MatchBench: Benchmarking Schema Matching Algorithms for Schematic Correspondences

Chenjuan Guo, Cornelia Hedeler, Norman W. Paton, and Alvaro A.A. Fernandes

School of Computer Science, University of Manchester, M13 9PL, UK
{guoc,chedeler,norm,alvaro}@cs.man.ac.uk

Abstract. Schema matching algorithms aim to identify relationships between database schemas, which are useful in many data integration tasks. However, the results of most matching algorithms are expressed as semantically inexpressive, 1-to-1 associations between pairs of attributes or entities, rather than semantically-rich characterisations of relationships. This paper presents a benchmark for evaluating schema matching algorithms in terms of their semantic expressiveness. The definition of such semantics is based on the classification of schematic heterogeneities of Kim *et al.*. The benchmark explores the extent to which matching algorithms are effective at diagnosing schematic heterogeneities. The paper contributes: (i) a wide range of scenarios that are designed to systematically cover several reconcilable types of schematic heterogeneities; (ii) a collection of experiments over the scenarios that can be used to investigate the effectiveness of different matching algorithms; and (iii) an application of the experiments for the evaluation of matchers from three well-known and publicly available schema matching systems, namely COMA++, Similarity Flooding and Harmony.

1 Introduction

Schema matching methods identify *matches* between elements of data sources that show similar properties (e.g., names, instances and structures) [4, 20]. Matching methods are not an end in themselves, but rather form part of other operations, such as *schema mapping* that refines matches into declarative but executable mappings (e.g., in SQL or XSLT) to specify the relationships between the data sources [11, 5]. Schema matching and mapping are important because a wide range of information management and integration tasks [13, 12], such as data exchange, evolution and distributed query processing, depend on a detailed understanding of the relationships between data sources.

Such integration tasks must be built on appropriate executable mappings, which, in turn, require clear characterisations of matches between data sources. However, although matches may be generated by a large number of different techniques, they are often numerous, uncertain and conflicting [3]. As such, when evaluating matches, it seems important to explore what kind of information is carried by matches that must be taken into account by executable programs.

We note that there have been several evaluation activities relating to schema matching/mapping in the data integration community in recent years, such as Ontology Alignment Evaluation Initiative (OAEI) [1], XBenchmark [10], eTuner [15] and STBenchmark [2]. The first three activities aim to evaluate schema or ontology matching

G. Gottlob et al. (Eds.): BNCOD 2013, LNCS 7968, pp. 92–106, 2013.

systems (e.g., [8, 17, 7]) in terms of accuracy or correctness, e.g., *precision*, *recall* and *F-measure*, of matches identified by the existing matching systems, while STBenchmark aims to compare schema mapping systems (e.g., [11, 5, 6]) in terms of the effectiveness of the support provided to mapping developers.

In this paper, we present a benchmark, called MatchBench, with the aim of understanding the effectiveness of schema matching systems in identifying *specific relationships* between elements of two schemas, rather than simply assessing the correctness of matches identified by the matching systems, thus differentiating MatchBench from previous evaluation activities. We characterise such relationships between schema elements using the classification of schematic heterogeneities of Kim *et al.* [14]. We do not intend to enumerate all kinds of relationships in MatchBench but try to exemplify a collection of semantic relationships based on the schematic heterogeneities.

Thus, the hypothesis behind MatchBench is that the effectiveness of matching systems in practice can be evaluated in terms of their ability to diagnose (or support the diagnosis of) such schematic heterogeneities as those proposed by Kim *et al.* The contributions of the paper are presented as follows:

1. A collection of scenarios, based on the schematic heterogeneities of Kim *et al.* [14], that systematically vary the amount and nature of the evidence available about heterogeneities.
2. An experiment design over the scenarios at (1) that can be used as a benchmark to investigate the contexts within which specific matchers are more, or less, effective.
3. The application of the benchmark to schema matching techniques is supported within three well-known and publicly available matching systems[1], namely COMA++ [8], Similarity Flooding [17] and Harmony [22].

The remainder of the paper is structured as follows. Section 2 introduces the schematic heterogeneities of Kim *et al.* [14]. Section 3 describes MatchBench, including the offered scenarios and the associated experiments. Sections 4 describes the application of MatchBench to matchers provided by COMA++, Similarity Flooding and Harmony, respectively. Section 5 reviews related work, in particular on the evaluation of schema matching techniques. Section 6 presents some overall conclusions.

2 Schematic Correspondences

The schematic heterogeneities proposed by Won Kim *et al.* [14] are defined as different symbolic representations of data that represent the same real world information. We essentially use the terms heterogeneity and correspondence as synonyms – a heterogeneity is an inconsistency between data sources in representation, and a correspondence is a description of the heterogeneity that allows it to be managed.

In this paper, we adopt the classification of schematic correspondences between relational schemas proposed by Won Kim *et al.* [14], and have refined the characteristics of many-to-many entity correspondences from [14] to distinguish horizontal and vertical partitioning. Before moving on to the details, let the following be the schemas of two

[1] In order to maintain the initial feature of matching systems, we decided not to re-implement the matching systems that are not publicly available.

independently designed relational databases RDB1 and RDB2[2].

