Benchmarking Matching Applications on the Semantic Web

Alfio Ferrara[1], Stefano Montanelli[1]
Jan Noessner[2], and Heiner Stuckenschmidt[2]

[1] Università degli Studi di Milano,
DICo - via Comelico 39, 20135 Milano, Italy
{ferrara,montanelli}@dico.unimi.it
[2] KR & KM Research Group
University of Mannheim, B6 26, 68159 Mannheim, Germany
{jan,heiner}@informatik.uni-mannheim.de

Abstract. The evaluation of matching applications is becoming a major issue in the semantic web and it requires a suitable methodological approach as well as appropriate benchmarks. In particular, in order to evaluate a matching application under different experimental conditions, it is crucial to provide a test dataset characterized by a controlled variety of different heterogeneities among data that rarely occurs in real data repositories. In this paper, we propose SWING (Semantic Web INstance Generation), a disciplined approach to the semi-automatic generation of benchmarks to be used for the evaluation of matching applications.

1 Introduction

In the recent years, the increasing availability of structured linked data over the semantic web has stimulated the development of a new generation of semantic web applications capable of recognizing identity and similarity relations among data descriptions provided by different web sources. This kind of applications are generally known as *matching applications* and are more and more focused on the specific peculiarities of instance and linked data matching [7]. Due to this situation, the evaluation of matching applications is becoming an emerging problem which requires the capability to measure the effectiveness of these applications in discovering the right correspondences between semantically-related data. One of the most popular approaches to the evaluation of a matching application consists in extracting a test dataset of linked data from an existing repository, such as those available in the linked data project, in deleting the existing links, and in measuring the capability of the application to automatically restore the deleted links. However, the datasets extracted from a linked data repository, suffer of three main limitations: i) the majority of them are created by acquiring data from web sites using automatic extraction techniques, thus both data and links are not validated nor checked; ii) the logical structure of these datasets is usually quite simple and the level of semantic complexity quite low; iii) the number and kind of dissimilarities among data is not controlled, so that it is difficult to tailor the evaluation on the specific problems that affect the currently considered matching application.

In this context, a reliable approach for the evaluation of matching applications has to address the following requirements: i) the test dataset used for the evaluation must be coherent to a given domain and must be represented according to the desired level of structural and semantic complexity; ii) the evaluation must be executed by taking into account a controlled variety of dissimilarities among data, including value, structural, and logical heterogeneity; iii) a ground-truth must be available, defined as a set of links among data that the matching application under evaluation is expected to discover. In order to address these requirements, we propose SWING (Semantic Web INstance Generation) a disciplined approach to the semi-automatic generation of benchmarks to be used for the evaluation of matching applications. A SWING benchmark is composed of a set of test cases, each one represented by a set of instances and their corresponding assertions (i.e., an OWL ABox) built from an initial dataset of real linked data extracted from the web. SWING aims at supporting the work of an *evaluation designer*, who has the need to generate a tailored benchmark for assessing the effectiveness of a certain matching application. The SWING approach has been implemented as a Java application and it is available at `http://code.google.com/p/swing`.

The paper is organized as follows. In Section 2, we discuss some related work on the subject of matching evaluation. In Section 3, we summarize our approach. In Section 4 and Section 5 we present the SWING acquisition and transformation techniques, respectively. In Section 6, we present the experimental results obtained by executing six different matching algorithms over the benchmark. In Section 7, we give our concluding remarks.

2 Related work

In the last years, significant research effort has been devoted to ontology matching with special attention on techniques for instance matching [7]. In the literature, most of the existing approaches/techniques use their individually created benchmarks for evaluation, which makes a comparison difficult or even impossible [10]. In the fields of object reconciliation, duplicate detection, and entity resolution, which are closely related to instance matching, a widely used set of benchmarks are proposed by the Transaction Processing Performance Council (TPC)[3] that focuses on evaluating transaction processing and databases. A number of benchmarks are also available for XML data management. Popular examples are presented in [3, 5]. Since these datasets are not defined in a Semantic Web language (e.g., OWL) their terminological complexity is usually very shallow. In the area of ontology matching, the Ontology Alignment Evaluation Initiative (OAEI) [6] organizes since 2005 an annual campaigns aiming at evaluating ontology matching technologies through the use of common benchmarks.

[3] `http://www.tpc.org`.

