper_author {paperTitle, author}
person_project {person, projectTitle}
project {projectTitle, description, link}
seminar {about, speaker, date, location}
student {student_id, studentName, advisor, email}
year {yr}

Second Schema

academic_staff {academic_staff_id, name, office, email, phone}
admin_staff {admin_staff_id, name, office, email, phone}
areas_of_interest {interest_id, area}
course {courseNumber, courseTitle, instructor, area, description, prerequisite}
student {student_id, student_name, email, supervisor}

Fig. 6.16. Schema Pair#4 (UNIV-1)

Step-1: First Schema of Schema Pair#5 + Second Schema of Schema Pair#5

At the first step of schema integration test, SASMINT system has integrated two schemas of the Schema Pair#5, shown in Figure 6.14, resulting in the integrated schema, elements of which are shown in Figure 6.17. During the integration process, one redundancy was automatically generated, which was the "UNIVERSITY_REF" column of the "department" table. Therefore, the result of minimality measure was 0.99, which is a substantial automated achievement. When key minimality is considered, one redundant foreign key relationship was defined on the same "UNIVERSITY_REF" column, which resulted in a key minimality of 97%. Although the resulting integrated schema had one redundant element and foreign key, it covered all the elements and keys of two source schemas. Therefore, the result is considered as 100% complete and 100% key complete, which is again a substantial automated achievement. Further details of this step are provided in Appendix F.

Step-2: Integrated Schema#1 + First Schema of Schema Pair#6

At this step, SASMINT integrated the Integrated Schema#1 and the first schema of the Schema Pair#6, generating the Integrated Schema#2. Figure 6.18 shows only newly added tables and those tables that had changes in their columns. Due to the redundant "UNIVERSITY_REF" column and the foreign key defined on it, the result of minimality measure was 0.99 and the key minimality measure was 0.97. However, since all the concepts and keys of the first three schemas integrated (first schema of the Schema Pair#5, second schema of the Schema Pair#5, and first schema of the Schema Pair#6) were covered in the integrated schema, completeness and key completeness were again 100%.

INTEGRATED_1:university {university_ID (PK), UNIVERSITY_NAME, UNIVERSITY_ESTABLISHMENT_DATE, UNIVERSITY_WEBSITE}

INTEGRATED_1:program{program_ID (PK), PROGRAM_NAME, PROGRAM_DESC}

INTEGRATED_1:*academic_programme*{academic_programme_ID (PK), ACADEMIC_YEAR, ACADEMIC_SEMESTER, PROGRAM_REF}

INTEGRATED_1:department{department_ID (PK), DEPT_NAME, UNIVERSITY_REF(FK), FACULTY_REF(FK)}

INTEGRATED_1:course{course_ID (PK), COURSE_NAME, COURSE_CREDITS, COURSE_PROVIDER (FK), COURSE_INSTRUCTOR(FK)}

INTEGRATED_1:academic_staff_member{academic_staff_member_ID (PK), STAFF_NAME, STAFF_IDENTIFICATION_NUM, STAFF_FAX, STAFF_BIRTHDATE, OFFICE_ADDRESS, STAFF_EMAIL, STAFF_PHONE}

INTEGRATED_1:campus{campus_ID (PK), CAMPUS_NAME, CAMPUS_LOCATION, UNVCAMPUS (FK)}

INTEGRATED_1:faculty{faculty_ID (PK), FACULTY_NAME, DEAN_REF(FK), UNIVERSITY_REF (FK)}

INTEGRATED_1:registration{registration_ID (PK), REGISTRATION_ACADEMICSTAFFMEMBER_REF(FK), REGISTRATION_COURSE_REF(FK), REGISTRATION_ACADEMICPROGRAMME_REF(FK)}

INTEGRATED_1:university_student{university_student_ID (PK), NAME, ELECTRONIC_MAIL, TELEPHONE}

