Ontology Matching
OM-2011

Proceedings of the ISWC Workshop

Introduction

Ontology matching[1] is a key interoperability enabler for the Semantic Web, as well as a useful tactic in some classical data integration tasks dealing with the semantic heterogeneity problem. It takes ontologies as input and determines as output an alignment, that is, a set of correspondences between the semantically related entities of these ontologies. These correspondences can be used for various tasks, such as ontology merging, data translation, query answering or navigation on the web of data. Thus, matching ontologies enables the knowledge and data expressed in the matched ontologies to interoperate.

The workshop has three goals:

- To bring together leaders from *academia*, *industry* and *user institutions* to assess how academic advances are addressing real-world requirements. The workshop will strive to improve academic awareness of industrial and final user needs, and therefore direct research towards those needs. Simultaneously, the workshop will serve to inform industry and user representatives about existing research efforts that may meet their requirements. The workshop will also investigate how the ontology matching technology is going to evolve.
- To conduct an extensive and rigorous evaluation of ontology matching approaches through the OAEI (Ontology Alignment Evaluation Initiative) 2011 campaign[2]. The particular focus of this year's OAEI campaign is on real-world specific matching tasks involving, e.g., linked open data and biomedical ontologies. Therefore, the ontology matching evaluation initiative itself will provide a solid ground for discussion of how well the current approaches are meeting business needs.
- To examine similarities and differences from database schema matching, which has received decades of attention but is just beginning to transition to mainstream tools.

We received 33 submissions for the technical track of the workshop. The program committee selected 7 submissions for oral presentation and 16 submissions for poster presentation. 18 matching systems participated in this year's OAEI campaign. Further information about the Ontology Matching workshop can be found at: `http://om2011.ontologymatching.org/`.

[1] `http://www.ontologymatching.org/`
[2] `http://oaei.ontologymatching.org/2011`

Acknowledgments. We thank all members of the program committee, authors and local organizers for their efforts. We appreciate support from the Trentino as a Lab (TasLab)[3] initiative of the European Network of the Living Labs[4] at Informatica Trentina SpA[5], the EU SEALS (Semantic Evaluation at Large Scale)[6] project and the Semantic Valley[7] initiative.

Pavel Shvaiko
Jérôme Euzenat
Tom Heath
Christoph Quix
Ming Mao
Isabel Cruz

October 2011

[3] http://www.taslab.eu
[4] http://www.openlivinglabs.eu
[5] http://www.infotn.it
[6] http://www.seals-project.eu
[7] http://www.semanticvalley.org/index_eng.htm

Organization

Organizing Committee

Pavel Shvaiko, TasLab, Informatica Trentina SpA, Italy
Jérôme Euzenat, INRIA & LIG, France
Tom Heath, Talis Systems Ltd, UK
Christoph Quix, RWTH Aachen University, Germany
Ming Mao, SAP Labs, USA
Isabel Cruz, The University of Illinois at Chicago, USA

Program Committee

Paolo Besana, University of Edinburgh, UK
Chris Bizer, Free University Berlin, Germany
Olivier Bodenreider, National Library of Medicine, USA
Paolo Bouquet, OKKAM, Italy
Marco Combetto, Informatica Trentina, Italy
Jérôme David, INRIA & LIG, France
Alfio Ferrara, University of Milan, Italy
Gabriele Francescotto, OpenContent, Italy
Fausto Giunchiglia, University of Trento, Italy
Bin He, IBM, USA
Eduard Hovy, ISI, University of Southern California, USA
Wei Hu, Nanjing University, China
Ryutaro Ichise, National Institute of Informatics, Japan
Antoine Isaac, Vrije Universiteit Amsterdam & Europeana, Netherlands
Krzysztof Janowicz, Pennsylvania State University, USA
Anja Jentzsch, Free University Berlin, Germany
Yannis Kalfoglou, Ricoh Europe plc, UK
Patrick Lambrix, Linköpings Universitet, Sweden
Monika Lanzenberger, Vienna University of Technology, Austria
Rob Lemmens, ITC, The Netherlands
Maurizio Lenzerini, University of Rome La Sapienza, Italy
Vincenzo Maltese, University of Trento, Italy
Fiona McNeill, University of Edinburgh, UK
Christian Meilicke, University of Mannheim, Germany
Peter Mork, The MITRE Corporation, USA
Nico Lavarini, Cogito, Italy
Andriy Nikolov, Open University, UK
Natasha Noy, Stanford University, USA
Leo Obrst, The MITRE Corporation, USA
Matteo Palmonari, University of Milan Bicocca, Italy
Yefei Peng, Google, USA

Evan Sandhaus, New York Times, USA
François Scharffe, LIRMM, France
Luciano Serafini, Fondazione Bruno Kessler - IRST, Italy
Kavitha Srinivas, IBM, USA
Umberto Straccia, ISTI-C.N.R., Italy
Ondřej Šváb-Zamazal, Prague University of Economics, Czech Republic
Cássia Trojahn, INRIA & LIG, France
Raphaël Troncy, EURECOM, France
Giovanni Tummarello, Fondazione Bruno Kessler - IRST, Italy
Lorenzino Vaccari, European Commission - Joint Research Center, Italy
Ludger van Elst, DFKI, Germany
Shenghui Wang, Vrije Universiteit Amsterdam, Netherlands
Baoshi Yan, LinkedIn, USA
Songmao Zhang, Chinese Academy of Sciences, China

Additional Reviewers

Robert Isele, Free University Berlin, Germany

Table of Contents

PART 2 - OAEI Papers

PART 3 - Posters

A Time-Efficient Hybrid Approach to Link Discovery

Axel-Cyrille Ngonga Ngomo[1]

Department of Computer Science
University of Leipzig
Johannisgasse 26, 04103 Leipzig
ngonga@informatik.uni-leipzig.de,
WWW home page: http://bis.uni-leipzig.de/AxelNgonga

Abstract. With the growth of the Linked Data Web, time-efficient Link Discovery frameworks have become indispensable for implementing the fourth Linked Data principle, i.e., the provision of links between data sources. Due to the sheer size of the Data Web, detecting links even when using trivial specifications based on a single property can be very time-demanding. Moreover, non-trivial Link Discovery tasks require complex link specifications and are consequently even more challenging to optimize with respect to runtime. In this paper, we present a novel hybrid approach to link discovery that combines two very fast algorithms. Both algorithms are combined by using original insights on the translation of complex link specifications to combinations of atomic specifications via a series of operations on sets and filters. We show in three experiments that our approach outperforms SILK by more than six orders of magnitude while abiding to the restriction of not losing any link.

Keywords: Linked Data, Link Discovery, Algorithms, Constraints

1 Introduction

The Linked Data Web has evolved from 12 knowledge bases in May 2007 to 203 knowledge bases in September 2010, i.e., in less than four years [6]. While the number of RDF triples available in the Linked Data Web has now surpassed 27 billion, less than 3% of these triples are links between knowledge bases [10]. Yet, links between knowledge bases play a key role in important tasks such as cross-ontology question answering [9], large-scale inferences [15] and data integration [2]. Given the enormous amount of information available on the Linked Data Web, time-efficient Link Discovery (LD) frameworks have become indispensable for implementing the fourth Linked Data principle, i.e., the provision of links between data sources [16, 10]. These frameworks rely on *link specifications*, which explicate conditions for computing new links between entities in knowledge bases. Due to the mere size of the Web of Data, detecting links even when using trivial specifications can be very time-demanding. Moreover, non-trivial LD tasks require complex link specifications for discovering accurate links between instances and are consequently even more challenging to optimize with respect to runtime. In this paper, we present a novel lossless hybrid approach to LD. Our approach is based on original insights on the distribution of property domain and ranges on the Web of Data. Based on these insights, we infer the requirements to efficient LD frameworks. We then use these requirements to specify the time-efficient approaches that underlie our framework, LIMES version 0.5[1]. We show that our framework outperforms state-of-the-art frameworks by several orders of magnitude with respect to runtime without losing links.

The contributions of this paper are as follows:

1. We present a formal grammar for link specifications that encompasses the functionality of state-of-the-art frameworks for LD.
2. Based on this grammar, we present a very time-efficient approach for LD that is based on translating complex link specifications into a combination of atomic specifications via a concatenation of operations on sets and filter operations.
3. We use this method to enable the PPJoin+ [18] algorithm to be used for processing complex link specifications.
4. We specify and evaluate the HYpersphere aPPrOximation algorithm HYPPO, a fully novel LD approach designed to operate on numeric values.
5. We evaluate our approach against SILK [7] within three experiments and show that we outperform it by up to six orders of magnitude with respect to runtime while abiding to the constraint of not losing links.

The rest of this paper is structured as follows: In Section 2, we give a brief overview of related work on LD and related research fields. Section 3 presents the preliminaries to our work. These preliminaries are the basis for Section 4, in which we specify a formal grammar for link specification and an approach to convert complex link specifications into an aggregation of atomic link specifications via set operations and filters. We subsequently present the core algorithms underlying our approach in Section 5. In section 6, we evaluate our approaches in three different large-scale experiments and show that we outperform the state-of-the-art approach SILK. After a discussion of our findings, we present our future work and conclude.

2 Related Work

Current frameworks for LD on the Web of Data can be subdivided into two categories: *domain-specific* and *universal* frameworks [10]. Domain-specific LD frameworks aim to discover links between knowledge bases from a particular domain. For example, the RKB knowledge base (RKB-CRS) [5] uses Universal Resource Identifier (URI) lists to compute links between universities and conferences. Another domain-specific tool is GNAT [12], which discovers links between

[1] LIMES stands for Link Discovery Framework for Metric Spaces. An online demo of the framework can be found at http://limes.sf.net

music data sets by using audio fingerprinting. Further simple or domain-specific approaches can be found in [14, 11].

Universal LD frameworks are designed to carry out mapping tasks independent from the domain of the source and target knowledge bases. For example, RDF-AI [13] implements a five-step approach that comprises the preprocessing, matching, fusion, interlinking and post-processing of data sets. SILK [7] (Version 2.3) implements a time-efficient and lossless approach that maps complex configurations to a multidimensional metric space. A blocking approach is then used in the metric space to reduce the number of comparisons by generating overlapping blocks. The original LIMES approach [10] presupposes that the datasets to link are in a metric space. It then uses the triangle inequality to portion the metric space so as to compute pessimistic approximations of distances. Based on these approximations, it can discard a large number of computations without losing links.

Although LD is closely related with record linkage [17, 4] and deduplication [3], it is important to notice that LD goes beyond these two tasks as LD aims to provide the means to link entities via arbitrary relations. Different blocking techniques such as standard blocking, sorted-neighborhood, bi-gram indexing, canopy clustering and adaptive blocking have been developed by the database community to address the problem of the quadratic time complexity of brute force comparison [8]. In addition, very time-efficient approaches have been proposed to compute string similarities for record linkage, including AllPairs [1], PPJoin and PPJoin+ [18]. However, these approaches alone cannot deal with the diversity of property values found on the Web of Data as they cannot deal with numeric values. In addition, most time-efficient string matching algorithms can only deal with simple link specifications, which are mostly insufficient when computing links between large knowledge bases.

The novel version of the LIMES framework goes beyond the state of the art (including previous versions of LIMES [10]) by integrating PPJoin+ and extending this algorithm so as to enable it to deal with complex configurations. In addition, LIMES0.5 integrates the fully novel HYPPO algorithm, which ensures that our framework can deal efficiently with numeric values and consequently with the whole diversity of data types found on the Web of Data.

3 Problem Definition

The goal of LD is to discover the set of pair of instances $(s, t) \in S \times T$ that are related by a relation R, where S and T are two not necessarily distinct sets of instances. One way to automate this discovery is to compare the $s \in S$ and $t \in T$ based on their properties using a (in general complex) similarity metric. Two entities are then considered to be linked via R if their similarity is superior to a threshold τ. We are aware that several categories of approaches can be envisaged for discovering links between instances, for example using formal inferences or semantic similarity functions. Throughout this paper, we will consider LD via properties. This is the most common definition of instance-based LD [10, 16], which translates into the following formal specification.