RDB1:
home_cust (id*, name, birth, a_id+, p_city, p_area, p_local)
oversea_cust (id*, name, birth, a_id+, p_city, p_area, p_local)
account (id*, name, balance, tax)

RDB2:
customer (id*, c_fname, c_lname, c_birth, account_id+)
cust_phone (id*+, city, area, local, extension)
cust_account (id*, account_name, account_balance)

Both RDB1 and RDB2 contain information about customers and their accounts. Even though they represent the information differently, it can be identified that they represent broadly the same real world information, and that correspondences exist between them at both entity and attribute levels:
(i) The *entity-level correspondences* indicate the equivalence between two (sets of) entities (e.g., tables), which can be decomposed into one-to-one and many-to-many entity correspondences, where

- *one-to-one entity correspondences* relate pairwise entities that represent the same information. For example, account in RDB1 and cust_account in RDB2 can be considered equivalent but show the following heterogeneities:
  - *name conflict*, which indicates that equivalent entities have different names. In the following, this conflict is called Different Names for the Same Entities (*DNSE*). When different entities happen to have the same name, we call the conflict Same Name for Different Entities (*SNDE*).
  - *missing attributes conflict*, which identifies attributes that are present in one entity but not in the other (e.g., attribute tax in account is a missing attribute of cust_account).
- *many-to-many entity correspondences* relate two sets of entities that represent the same information. For example, home_cust and oversea_cust together in RDB1 describe the same information about customers as customer and cust_phone in RDB2. It can be seen that these two sets of entities in RDB1 and RDB2 do not have the same structure, but the underlying information is similar. This difference results in distinct types of many-to-many conflicts. Borrowing terminology from distributed database systems [19],we classify them as follows:
  - *horizontal partitioning* (*HP*), where one single entity is partitioned along its instances into multiple entities in another schema. As such, all attributes of the former entity are present in each of the corresponding entities in the latter (e.g., home_cust and oversea_cust in RBD1).
  - *vertical partitioning* (*VP*), where a single entity is partitioned into multiple entities in another schema, where the attributes in each of the latter constitute subsets of the attributes in the former. The primary key of the vertically partitioned entity appears as an equivalent attribute in every one of its vertical partitions in the other schema, whereas other attributes of the former entity are present only once in the latter (e.g., customer and cust_phone in RDB2).

[2] Symbols * and + indicate primary key and foreign key attributes, respectively.

Given the above information, we are then able to enumerate 4 types of many-to-many entity correspondences: *HP* vs *HP*, *HP* vs *VP*, *VP* vs *HP* and *VP* vs *VP* correspondences. For example, the correspondence between entity sets {home_cust, oversea_cust} and {customer, cust_phone} is a *HP* vs *VP* correspondence. Readers may follow the definitions to enumerate other partitioning types, such as hybrid partitioning, which refers to the cases where HP and VP appear together in the source (or target).

(ii) The *attribute-level correspondences* indicate the equivalence between two (sets of) attributes. For the remainder of the paper, we assume that attributes associated by attribute-level correspondences belong to entities that are participating in some entity-level correspondence. Similar the entity-level correspondence, the attribute-level correspondences can be decomposed into one-to-one and many-to-many correspondences, where

- *one-to-one attribute correspondences* relate pairwise attributes. Equivalent attributes may have different names, so such a conflict is called Different Names for the Same Attributes (*DNSA*) (e.g., account.name in RDB1 and cust_account.account_name in RDB2). By contrast, attributes that are different may have the same name, giving rise to Same Name for Different Attributes (*SNDA*) correspondences.
- *many-to-many attribute correspondences* associate two sets of attributes that present the same property of equivalent entities. For example, the single attribute home_cust.
  name in RDB1 and the set of attributes customer.c_fname and customer.c_lname in RDB2 represent names of customers.

3 Benchmark Description

This section describes the benchmark[3] consisting of: (i) a collection of scenarios, in which schematic heterogeneities are systematically injected into an initial database; and (ii) a collection of experiments that investigate the ability of matching methods to identify specific properties exhibited by the scenarios.

3.1 Scenario

The scenarios build upon the *source* and *target* databases illustrated in Fig. 1, derived from TPC-E (http://www.tpc.org/tpce/), which are manipulated in a controlled manner.

There are *positive* and *negative* scenarios. In the *positive* scenarios, the starting point is that the *source* and *target* databases have a single table in common, into which schematic heterogeneities described in Section 2 are systematically introduced. In the implementation, the initial *target* database is extended with a single table from the *source* (see Fig. 1). In the *negative* scenarios, the starting point is that the source and

[3] This paper only means to demonstrate a general idea of MatchBench. Readers may find the complete version for all available scenarios and experiments from `http://code.google.com/p/matchbench/`.