Fig. 6.17. Elements of Integrated Schema#1

INTEGRATED_2:payrate{Rank (PK), HrRate}

INTEGRATED_2:workson{Name, Proj, Hrs, ProjRank (FK)}

INTEGRATED_2:address{Id (PK), Street, City, PostalCode}

INTEGRATED_2:academic_staff_member{academic_staff_member_ID (PK), STAFF_NAME, STAFF_IDENTIFICATION_NUM, STAFF_FAX, STAFF_BIRTHDATE, STAFF_EMAIL, STAFF_PHONE, Sal, addr(FK)}

INTEGRATED_2:university_student{university_student_ID (PK), NAME, ELECTRONIC_MAIL, TELEPHONE, GPA, Yr}

Fig. 6.18. New Elements of Integrated Schema#2

Step-3: Integrated Schema#2 + Second Schema of Schema Pair#6

At Step-3, SASMINT generated Integrated Schema#3, by integrating the Integrated Schema#2 and the second schema of the Schema Pair#6. The only change in the new integrated schema was the addition of one new column, called "Expenses" to the "workson" table. Due to the redundant "UNIVERSITY_REF" column and the foreign key defined on it, the resulting schema was again 99% minimal and 97% key minimal. However, it was again 100% complete considering both the concepts and keys.

Step-4: Integrated Schema#3 + First Schema of Schema Pair#4

In Step-4, SASMINT integrated the Integrated Schema#3 and the first schema of the Schema Pair#4, resulting in the Integrated Schema#4. Figure 6.19 shows only the newly added tables and those tables that had changes in their columns at this step. Minimality and key minimality were 0.99 and 0.98 respectively, because of the redundant "UNIVERSITY_REF" column and the foreign key. Considering the concepts, schema was 100% complete, but since three foreign keys were missed, as explained in Appendix F, the key completeness was 0.95 after this step.

INTEGRATED_4:academic_staff_member{academic_staff_member_ID (PK), STAFF_NAME, STAFF_IDENTIFICATION_NUM, STAFF_FAX, STAFF_BIRTHDATE, STAFF_EMAIL, STAFF_PHONE, Sal, addr(FK) personTitle, homepage, researchInterest}

INTEGRATED_4:university_student{university_student_ID (PK), NAME, ELECTRONIC_MAIL, TELEPHONE, GPA, Yr, advisor(FK)}

INTEGRATED_4:paper{paperTitle (PK), description, publicationYear(FK)}

INTEGRATED_4:paper_author{paperTitle(PK)(FK), author(PK)(FK) }

INTEGRATED_4:person_project{person (PK)(FK), projectTitle (PK)(FK)}

INTEGRATED_4:seminar{about (PK), speaker (FK), location, date}

INTEGRATED_4:year{yr (PK)}

INTEGRATED_4:project{projectTitle (PK), description, link}

INTEGRATED_4:workson{Name, Hrs, ProjRank, Expenses}

Fig. 6.19. New Elements of Integrated Schema#4

Step-5: Integrated Schema#4 + Second Schema of Schema Pair#4

In the final step of schema integration, SASMINT integrated the Integrated Schema#4 and the second schema of the Schema Pair#4. Final integrated schema is called Integrated Schema#5. Figure 6.20 shows the elements of the final integrated schema. This schema was 99% minimal and 99% key minimal. Redundancy was again due to the "UNIVERSITY_REF" column and the foreign key defined on it. Although all the concepts of six schemas integrated were covered in the final schema, resulting in 100% completeness, two more foreign keys were

INTEGRATED_5:university {university_ID (PK), UNIVERSITY_NAME, UNIVERSITY_ESTABLISHMENT_DATE, UNIVERSITY_WEBSITE}

INTEGRATED_5:program{program_ID (PK), PROGRAM_NAME, PROGRAM_DESC}

INTEGRATED_5:*academic_programme*{academic_programme_ID (PK), ACADEMIC_YEAR, ACADEMIC_SEMESTER, PROGRAM_REF}

INTEGRATED_5:department{department_ID (PK), DEPT_NAME, UNIVERSITY_REF(FK), FACULTY_REF(FK)}