Definition 1 (Link Discovery). *Given two sets S (source) and T (target) of instances, a (complex) similarity measure σ over the properties of $s \in S$ and $t \in T$ and a similarity threshold $\tau \in [0, 1]$, the goal of LD is to compute the set of pairs of instances $(s, t) \in S \times T$ such that $\sigma(s, t) \geq \tau$.*

This problem can be expressed equivalently as follows:

Definition 2 (Link Discovery on Distances). *Given two sets S and T of instances, a (complex) distance measure δ over the properties of $s \in S$ and $t \in T$and a distance threshold $\theta \in [0, \infty[$, the goal of LD is to compute the set of pairs of instances $(s, t) \in S \times T$ such that $\delta(s, t) \leq \theta$.*

Note that a normed similarity function σ can always be derived from a distance function δ by setting $\sigma(x, y) = (1 + \delta(x, y))^{-1}$. Furthermore, the distance threshold θ can be transformed into a similarity threshold τ by setting $\tau = (1 + \theta)^{-1}$. Consequently, distance and similarities are used interchangeably within our framework.

Although it is sometimes sufficient to define atomic similarity functions (i.e., similarity functions that operate on exactly one property pair) for LD, many LD problems demand the specification of complex similarity functions over several datatypes (numeric, strings, ...) to return accurate links. For example, while the name of bands can be used for detecting duplicate bands across different knowledge bases, linking cities from different knowledge bases requires taking more properties into consideration (e.g., the different names of the cities as well as their latitude and longitude) to compute links accurately. Consequently, linking on the Data Web demands frameworks that support complex link specifications.

4 Link Specifications as Operations on Sets

In state-of-the-art LD frameworks, the condition for establishing links is usually expressed by using combinations of operations such as MAX (maximum), MIN (minimum) and linear combinations on binary similarity measures that compare property values of two instances $(s, t) \in S \times T$. Note that transformation operations may be applied to the property values (for example a lower-case transformation for strings) but do not affect our formal model. We present a formal grammar that encompasses complex link specifications as found in current LD frameworks and show how complex configurations resulting from this grammar can be translated into a sequence of set and filter operations on simple configurations. We use $\rightsquigarrow$ to denote generation rules for metrics and specifications, $\equiv$ to denote the equivalence of two specifications and $A \sqsubseteq B$ to denote that the set of links that results from specification A is a subset of the set of links that results from specification B.

Our definition of a link specification relies on the definition of *atomic similarity measures* and *similarity measures.* Generally, a similarity measure m is a function such that $m : S \times T \to [0,1]$. We call a measure atomic (dubbed $atomicMeasure$) when it relies on exactly one similarity measure σ (e.g., trigrams similarity for strings) to compute the similarity of two instances s and t. A similarity measure m is either an atomic similarity measure $atomicMeasure$ or the combination of two similarity measures via operators OP such as MAX, MIN or linear combinations as implemented in LIMES. Thus, the following rule set for constructing metrics holds:

1. $m \rightsquigarrow atomicMeasure$
2. $m \rightsquigarrow OP(m_1, m_2)$

Note that frameworks differ in the type of operators they implement.

We call a link specification atomic (*atomicSpec*) if it compares the value of a measure m with a threshold τ, thus returning the pairs (s,t) that satisfy the condition $\sigma(s,t) \geq \tau$. A link specification $spec(m,\tau)$ is either an atomic link specification or the combination of two link specifications via operations such as AND (the conditions of both specifications must be satisfied, equivalent to set intersection), OR (set union), XOR (symmetric set difference), or $DIFF$ (set difference). Thus, the following grammar for specifications holds :

1. $spec(m,\theta) \rightsquigarrow atomicSpec(m,\theta)$
2. $spec(m,\theta) \rightsquigarrow AND(spec(m_1,\theta_1), spec(m_2,\theta_2))$
3. $spec(m,\theta) \rightsquigarrow OR(spec(m_1,\theta_1), spec(m_2,\theta_2))$
4. $spec(m,\theta) \rightsquigarrow XOR(spec(m_1,\theta_1), spec(m_2,\theta_2))$
5. $spec(m,\theta) \rightsquigarrow DIFF(spec(m_1,\theta_1), spec(m_2,\theta_2))$

Most very time-efficient algorithms such as PPJoin+ operate solely on atomic measures and would not be usable if specifications could not be reduced to run only on atomic measures. For the operators MIN, MAX and linear combinations, we can reduce configurations that rely on complex measures to operations on configurations that rely on atomic measures via the following rules:

1. $spec(MAX(m_1,m_2),\theta) \equiv OR(spec(m_1,\theta), spec(m_2,\theta))$
2. $spec(MIN(m_1,m_2),\theta) \equiv AND(spec(m_1,\theta), spec(m_2,\theta))$
3. $spec(\alpha m_1 + \beta m_2, \theta) \sqsubseteq AND(spec(m_1,(\theta-\beta)/\alpha), spec(m_2,(\theta-\alpha)/\beta))$

Note that while we can derive equivalent conditions on a smaller number of dimensions for the first two operations, the simpler linking specifications that can be extracted for linear combinations are necessary to fulfill their premise, but not equivalent to the premise. Thus, in the case of linear combinations, it is important to validate the final set of candidates coming from the intersection of the two sets specified on a smaller number of dimensions against the premise by using *filters*. Given these transformations, we can reduce all complex specifications that abide by our grammar to a sequence of set and filter operations on the results of atomic measures. Consequently, we can apply very time-efficient approaches designed for atomic measures on each category of data types to process even highly complex link specifications on the Web of Data. In the following, we present the approaches used by our framework on strings and numerical values.

5 Processing Simple Configurations

Our framework implements a hybrid approach to LD. The first approach implemented in our framework deals exclusively with strings by harnessing the near-duplicate detection algorithm PPJoin+ [18]. Instead of mapping strings to a vector space, PPJoin+ uses a combination of three main insights to implement a very time-efficient string comparison approach. First, it uses the idea that strings with a given similarity must share a certain number of characters in their prefix to be able to have a similarity beyond the user-specified threshold. A similar intuition governs the suffix filtering implemented by PPJoin+. Finally, the algorithm makes use of the position of each word w in the index to retrieve a lower and upper bound of the index of the terms with which w might be similar. By combining these three approaches, PPJoin+ can discard a large number of non-matches. The integration of the PPJoin+ algorithm into our framework ensures that we can mitigate the pitfall of the time-demanding transformation of strings to vector spaces as implemented by multidimensional approaches. The main drawback of PPJoin+ is that it can only operate on one dimension [8]. However, by applying the transformations of configurations specified above, we make PPJoin+ applicable to link discovery tasks with complex configurations. While mapping strings to a vector space demands some transformation steps and can be thus computationally demanding, all numeric values explicitly describe a vector space. The second approach implemented in our framework deals exclusively with numeric values and implements a novel approach dubbed *HYPPO*.

The HYPPO algorithm addresses the problem of efficiently mapping instance pairs $(s, t) \in S \times T$ described by using exclusively numeric values in a n-dimensional metric space. The approach assumes a distance metric δ for measuring the distance between objects and returns all pairs such that $\delta(s, t) \leq \theta$, where θ is a distance threshold. Let $\omega = (\omega_1, ..., \omega_n)$ and $x = (x_1, ..., x_n)$ be points in the n-dimensional space $\Omega = S \cup T$. The observation behind HYPPO is that in spaces (Ω, δ) with orthogonal, i.e., uncorrelated dimensions, distance metrics can be decomposed into the combination of functions $\phi_{i,i\in\{1...n\}}$ which operate on exactly one dimension of $\Omega : \delta = f(\phi_1, ..., \phi_n)$. For example, for Minkowsky distances of order $p > 1$, $\phi_i(x, \omega) = |x_i - \omega_i|$ for all values of i and $\delta(x, \omega) = \sqrt[p]{\sum \phi_i(x, \omega)^p}$. Note that the Euclidean distance is the Minkowsky distance of order 2. The Minkowsky distance can be extended further by weighting the different axes of Ω. In this case, $\delta(x, \omega) = \sqrt[p]{\sum \gamma_{ii}^p \phi_i(x, \omega)^p}$ and $\phi_i(x, \omega) = \gamma_{ii}|x_i - \omega_i|$, where γ_{ii} are the entries of a positive diagonal matrix.

Some distances do exist, which do not assume an orthogonal basis for the metric space. Mahalanobis distances for example are characterized by the equation $\delta(x, \omega) = \sqrt{(x - \omega)\Gamma(x - \omega)^T}$, where Γ is a $n \times n$ covariance matrix. However, given that each space with correlated dimensions can always be transformed into an affine space with an orthonormal basis, we will assume in the remainder of this paper that the dimensions of Ω are independent. Given this assumption, it is important to notice that the following inequality holds:

$$\phi_i(x, \omega) \leq \delta(x, \omega), \tag{1}$$

ergo, $\delta(x, \omega)$ is the upper bound of $\phi_i(x, \omega)$. Note that this is the sole condition that we pose upon δ for HYPPO to be applicable. Also note that this condition can always be brought about in a metric space by transforming its basis into an orthogonal basis.

The basic intuition behind HYPPO is that the hypersphere $H(\omega, \theta) = \{x \in \Omega : \delta(x, \omega) \leq \theta\}$ is a subset of the hypercube V defined as $V(\omega, \theta) = \{x \in \Omega : \forall i \in \{1...n\}, \phi_i(x_i, \omega_i) \leq \theta\}$ due to inequality 1. Consequently, one can reduce the number of comparisons necessary to detect all elements of $H(\omega, \theta)$ by discarding all elements which are not in $V(\omega, \theta)$ as non-matches. HYPPO uses this intuition by implementing a two-step approach to LD. First, it tiles Ω into hypercubes of the same volume. Second, it compares each $s \in S$ with those $t \in T$ that lie in cubes at a distance below θ. Note that these two steps differ from the steps followed by similar algorithms (such as blocking) in two ways. First, we do not use only one but several hypercubes to approximate $H(\omega, \theta)$. Most blocking approach rely on finding *one block* that contains the elements that are to be compared with ω [8]. In addition, HYPPO is guaranteed not to lose any link, as H is completely enclosed in V, while most blocking techniques are not lossless.

Formally, let $\Delta = \theta/\alpha$. We call $\alpha \in \mathbb{N}$ the granularity parameter. HYPPO first tiles Ω into the adjacent hypercubes (short: cubes) C that contain all the points ω such that $\forall i \in \{1...n\}, c_i\Delta \leq \omega_i < (c_i + 1)\Delta$, $(c_1, ..., c_n) \in \mathbb{N}^n$. We call the vector $(c_1, ..., c_n)$ the coordinates of the cube C. Each point $\omega \in \Omega$ lies in the cube $C(\omega)$ with coordinates $(\lfloor \omega_i/\Delta \rfloor)_{i=1...n}$. Given such a space tiling and inequality (1), it is obvious that all elements of $H(\omega, \theta)$ lie in the set $\mathfrak{C}(\omega, \alpha)$ of cubes such that $\forall i \in \{1...n\} : |c_i - c(\omega)_i| \leq \alpha$. Figure 1 shows examples of space tilings for different values of α.

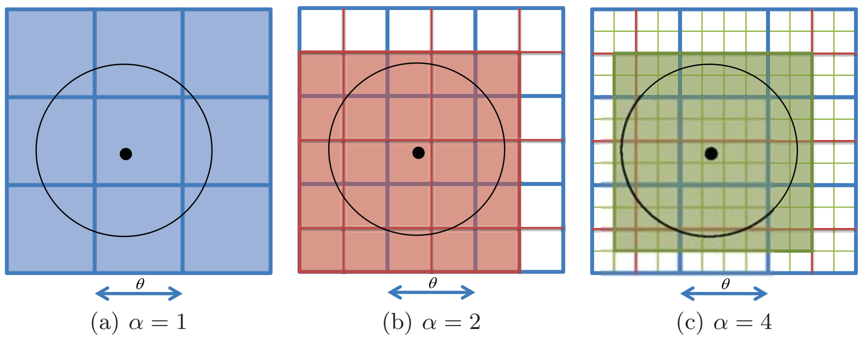


Fig. 1. Space tiling for different values of α. The colored squares show the set of elements that must be compared with the instance located at the black dot. The points within the circle lie within the distance θ of the black dot.