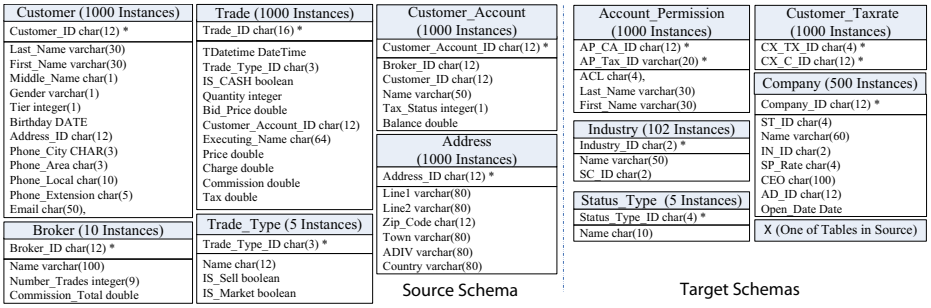


Fig. 1. The *source* and *target* databases used as a basis for scenario generation, where primary keys are marked with ∗

target databases have no single table in common, but similarities have been systematically introduced, giving rise to scenarios where tables should not be matched, but where there are some similarities between tables.

Positive Scenarios for One-to-One Entity Correspondences. Fig. 2 describes the space of positive scenarios where heterogeneities are introduced into one-to-one identical entities[4]. In the figure, boxes represent scenario sets and arrows represent the systematic introduction of heterogeneities into the scenario sets. Each *scenario set* consists of a collection of databases each of which manifests examples of the heterogeneities named in the corresponding box, the definitions of which are provided below. For example, *Scenario Set 1* represents the starting point for the introduction of the heterogeneities, and the arrow leading to *Scenario Set 5* indicates that it has been derived from *Scenario Set 1* through the changing of entity names.

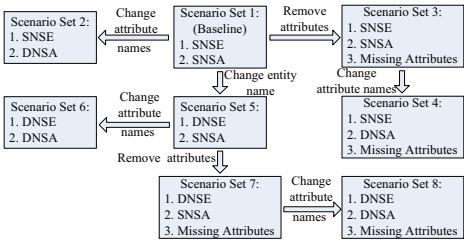


Fig. 2. Positive scenarios for one-to-one entity correspondences

In what follows, where names are described as the *same* they are identical, and where they are described as *similar* their strings overlap; neither of these properties hold for *different* names. Following the terminology introduced in Section 2, terms used in Fig. 2 include *SNSE* as Same Name for Same Entity; *DNSE* as Different Names for Same Entity; *SNSA* as Same Name for Same Attribute; and *DNSA* as Different Names for Same Attribute. As such, a scenario set that exhibits one-to-one entity heterogeneities may also exhibit one-to-one attribute heterogeneities.

In each scenario set, the extents of equivalent entities either contain the *same instances* (SI) or *disjoint instances* (DI). The *disjoint instances* are generated by partitioning instances of an original entity into two disjoint sets of instances, thus forming

[4] The order of introducing different types of heterogeneities is insignificant.

disjoint instances of two equivalent entities. Overlapping instances are also possible real world cases, but are not implemented in MatchBench.

Negative Scenarios for One-to-One Entity Correspondences. The space of negative scenarios for one-to-one different entities, where pairs of entities represent different real world information, is described in Fig. 3. Terms used to describe the properties of the scenario sets include *DNDE* as Different Names for Different Entities; *SNDE* as Same Name for Different Entities; *DNDA* as Different Names for Different Attributes; *SNSA* as Same Name for Same Attribute; *DNSA* as Different Names for the Same Attributes; and *SNDA* as Same Name for Different Attributes.

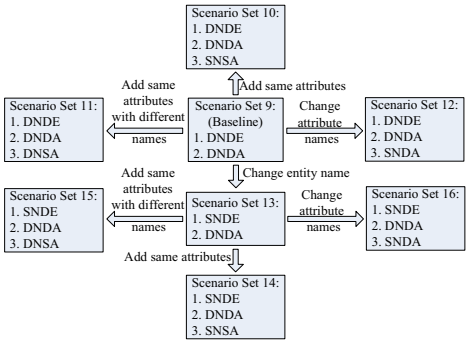


Fig. 3. Negative scenarios for one-to-one entity correspondences

Positive Scenarios for Attribute Many-to-One Correspondences. In Fig. 4, the space of attribute many-to-one correspondences is described, where a set of attributes and a single attribute that belong to equivalent entities represent the same real world information. We note that most schema matchers only handle many-to-one attribute correspondences, and thus we set up a task that existing matchers can manage. MatchBench includes three different types of attribute many-to-one correspondences shown as follows.

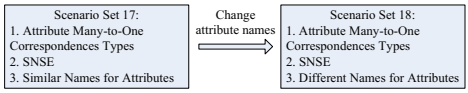


Fig. 4. Positive scenarios for attribute many-to-one correspondences.

1. numeric operation: (*price* + *charge* + *commission*) × (1 + *tax*) = *price*
2. string concatenation: Concat (*first_name*, *middle_name*, *last_name*) = *name*
3. numeric concatenation:
   Concat (*phone_city*, *phone_area*, *phone_local*, *phone_extension*) = *phone*

Similar to Fig. 2, extents of equivalent entities are generated that give rise to SI and DI cases for scenario set 17. Scenario set 18 only contains SI cases but not DI, in order to retain a certain level of similarity between attributes.