However, the main focus of the past OAEI benchmarks was to compare and evaluate schema-level ontology matching tools. From 2009, a new track specifically focused on instance matching applications has been introduced in OAEI and a benchmark has been developed to this end [8]. The weakness of this 2009 OAEI benchmark is the basic level of flexibility enforced during the dataset creation and the limited size of the generated test cases. The benchmark provided by Yatskevich et al. [13] is based on real-world data, using the taxonomy of Google and Yahoo as input. In [13], the limit of the proposed approach is the problem to create an error-free gold standard, since the huge size of the datasets prevents a manual alignment. Intelligent semi-automatic approximations are used to overcome such a weakness, however it is not possible to guarantee that all the correct correspondences are found and that none of the found correspondences is incorrect. The same problem raises with the linked data benchmarks DI[4] and VLCR[5]. Alexe et al. [1] provide a benchmark for mapping systems, which generate schema files out of a number of given parameters. Their automatic generation process ensures that a correct gold standard accrues. However, real-world data are not employed and artificial instances with meaningless content are mainly considered. Other benchmarks in the area of ontology and instance matching are presented in [13] and [9]. In these cases, the weak point is still the limited degree of flexibility in generating the datasets of the benchmark. We stress that the proposed SWING approach provides a general framework for creating benchmarks for instance matching applications starting with a linked data source and ending with various transformed ABox ontologies. In particular, the SWING approach combines the strength of both benchmarks [1] and [12] by taking real-world data from the linked data cloud as input and by performing transformations on them which ensure that the gold standard must be correct in all cases. Moreover, a further contribution of the SWING approach is the high level of flexibility enforced in generating the datasets through data transformations that is a widely recognized weak point of the other existing benchmarks.

3 The SWING approach

The SWING approach is articulated in three phases as shown in Figure 1.

Data acquisition techniques. SWING provides a set of techniques for the acquisition of data from the repositories of linked data and their representation as a reference OWL ABox. In SWING, we work on open repositories by addressing two main problems featuring this kind of datasources. First, we support the evaluation designer in defining a subset of data by choosing both the data categories of interest and the desired size of the benchmark. Second, in the data enrichment activity, we add semantics to the data acquired. In particular, we adopt specific ontology design patterns that drive the evaluation designer in defining a data

[4] `http://www.instancematching.org/oaei/imei2010.html`.
[5] `http://www.cs.vu.nl/~laurah/oaei/2010/`.

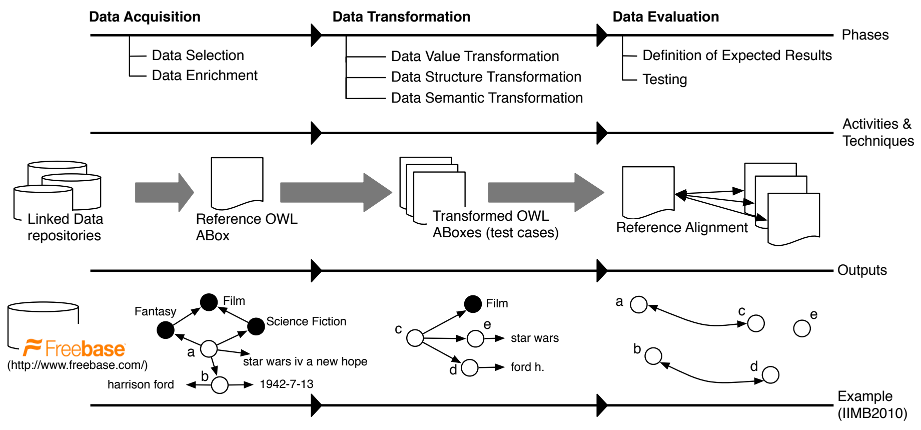


Fig. 1. The SWING approach

description scheme capable of supporting the simulation of a wide spectrum of data heterogeneities.

Data transformation techniques. In the subsequent *data transformation* activity the TBox is unchanged, while the ABox is modified in several ways by generating a set of new ABoxes, called *test cases*. Each test case, is produced by transforming the individual descriptions in the reference ABox in new individual descriptions that are inserted in the test case at hand. The goal of transforming the original individuals is twofold: on one side, we provide a simulated situation where data referred to the same objects are provided in different datasources; on the other side, we generate a number of datasets with a variable level of data quality and complexity.

Data evaluation techniques. Finally, in the *data evaluation* activity, we automatically create a ground-truth as a reference alignment for each test case. A reference alignment contains the mappings (in some contexts called "links") between the reference ABox individuals and the corresponding transformed individuals in the test case. These mappings are what an instance matching application is expected to find between the original ABox and the test case.

As a running example illustrating our approach, in this paper, we present the IIMB 2010 benchmark[6], which has been created by applying our SWING approach. IIMB 2010 has been used in the instance matching track of OAEI 2010. IIMB 2010 is a collection of OWL ontologies consisting of 29 concepts, 20 object properties, 12 data properties and thousands of individuals divided into 80 test cases. In fact in IIMB 2010, we have defined 80 test cases, divided into 4 sets of 20 test cases each. The first three sets are different implementations of data

[6] `http://www.instancematching.org/oaei/imei2010.html`

value, data structure, and data semantic transformations, respectively, while the fourth set is obtained by combining together the three kinds of transformations. IIMB 2010 is created by extracting data from Freebase [2], an open knowledge base that contains information about 11 Million real objects including movies, books, TV shows, celebrities, locations, companies and more. Data extraction has been performed using the query language JSON together with the Freebase JAVA API[7].