The accuracy of the approximation performed by HYPPO can be computed easily: The number of cubes that approximate $H(\omega, \theta)$ is $(2\alpha + 1)^n$, leading to a total volume $V_{\mathfrak{C}}(\alpha,\theta) = ((2\alpha+1)\Delta)^n = \left(\frac{2\alpha+1}{\alpha}\theta\right)^n$ that approximates $H(\omega,\theta)$. The volume $V_H(\theta)$ of $H(\omega,\theta)$ is given by $\frac{S_n\theta^n}{n}$, where S_n is the volume of a unit sphere in n dimensions, i.e., 2 for $n = 1$, π for $n = 2$, $\frac{4\pi}{3}$ for $n = 3$ and so on. The approximation ratio

$$\frac{V_{\mathfrak{C}}(\alpha,\theta)}{V_H(\theta)} = \frac{n}{S_n}\left(\frac{2\alpha+1}{\alpha}\right)^n, \tag{2}$$

permits to determine the accuracy of HYPPO's approximation as shown in Figure 2 for dimensions between 1 and 3 and values of α up to 10. Note that $V_{\mathfrak{C}}$ and V_H do not depend on ω and that $\frac{V_{\mathfrak{C}}(\alpha,\theta)}{V_H(\theta)}$ does not depend on θ. Furthermore, note that the higher the value of α, the better the accuracy of HYPPO. Yet, higher values of α also lead to an exponentially growing number of hypercubes $|\mathfrak{C}(\omega,\alpha)|$ and thus to longer runtimes when constructing $\mathfrak{C}(\omega,\alpha)$ to approximate $H(\omega,\theta)$. Once the space tiling has been completed, all that remains to do is to compare each $s \in S$ with all the $t \in T \cap (\bigcup C \in \mathfrak{C}(\omega,\alpha))$ and to return those pairs of entities such that $\delta(s,t) \leq \theta$. Algorithm 1 shows HYPPO's pseudocode.

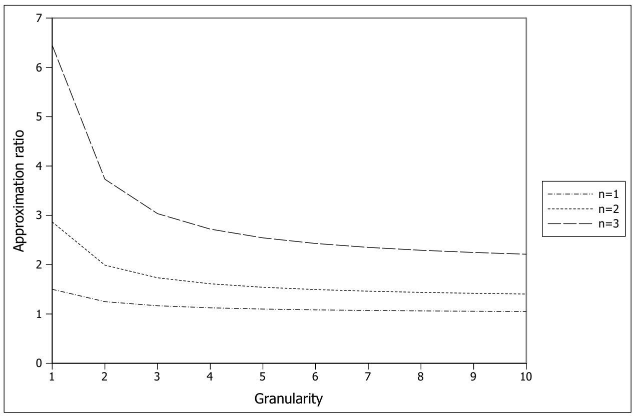


Fig. 2. Approximation ratio for $n \in \{1,2,3\}$. The x-axis shows values of α while the y-axis shows the approximation ratios.

6 Evaluation

We compared our framework (i.e., LIMES Version 0.5) with SILK version 2.3. in three large-scale experiments of different complexity based on geographic data. We chose SILK because (to the best of our knowledge) it is the only other LD framework that allows the specification of such complex linking experiments. We ran all experiments on the same computer running a Windows 7 Enterprise 64-bit installation on a 2.8GHz i7 processor with 8GB RAM. The JVM was

Algorithm 1 Current implementation of HYPPO

```
Require: $S$, $T$, $\theta$, $\delta$, $\alpha$ as defined above
  Mapping $M := \emptyset$
  $\Delta = \theta/\alpha$
  for $\omega \in S \cup T$ do
    $C(\lfloor\omega_1/\Delta\rfloor, ..., \lfloor\omega_n/\Delta\rfloor) := C(\lfloor\omega_1/\Delta\rfloor, ..., \lfloor\omega_n/\Delta\rfloor) \cup \{\omega\}$
  end for
  for $s \in S$ do
    for $C \in \mathfrak{C}(s, \alpha)$ do
      for $t \in C \cap T$ do
        if $\delta(s, t) \leq \theta$ then
          $M := M \cup (s, t)$
        end if
      end for
    end for
  end for
  return $M$
```

allocated 7.4GB RAM. For each tool we measured exclusively the time needed for computing the links. All experiments were carried out 5 times except when stated otherwise. In all cases, we report best runtimes. Experiments marked with an asterisk would have lasted longer than 48 hours when using SILK and were not computed completely. Instead, SILK's runtime was approximated by extrapolating the time needed by the software to compute 0.1% of the links.

We chose to use geographic datasets because they are large and allow the use of several attributes for linking. In the first experiment, we computed links between *villages* in DBpedia and LinkedGeoData based on the `rdfs:label` and the `population` of instances. The link condition was twofold: (1) the difference in population had to be lower or equal to θ and (2) the labels had to have a trigram similarity larger or equal to τ. In the second experiment, we aimed to link towns and *cities* from DBpedia with populated places in Geonames. We used the names (`gn:name`), alternate names (`gn:alternateName`) and `population` of cities as criteria for the comparison. Finally, we computed links between *Geo-locations* in LinkedGeoData and GeoNames by using 4 combinations of criteria for comparing entities: their longitude (`wgs84:long`), latitude (`wgs84:lat`), preferred names and names.

Experiment	$\|S\|$	$\|T\|$	Dims	Complexity	Source	Target	Thresholds
Villages*	26717	103175	2	3.8 $\times 10^9$	DBpedia	LGD	τ_s, θ_p
Cities*	36877	39800	3	1.5 $\times 10^9$	Geonames	DBpedia	τ_s, θ_p
Geo-Locations*	50031	74458	4	3.7 $\times 10^9$	LGD	GeoNames	τ_s, θ_p, θ_l

Table 1. Summary of experimental setups for LIMES and SILK. Dims stands for dimensions.

The setup of the experiments is summarized in Table 1. We used two threshold setups. In the *strict setup*, the similarity threshold τ_s on strings was set to 0.9, the maximal difference in population θ_p was set to 9 and the maximal difference in latitude and longitude θ_l was set to 1. In the *lenient setup*, τ_s was set to 0.7 and θ_p to 19. The lenient setup was not used in the Geo-Locations experiments because it led to too many links, which filled up the 7.4G of RAM allocated to both tools and led to swapping, thus falsifying the evaluation of the runtimes. In all setups, we use the trigrams similarity metrics for strings and the euclidean distance for numeric values.

Our results (see Figure 3) confirm that we outperform SILK by several orders of magnitude in all setups. In the Cities experiment, we are more than 6 orders of magnitude faster than SILK. We compared the runtimes of LIMES for different values of α as shown in Figure 4. Our results show that our assumption on the relation between α and runtimes is accurate as finding the right value for α can reduce the total runtime of the algorithm by approximately 40% (see Geo-Locations, $\alpha = 4$). In general, setting α to values between 2 and 4 leads to an improved performance in all experiments.

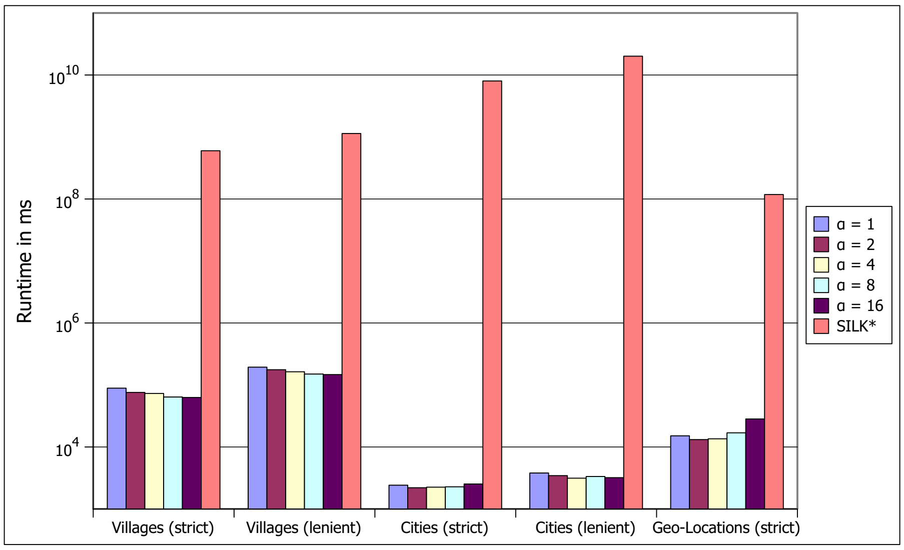


Fig. 3. Comparison of the runtime of LIMES and SILK on large-scale link discovery tasks.

7 Discussion and Future Work

In this paper, we introduced and evaluated a novel hybrid approach to LD. We presented original insights on the conversion of complex link specifications

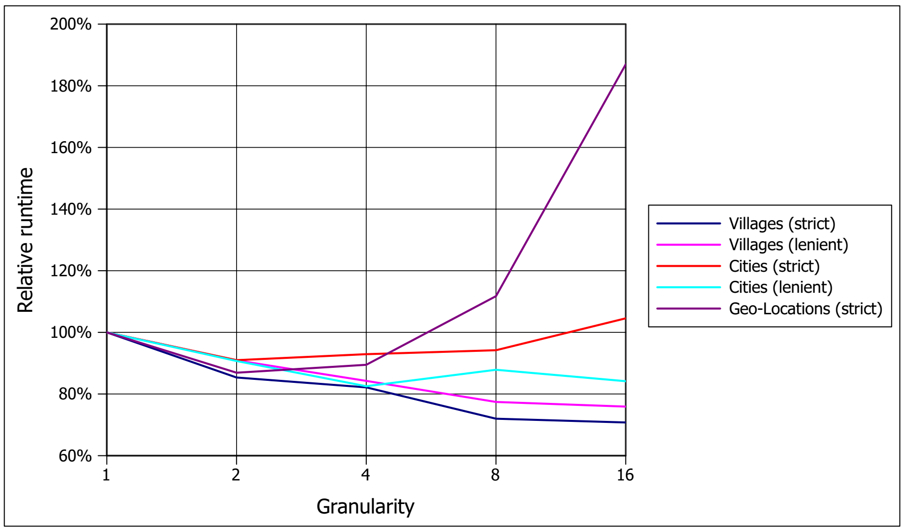


Fig. 4. Runtimes of LIMES relatively to the runtime for $\alpha = 1$.

into simple link specifications. Based on these conversions, we inferred that efficient means for processing simple link specifications are the key for time-efficient linking. We then presented the two time-efficient approaches implemented in LIMES0.5 and showed how these approaches can be combined for time-efficient linking. A thorough evaluation of our framework in three large-scale experiments showed that we outperform SILK by more than 6 orders of magnitude while not losing a single link.

One of the central innovations of this paper is the HYpersphere aPPrOximation algorithm HYPPO. Although it was defined for numeric values, HYPPO can be easily generalized to the efficient computation of the similarity of pairs of entities that are totally ordered, i.e., to all sets of entities $e = (e_1, ..., e_n) \in E$ such that a real function f_i exists, which preserves the order $\succ$ on the ith dimension of E, ergo $\forall e, e' \in E : e_i \succ e'_i \to f(e_i) > f(e'_i)$. Yet, it is important to notice that such a function can be very complex and thus lead to overheads that may nullify the time gain of HYPPO. In future work, we will aim to find such functions for different data types. In addition, we will aim to formulate an approach for determining the best value of α for any given link specification. The new version of LIMES promises to be a stepping stone for the creation of a multitude of novel semantic applications, as it is time-efficient enough to make complex interactive scenarios for link discovery possible even at large scale.

Acknowledgement

This work was supported by the Eurostars grant SCMS E!4604 and a fellowship grant of the Mainz University.