INTEGRATED_5:course{course_ID (PK), COURSE_NAME, COURSE_CREDITS, COURSE_PROVIDER (FK), COURSE_INSTRUCTOR(FK), description, prerequisites, area}

INTEGRATED_5:campus{campus_ID (PK), CAMPUS_NAME, CAMPUS_LOCATION, UNVCAMPUS (FK)}

INTEGRATED_5:faculty{faculty_ID (PK), FACULTY_NAME, DEAN_REF(FK), UNIVERSITY_REF (FK)}

INTEGRATED_5:registration{registration_ID (PK), REGISTRATION_ACADEMICSTAFFMEMBER_REF(FK), REGISTRATION_COURSE_REF(FK), REGISTRATION_ACADEMICPROGRAMME_REF(FK)}

INTEGRATED_5:payrate{Rank (PK), HrRate}

INTEGRATED_5:workson{Name, Hrs, ProjRank, Expenses}

INTEGRATED_5:address{Id (PK), Street, City, PostalCode}

INTEGRATED_5:academic_staff_member{academic_staff_member_ID (PK), STAFF_NAME, STAFF_IDENTIFICATION_NUM, STAFF_FAX, STAFF_BIRTHDATE, STAFF_EMAIL, STAFF_PHONE, Sal, addr(FK) personTitle, homepage}

INTEGRATED_5:university_student{university_student_ID (PK), NAME, ELECTRONIC_MAIL, TELEPHONE, GPA, Yr, advisor(FK)}
INTEGRATED_5:paper{paperTitle (PK), description, publicationYear(FK)}

INTEGRATED_5:paper_author{paperTitle (PK)(FK), author (PK)(FK) }

INTEGRATED_5:person_project{person (PK)(FK), projectTitle (PK)(FK)}

INTEGRATED_5:seminar{about (PK), speaker (FK), location, date}

INTEGRATED_5:year{yr (PK)}

INTEGRATED_5:project{projectTitle (PK), description, link}

INTEGRATED_5:admin_staff{admin_staff_id (PK), name, email, phone}

INTEGRATED_5:areas_of_interest{interest_id (PK), area (PK)}

Fig. 6.20. Elements of the **Final Integrated Schema**

missed in this step, in addition to the ones in the previous step. Therefore, the key completeness was 0.93, as explained in detail in Appendix F.

6.8 Conclusions

This chapter presents the results of our evaluation of the SASMINT system. In this chapter, first the state of the art in the schema matching evaluations is addressed, and then the quality measures that were applied during our experiments are explained. After that, the set of six test schemas that were used for evaluating the SASMINT system are introduced. Since there was not any benchmark for relational schema matching systems, we generated our own test schemas, a number of which were the same or modified versions of schemas from the evaluations of similar matching systems in related research.

After the introductory part, the results of our experiments are presented in this chapter. Schema matching in SASMINT was compared against one leading state of the art schema matching system, the COMA++. A brief summary of this comparison based on the input, the combination of matchers, the output, the persistence store, and the quality criteria is given below:

- *Input:* SASMINT accepts relational schemas, bearing in mind that most data are still stored in relational databases and corresponding schemas are represented as relational DDLs. As stated in Chapter 7 about the Future Steps, it may be possible to extend SASMINT to also support matching of XML Schema. The COMA++ accepts relational schema, XML Schema, and OWL as input to its matching procedure. In addition to the schemas to be matched, SASMINT uses a number of auxiliary inputs. A file consisting of a number of well-known abbreviations is exploited. Users can update (extend) this file with other abbreviations from the domain of schemas. As the second auxiliary input, SASMINT uses the WordNet for identifying semantic relationships between schema elements. Similar to SASMINT, COMA++ also utilizes a user-modifiable list of abbreviations. On the other hand, in order to detect synonymy relationships, COMA++ requires a user-provided list of synonyms. The disadvantage of this approach is that users are required to continuously update this list with pairs of synonyms from the domain of schemas.