4 Data acquisition

The SWING data acquisition phase is articulated in two tasks, called data selection and data enrichment. The task of *data selection* has the aim to find the right balance between the creation of a realistic benchmark and the manageability of the dataset therein contained. In SWING, the data selection task is performed according to an initial query that is executed against a linked data repository with the supervision of the evaluation designer. In particular, the size of the linked data source is narrowed down by i) selecting a specific subset of all available linked data classes and ii) limit the individuals belonging to these selected classes. With the latter selection technique, we can easily scale the number of individuals from hundreds to millions only by adjusting one single parameter. The goal of the *data enrichment* task is to provide a number of data enrichment techniques which can be applied to any linked data source for extending its structural and semantic complexity from the description logic $\mathcal{ALE}(\mathcal{D})$ up to $\mathcal{ALCHI}(\mathcal{D})$. This data enrichment has to be realized, because in the open linked data cloud the concept hierarchies are usually very low and disjointness axioms or domain and range restrictions are rarely defined. The limited level of semantic complexity is a distinguishing feature of linked data. Nevertheless, many matching applications are capable of dealing with data at different levels of OWL expressiveness.

To illustrate the SWING enrichment techniques, we will refer to a small snippet of the IIMB 2010 benchmark displayed in Figure 4. The black colored nodes, arrows, and names represent information that has been extracted from Freebase, while the gray colored information has been added according to the following enrichment techniques of our SWING approach.

Add Super Classes and Super Properties. The designer can adopt two different approaches for determining new super classes. The first one is a bottom-up approach where new super classes are created by aggregating existing classes. Thereby, the evaluation designer has to define a super class name which encompasses all the classes to include. The second approach is top-down and it requires to define how to split a class into more subclasses. The same approaches can be applied for determining super object properties, respectively. This task is mainly

[7] `http://code.google.com/p/freebase-java/`. However, we stress that any kind of linked data-compatible source can be used to implement the SWING approach.

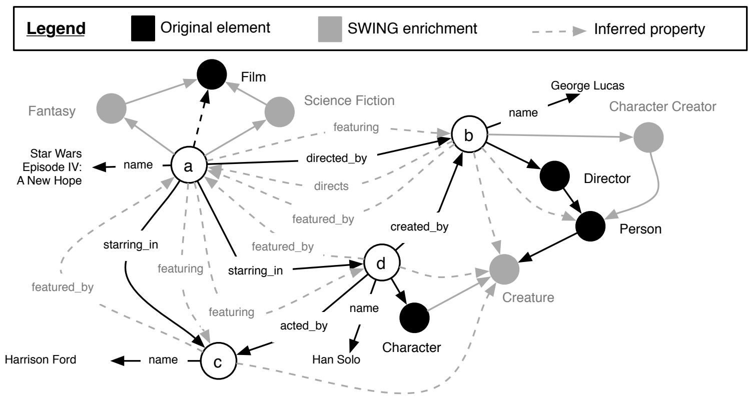


Fig. 2. A portion of IIMB 2010

performed manually by the designer, with the support of the system to avoid the insertion of inconsistency errors.

In IIMB we added for instance following statements for classes and object properties:

$$(Person \sqcup Character) \sqsubseteq Creature$$

$$(directed\_by \sqcup acted\_by \sqcup starring\_in) \sqsubseteq featuring$$

Convert Attributes to Class Assertions. Sometimes, linked data sources contain string attributes which can be "upgraded" to classes. A good indicator for such attributes is if the data values are restricted to a number of predefined expressions like for example $male$ and $female$ or a restricted number of terms denoting concepts, like for example *red*, *green*, and *yellow*. In this case, an external lexical system such as for example WordNet can be used to support the designer in finding those terms that can be considered as candidates for the class assertion conversion. For example, in Freebase, every film has an attribute *genre* with values like *Fantasy*, *Science Fiction*, *Horror*, and many more. This attribute was used in IIMB 2010 to derive additional class assertions as subclasses of *Film* as shown in Figure 4.

$$(Fantasy \sqcup Science\ Fiction \sqcup Horror \sqcup ...) \sqsubseteq Film$$

Determine Disjointness Restrictions. The challenge of this task is to add as many disjointness restrictions as possible while ensuring the ontology consistency. This could be realized by trying to add all possible disjointness combinations and by checking the consistency of the ontology after each combination. If the ontology does not turn to inconsistency we integrate the axiom, otherwise, we discard it. In general, disjointness axioms can not only be added for classes, but also for object properties. For the sake of readability, disjointness axioms are not shown in Figure 4. However, in IIMB 2010, the classes *Film*, *Location*, *Language*, *Budget*, and *Creature* were set to be pairwise disjoint.

Enrich with Inverse Properties. We perform the insertion of inverse properties by creating an object property with both the active and the passive verb forms used as property names. To automate this procedure, we rely on a lexical systems (e.g., WordNet) to determine the active/passive verb forms to insert. Moreover, the evaluation designed can manually insert the inverse property for those property names that are not retrieved in WordNet. For example, in IIMB 2010, we added the property *directs* as the inverse of the existing property *directed_by*.