References

1. Roberto J. Bayardo, Yiming Ma, and Ramakrishnan Srikant. Scaling up all pairs similarity search. In *WWW*, pages 131–140, 2007.
2. David Ben-David, Tamar Domany, and Abigail Tarem. Enterprise data classification using semantic web technologies. In *ISWC*, 2010.
3. Jens Bleiholder and Felix Naumann. Data fusion. *ACM Comput. Surv.*, 41(1):1–41, 2008.
4. Ahmed K. Elmagarmid, Panagiotis G. Ipeirotis, and Vassilios S. Verykios. Duplicate record detection: A survey. *IEEE Transactions on Knowledge and Data Engineering*, 19:1–16, 2007.
5. Hugh Glaser, Ian C. Millard, Won-Kyung Sung, Seungwoo Lee, Pyung Kim, and Beom-Jong You. Research on linked data and co-reference resolution. Technical report, University of Southampton, 2009.
6. Tom Heath and Christian Bizer. *Linked Data: Evolving the Web into a Global Data Space*. Morgan & Claypool, 2011.
7. R. Isele, A. Jentzsch, and C. Bizer. Efficient Multidimensional Blocking for Link Discovery without losing Recall. June 2011.
8. Hanna Köpcke, Andreas Thor, and Erhard Rahm. Comparative evaluation of entity resolution approaches with fever. *Proc. VLDB Endow.*, 2(2):1574–1577, 2009.
9. Vanessa Lopez, Victoria Uren, Marta Reka Sabou, and Enrico Motta. Cross ontology query answering on the semantic web: an initial evaluation. In *K-CAP '09: Proceedings of the fifth international conference on Knowledge capture*, pages 17–24, New York, NY, USA, 2009. ACM.
10. Axel-Cyrille Ngonga Ngomo and Sören Auer. A time-efficient approach for large-scale link discovery on the web of data. In *IJCAI*, 2011.
11. George Papadakis, Ekaterini Ioannou, Claudia Niedere, Themis Palpanasz, and Wolfgang Nejdl. Eliminating the redundancy in blocking-based entity resolution methods. In *JCDL*, 2011.
12. Yves Raimond, Christopher Sutton, and Mark Sandler. Automatic interlinking of music datasets on the semantic web. In *Proceedings of the 1st Workshop about Linked Data on the Web*, 2008.
13. Franois Scharffe, Yanbin Liu, and Chuguang Zhou. Rdf-ai: an architecture for rdf datasets matching, fusion and interlink. In *Proc. IJCAI 2009 workshop on Identity, reference, and knowledge representation (IR-KR), Pasadena (CA US)*, 2009.
14. Jennifer Sleeman and Tim Finin. Computing foaf co-reference relations with rules and machine learning. In *Proceedings of the Third International Workshop on Social Data on the Web*, 2010.
15. Jacopo Urbani, Spyros Kotoulas, Jason Maassen, Frank van Harmelen, and Henri Bal. Owl reasoning with webpie: calculating the closure of 100 billion triples. In *Proceedings of the ESWC 2010*, 2010.
16. Julius Volz, Christian Bizer, Martin Gaedke, and Georgi Kobilarov. Discovering and maintaining links on the web of data. In *ISWC*, pages 650–665, 2009.
17. William Winkler. Overview of record linkage and current research directions. Technical report, Bureau of the Census - Research Report Series, 2006.
18. Chuan Xiao, Wei Wang, Xuemin Lin, and Jeffrey X. Yu. Efficient similarity joins for near duplicate detection. In *WWW*, pages 131–140, 2008.

Learning Linkage Rules using Genetic Programming

Robert Isele and Christian Bizer

Freie Universität Berlin, Web-based Systems Group
Garystr. 21, 14195 Berlin, Germany
mail@robertisele.com, chris@bizer.de

Abstract. An important problem in Linked Data is the discovery of links between entities which identify the same real world object. These links are often generated based on manually written linkage rules which specify the condition which must be fulfilled for two entities in order to be interlinked. In this paper, we present an approach to automatically generate linkage rules from a set of reference links. Our approach is based on genetic programming and has been implemented in the Silk Link Discovery Framework. It is capable of generating complex linkage rules which compare multiple properties of the entities and employ data transformations in order to normalize their values. Experimental results show that it outperforms a genetic programming approach for record deduplication recently presented by Carvalho et. al. In tests with linkage rules that have been created for our research projects our approach learned rules which achieve a similar accuracy than the original human-created linkage rule.

Keywords: Genetic Programming, Linked Data, Link Discovery, Duplicate Detection, Deduplication, Record Linkage

1 Introduction

In the decentralized Web of Data, many data sources use different URIs for the same real world object. Identifying these URI aliases, is a central problem in Linked Data. Two approaches are widely used for that purpose: The first category includes fully automatic tools which identify links using unsupervised learning [11]. The second category includes tools which improve the accuracy of the generated links using user-provided *linkage rules*. A linkage rule [30], specifies the conditions two entities must fulfill in order to be interlinked. For this purpose, a linkage rule typically uses one or more distance measures to compare the properties of the entities. If the data sources use different data types, the property values may be normalized by applying transformations prior to the comparison. Linkage rules aggregate multiple similarity measures into one compound similarity value. As these conditions are strongly domain dependent, a separate linkage rule is typically used for each type of entities.

In this paper, we present an approach to automatically learn linkage rules from a set of reference links. The approach is based on genetic programming and generates linkage rules that can be understood and further improved by humans. Our approach has been implemented and evaluated in Silk [16], a link discovery framework which generates RDF links between data items based on linkage rules which are expressed using the Silk Link Specification Language (Silk-LSL). The current version of Silk which includes the presented learning method can be downloaded from the project homepage[1] under the terms of the Apache Software License.

The experimental evaluation shows that it produces better results than a recently developed genetic programming approach by Carvalho et. al. [8]. In tests with linkage rules that have been created for our research projects our approach learned rules which achieve a similar accuracy than the original human-created linkage rule.

This paper is organized as follows: The following Section gives an overview of related work. Section 3 describes the proposed approach in detail. Afterwards, Section 4 presents the results of the experimental evaluation of the learning algorithm. Finally, Section 5 concludes this paper.

2 Related Work

Supervised learning of linkage rules in the context of linked data can build on previous results in record linkage. In literature many approaches suitable for learning binary classifiers have been adapted for learning linkage rules [18]. This section gives an overview of the most widely used approaches.

Naive Bayes. Based on the original Fellegi-Sunter statistical model [13] of record linkage, methods from Bayesian statistics such as *Naive Bayes* classifiers [31] have been used to learn linkage rules. The main disadvantage of Naive Bayes classifiers from a practical point of view is that they represent a black box system to the user. This means that the user can not easily understand and improve the learned linkage rules.

Support Vector Machines. Another widely used approach is to learn parameters of the linkage rule using *Support Vector Machines* (SVM) [5]. A SVM is a binary linear classifier which maps the input variables into a high-dimensional space where the two classes are separated by a hyperplane via a kernel function [2]. In the context of learning linkage rules, SVMs are often employed to learn specific parameters of a linkage rule such as the weights of different similarity measures. One popular example is MARLIN (Multiply Adaptive Record Linkage with INduction) [1], which uses SVMs to learn how to combine multiple similarity measures.

Decision Trees. Linkage rules can also be modeled using *Decision Trees* which can be learned by a variety of algorithms including genetic algorithms. The main advantage of Decision Trees is that they provide explanations for each classification and thus can be understood and improved manually. Active Atlas [28, 29]

[1] `http://www4.wiwiss.fu-berlin.de/bizer/silk/`

learns mappings rules consisting of a combination of predefined transformations and similarity measures. TAILOR [10] is another tool which employs decision trees to learn linkage rules.

Genetic Programming. Another approach which is more expressive than decision trees and promising to learn complex linkage rules is *genetic programming* (GA). Genetic programming is an extension of the genetic algorithm [15] which has been first proposed by Cramer [6]. Similar to a genetic algorithm, it starts with a randomly created population of individuals. Each individual is represented by a tree which is a potential solution to the given problem. From that starting point the algorithm iteratively transforms the population into a population with better individuals by applying a number of genetic operators. These operations are applied to individuals which have been selected based on a fitness measure which determines how close a specific individual is to the desired solution. The three genetic operators typically used in genetic programming are [19]:

Reproduction: An individual is copied without modification.
Crossover: Two selected individuals are recombined into a new individual.
Mutation: A random modification is applied to the selected individual.

The algorithm stops as soon as either the configured maximum number of iterations or a user-defined stop condition is reached.

Genetic programming has been applied to many problems in a variety of domains [26]. In many of these areas genetic programming is capable of producing human-competitive results [23, 21, 22]. Examples include the synthesis of electrical circuits [20], the creation of quantum algorithms [27], and the development of controllers [22].

To the best of our knowledge, genetic programming for learning linkage rules has only been applied by Carvalho et. al. so far [7, 4, 8]. Their approach uses genetic programming to learn how to combine a set of presupplied pairs of the form `<attribute, similarity function>` (e.g. `<name, Jaro>`) into a linkage rule. These pairs can be combined by the genetic programming method arbitrarily by using mathematical functions (e.g. +, -, *, /, exp) and constants. Carvalho et. al. show that their method produces better results than the state-of-the-art SVM based approach by MARLIN [8]. Their approach is very expressive although it cannot express data transformations. On the downside, using mathematical functions to combine the similarity measures does not fit any commonly used linkage rule model [12] and leads to complex and difficult to understand linkage rules.

We are not aware of any previous application of genetic programming to learn linkage rules in the context of Linked Data.

3 Approach

This Section explains our approach of learning linkage rules using genetic programming. It is organized as follows: First of all, in order to learn a linkage rule using genetic programming, a rule must be represented as a tree structure. Thus, Section 3.1 describes our approach of representing a linkage rule using 4 basic operators. For each candidate solution the fitness function described in Section 3.2 is used to determine the performance of a linkage rule. Section 3.3 describes how the initial population of candidate solutions is generated. After the initial population has been generated, the candidate solutions are iteratively transformed into better ones by breeding the population according to the rules described in Section 3.4. Finally, Section 3.5 describes our approach to avoid the occurrence of bloat in linkage rules.

3.1 Representation of a Linkage Rule

We represent a linkage rule as a tree which is built from 4 basic operators:

Property: Creates a set of values to be used for comparison by retrieving all values of a specific property of the entity.

Transformation: Transforms the input values according to a specific data transformation function.

Comparison: Evaluates the similarity between the values of two input operators according to a specific distance measure. A user-specified threshold specifies the maximum distance. If the underlying properties do not provide any values for a specific entity, no similarity value is returned.

Aggregation: Aggregates the similarity values from multiple operators into a single value according to a specific aggregation function. Aggregation functions such as the weighted average may take the weight of the operators into account. If an operator is marked as required, the aggregation will only yield a value if the operator itself provides a similarity value.

The resulting linkage rule forms a tree where the terminals are given by the properties and the nodes are represented by transformations, comparisons and aggregations. The linkage rule tree is strongly typed [25] i.e. it does not allow arbitrary combinations of its four basic operators. Figure 1 specifies the valid structure of a linkage rule. Figure 2 shows a simple example of a linkage rule.

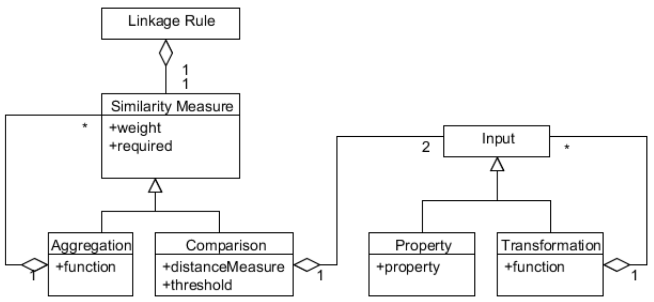


Fig. 1. Structure of a linkage rule

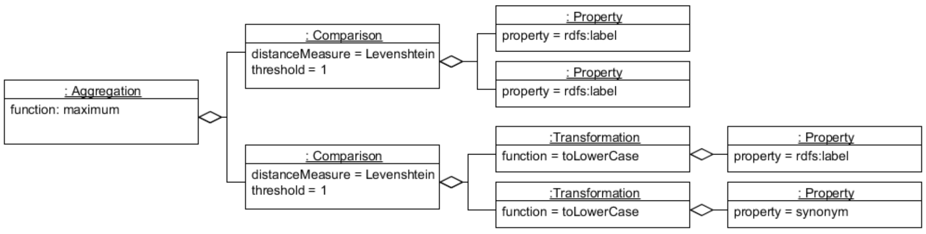


Fig. 2. Example linkage rule

3.2 Fitness Function

The quality of a linkage rule is assessed by the fitness function based on user-provided training data. The training data consists of a set of positive reference links (connecting entities which identify the same real world object) and a set of negative reference links (connecting entities which identify different objects). The prediction of the linkage rule is compared with the positive reference links while counting *true positives* (TP) and *false negatives* (FN) and the negative reference links while counting *false positives* (FP) and *true negatives* (TN). Based on these counts, a fitness value between -1 and 1 is assigned to the linkage rule by calculating *Matthews correlation coefficient* (MCC):

$$MCC = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}}$$

In contrast to many other popular fitness measures such as the F-measure (i.e. the harmonic mean of precision and recall), Matthews correlation coefficient yields good results even for heavily unbalanced training data.