Positive Scenarios for Entity Many-to-Many Correspondences. Two sets of entities, shown in Fig. 5, represent the same real world information. Three different types of many-to-many entity correspondences are included in MatchBench:

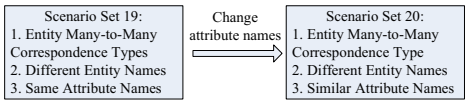


Fig. 5. Positive scenarios for many-many entity correspondences

- *HP vs HP*, where the two sets are related by horizontal partitioning.
- *VP vs VP*, where the two sets are related by vertical partitioning.
- *HP vs VP*, where the two sets are related by horizontal and vertical partitioning.

3.2 Experiments

Effectiveness Measures. The effectiveness of matching systems is evaluated in MatchBench by identifying whether the systems meet specific requirements for diagnosing each type of schematic heterogeneity. Each such requirement is met if results of the systems are close to the correct matches provided by the scenarios presented in this section. Following terms in the standard definitions [9] of information retrieval, we call the correct matches as ground truth. We compare the results of the systems with the ground truth, and report recall, precision and F-measure of the results following to shown the effectiveness of the matching systems.

Experiment Design. Building on the scenarios, we designed 10 experiments to measure how effectively matching systems identify the presence of schematic heterogeneities. Due to the space limitations, only 4 experiments are presented in this paper. More experiments are included in the technical report.

In Experiments 1, 3 and 4, where schematic heterogeneities are exhibited in the chosen scenarios, the F-measure is reported in the vertical axis drawn in the figures produced in Section 4.2. The higher the F-measure reports, the better is the matching system for diagnosing the heterogeneity. In Experiment 2, which involves negative scenarios, where there are no such heterogeneities in the chosen scenarios, 1 – F-measure is reported on the vertical axis so that larger values also reflect the better effectiveness of the matching system on not reporting the heterogeneities.

Experiment 1: *Identifying when the same entity occurs in positive scenarios.* This experiment involves *Scenario Sets 1* to *8* in Fig. 2, and reports on the ability of the matchers to meet two requirements:

- *Requirement R1:* Equivalent entities are matched, where the ground truth is the set of pairwise entity matches between equivalent entities.

- *Requirement R2:* Equivalent attributes are matched, where the ground truth is the collection of pairwise attribute matches between equivalent attributes.

Experiment 2: *Identifying when the same entity occurs in negative scenarios.* This experiment involves *Scenario Sets 9* to *16* in Fig. 3, and reports on the ability of matching systems in scenarios where no correspondences exist:
- *Requirement R1:* Different entities are not matched, where the ground truth is that there are no pairwise entity matches between different entities.
- *Requirement R2:* Different attributes are not matched, where the ground truth is that there are no pairwise attribute matches between pairs of attributes.

Experiment 3: *Identifying many-to-one attribute correspondences in positive scenarios*. This experiment involves *Scenario Sets 17* and *18* in Fig. 4, where each element in the ground truth is a collection of attribute matches between each attribute in the set and the single attribute.

Experiment 4: *Identifying many-to-many entity correspondences in positive scenarios*. This experiment involves *Scenario Sets 19* and *20* in Fig. 5.

- *Requirement R1*: Each entity in the source set should be matched to all entities in the target set. The ground truth is the collection of pairwise entity matches between each entity in the source set and all entities in the target set.

The following two requirements are investigated only when the evaluated systems are able to meet R1.
- *Requirement R2*: Primary key attributes in each entity in the source set should be matched to primary key attributes in all entities in the target set. The ground truth is the collection of pairwise attribute matches between primary key attributes in each entity in the source set and primary key attributes in all entities in the target set.
- *Requirement R3*: Partitions in the source schema are matched against partitions in the target schema, with a view to identifying specific types of many-to-many correspondences. For each type, the ground truth is the collection of pairwise attribute matches between attributes as described below:

– *Horizontal Partitioning vs Horizontal Partitioning:* Each non-key attribute in each entity in the source (target) set should be matched to a single non-key attribute in every entity in the target (source) set.
– *Vertical Partitioning vs Vertical Partitioning:* Each non-key attribute in each entity in the source (target) set should be matched to a single non-key attribute in an entity in the target (source) set.
– *Horizontal Partitioning vs Vertical Partitioning:* Each non-key attribute in each entity in the source set should be matched to a single non-key attribute in an entity in the target set; but each non-key attribute in each entity in the target set should be matched to a single non-key attribute in each entity in the source set.

4 Application of MatchBench

4.1 Matching Systems

In general, we follow the advice of the developers when configuring matching systems, for example, by employing the settings suggested in the published papers or in private communication with the authors. In addition, we take all steps that are available to us in order to help the systems to perform well, e.g., by plugging an instance-level matcher into Similarity Flooding and Harmony, which were both originally supplied with only schema-level matchers.

COMA++ [8] is a schema matching platform that supports the composition of schema and instance level matchers from a substantial library. In particular, we applied *AllContext* as the matching strategy, selected matchers *Name*, *NamePath*, *Leaves* and *Parents* at the schema-level and *Content-based* at the instance-level, and employed *Average* for aggregation, *Both* for direction, *Threshold+MaxDelta* for selection and *Average* for combination, as they are demonstrated to be effective in published experimental evaluations [8]. As experience with COMA++ has not given rise to consistent recommendations for Threshold and Delta [16, 8], we decided to employ the default settings of Threshold and Delta (i.e., 0.1 and 0.01) provided with the COMA++ tool.