Specify Domain and Range Restrictions. For all existing properties, the challenge is to find the best - that means the narrowest - domain and range without turning the ontology to inconsistency. For an object property, all the possible domain and range restrictions can be determined by attempting to assign all the existing classes to be the potential domain/range. If the ontology is still consistent after the assignment of this new domain/range restriction, the change is saved, otherwise it is discarded. In the example of IIMB 2010, among the others, the following domain and range restrictions have been added to the object property *created_by*:

$$\exists created\_by \sqsubseteq Character \quad \wedge \quad \exists created\_by^{-} \sqsubseteq Character\ Creator$$

5 Data transformation

Once the data acquisition phase is executed and a reference ontology $\mathcal{O}$ is produced, we start the SWING data transformation phase that has the goal of producing a set $\mathcal{T} = \{\mathcal{O}_1, \mathcal{O}_2, \ldots, \mathcal{O}_{n-1}, \mathcal{O}_n\}$ of new ontologies, called test cases. Each test case $\mathcal{O}_i \in \mathcal{T}$ has the same schema (i.e., TBox) of $\mathcal{O}$ but a different set of instances (ABox) generated by transforming the ABox $\mathcal{A}_{\mathcal{O}}$ of $\mathcal{O}$. In detail, the input of each transformation is a reference ontology $\mathcal{O}$ and a configuration scheme $\mathcal{C}$, which contains the specification of properties involved in the transformations process, the kind of transformations enforced, and the parameters required by the transformation functions. The output is a new ontology $\mathcal{O}_i$. The implementation of each ontology transformation can be described in terms of a transformation function $\theta : \mathcal{A}_{\mathcal{O}} \rightarrow \mathcal{A}^i_{\mathcal{O}}$, where $\mathcal{A}_{\mathcal{O}}$ and $\mathcal{A}^i_{\mathcal{O}}$ denote two ABoxes consistent with the TBox $\mathcal{T}_{\mathcal{O}}$ of the ontology $\mathcal{O}$. The transformation function θ maps each assertion $\alpha_k \in \mathcal{A}_{\mathcal{O}}$ into a new set of assertions $\theta(\alpha_k)_{\mathcal{C}}$ according to the configuration scheme $\mathcal{C}$. Thus, given a configuration scheme $\mathcal{C}$ and an ontology $\mathcal{O}$, a test case $\mathcal{O}_i$ is produced as follows:

$$\mathcal{O}_i = \mathcal{T}_{\mathcal{O}} \cup \mathcal{A}^i_{\mathcal{O}} \ with \ \mathcal{A}^i_{\mathcal{O}} = \bigcup_{k=1}^{N} \theta(\alpha_k)_{\mathcal{C}}$$

where N is the number of assertions in $\mathcal{A}_{\mathcal{O}}$. In SWING, we take into account two kinds of assertions for the transformation purposes, namely class assertions and property assertions. A class assertion has the form $C(x)$ and denotes the fact that and individual x is an instance of class C (i.e., the type of x is C). A property assertion has the form $P(x, y)$ and denotes the fact that an individual x is featured by a property P which has value y. P may be either an object property or a data property. In the first case, the value y is another individual, while in the second case y is a concrete value. As an example, in the reference ontology used for IIMB 2010, we have the following assertions:

$$\alpha_1 : \mathit{Director}(b),\ \alpha_2 : \mathit{name}(b, \text{“George Lucas”}),\ \alpha_3 : \mathit{created\_by}(d, b)$$

denoting the fact that b is a $\mathit{Director}$ whose name is represented by the string “George Lucas”. Moreover the object denoted by the individual d is created by the individual b (d denotes the character “Han Solo” as shown in Figure 2). Both the kinds of assertions taken into account in SWING can be described in terms of an assertion subject, denoted α^s, that is the individual which the assertion α is referred to, an assertion predicate, denoted α^p, that is the RDF property $\mathit{rdf} : \mathit{type}$ in case of class assertions or the property involved in the assertion in case of property assertions, and an assertion object, denoted α^o, that is a class in case of class assertions and a concrete or abstract value in case of property assertions. For example, referring to the IIMB 2010 example above, we have $\alpha_1^s = b$, $\alpha_1^p = \mathit{rdf} : \mathit{type}$, $\alpha_1^o = \mathit{Director}$ and $\alpha_2^s = b$, $\alpha_2^p = \mathit{name}$, $\alpha_2^o =$ “George Lucas”. According to this notation, we can define the individual description D_j of and individual j into an ABox $\mathcal{A}_{\mathcal{O}}$ as follows:

$$D_j = \{\alpha_k \in \mathcal{A}_{\mathcal{O}} \mid \alpha_k^s = j\}$$

that is the set of all assertions in $\mathcal{A}_{\mathcal{O}}$ having j as subject. According to this notion of individual description, we define also the notion of individual transformation $\theta(j)$ as the result of the transformation $\theta(\alpha_k)$ of each assertion α_k in the definition D_j of j.