3.3 Generating the Initial Population

This section explains our approach of generating the initial population: Before the population is generated, we build a list of property pairs which hold similar values as described below. Based on that, random linkage rules are built by selecting property pairs from the list and applying data transformations, comparisons and aggregations. Finally, we seed the population with common comparison patterns in order to increase the efficiency of the algorithm.

Finding Compatible Properties Prior to generating the population, we generate a list of pairs of properties which hold similar values. For this purpose, the datasets are preprocessed in order to find the 100 most frequent properties in the data set where the entity is in subject position and the 10 most frequent properties where the entity is in object position. The selection of the `owl:sameAs` property has been disallowed as it usually is the result of an existing run of a link discovery tool. For each possible property pair, the values of the entities referenced by the positive reference links as well as the negative reference links are analyzed. This is done by tokenizing the values and counting the reference links for which there is a distance measure in the list of functions configured to be used for learning linkage rules according to which both values are similar (given a certain threshold). Finally, the list of compatible properties is generated by collecting all pairs of properties for which more positive than negative links are counted.

Generating a Random Linkage Rule A random linkage rule is generated according to the following rules: First of all, a linkage rule is built consisting of a random aggregation and up to two comparisons. For each comparison a random pair from the pre-generated list of compatible properties is selected. In addition, with a possibility of 50% a random transformation is appended to each property.

Note that this does not limit the algorithm to learn more complex linkage rules as it is the purpose of the genetic operators to generate more complex linkage rules from the ones in the initial population.

Seeding with Common Comparison Patterns Analyzing a set of linkage rules manualy developed for the LATC EU project (`http://latc-project.eu/`) revealed that certain patterns occur in many linkage rules. Most noticeable, 84 % of the analyzed linkage rules compare the labels and 66 % the geographic coordinates of the entities. For that reason, the population has been seeded not only with fully random linkage rules, but also with linkage rules which contain these two special cases. While these patterns can also be learned by the algorithm, previous work [26, 14] shows that seeding them in the initial population can improve the efficiency.

3.4 Breeding

In order to improve the population our approach employs all three common genetic operations: reproduction, crossover and mutation. At first, 1% of the individuals with the highest fitness are directly selected for reproduction following a elitist strategy [9]. After this, new individuals are generated using crossover and mutation until the population size is reached.

Instead of using subtree crossover, which is commonly used in genetic programming, our approach uses a set of specific crossover operators which are tailored to the domain. For each crossover operation an operator from this set is selected randomly and applied to two selected individuals. Each operator learns one aspect of the linkage rule. For our experiments, we used the following operators:

Function Crossover Used to find the best similarity, transformation or aggregation function. Selects one operator at random in each linkage rule and interchanges the functions. For example, it may select a comparison with the levensthein distance function in the first linakge rule and a comparison with the jaccard distance function in the second linkage rule and than interchange these two functions.

Operators Crossover As a linkage rule usually needs to combine multiple comparisons, this operator combines aggregations from both linkage rules. For this, it selects two aggregations, one from each linkage rule and combines theirs comparisons. The comparisons are combined by selecting all comparisons from both aggregations and removing each comparison with a propability of 50%. For example, it may select an aggregation of a label comparison and a date comparison in the first linkage rule and an aggregation of a label comparison and a comparison of the geographic coordinates in the second linkage rule. In this case the operator replaces the selected aggregations with a new aggregation which contains all 4 comparisons and then removes each comparison with a propability of 50%. Note that the comparisons are exchanged including the complete subtree i.e. the distance functions as well as existing transformations are retained.

Aggregation Crossover While most linkage rules are linear i.e. can be expressed using a single weighted average aggregation, some linkage rules need more complex aggregation hierarchies. In order to learn these hierachies, aggregation crossover selects a random aggregation or comparison operator in the first linkage rule and replaces it with a random aggregation or comparison operator from the second linkage rule. This way, the operator builds a hierachy as it may select operators from different levels in the tree. For example, it may select a comparison in the first linkage rule and replace it with a aggregation of multiple comparisons from the second linakge rule.

Transformation Crossover This operator is used to learn complex transformations by selecting a random path of transformations in both linkage rules. It then combines both paths by executing a two point crossover.

Threshold Crossover This operator is used to find the optimal thresholds. For this, one comparison operator is selected at random in each individual. The new threshold is then set to the average of both comparisons.

Weight Crossover Finds the optimal weights analog to the treshold crossover.

Mutation is implemented similarly by selecting a random crossover operator and executing a headless chicken crossover [17] i.e. crossing an individual from the population with a randomly generated individual.

3.5 Avoiding Bloat

One well-known problem in genetic programming is that over time the individuals may develop redundant parts which do not contribute to their overall fitness [3, 24]. One possibility to control this bloating is to penalize big trees in order to force the algorithm to favor smaller trees over bigger ones. In literature this method is known as parsimony pressure [32]. Another more sophisticated method is to automatically analyze the trees and remove redundant parts. For that purpose a simplification algorithm has been developed which detects redundant parts in the linkage rule and removes them. In order to avoid bloated linkage rules the simplification algorithm is executed every 5 generations.

4 Evaluation

The proposed learning approach has been evaluated in 3 different experiments. Because genetic algorithms are non-deterministic and may yield different results in each run, all experiments have been run 10 times. For each run the reference links have been randomly split into 2 folds for cross-validation. The results of all runs have been averaged and the standard deviation has been computed. For each experiment, we provide the evaluation results with respect to the training data set as well as the validation dataset. All experiments have been run on a 3GHz Intel(R) Core i7 CPU with 4 cores while the Java heap space has been restricted to 1GB.

4.1 Parameters

Table 1 lists the parameters which have been used in all experiments. As it is the purpose of the developed method to work on arbitrary datasets without the need to tailor its parameters to the specific datasets that should be interlinked, the same parameters have been used for all experiments.

Table 1. Learning Parameters

Parameter	Value
Population size	500
Maximum number of generations	50
Selection method	Tournament selection with a group size of 5
Probability of Crossover	75%
Probability of Mutation	25%
Stop Condition	MCC = 1.0

Our approach is independent of any specific aggregation functions, distance measures or data transformations. Thus, it can learn linkage rules with any functions provided to it. Table 2 shows the set of functions which has been used by us in all experiments. The details about the functions are provided in the Silk user manual on the website.

Table 2. Set of functions used in all experiments

Aggregation Functions	Distance Measures	Transformations
Average similarity	Levenshtein distance	Convert to lower case
Maximum similarity	Jaccard index	Tokenize the string
Minimum similarity	Numeric distance	Strip the URI prefix
	Geographic distance	

4.2 Experiment 1: Comparison with related work

At first, we evaluated how our approach compares to the genetic programming approach by Carvalho et. al. [8], which claims to produce better results than the state-of-the-art SVM based approach by MARLIN. One dataset commonly used for evaluating different record deduplication approaches is *Cora*. The Cora dataset contains citations to research papers from the Cora Computer Science research paper search engine. For the purpose of evaluating our approach, we converted the Cora dataset provided at [2] to RDF.

For evaluation we used 1617 randomly selected positive links and 1617 randomly selected negative reference links. Table 4.2 summarizes the cross validation results. On average, our approach achieved an F-measure of 96.9% against the training set and 93.6% against the validation set and needed less than 5 minutes to perform all 50 iterations on the test machine. The learned linkage rules compared by title, author and date. For the same dataset, Carvalho et. al. report an F-measure of 90.0% against the training set and 91.0% against the validation set [8].

Table 3. Average results of all learning runs. The last row contains the best results of Carvalho et. al. for comparison.

Iter.	Time in s (σ)	Train. F1 (σ)	Train. MCC (σ)	Val. F1 (σ)	Val. MCC (σ)
1	4.0 (0.3)	0.896 (0.022)	0.806 (0.042)	0.896 (0.021)	0.805 (0.041)
10	31.1 (3.9)	0.956 (0.013)	0.912 (0.026)	0.954 (0.015)	0.907 (0.029)
20	71.4 (18.3)	0.964 (0.008)	0.928 (0.017)	0.960 (0.010)	0.919 (0.020)
30	132.5 (48.5)	0.965 (0.007)	0.931 (0.013)	0.962 (0.007)	0.924 (0.015)
40	217.6 (106.7)	0.968 (0.004)	0.936 (0.008)	0.945 (0.036)	0.900 (0.053)
50	271.1 (140.1)	0.969 (0.003)	0.938 (0.007)	0.936 (0.056)	0.902 (0.057)
Ref.	-	0.900 (0.010)	-	0.910 (0.010)	-

4.3 Experiment 2: Learning linkage rules for geographic datasets

With 16 datasets in the LOD cloud[3], interlinking geographic datasets is a very common problem. For this reason we evaluated our approach by learning a linkage rule for interlinking cities in DBpedia and LinkedGeoData.

In order to evaluate the learned linkage rules we used a manually collected set of 100 positive and 100 negative reference links. Special care has been taken to include rare corner cases such as for example cities which share the same name but don't represent the same city and cities which are very closely located so that. Table 4.3 summarizes the cross validation results. In all runs, the stop condition (i.e. an MCC of 100%) has been reached before the 25th iteration.

[2] `http://www.hpi.uni-potsdam.de/naumann/projekte/repeatability/datasets/cora_dataset.html`

[3] `http://www4.wiwiss.fu-berlin.de/lodcloud/state/`

Table 4. Average results of all learning runs.

Iter.	Time in s (σ)	Train. F1 (σ)	Train. MCC (σ)	Val. F1 (σ)	Val. MCC (σ)
1	2.6 (1.0)	0.984 (0.025)	0.970 (0.046)	0.932 (0.059)	0.883 (0.099)
10	3.8 (2.1)	0.996 (0.007)	0.993 (0.013)	0.932 (0.059)	0.883 (0.099)
20	3.9 (2.3)	0.998 (0.004)	0.996 (0.007)	0.964 (0.032)	0.945 (0.056)
25	4.0 (2.4)	1.000 (0.000)	1.000 (0.000)	1.000 (0.000)	1.000 (0.000)

4.4 Experiment 3: Learning complex linkage rules

While the vast majority of linkage rules commonly used are very simple, a few of them employ more complex structures. Interlinking drugs in DBpedia and Drugbank is an example where the original linkage rule which has been produced by humans is very complex. In order to match two drugs, it compares the drug names and their synonyms as well as a list of well-known and used identifiers (e.g. the CAS number[4]). In total, the manually written linkage rule uses 13 comparisons and 33 transformations. This includes complex transformations such as replacing specific parts of the strings.

All 1,403 Links which have been generated by executing the original linkage rule have been used as positive reference links. Negative reference links have been generated by shuffling the target URIs of the positive links.

Table 5 shows the averaged results of all runs. The learned linkage rules yield an F-Measure of 99.8% for the training data and 99.4% for the validation data. Figure 3 shows that from the 30th iteration the generated linkage rules on average only use 5.6 comparisons and 3.2 transformations and the simplification algorithm successfully avoids bloating in the subsequent iterations. Thus, the learned linkage rules use less than half of the comparisons and only one-tenth of the transformations of the manually written linkage rules.

Table 5. Average results of all learning runs

Iter.	Time in s (σ)	Train. F1 (σ)	Train. MCC (σ)	Val. F1 (σ)	Val. MCC (σ)
1	67.5 (2.2)	0.929 (0.026)	0.876 (0.042)	0.928 (0.029)	0.874 (0.045)
10	334.1 (157.4)	0.994 (0.002)	0.987 (0.003)	0.991 (0.003)	0.983 (0.006)
20	1014.1 (496.8)	0.996 (0.001)	0.992 (0.002)	0.988 (0.010)	0.977 (0.017)
30	1829.7 (919.3)	0.997 (0.001)	0.994 (0.002)	0.985 (0.016)	0.973 (0.027)
40	2685.4 (1318.9)	0.998 (0.001)	0.996 (0.002)	0.994 (0.002)	0.988 (0.004)
50	3222.2 (1577.7)	0.998 (0.001)	0.996 (0.001)	0.994 (0.002)	0.989 (0.004)

5 Conclusion and Outlook

We presented an approach for learning complex linkage rules which compare multiple properties of the entities and employ data transformations in order

[4] A unique numerical identifier assigned by the "Chemical Abstracts Service"

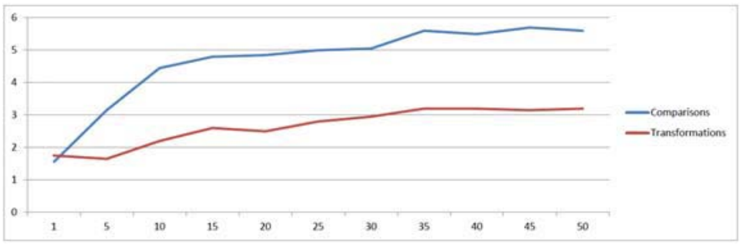


Fig. 3. Average number of comparisons and transformations

to normalize their values. As the current algorithm requires manually supplied reference links, future work will focus on the efficient generation of these. For this, we will investigate into semi-supervised learning and active learning in order to minimize the user effort to generate the reference links.