Similarity Flooding (SF) [17] is a schema matching operator used by the model management platform, Rondo [18]. SF applies a name matcher *NGram* and a similarity flooding algorithm to generate candidate matches, and selects a best match for each element from the candidate matches under the constraint that each element can only be associated with a single match.

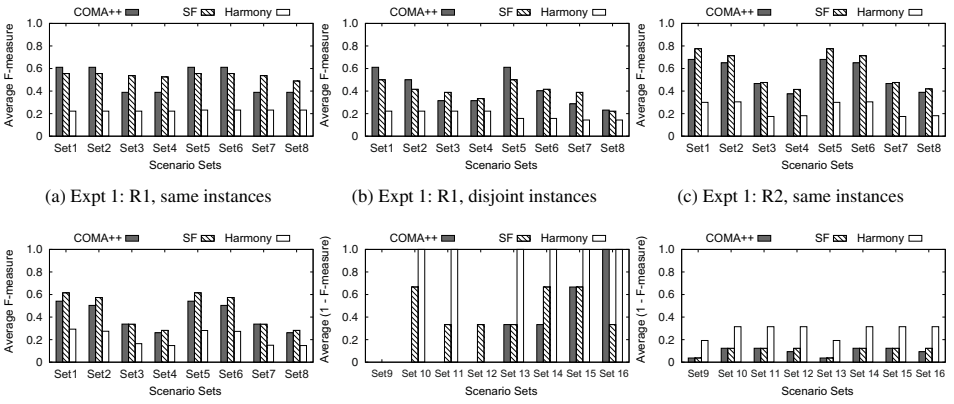
For the evaluation of SF using MatchBench, the *NGram* matcher and an instance matcher (i.e., the *Content-based* matcher of COMA++) are used together to enable SF making use of instance-level information. This improvement turns out to be important for identifying schematic correspondences.

Harmony [22] is an interactive matching tool contained in a suite of data integration tools, called OpenII [21]. For the evaluation using MatchBench, we chose the *EditDistance*, *Documentation* and *Exact* matchers provided by Harmony but we left out the *Mapping* matcher as we do not consider previous matches during the evaluation.

Harmony returns all candidate matches and allows the user to slide a threshold bar while visually observing which matches pass different thresholds. However, there are a large number of scenarios in MatchBench, thus selecting a threshold manually for each of them is not practical. Therefore, we decided to follow the recommendation of the OpenII authors. We use the top matches associated with each element while not restricting the number of matches associated with an element. In addition, as Harmony only works at the schema-level, we combine it with the *Content-based* matcher of COMA++, to provide the same basis in terms of instance-based matches as COMA++ and SF.

4.2 Effectiveness Comparison

Experiment 1: *Identifying when the same entity occurs in positive scenarios.* The results of this experiment are presented in Fig. 6(a) to (d). The following can be observed: (i) All three systems have been reasonably successful at matching equivalent entities and equivalent attributes when they have the same instances (*recalls* reported are fairly high, though not illustrated here), but have been less successful for disjoint instances.



(a) Expt 1: R1, same instances (b) Expt 1: R1, disjoint instances (c) Expt 1: R2, same instances

(d) Expt 1: R2, disjoint instances (e) Expt 2: R1 (f) Expt 2: R2

Fig. 6. Experiments 1 and 2 for COMA++, Similarity Flooding (SF) and Harmony

(ii) A significant number of false positives between different entities and between different attributes have been generated by all systems (the *F-measures* reported in Fig. 6(a) to (d) are fairly low, given high *recalls*). This is due to the selection strategies these

platforms employ: for COMA++, the *MaxDelta* method always chooses a few of the top matches associated with an element, even though the scores of these matches may be fairly low due to the low threshold of 0.1; SF only returns 1-to-1 matches by selecting a best match for each element, regardless of its similarity score; and Harmony keeps a match as long as it is the top match for either of its associated elements irrespective of the match scores, resulting in a large number of incorrect matches, which makes it perform worst among the three platforms.
(iii) Changing the names of equivalent entities into similar or different has almost no impact on the three platforms on matching equivalent attributes (Fig. 6(c) and (d)).

Experiment 2: *Identifying when the same entity occurs in negative scenarios.* The results of this experiment are presented in Fig. 6(e) and (f). The following can be observed: (i) All three systems have matched the two different entities when similarities have been injected into their names or their attributes (Average($1 - F\text{-}measure$) > 0 in Sets *13* to *16* in Fig. 6(e)). This is because all three systems choose the top candidate matches for each element, and this also indicates that entities are matched because they are more similar to each other than to other entities, but not because they represent the same real world notion. (ii) COMA++ and SF perform satisfactorily in not matching different attributes (Fig. 6(f)). Where attributes are matched, this is normally because similar attributes have been introduced, and the remainder results from overlaps in the instances or the names of non-equivalent attributes. Harmony has matched several different attributes even in the baseline scenarios where no similarities have been injected. This shows that its selection strategy that keeps a top match for each element is not effective in deciding the final matches.