5.1 Data Transformation Procedure

The data transformation of an ontology $\mathcal{O}$ into an ontology $\mathcal{O}_i$ is run as a procedure articulated in three steps:

Preprocessing of the Initial Ontology. The preprocessing step has the goal of adding some axioms to the ontology TBox $\mathcal{O}$, that will be the reference TBox for the rest of the transformation and will be identical for all the test cases. These additions are required in order to implement some of the subsequent data transformations without altering the reference TBox. In particular, we add two kind of axioms. As a first addition, we take into account all the data properties $P_i \in \mathcal{O}$ and, for each property, we add a new object property R_i, such that $\mathcal{O} = \mathcal{O} \cup R_i$. Moreover, we add a data property $\mathit{has\_value}$ to $\mathcal{O}$. These additions are required for transforming data property assertions into object property assertions. The second addition is performed only if the semantic complexity of the ontology chosen by the designer allows the usage of inverse properties. In this case, we take into account all the object properties R_i that are not already associated with an inverse property and we add to $\mathcal{O}$ a new property K_i such that $K_i \equiv R_i^-$.

Deletion/Addition of Individuals. The SWING approach allows the evaluation designer to select a portion of individuals that must be deleted and/or duplicated in the new ontology. The reason behind this functionality is to obtain a new ontology where each original individual can have none, one, or more matching counterparts. The goal is to add some noise in the expected mappings in such a way that the resulting benchmark contains both test cases where each original instance has only one matching counterpart (i.e., one-to-one mappings) and test cases where each original instance may have more than one matching counterpart (i.e., one-to-many mappings). This is a required feature in a benchmark to avoid that those matching applications that produce only one-to-one mappings are favored. In particular, the evaluation designer can choose the percentage t_d (expressed in the range [0,1]) of individuals that are candidate for deletion and the percentage t_a of individuals that are candidate for addition. Then, given the number N_I of individuals in the initial ontology $\mathcal{O}$, SWING calculates the number C_I of individuals that have to be deleted as $C_I = \lfloor t_d \cdot N_I \rfloor$. Given C_I, two strategies, called *deterministic* and *non-deterministic* strategies, are defined to randomly choose the individuals to eliminate. In the deterministic strategy, we randomly choose C_I individuals from $\mathcal{O}$. The assertions in the descriptions of the chosen individuals are simply not submitted to transformation and, thus, do not appear in the new ontology $\mathcal{O}_i$. In the non-deterministic strategy, we take into account all the individuals in $\mathcal{O}$ once at a time. For each individual, we generate a random value r in the range [0,1]. If $r \leq t_d$, the individual is not submitted to transformation. Every time the transformation is not executed, we add 1 to a counter c. This procedure is iterated until $c < C_I$ or all the individuals in $\mathcal{O}$ have been considered. The advantage of the deterministic strategy is that it is possible to control the exact number of transformations that are generated. The advantage of a non-deterministic choice is to keep the transformation process partially blind even for the evaluation designer. Similarly, the number A_I of individuals to be added is calculated as $A_I = \lfloor t_a \cdot (N_I - C_I) \rfloor$. We randomly select the individuals to be added and for each of these individuals i, we create a new individual i' in the text case by substituting the individual identifier i with a new randomly generated identifier i'. Then, each assertion $\alpha_k \in D_i$ is transformed by substituting any reference to i with i'. In such a way, in the test case we will have a copy of each individual description plus the individual description transformation $\theta(i)$.

Individuals Transformation. For each individual description D_i and for each assertion $\alpha_j \in D_i$, we calculate the transformation $\theta(\alpha_j)_{\mathcal{C}}$ according to the configuration scheme $\mathcal{C}$, that is defined by the evaluation designer. Every transformation $\theta(\alpha_j)_{\mathcal{C}}$ is seen as a ordered sequence of transformation operations. Each operation takes a set A of assertions in input and returns a set A' of transformed assertions as output. The input of the first transformation operation is the singleton set $\{\alpha_j\}$, while the output of the final operation in the sequence is the transformation $\theta(\alpha_j)_{\mathcal{C}}$. Transformation operations are distinguished in three categories, namely *data value transformation*, *data structure transformation*, and *data semantic transformation.* Due to space reasons, we cannot describe operations in detail, but we summarize them in Table 1 and we will provide an example of their application in the following.

Table 1. Summary of data transformation operations provided by SWING

	Data Value	Data Structure	Data Semantic
Add	γ	σ, ζ	ι
Delete	ρ	δ	λ, π, ι
Modify	ρ, κ	τ, ζ	λ, ω, π

γ = Random token/character addition	τ = Property type transformation
ρ = Random token/character modification	ι = Creation of inverse property assertions
κ = Specific data modification	λ = Deletion/modification of class assertions
σ = Property assertion addition	π = Creation of super-property assertions
ζ = Property assertion splitting	ω = Classification of individuals in disjoint classes
δ = Property assertion deletion	

Data value transformation operations work on the concrete values of data properties and their datatypes when available. The output is a new concrete value. An example of data value transformation, is shown in Table 2.