References

1. M. Bilenko and R. Mooney. Adaptive duplicate detection using learnable string similarity measures. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 39–48. ACM, 2003.
2. M. Bilenko and R. J. Mooney. Learning to combine trained distance metrics for duplicate detection in databases. Technical report, 2002.
3. T. Blickle and L. Thiele. Genetic Programming and Redundancy. 1994.
4. M. Carvalho, A. Laender, M. Gonçalves, and A. da Silva. Replica identification using genetic programming. In *Proceedings of the 2008 ACM symposium on Applied computing*, pages 1801–1806. ACM, 2008.
5. C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, 20:273–297, 1995. 10.1007/BF00994018.
6. N. Cramer. A representation for the adaptive generation of simple sequential programs. In *Proceedings of the First International Conference on Genetic Algorithms*, volume 183, page 187, 1985.
7. M. G. de Carvalho, M. A. Gonçalves, A. H. F. Laender, and A. S. da Silva. Learning to deduplicate. In *Proceedings of the 6th ACM/IEEE-CS joint conference on Digital libraries*, JCDL '06, pages 41–50, New York, NY, USA, 2006. ACM.
8. M. G. de Carvalho, A. H. F. Laender, M. A. Goncalves, and A. S. da Silva. A genetic programming approach to record deduplication. *IEEE Transactions on Knowledge and Data Engineering*, 99(PrePrints), 2010.
9. K. A. De Jong. *An analysis of the behavior of a class of genetic adaptive systems.* PhD thesis, Ann Arbor, MI, USA, 1975.
10. M. Elfeky, V. Verykios, and A. Elmagarmid. Tailor: A record linkage toolbox. In *Data Engineering, 2002. Proceedings. 18th International Conference on*, pages 17–28. IEEE, 2002.
11. J. Euzenat, A. Ferrara, C. Meilicke, et al. First Results of the Ontology Alignment Evaluation Initiative 2010. *Ontology Matching*, page 85, 2010.
12. J. Euzenat and P. Shvaiko. *Ontology matching*. Springer-Verlag, Heidelberg (DE), 2007.

13. I. P. Fellegi and A. B. Sunter. A Theory for Record Linkage. *Journal of the American Statistical Association*, 64(328), 1969.
14. C. Henrik Westerberg and J. Levine. Investigation of different seeding strategies in a genetic planner. *Applications of Evolutionary Computing*, pages 505–514, 2001.
15. J. Holland. Adaptation in natural and artificial systems. 1975.
16. R. Isele, A. Jentzsch, and C. Bizer. Silk server - adding missing links while consuming linked data. In *1st International Workshop on Consuming Linked Data (COLD 2010), Shanghai*, 2010.
17. T. Jones. Crossover, macromutation, and population-based search. In *Proceedings of the Sixth International Conference on Genetic Algorithms*, pages 73–80, 1995.
18. H. Köpcke and E. Rahm. Frameworks for entity matching: A comparison. *Data & Knowledge Engineering*, 69(2):197 – 210, 2010.
19. J. Koza. *Genetic programming: on the programming of computers by means of natural selection.*
20. J. Koza, F. Bennett III, F. Bennett, D. Andre, and M. Keane. Genetic Programming III: Automatic programming and automatic circuit synthesis, 1999.
21. J. Koza, M. Keane, and M. Streeter. What's AI done for me lately? Genetic programming's human-competitive results. *Intelligent Systems, IEEE*, 18(3):25–31, 2003.
22. J. Koza, M. Keane, M. Streeter, W. Mydlowec, J. Yu, and G. Lanza. *Genetic programming IV: Routine human-competitive machine intelligence.* Springer Verlag, 2005.
23. J. Koza, M. Keane, J. Yu, F. Bennett, and W. Mydlowec. Automatic creation of human-competitive programs and controllers by means of genetic programming. *Genetic Programming and Evolvable Machines*, 1(1):121–164, 2000.
24. W. Langdon and R. Poli. Fitness causes bloat. *Soft Computing in Engineering Design and Manufacturing*, 1:13–22, 1997.
25. D. Montana. Strongly typed genetic programming. *Evolutionary computation*, 3(2):199–230, 1995.
26. R. Poli, W. Langdon, and N. McPhee. *A field guide to genetic programming.* Lulu Enterprises Uk Ltd, 2008.
27. L. Spector, H. Barnum, H. Bernstein, and N. Swamy. Finding a better-than-classical quantum AND/OR algorithm using genetic programming. In *Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on*, volume 3. IEEE, 1999.
28. S. Tejada, C. Knoblock, and S. Minton. Learning object identification rules for information integration. *Information Systems*, 26(8):607–633, 2001.
29. S. Tejada, C. A. Knoblock, and S. Minton. Learning domain-independent string transformation weights for high accuracy object identification. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '02, pages 350–359, New York, NY, USA, 2002. ACM.
30. W. E. Winkler. Matching and Record Linkage. In *Business Survey Methods*, pages 355–384, 1995.
31. W. E. Winkler. Methods for record linkage and bayesian networks. Technical report, Series RRS2002/05, U.S. Bureau of the Census, 2002.
32. B. Zhang and H. M
"uhlenbein. Balancing accuracy and parsimony in genetic programming. *Evolutionary Computation*, 3(1):17–38, 1995.

RAVEN – Active Learning of Link Specifications

Axel-Cyrille Ngonga Ngomo, Jens Lehmann, Sören Auer, Konrad Höffner

Department of Computer Science, University of Leipzig
Johannisgasse 26, 04103 Leipzig, Germany
{ngonga|lehmann|auer}@informatik.uni-leipzig.de,
konrad.hoeffner@uni-leipzig.de
WWW home page: http://aksw.org

Abstract. With the growth of the Linked Data Web, time-efficient approaches for computing links between data sources have become indispensable. Yet, in many cases, determining the right specification for a link discovery problem is a tedious task that must still be carried out manually. We present RAVEN, an approach for the semi-automatic determination of link specifications. Our approach is based on the combination of stable solutions of matching problems and active learning with the time-efficient link discovery framework LIMES. RAVEN aims at requiring a small number of interactions with the user to generate classifiers of high accuracy. We focus on using RAVEN to compute and configure boolean and weighted classifiers, which we evaluate in three experiments against link specifications created manually. Our evaluation shows that we can compute linking configurations that achieve more than 90% F-score by asking the user to verify at most twelve potential links.

Keywords: Linked Data, Link Discovery, Algorithms, Constraints

1 Introduction

The core idea behind the Linked Data paradigm is to facilitate the transition from the document-oriented to the Semantic Web by extending the current Web with a commons of interlinked data sources [2]. One of the key challenges that arise when trying to discover links between two data sources lies in the *specification* of an appropriate configuration for the tool of choice [10]. Such a specification usually consists of a set of restrictions on the source and target knowledge base, a list of properties of the source and target knowledge base to use for similarity detection, a combination of suitable similarity measures (e.g., Levenshtein [9]) and similarity thresholds. Specifying link configurations is a tedious process, as the user does not necessarily know which combinations of properties lead to an accurate linking configuration. The difficulty of devising suitable link discovery specifications is amplified on the Web of Data by the *sheer size* of the knowledge bases (which often contain millions of instances) and their *heterogeneity* (i.e., by the complexity of the underlying ontologies, which can contain thousands of different types of instances and properties) [2].

In this paper, we present the RApid activE liNking (RAVEN) approach. RAVEN is the first approach to apply active learning techniques for the semi-automatic detection of specifications for link discovery. Our approach is based on a combination of stable matching and a novel active learning algorithm derived from perceptron learning. RAVEN allows to determine (1) a sorted matching of classes to interlink; this matching represents the set of restrictions of the source and target knowledge bases, (2) a stable matching of properties based on the selected restrictions that specifies the similarity space within which the linking is to be carried out and (3) a highly accurate link specification via active learning. Our evaluation with three series of experiments shows that we can compute linking configurations that achieve more than 90% F-score by asking the user to verify at most twelve potential links. RAVEN is generic enough to be employed with any link discovery framework that supports complex link specifications. The results presented herein rely on the LIMES framework for linking. We chose LIMES because it implements lossless approaches and is very time-efficient.

2 Related Work

Current approaches for link discovery on the Web of Data can be subdivided into two categories: *domain-specific* and *universal* approaches. Domain-specific link discovery frameworks aim at discovering links between knowledge bases from a particular domain. For example, the approach implemented in RKB knowledge base (RKB-CRS) [5] focuses on computing links between universities and conferences while GNAT [12] discovers links between music data sets. Further simple or domain-specific approaches can be found in [7, 17, 11].

Universal link discovery frameworks are designed to carry out mapping tasks independently from the domain of the source and target knowledge bases. For example, RDF-AI [15] implements a five-step approach that comprises the preprocessing, matching, fusion, interlinking and post-processing of data sets. SILK [18] is a time-optimized tool for link discovery. It implements a multi-dimensional blocking approach that projects the instances to match in a multi-dimensional metric space. Subsequently, this space is subdivided into to overlapping blocks that are used to retrieve matching instances without loosing links. Another lossless Link Discovery framework is LIMES [10], which addresses the scalability problem by utilizing the *triangle inequality* in metric spaces to compute pessimistic estimates of instance similarities. Based on these approximations, LIMES can filter out a large number of non-matches.

The task of discovering links between knowledge bases is closely related with record linkage and deduplication [3]. The database community has produced a vast amount of literature on efficient algorithms for solving these problems. Different blocking techniques such as standard blocking, sorted-neighborhood, bi-gram indexing, canopy clustering and adaptive blocking (see, e.g., [8]) have been developed to address the problem of the quadratic time complexity of brute force comparison methods. Active learning has been employed in the database community [13, 14, 1] to address the configuration problem because active learning approaches usually present only few match candidates to the user for manual verification. The technique is particularly efficient in terms of required user input [16], because the user is only confronted with those match candidates which provide a high benefit for the underlying learning algorithm.

The RAVEN approach goes beyond the state of the art in several ways: It is the first active learning algorithm and RDF-based approach to use machine learning for obtaining link specifications. Moreover, it is the first approach to detect corresponding classes and properties automatically for the purpose of Link Discovery. Note that similar approaches developed for databases assume the mapping of columns to be known [1]. Yet, this assumption cannot be made when trying to link knowledge bases from the Web of Data because of the possible size of the underlying ontology. By supporting the automatic detection of links, we are able to handle heterogeneous knowledge bases with large schemata.

3 Preliminaries

Our approach to the active learning of linkage specifications extends ideas from several research areas, especially classification and stable matching problems. In the following, we present the notation that we use throughout this paper and explain the theoretical framework underlying our work.

3.1 Problem Definition

The link discovery problem, which is similar to the record linkage problem, is an ill-defined problem and is consequently difficult to model formally [1]. In general, link discovery aims to discover pairs $(s, t) \in S \times T$ related via a relation R.