Experiment 3: *Identifying many-to-one attribute correspondences in positive scenarios*. The results of this experiment are presented in Fig. 7. COMA++ and SF have failed in this experiment. In contrast to SF, which only identifies 1-to-1 matches, the *Threshold+MaxDelta* method COMA++ uses allows the identification of n-to-1 matches. However, given the *delta* value of 0.01, the *MaxDelta* method sets a fairly small tolerance range below the top match of an attribute, thus only being able to return matches whose similarities are close to the top match. Harmony has identified some n-to-1 attribute correspondences, where the *n* attributes and the *1* attribute have similar names (Sets *17 SI* and *17 DI* in Fig. 7), because Harmony chooses a best match for each element and allows a match to be kept as long as it is the best match for either of its associated elements. When the *n* attributes and the *1* attribute have similar names, the matches between the *n* attributes and the *1* attribute are usually the top matches for the *n* attributes, and thus are selected by Harmony.

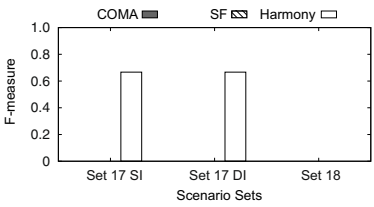
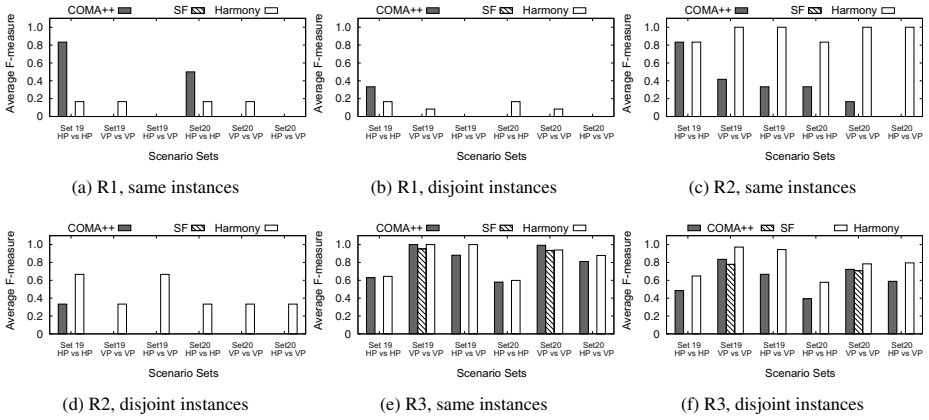


Fig. 7. Experiment 3 for COMA++, Similarity Flooding (SF) and Harmony

Nevertheless, an n-to-1 attribute correspondence refers to a transformation of instances (e.g., string concatenation or numeric operation) between the *n* attributes and the *1* attribute rather than a selection of matches whose similarities are close or the top, as determined by comparing names or instances of pairwise attributes. We anticipate

that iMAP [7] could identify the n-to-1 attribute correspondences, however, the system is not publicly available.

Experiment 4: *Identifying many-to-many entity correspondences in positive scenarios*. In this experiment, SF is not able to carry out the task due to its focus on 1-to-1 matches. Where SF identifies a few matches (Fig. 8(e) and (f)), it is because the ground truth is the 1-to-1 matches in vertical partitioning. COMA++ and Harmony have performed rather patchily in seeking to meet requirement *R1*, as presented in Fig. 8(a) and (b), though COMA++ and Harmony have performed satisfactorily on investigating requirements *R2* and *R3*. The following can be observed for COMA++ and Harmony.
(i) COMA++ has only been able to associate the n-to-m entities, i.e., to satisfy requirement *R1*, where the same instances are being represented in the horizontal partitioning models (Set *19 HP vs HP* and Set *20 HP vs HP* in Fig. 8(a)), but has failed in other partitioning models or in disjoint instances. This is because only when the two original entities that have the same instances are horizontally partitioned, the similarities between each pair of entities in the source and target sets are close, and as such are selected by the *MaxDelta* method. Harmony has performed slightly better than the others. However, it has been fairly generous (the *recalls* are always high, but the *precisions* are fairly low). Therefore, the patchy results shown by Harmony are because the equivalent n-to-m entities have also been matched to different entities.



(a) R1, same instances (b) R1, disjoint instances (c) R2, same instances

(d) R2, disjoint instances (e) R3, same instances (f) R3, disjoint instances

Fig. 8. Experiment 4 for COMA++, Similarity Flooding (SF) and Harmony

(ii) Similar to requirement *R1*, when the alternatively fragmented entities have the same instances and no changes have been made to attribute names, and thus the similarities of matches for many-to-many primary key attributes are close, COMA++ has generally been successful in satisfying requirement *R2*, as shown in Fig. 8(c). Harmony has performed fairly satisfactorily in satisfying requirement *R2* in the SI case, however, for cases where there is less evidence (e.g., the DI case), equivalent primary key attributes are matched to different attributes (Fig. 8(c) and (d)).