Table 2. Examples of data value transformations

Operation	Original value	Transformed value
Standard transformation	Luke Skywalker	L4kd Skiwaldek
Date format	1948-12-21	December 21, 1948
Name format	Samuel L. Jackson	Jackson, S.L.
Gender format	Male	M
Synonyms	Jackson has won multiple awards [...]	Jackson has gained several prizes [...]
Integer	10	110
Float	1.3	1.30

Data structure transformation operations change the way data values are connected to individuals in the original ontology graph and change the type and number of properties associated with a given individual. A comprehensive example of data structure transformation is shown in Table 3, where an initial set of assertions A is transformed in the corresponding set of assertions A' by applying the property type transformation, property assertion deletion/addition, and property assertion splitting.

Table 3. Example of data structure transformations

A	A'
$name(n,$ "Natalie Portman"$)$	$name(n,$ "Natalie"$)$
$born\_in(n, m)$	$name(n,$ "Portman"$)$
$name(m,$ "Jerusalem"$)$	$born\_in(n, m)$
$gender(n,$ "Female"$)$	$name(m,$ "Jerusalem"$)$
$date\_of\_birth(n,$ "1981-06-09"$)$	$name(m,$ "Auckland"$)$
	$obj\_gender(n, y)$
	$has\_value(y,$ "Female"$)$

Finally, data semantic transformation operations are based on the idea of changing the way individuals are classified and described in the original ontology. For the sake of brevity, we illustrate the main semantic transformation operations by means of the following example, by taking into account the portion of $\mathcal{T}_\mathcal{O}$ and the assertions sets A and A' shown in Table 4.

Table 4. Example of data semantic transformations

$\mathcal{T}_\mathcal{O}$
$Character \sqsubseteq Creature$, $created\_by \equiv creates^-$, $acted\_by \sqsubseteq featuring$, $Creature \sqcap Country \sqsubseteq \bot$

A	A'
$Character(k)$	$Creature(k)$
$Creature(b)$	$Country(b)$
$Creature(r)$	$\top(r)$
$created\_by(k, b)$	$creates(b, k)$
$acted\_by(k, r)$	$featuring(k, r)$
$name(k,$ "Luke Skywalker"$)$	$name(k,$ "Luke Skywalker"$)$
$name(b,$ "George Lucas"$)$	$name(b,$ "George Lucas"$)$
$name(r,$ "Mark Hamill"$)$	$name(r,$ "Mark Hamill"$)$

In the example, we can see how the combination of all the data semantic operations may change the description of the individual k. In fact, in the original set A, k (i.e., the Luke Skywalker of Star Wars) is a character created by the individual b (i.e., George Lucas) and acted by r (i.e., Mark Hamill). In A' instead, k is a more generic "creature" and also the relation with r is more generic (i.e., "featuring" instead of "acted_by"). Moreover, individual k is not longer *created_by* b as it was in A, but it is b that *creates* k. But the individual b of A' cannot be considered the same than $b \in A$, since the class *Creature* and *Country* are disjoint.

According to Table 1, data transformation operations may also be categorized as operations that *add*, *delete* or *modify* the information originally provided by the initial ontology. Table 1 shows also how some operations are used in order to implement more than one action over the initial ontology, such as in case of deletion and modifications of string tokens that are both implemented by means of the operation ρ. Moreover, some operations cause more than one consequence on the initial ontology. For example, the property assertion splitting ζ causes both the modification of the original property assertion and the addition of some new assertions in the new ontology.

5.2 Combining Transformations and Defining the Expected Results

When a benchmark is generated with SWING it is usually a good practice to provide a set of test cases for each category of data transformation plus a fourth bunch of test cases where data value, data structure, and data semantic transformations are combined together. The combination of different transformations in SWING is easy both in case of transformations of the same category and in case of cross-category transformations. In fact, all the transformation operations work on assertions sets and produce other assertions sets. Thus, the combination is obtained by executing the desired transformations one over the output of the other. As an example, we consider the initial assertion set $A = \{name(b, \text{“George Lucas”})\}$ and the transformation sequence $A \rightarrow \rho(A, 0.5) \rightarrow A' \rightarrow \tau(A', 1.0) \rightarrow A''\iota(A'', 1.0) \rightarrow A'''$ that produces the following results: $A = name(b, \text{“George Lucas”})$, $A' = name(b, \text{“YFsqap Lucas”})$, $A'' = obj\_name(b, x)$, $has\_value(x, \text{“YFsqap Lucas”})$, $A''' = obj\_name^{-}(x, b)$, $has\_value(x, \text{“YFsqap Lucas”})$.