Definition 1 (Link Discovery). *Given two sets S (source) and T (target) of entities, compute the set $\mathcal{M}$ of pairs of instances $(s, t) \in S \times T$ such that $R(s, t)$.*

The sets S resp. T are usually subsets of the instances contained in two knowledge bases $\mathcal{K}_S$ resp. $\mathcal{K}_T$. In most cases, the computation of whether $R(s, t)$ holds for two elements is carried out by projecting the elements of S and T based on their properties in a similarity space $\mathfrak{S}$ and setting $R(s, t)$ iff some similarity condition is satisfied. The specification of the sets S and T and of this similarity condition is usually carried out within a *link specification.*

Definition 2 (Link Specification). *A link specification consists of three parts: (1) two sets of restrictions $\mathcal{R}_1^S$ ... $\mathcal{R}_m^S$ resp. $\mathcal{R}_1^T$ ... $\mathcal{R}_k^T$ that specify the sets S resp. T, (2) a specification of a complex similarity metric σ via the combination of several atomic similarity measures σ_1, ..., σ_n and (3) a set of thresholds τ_1, ..., τ_n such that τ_i is the threshold for σ_i.*

A restriction $\mathcal{R}$ is generally a logical predicate. Typically, restrictions in link specifications state the `rdf:type` of the elements of the set they describe, i.e., $\mathcal{R}(x) \leftrightarrow x$ `rdf:type someClass` or the features the elements of the set must have, e.g., $\mathcal{R}(x) \leftrightarrow (\exists y : x \ \texttt{someProperty} \ y)$. Each $s \in S$ must abide by each of the restrictions $\mathcal{R}_1^S \ ... \ \mathcal{R}_m^S$, while each $t \in T$ must abide by each of the restrictions $\mathcal{R}_1^T \ ... \ \mathcal{R}_k^T$. Note that the atomic similarity functions $\sigma_1, ..., \sigma_n$ can be combined to σ by different means. In this paper, we will focus on using boolean operators and real weights combined as conjunctions.

According to the formalizations of link discovery and link specifications above, finding matching pairs of entities can be defined equivalently as a classification task, where the classifier $\mathcal{C}$ is a function from $S \times T$ to $\{-1, +1\}$.

Definition 3 (Link Discovery as Classification). *Given the set $S \times T$ of possible matches, the goal of link discovery is to find a classifier $\mathcal{C} : S \times T \rightarrow \{-1, +1\}$ such that $\mathcal{C}$ maps non-matches to the class -1 and matches to $+1$.*

In general, we assume classifiers that operate in an n-dimensional similarity space $\mathfrak{S}$. Each of the dimensions of $\mathfrak{S}$ is defined by a similarity function σ_i that operates on a certain pair of attributes of s and t. Each classifier $\mathcal{C}$ on $\mathfrak{S}$ can be modeled via a specific function $\mathcal{F}_\mathcal{C}$. $\mathcal{C}$ then returns $+1$ iff the logical statement $\mathcal{P}_\mathcal{C}(\mathcal{F}_\mathcal{C}(s,t))$ holds and -1 otherwise, where $\mathcal{P}_\mathcal{C}$ is what we call the specific predicate of $\mathcal{C}$. In this work, we consider two families of classifiers: *linear weighted* classifiers $\mathcal{L}$ and *boolean conjunctive* classifiers $\mathcal{B}$. The specific function of linear weighted classifiers is of the form

$$\mathcal{F}_\mathcal{L}(s,t) = \sum_{i=1}^{n} \omega_i \sigma_i(s,t), \tag{1}$$

where $\omega_i \in \mathbb{R}$. The predicate $\mathcal{P}_\mathcal{L}$ for a linear classifier is of the form $\mathcal{P}_\mathcal{L}(X) \leftrightarrow (X \geq \tau)$, where $\tau = \tau_1 = ... = \tau_n \in [0,1]$ is the similarity threshold. A boolean classifier $\mathcal{B}$ is a conjunction of n atomic linear classifiers $\mathcal{C}_1$, ... ,$\mathcal{C}_n$, i.e., a conjunction of classifiers that each operate on exactly one of the n dimensions of the similarity space $\mathfrak{S}$. Thus, the specific function $\mathcal{F}_\mathcal{B}$ is as follows:

$$\mathcal{F}_\mathcal{B}(s,t) = \bigwedge_{i=0}^{n} (\sigma_i(s,t) \geq \tau_i) \tag{2}$$

and the specific predicate is simply $\mathcal{P}_\mathcal{B}(X) = X$. Note that given that classifiers are usually learned by using iterative approaches, we will denote classifiers, weights and thresholds at iteration t by using superscripts, i.e., $\mathcal{C}^t$, ω_i^t and τ_i^t.

Current approaches to learning in record matching assume that the similarity space $\mathfrak{S}$ is given. While this is a sensible premise for mapping problems which rely on simple schemas, the large schemas (i.e., the ontologies) that underlie many data sets in the Web of Data do not allow such an assumption. The DBpedia ontology (version 3.6) for example contains 275 classes and 1335 properties. Thus, it would be extremely challenging for a user to specify the properties to map when carrying out a simple deduplication analysis, let alone more complex tasks using the DBpedia data set. In the following, we give a brief overview of stable matching problems, which we use to solve the problem of suggesting appropriate sets of restrictions on data and matching properties.

3.2 Stable Matching Problems

The best known stable matching problem is the stable marriage problem $\mathcal{SM}$ as formulated by [4]. Here, one assumes two sets M (males) and F (females) such that $|M| = |F|$ and two functions $\mu : M \times F \rightarrow \{1, ..., |F|\}$ resp. $\gamma : M \times F \rightarrow \{1, ..., |M|\}$, that give the degree to which a male likes a female resp. a female a male. $\mu(m, f) > \mu(m, f')$ means that m prefers f to f'. Note, that for all f and f' where $f \neq f'$ holds, $\mu(m, f) \neq \mu(m, f')$ must also hold. Analogously, $m \neq m'$ implies $\gamma(m, f) \neq \gamma(m', f)$. A bijective function $s : M \rightarrow F$ is called a stable matching iff for all m, m', f, f' the following holds:

$$(s(m) = f) \wedge (s(m') = f') \wedge (\mu(m, f') > \mu(m, f)) \rightarrow (\gamma(m', f') > \gamma(m, f')) \quad (3)$$

In [4] an algorithm for solving this problem is presented and it is shown how it can be used to solve the well-know Hospital/Residents ($\mathcal{HR}$) problem. Formally, $\mathcal{HR}$ extends $\mathcal{SM}$ by assuming a set R of residents (that maps M) and a set of hospitals (that maps F) with $|R| \neq |F|$. Each hospital $h \in H$ is assigned a capacity $c(h)$. A stable solution of the Hospital/Residents problem is a mapping of residents to hospitals such that each hospital accepts maximally $c(h)$ residents and that fulfills Equation 3. Note that we assume that there are no ties, i.e., that the functions μ and γ are injective.

4 The RAVEN Approach

Our approach, dubbed RAVEN (RApid actiVE liNking), addresses the task of linking two knowledge bases S and T by using the active learning paradigm within the pool-based sampling setting [16]. Overall, the goal of RAVEN is to find the best classifier $\mathcal{C}$ that achieves the highest possible precision, recall or F_1 score as desired by the user. The algorithm also aims to minimize the burden on the user by limiting the number of link candidates that must be labeled by the user to a minimum.

Algorithm 1 The RApid actiVE liNking (RAVEN) algorithm

```
Require: Source knowledge base $\mathcal{K}_S$
Require: Target knowledge base $\mathcal{K}_T$
  Find stable class matching between classes of $\mathcal{K}_S$ and $\mathcal{K}_T$
  Find stable property matching for the selected classes
  Compute sets $S$ and $T$; Create initial classifier $\mathcal{C}^0$; $t := 0$
  while termination condition not satisfied do
    Ask the user to classify $2\alpha$ examples; Update $\mathcal{C}^t$ to $\mathcal{C}^{t+1}$; t := t+1
  end while
  Compute set $\mathcal{M}$ of links between $S$ and $T$ based on $\mathcal{C}^t$
  return $\mathcal{M}$
```

An overview of our approach is given in Algorithm 1. In a first step, RAVEN aims to detect the restrictions that will define the sets S and T. To achieve this goal, it tries to find a stable matching of pairs of classes, whose instances are to be linked. The second step of our approach consists of finding a stable matching between the properties that describe the instances of the classes specified in the first step. The user is also allowed to alter the suggested matching at will. Based on the property mapping, we compute S and T and generate an initial classifier $\mathcal{C} = \mathcal{C}^0$ in the third step. We then refine $\mathcal{C}$ iteratively by asking the user to classify pairs of instances that are most informative for our algorithm. $\mathcal{C}$ is updated until a termination condition is reached, for example $\mathcal{C}^t = \mathcal{C}^{t+1}$. The final classifier is used to compute the links between S and T, which are returned by RAVEN. In the following, we expand upon each of these three steps.

4.1 Stable Matching of Classes

The first component of a link specification is a set of restrictions that must be fulfilled by the instances that are to be matched. We use a two-layered approach for matching classes in knowledge bases. Our *default approach* begins by selecting a user-specified number of `sameAs` links between the source and the target knowledge base randomly. Then, it computes μ and γ on the classes C_S of $\mathcal{K}_S$ and C_T of $\mathcal{K}_T$ as follows[1]:

$$\mu(C_S, C_T) = \gamma(C_S, C_T) = |\{s \texttt{ type } C_s \wedge s \texttt{ sameAs } t \wedge t \texttt{ type } C_T\}|. \quad (4)$$

In the case when no `sameAs` links are available, we run our *fallback* approach. It computes μ and γ on the classes of S and T as follows:

$$\mu(C_S, C_T) = \gamma(C_S, C_T) = |\{s \texttt{ type } C_s \wedge s \texttt{ p } x \wedge t \texttt{ type } C_T \wedge t \texttt{ q } x\}|, \quad (5)$$

where `p` and `q` can be any property. Let $c(S)$ be the number of classes C_S of S such that $\mu(C_S, C_T) > 0$ for any C_T. Furthermore, let $c(T)$ be the number of classes C_T of T such that $\gamma(C_S, C_T) > 0$ for any C_S. The capacity of each C_T is set to $\lceil c(S)/c(T) \rceil$, thus ensuring that the hospitals provide enough capacity to map all the possible residents. Once μ, γ and the capacity of each hospital has been set, we solve the equivalent $\mathcal{HR}$ problem.

4.2 Stable Matching of Properties

The detection of the best matching pairs of properties is very similar to the computation of the best matching classes. For properties `p` and `q`, we set:

$$\mu(p, q) = \gamma(p, q) = |\{s \texttt{ type } C_s \wedge s \texttt{ p } x \wedge t \texttt{ type } C_T \wedge t \texttt{ q } x\}|. \quad (6)$$

The initial mapping of properties defines the similarity space in which the link discovery task will be carried out. Note that none of the prior approaches to active learning for record linkage or link discovery automatized this step. We associate each of the basis vectors σ_i of the similarity space to exactly one of the pairs (`p`, `q`) of mapping properties. Once the restrictions and the property mapping have been specified, we can fetch the elements of the sets S and T.

[1] Note that we used `type` to denote `rdf:type` and `sameAs` to denote `owl:sameAs`.

4.3 Initial Classifier

The specific formula for the initial linear weighted classifier $\mathcal{L}^0$ results from the formal model presented in Section 3 and is given by

$$\mathcal{F}_{\mathcal{L}}{}^0(s,t) = \sum_{i=1}^{n} \omega_i^0 \sigma_i(s,t). \tag{7}$$

Several initialization methods can be used for ω_i^0 and the initial threshold τ^0 of $\mathcal{P}_{\mathcal{L}}$. In this paper, we chose to use the simplest possible approach by setting $\omega_i^0 := 1$ and $\tau^0 := \kappa n$, where $\kappa \in [0,1]$ is a user-specified *threshold factor*. Note that setting the overall threshold to κn is equivalent to stating that the arithmetic mean of the $\sigma_i(s,t)$ must be equal to κ.

The equivalent initial boolean classifier $\mathcal{B}^0$ is given by

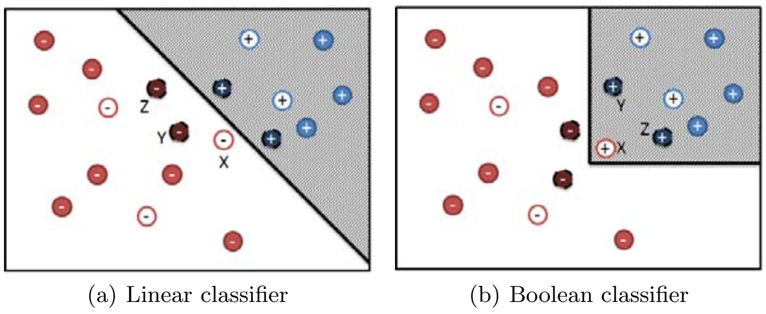
$$\mathcal{F}_{\mathcal{B}}{}^0(s,t) = \bigwedge_{i=0}^{n} (\sigma_i^0(s,t) \geq \tau_i^0) \text{ where } \tau_i^0 := \kappa. \tag{8}$$

4.4 Updating Classifiers

RAVEN follows an iterative update strategy, which consists of asking the user to classify 2α elements of $S \times T$ (α is explained below) in each iteration step t and using these to update the values of ω_i^{t-1} and τ_i^{t-1} computed at step $t-1$. The main requirements to the update approach is that it computes those elements of $S \times T$ whose classification allow to maximize the convergence of $\mathcal{C}^t$ to a good classifier and therewith to minimize the burden on the user. The update strategy of RAVEN varies slightly depending on the family of classifiers. In the following, we present how RAVEN updates linear and boolean classifiers.