(iii) COMA++ has been generally successful at matching non-key attributes, i.e., satisfying requirement *R3*, in both scenario sets where the same instances are represented, but has performed slightly worse in the presence of disjoint instances. COMA++ has performed particularly well in the vertical partitioning scenarios (Set *19 VP vs VP* and Set *20 VP vs VP* in Fig. 8(e)), as the non-key attributes only have single corresponding attributes; but has performed less well in the horizontal partitioning scenarios (Set *19 HP vs HP* and Set *20 HP vs HP* in Fig. 8(e)) where many-to-many correspondences between non-key attributes should be identified. This indicates that COMA++ is more suited to identifying one-to-one correspondences than to many-to-many correspondences. Harmony has been competitive with COMA++ in the SI case, but has performed better in the DI case (Fig. 8(e) and (f)), as the lack of a threshold means that Harmony tends to return more matches, some of which are true positives.

4.3 Summary

The following lessons have been learned from the application of the representative matchers to the benchmark:

(1) The existing schema matching methods were designed principally to associate similar schema elements, and have been shown to perform rather better at this task than at diagnosing the schematic heterogeneities of Kim *et al.* [14].
(2) The existing schema matching methods were more designed for identifying one-to-one matches than for identifying many-to-many schematic correspondences.
(3) The strategy for selecting candidate matches influences the overall effectiveness of schema matching methods significantly.
(4) COMA++ offers alternative choices for different matching tasks. We anticipate that with more appropriate *threshold* and *delta* values, COMA++ would have performed better in experiments provided in MatchBench [15]. However, as a well-known problem, this presents practical challenges that setting any parameters generally requires access to at least some training data.
(5) SF always identifies one-to-one matches between elements of data sources, and thus cannot be used in diagnosing many-to-many schematic heterogeneities.
(6) Designed as an interactive tool, Harmony seems unsuitable for scenarios where a very large number of matching tasks are required and where the automatic generation of matches are demanded, since it is not practical to manually choose matches in such scenarios for individual human users.

5 Related Work

This section reviews work that is related to that carried out here, and considers in particular *experimental evaluation practice for schema matching*, *generation of test cases for schema matching and mapping*, and *existing benchmarks for matching*.

In terms of *experimental evaluation practice for schema matching*, most results have been presented in the context of specific matching proposals, as compared by Do *et al.* [9]. This comparison makes explicit that the diversity in evaluation practice is

problematic, thus providing motivation for the development of benchmarks [9]. Overall, the comparison indicates that most evaluations have been carried out using representative real-world schemas; while this approach provides insights into the effectiveness of techniques in specific contexts, the lack of fine-grained control over properties of the matched schemas can make it difficult to understand precisely the circumstances in which methods are effective. Rather than revisiting the ground covered by Do *et al.*, here we focus on the published evaluations of COMA++, SF and Harmony, to which MatchBench is applied in Section 4.

The most comprehensive comparative activity of relevance to MatchBench is the Ontology Alignment Evaluation Initiative (OAEI) [1], which runs an annual event on evaluating ontology matching systems. Whereas, MatchBench is designed to assess whether specific relationships, i.e., schematic correspondences, can be identified by schema matching systems.

In terms of *generation of test cases for schema matching and mapping*, test cases have been generated to support both tuning of matching systems in eTuner [15] and evaluation of schema mapping platforms in STBenchmark [2]. The test schemas over which eTuner is evaluated are generated by applying a number of rules for introducing perturbations into existing schemas. These perturbations overlap with those described in Section 3, but differ in the following respects: (i) they do not follow an established classification of schematic correspondences; (ii) the emphasis is on 1-to-1 matches; (iii) no negative scenarios are described; and (iv) there is no specific identification of collections of test scenarios.

STBenchmark [2] is a benchmark for comparing visual interactive mapping construction systems that aim at assisting an expert in generating a precise specification of mappings between two schemas with less effort. STBenchmark provides rules for generating mapping scenarios and evaluates the degree of effort supported by a schema mapping system in specifying mappings. In essence, these rules overlap with those described in Section 3. However, selecting types of variations and generating evaluation scenarios is the responsibility of users. On the other hand, MatchBench supports the developers of matchers through the provision of immediately usable scenarios.

In terms of *existing benchmarks for matching*, XBenchmark [10] has been developed in the context of XML schema matching, and STBenchmark has been used for schema mapping generation [2]. XBenchmark reports results at a very coarse grain, and is agnostic as to the test cases used. In contrast, we systematically generate test cases to assess the capabilities of matchers in specific scenarios with known properties, and have an overall objective of ascertaining whether or not the matchers provide the diagnostic information required to identify specific schematic heterogeneities. STBenchmark aims for evaluating interactive tools for constructing mappings from matchings, such as Clio [11] or BizTalk Mapper[5], and thus the benchmark measures the amount of human effort involved in addressing specific mapping scenarios given specific matchings. As such, STBenchmark is complementary to MatchBench; indeed, insights from MatchBench may inform the development of helper components for interactive mapping tools that suggest to users what mappings may be most appropriate in a given setting.