As a last step in the benchmark generation, for each test case $\mathcal{O}_i$ we define a set of mappings $\mathcal{M}_{\mathcal{O},\mathcal{O}_i}$ that represents the set of correspondences between the individuals of the ontology $\mathcal{O}$ and the individuals of the the test case $\mathcal{O}_i$ that a matching application is expected to find when matching $\mathcal{O}$ against $\mathcal{O}_i$. For each individual j in $\mathcal{O}$ we define a new individual j' and we substitute any reference to j in $\mathcal{O}_i$ with a reference to j'. Then, if the pair j, j' is not involved in any operation of classification of individuals in disjoint classes, we insert $m(j, j')$ in $\mathcal{M}_{\mathcal{O},\mathcal{O}_i}$.

A comprehensive example of data transformation taken form IIMB 2010 is shown in Figure 3, together with the expected results generated for the test case at hand.

6 Experimental Results

In order to evaluate the applicability of the SWING approach to the evaluation of matching applications, we have executed six different matching algorithms against IIMB 2010 and we have compared precision and recall of each algorithm against the different test cases of the benchmark. The six algorithms have been chosen to represent some of the most popular and reliable matching techniques in the field of instance matching. We recall that the goal of our evaluation is to verify the capability of the IIMB 2010 benchmark generated with SWING to provide a reliable and sufficiently complete dataset to measure the effectiveness of different and often complementary matching algorithms/applications. The considered matching algorithms are divided into two categories, namely *simple matching* and *complex matching*. Simple matching algorithms are three

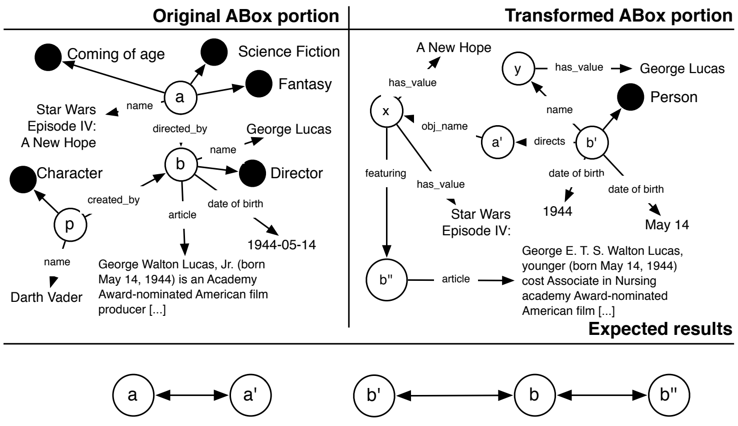
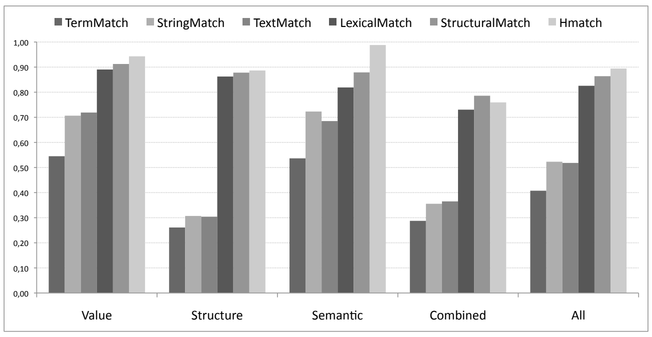


Fig. 3. Example of data transformation taken from IIMB 2010

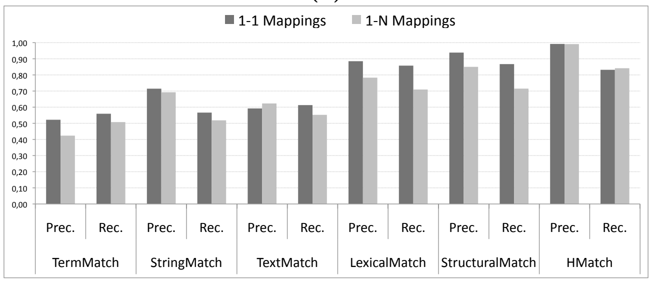
variations of string matching functions that are used for the comparison of a selected number of property values featuring the test case instances. Complex matching algorithms work on the structure and semantics of test cases and on the expected cardinality of resulting mappings. In this category we executed the algorithms *LexicalMatch*, *StructuralMatch*, and *HMatch*. LexicalMatch and StructuralMatch [11] use integer linear programming to calculate the optimal one-to-one alignment based on the sum of the lexical similarity values. In LexicalMatch, no structural information is considered. StructuralMatch uses both, lexical and structural information. Finally, HMatch [4] is a flexible matching suite where a number of matching techniques are implemented and organized in different modules providing linguistic, concept, and instance matching techniques that can be invoked in combination.

A summary of the matching results is reported in Figure 4(a), where we show the average values of the harmonic mean of precision and recall (i.e., FMeasure) for each algorithm over data value, structure and semantic transformation test cases, respectively. The last two chunks of results refer to a combination of transformations and to the benchmark as a whole, respectively.