- *Combination of Matchers:* SASMINT and COMA++ both provide a library of matchers. SASMINT provides the possibility of user assigned weights to different algorithms and a Sampler component, which helps the user to identify the appropriate weight for each linguistic matching metric. On the other hand, COMA++ supports different alternatives for combining, aggregating, and selecting match results from different metrics. But the user should decide and select the approaches to be applied. This feature makes it difficult for an inexperienced user to identify the best combination.

- *Output:* The output of a match system is a mapping, indicating which elements of the recipient and donor schemas correspond to each other. Both SASMINT and COMA++ represent these correspondences using a value between 0 and 1. Furthermore, they both can support 1-to-1, 1-to-n, n-to-1, and m-to-n types of matches.

- *Persistence store for the results:* For matching and integration of schemas, SASMINT stores the results based on SDML. This allows the results to be used for federated query processing and for decomposition of queries to be sent to different local schemas, as well as for formal representation of the semi-automatically generated integrated schema from the recipient and donor schemas. COMA++ has an internal repository for the results, but users cannot see in which format results are stored and it is not clear how to use these results outside of the system.
- *Quality of Schema Matching*: The quality of schema matching supported by SASMINT and COMA++ was compared using their default settings for the combination of different matchers. SASMINT's default approach for combining linguistic and structure matching metrics calculates their weighted sum. However, then the Linguistic metrics have a higher impact (0.7) than the structure ones (0.3), on the final result. But in the evaluation between the two systems, each metric in groups of the linguistic matching and structure matching was considered with equal weight. Namely, in order to make a fair comparison with COMA++, we did not give higher weights to the metrics that could be more appropriate for some schema types. COMA matcher combines name, path, leaves, parents, and siblings matchers by averaging them. We updated the abbreviation lists of both systems with new abbreviations related to schemas. However, we did not update the synonyms list of COMA++, because manually adding into this list some complex semantic correspondences would also lead to unfair comparisons. We carried out experiments based on two types of result selection strategies that we call as: 1) Select all above threshold and 2) select max above threshold. Both systems performed better in the second approach. When the first approach was used, results for COMA++ were worse than those of SASMINT. For the second approach, the systems performed the same for some schema pairs, for the remaining pairs, SASMINT performed better than COMA++.

In order to evaluate the Sampler component of SASMINT, we performed some tests using the same set of test schemas. For this purpose, after setting the threshold value, we provided the Sampler component with a number of similar pairs from the two schemas being compared. We performed schema matching using both the Sampler’s computed weights as well as the equal weights for linguistic matching algorithms. In some cases, using Sampler’s computed weights resulted in an increase in the number of false positives, and thus a decrease in the precision. However, in every such case since Sampler identifies higher number of correct matches, by assigning appropriate weights, the corresponding recall was much better than the case where Sampler was not used. Therefore, even in these cases, this resulted in an increase in f-measure and overall performance of SASMINT. Therefore, using Sampler was shown to improve the quality of match results.

After evaluating the schema matching approach of SASMINT against the leading system COMA++, we evaluated the schema integration approach of SASMINT. Since COMA++’s schema merging feature is very primitive and there were no other systems at the level of SASMINT, which can use their schema matching results for semi-automatic schema integration, we could unfortunately not compare the results of schema integration approach of SASMINT with any other system. Nevertheless, we performed the incremental integration of six schemas to be able to evaluate SASMINT against the state of the art criteria defined for automated schema integration. During the empirical evaluation, SASMINT achieved a high percentage of minimality and completeness for its integrated schemas procedure, which applies its user-validated matches.

To sum up, schema matching and schema integration are two challenging tasks in SASMINT. Different types of schema heterogeneities, such as semantic and structural, make these tasks more difficult to achieve automatically. A semi-automatic system might perform badly on such schemas. Evaluation data sets need to be carefully selected to cover different types of schema heterogeneities. Furthermore, in order to fairly evaluate the schema matching and schema integration systems, measures need to be carefully selected and defined to consider all aspects of a system in evaluation, such as quality of the match and integration results, how the results are represented, how easily these results can be modified/corrected by the user, and whether it is possible to use these results in other processes like query decomposition in federated query processing.