[5] www.microsoft.com/biztalk

6 Conclusions

This paper has presented a benchmark for schema matching methods that identifies the extent to which these methods are successful at identifying correspondences between schemas in the presence of the schematic heterogeneities of Kim *et al.* [14]. This is in contrast to most reported evaluations of matching methods, where the focus is on the identification of 1-to-1 matches between individual schema elements, where the ability to combine these observations to draw higher level conclusions has not been investigated.

The objective of the benchmark is not to seek to identify which matching methods are "better" than others, but rather to enhance understanding of when and why specific matching methods are suitable for a given task, with a view to guiding matcher selection and configuration. In providing a revised focus for the evaluation of matching methods, on diagnosing the heterogeneities that mappings must resolve, the benchmark both supports the reassessment of existing proposals and timely evaluation of new techniques.

References

[1] Ontology Alignment Evaluation Initiative (OAEI), `http://oaei.ontologymatching.org/`

[2] Alexe, B., Tan, W.C., Velegrakis, Y.: Stbenchmark: towards a benchmark for mapping systems. PVLDB 1(1), 230–244 (2008)

[3] Bernstein, P., Melnik, S.: Model management 2.0: manipulating richer mappings. ACM SIGMOD, 1–12 (2007)

[4] Bernstein, P.A., Madhavan, J., Rahm, E.: Generic schema matching, ten years later. PVLDB 4(11), 695–701 (2011)

[5] Bonifati, A., Chang, E.Q., Ho, T., Lakshmanan, L.V.S., Pottinger, R., Chung, Y.: Schema mapping and query translation in heterogeneous p2p xml databases. VLDB J. 19(2), 231–256 (2010)

[6] Bonifati, A., Mecca, G., Pappalardo, A., Raunich, S., Summa, G.: Schema mapping verification: the spicy way. In: EDBT, pp. 85–96 (2008)

[7] Dhamankar, R., Lee, Y., Doan, A., Halevy, A.Y., Domingos, P.: imap: Discovering complex mappings between database schemas. In: SIGMOD Conference, pp. 383–394 (2004)

[8] Do, H., Rahm, E.: Matching large schemas: Approaches and evaluation. Information Systems 32(6), 857–885 (2007)

[9] Do, H.-H., Melnik, S., Rahm, E.: Comparison of schema matching evaluations. In: Chaudhri, A.B., Jeckle, M., Rahm, E., Unland, R. (eds.) NODe-WS 2002. LNCS, vol. 2593, pp. 221–237. Springer, Heidelberg (2003)

[10] Duchateau, F., Bellahsene, Z., Hunt, E.: Xbenchmatch: a benchmark for xml schema matching tools. In: VLDB, pp. 1318–1321 (2007)

[11] Fagin, R., Haas, L.M., Hernández, M., Miller, R.J., Popa, L., Velegrakis, Y.: Clio: Schema mapping creation and data exchange. In: Borgida, A.T., Chaudhri, V.K., Giorgini, P., Yu, E.S. (eds.) Conceptual Modeling: Foundations and Applications. LNCS, vol. 5600, pp. 198–236. Springer, Heidelberg (2009)

[12] Franklin, M., Halevy, A., Maier, D.: From databases to dataspaces: a new abstraction for information management. SIGMOD Record 34(4), 27–33 (2005)

[13] Haas, L.: Beauty and the beast: The theory and practice of information integration. In: Schwentick, T., Suciu, D. (eds.) ICDT 2007. LNCS, vol. 4353, pp. 28–43. Springer, Heidelberg (2006)

[14] Kim, W., Seo, J.: Classifying schematic and data heterogeneity in multidatabase systems. IEEE Computer 24(12), 12–18 (1991)
[15] Lee, Y., Sayyadian, M., Doan, A., Rosenthal, A.: etuner: tuning schema matching software using synthetic scenarios. VLDB J. 16(1), 97–122 (2007)
[16] Massmann, S., Engmann, D., Rahm, E.: Coma++: Results for the ontology alignment contest oaei 2006. Ontology Matching (2006)
[17] Melnik, S., Garcia-Molina, H., Rahm, E.: Similarity flooding: a versatile graph matching algorithm and itsapplication to schema matching. In: ICDE, pp. 117–128 (2002)
[18] Melnik, S., Rahm, E., Bernstein, P.: Rondo: a programming platform for generic model management. In: ACM SIGMOD, pp. 193–204 (2003)
[19] Ozsu, M.T., Valduriez, P.: Principles of distributed database systems. Addison-Wesley, Reading Menlo Park (1989)
[20] Rahm, E., Bernstein, P.: A survey of approaches to automatic schema matching. The VLDB Journal The International Journal on Very Large Data Bases 10(4), 334–350 (2001)
[21] Seligman, L., Mork, P., Halevy, A.Y., Smith, K., Carey, M.J., Chen, K., Wolf, C., Madhavan, J., Kannan, A., Burdick, D.: Openii: an open source information integration toolkit. In: SIGMOD Conference, pp. 1057–1060 (2010)
[22] Smith, K., Morse, M., Mork, P., Li, M.H., Rosenthal, A., Allen, D., Seligman, L.: The role of schema matching in large enterprises. In: CIDR (2009)