The goal of our experiments is to evaluate the IIMB 2010 *effectiveness*, that is the capability of distinguishing among different algorithms where they are tested on different kinds of transformations. To this end, we observe that IIMB 2010 allows to stress the difference between simple and complex matching algorithms. In fact, in Figure 4(a), the FMeasure for simple matching algorithms is between 0.4 and 0.5, while we obtain values in the range 0.8-0.9 with complex algorithms. It is interesting to see how simple algorithms have best performances on value transformations and worst performances on structural transformations. This result is coherent with the fact that simple matching does not take into account neither the semantics nor the structure of individuals, and proves that SWING simulates structural transformation in a correct way. In case of seman-



(a)



(b)

Fig. 4. Results for the execution of different matching algorithms against IIMB 2010

tic transformations instead, simple algorithms have quite good performances because many semantic transformations affect individual classification, which is an information that is ignored by simple algorithms. In Figure 4(b), we show the values of precision and recall of the considered algorithms in the test cases where all the expected mappings were one-to-one mappings (i.e., 1-1 Mappings) and in the test cases where one-to-many mappings were expected for 20% of the individuals (i.e., 1-N Mappings). We note that the algorithms that are not based on the assumption of finding one-to-one mappings (i.e., simple matching algorithms and HMatch) have similar results in case of 1-1 and 1-N mappings. Instead, LexicalMatch and StructuralMatch are based on the idea of finding the best 1-1 mapping set. Thus, precision and recall of these algorithms are lower when 1-N mappings are expected. The number of individuals corresponding to more than one individual is about 20% of the total number of individuals and this percentage corresponds to the degradation of results that we can observe for LexicalMatch and StructuralMatch.

7 Concluding Remarks

In this paper we have presented SWING, our approach to the supervised generation of benchmarks for the evaluation of matching applications. Experiments presented in the paper show that SWING is applicable to the evaluation of real matching applications with good results. Our future work is focused on collecting more evaluations results in the instance matching evaluation track of OAEI 2010, where SWING has been used to generate the IIMB 2010 benchmark. Moreover, we are interested in studying the problem of extending SWING to the creation of benchmarks for evaluation of ontology matching applications in general, by providing a suite of comprehensive evaluation techniques and tools tailored for the specific features of TBox constructs.

Acknowledgements. Heiner Stuckenschmidt is partially supported by the European project SEALS (IST-2009-238975).

References

1. Alexe, B., Tan, W., Velegrakis, Y.: STBenchmark: towards a Benchmark for Mapping Systems. Proc. of the VLDB Endowment 1(1), 230–244 (2008)
2. Bollacker, K., Evans, C., Paritosh, P., Sturge, T., Taylor, J.: Freebase: a Collaboratively Created Graph Database for Structuring Human Knowledge. In: Proc. of the ACM SIGMOD Int. Conference on Management of Data. pp. 1247–1250 (2008)
3. Bressan, S., Li Lee, M., Guang Li, Y., Lacroix, Z., Nambiar, U.: The XOO7 Benchmark. In: Proc. of the 1st VLDB Workshop on Efficiency and Effectiveness of XML Tools, and Techniques (EEXTT 2002) (2002)
4. Castano, S., Ferrara, A., Montanelli, S.: Matching Ontologies in Open Networked Systems: Techniques and Applications. Journal on Data Semantics V (2006)
5. Duchateau, F., Bellahse, Z., Hunt, E.: XBenchMatch: a Benchmark for XML Schema Matching Tools. In: Proc. of the 33rd Int. Conference on Very Large Data Bases (VLDB 2007) (2007)
6. Euzenat, J., Ferrara, A., Hollink, L., et al.: Results of the Ontology Alignment Evaluation Initiative 2009. In: Proc. of the 4th Int. Workshop on Ontology Matching (OM 2009) (2009)
7. Euzenat, J., Shvaiko, P.: Ontology Matching. Springer-Verlag (2007)
8. Ferrara, A., Lorusso, D., Montanelli, S., Varese, G.: Towards a Benchmark for Instance Matching. In: Proc. of the ISWC Int. Workshop on Ontology Matching (OM 2008) (2008)
9. Guo, Y., Pan, Z., Heflin, J.: An Evaluation of Knowledge Base Systems for Large OWL Datasets. In: Proc. of the 3rd Int. Semantic Web Conference (ISWC 2004) (2004)
10. Koepcke, H., Thor, A., Rahm, E.: Evaluation of Entity Resolution Approaches on Real-World Match Problems. In: Proc. of the 36th Int. Conference on Very Large Data Bases (VLDB 2010) (2010)
11. Noessner, J., Niepert, M., Meilicke, C., Stuckenschmidt, H.: Leveraging Terminological Structure for Object Reconciliation. The Semantic Web: Research and Applications pp. 334–348 (2010)
12. Perry, M.: TOntoGen: A Synthetic Data Set Generator for Semantic Web Applications. AIS SIGSEMIS Bulletin 2(2), 46–48 (2005)
13. Yatskevich, M., Giunchiglia, F., Avesani, P.: A Large Scale Dataset for the Evaluation of Matching Systems. In: Proc. of the 4th European Semantic Web Conference (ESWC 2007), Poster Session (2007)