Updating linear classifiers. The basic intuition behind our update approach is that we aim to present the user with those elements from $S \times T$ whose classification is most unsure. We call the elements presented to the user *examples*. We call an example *positive* when it is assumed by the classifier to belong to $+1$. Else we call it *negative*. Once the user has provided us with the correct classification for all examples, the classifier can be updated effectively so as to better approximate the target classifier. In the following, we will define the notion of *most informative example* for linear classifiers before presenting our update approach.

When picturing a classifier as a boundary in the similarity space $\mathfrak{S}$ that separates the classes $+1$ and -1, the examples whose classification is most uncertain are those elements from $S \times T$ who are closest to the boundary. Note that we must exclude examples that have been classified previously, as presenting them to the user would not improve the classification accuracy of RAVEN while generating extra burden on the user, who would have to classify the same link candidate twice. Figure 1 depicts the idea behind most informative examples. In both subfigures, the circles with a dashed border represent the 2 most informative positive and negatives examples, the solid disks represent elements from $S \times T$ and the circles are examples that have already been classified by the users. Note that while X is closer to the boundary than Y and Z, it is not a most informative example as it has already been classified by the user.



(a) Linear classifier (b) Boolean classifier

Fig. 1. Most informative examples for linear and boolean classifiers. The current elements of the classes -1 resp. $+1$ are marked with $-$ resp. $+$.

Formally, let $\mathcal{M}^t$ be the set of $(s,t) \in S \times T$ classified by $\mathcal{L}^t$ as belonging to $+1$. Furthermore, let $\mathcal{P}^{t-1}$ (resp. $\mathcal{N}^{t-1}$) be the set of examples that have already been classified by the user as being positive examples, i.e, links (resp. negative examples, i.e., wrong links). We define a set Λ as being a set of most informative examples λ for $\mathcal{L}^{t+1}$ when the following conditions hold:

$$\forall \lambda \in S \times T \ (\lambda \in \Lambda \rightarrow \lambda \notin \mathcal{P}^{t-1} \cup \mathcal{N}^{t-1}) \tag{9}$$

$$\forall \lambda' \notin \mathcal{P}^{t-1} \cup \mathcal{N}^{t-1} : \lambda' \neq \lambda \rightarrow |\mathcal{F_L}^t(\lambda') - \tau^t| \geq |\mathcal{F_L}^t(\lambda) - \tau^t|. \tag{10}$$

Note that there are several sets of most informative examples of a given magnitude. We denote a set of most informative examples of magnitude α by Λ_α. A set of most informative positive examples, Λ^+, is a set of pairs such that

$$\forall \lambda \notin \Lambda^+ \cup \mathcal{P}^{t-1} \cup \mathcal{N}^{t-1} : (\mathcal{F_L}^t(\lambda) < \tau^t) \vee (\forall \lambda^+ \in \Lambda^+ : \mathcal{F_L}^t(\lambda) > \mathcal{F_L}^t(\lambda^+)). \tag{11}$$

In words, Λ^+ is the set of examples that belong to class $+1$ according to $\mathcal{C}$ and are closest to $\mathcal{C}$'s boundary. Similarly, the set of most informative negative examples, Λ^-, is the set of examples such that

$$\forall \lambda \notin \Lambda^- \cup \mathcal{P}^{t-1} \cup \mathcal{N}^{t-1} : (\mathcal{F_L}^t(\lambda) \geq \tau^t) \vee (\forall \lambda^- \in \Lambda^- : \mathcal{F_L}^t(\lambda) < \mathcal{F_L}^t(\lambda^-)). \tag{12}$$

We denote a set of most informative (resp. negative) examples of magnitude α as Λ^+_α (resp. Λ^-_α). The 2α examples presented to the user consist of the union $\Lambda^+_\alpha \cup \Lambda^-_\alpha$, where Λ^+_α and Λ^-_α are chosen randomly amongst the possible sets of most informative positive resp. negative examples .

The update rule for the weights of $\mathcal{L}^t$ is derived from the well-known Perceptron algorithm, i.e.,

$$\omega_i^{t+1} = \omega_i^t + \eta^+ \sum_{\lambda \in \Lambda^+} \rho(\lambda)\sigma_i(\lambda) - \eta^- \sum_{\lambda \in \Lambda^-} \rho(\lambda)\sigma_i(\lambda), \tag{13}$$

where η^+ is the learning rate for positives examples, η^- is the learning rate for negative examples and $\rho(\lambda)$ is 0 when the classification of λ by the user and $\mathcal{L}^t$ are the same and 1 when they differ.

The threshold is updated similarly, i.e,

$$\tau_i^{t+1} = \tau_i^t + \eta^+ \sum_{\lambda \in \Lambda_\alpha^+} \rho(\lambda)\mathcal{F}_{\mathcal{L}}{}^t(\lambda) - \eta^- \sum_{\lambda \in \Lambda_\alpha^-} \rho(\lambda)\mathcal{F}_{\mathcal{L}}{}^t(\lambda). \quad (14)$$

Note that the weights are updated by using the dimension which they describe while the threshold is updated by using the whole specific function. Finally, the sets $\mathcal{P}^{t-1}$ and $\mathcal{N}^{t-1}$ are updated to

$$\mathcal{P}^t := \mathcal{P}^{t-1} \cup \Lambda_\alpha^+ \text{ and } \mathcal{N}^t := \mathcal{N}^{t-1} \cup \Lambda_\alpha^-. \quad (15)$$

Updating boolean classifiers. The notion of *most informative example* differs slightly for boolean classifiers. λ is considered a most informative example for $\mathcal{B}$ when the conditions

$$\lambda \notin \mathcal{P}^{t-1} \cup \mathcal{N}^{t-1} \quad (16)$$

and

$$\forall \lambda' \notin \mathcal{P}^{t-1} \cup \mathcal{N}^{t-1} : \lambda' \neq \lambda \rightarrow \sum_{i=1}^{n} |\sigma_i^t(\lambda') - \tau_i^t| \geq \sum_{i=1}^{n} |\sigma_i^t(\lambda) - \tau_i^t| \quad (17)$$

hold. The update rule for the thresholds τ_i^t of $\mathcal{B}$ is then given by

$$\tau_i^{t+1} = \tau_i^t + \eta^+ \sum_{\lambda \in \Lambda_\alpha^+} \rho(\lambda)\sigma_i(\lambda) - \eta^- \sum_{\lambda \in \Lambda_\alpha^-} \rho(\lambda)\sigma_i(\lambda), \quad (18)$$

where η^+ is the learning rate for positives examples, η^- is the learning rate for negative examples and $\rho(\lambda)$ is 0 when the classification of λ by the user and $\mathcal{C}_{t-1}$ are the same and 1 when they differ. The sets $\mathcal{P}^{t-1}$ and $\mathcal{N}^{t-1}$ are updated as given in Equation 15.

5 Experiments and Results

5.1 Experimental Setup

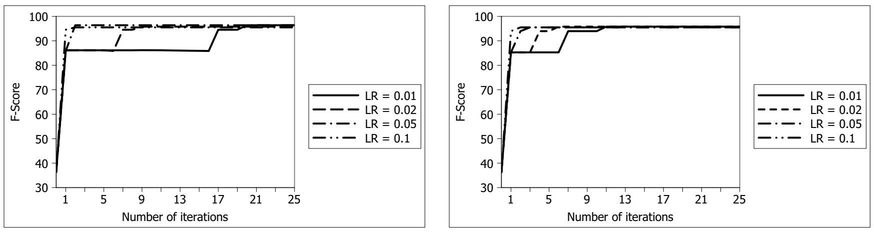
We carried out three series of experiments to evaluate our approach. In our first experiment, dubbed *Diseases*, we aimed to map diseases from DBpedia with diseases from Diseasome. In the *Drugs* experiments, we linked drugs from Sider with drugs from Drugbank. Finally, in the *Side-Effects* experiments, we aimed to link side-effects of drugs and diseases in Sider and Diseasome.

In all experiments, we used the following setup: The learning rates η^+ and η^- were set to the same value η, which we varied between 0.01 and 0.1. We set the number of inquiries per iteration to 4. The threshold factor κ was set to

0.8. In addition, the number of instances used during the automatic detection of class resp. property matches was set to 100 resp. 500. The fallback solution was called and compared the property values of 1000 instances chosen randomly from the source and target knowledge bases. We used the trigrams metric as default similarity measure for strings and the Euclidean similarity as default similarity measure for numeric values. To measure the quality of our results, we used *precision*, *recall* and *F-score*. We also measured the total number of inquiries that RAVEN needed to reach its maximal F-Score. As reference data, we used the set of instances that mapped perfectly according to a configuration created manually.

5.2 Results

The results of our experiments are shown in Figures 2 and 3. The first experiment, Diseases, proved to be the most difficult for RAVEN. Although the `sameAs` links between `Diseasome` and `DBpedia` allowed our experiment to run without making use of the fallback solution, we had to send 12 inquiries to the user when the learning rate was set to 0.1 to determine the best configuration that could be learned by linear and boolean classifiers. Smaller learning rates led to the system having to send even up to 80 inquiries ($\eta = 0.01$) to determine the best configuration. In this experiment linear classifiers outperform boolean classifiers in all setups by up to 0.8% F-score.

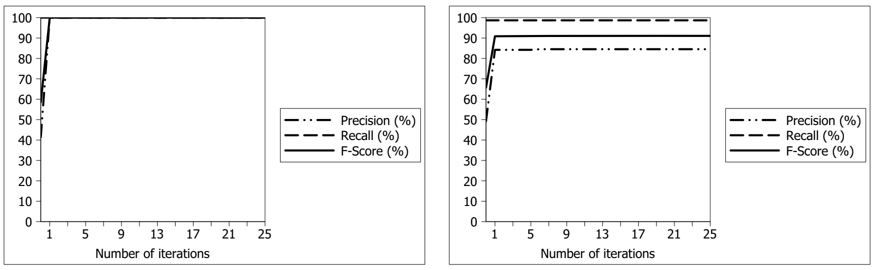


(a) Linear classifier (b) Boolean classifier

Fig. 2. Learning curves on Diseases experiments. LR stands for learning rate.

The second and the third experiment display the effectiveness of RAVEN. Although the fallback solution had to be used in both cases, our approach is able to detect the right configuration with an accuracy of even 100% in the Side-Effects experiment by asking the user no more than 4 questions. This is due to the linking configuration of the user leading to two well-separated sets of instances. In these cases, RAVEN converges rapidly and finds a good classifier rapidly. Note that in these two cases, all learning rates in combination with both linear and boolean classifiers led to the same results (see Figures 3(b) and 3(a)).

Although we cannot directly compare our results to other approaches as it is the first active learning algorithm for learning link specifications, results reported



(a) Learning curve in the Side-Effects experiment (b) Learning curve in the Drug experiment

Fig. 3. Learning curve in the Side-Effects and Drugs experiments

in the database area suggest that RAVEN achieves state-of-the-art performance. The runtimes required for each iteration ensure that our approach can be used in real-world interactive scenarios. In the worst case, the user has to wait for 1.4 seconds between two iterations. The runtime for the computation of the initial configuration depends heavily on the connectivity to the SPARQL endpoints. In our experiments, the computation of the initial configuration demanded 60s when the default solution was used. The fallback solution required up to 90s.

6 Conclusion and Future Work

In this paper, we presented RAVEN, the first active learning approach tailored towards semi-automatic Link Discovery on the Web of Data. We showed how RAVEN uses stable matching algorithms to detect initial link configurations. We opted to use the solution of the hospital residence problem ($\mathcal{HR}$) without ties because of the higher time complexity of the solution of $\mathcal{HR}$ with ties, i.e., L^4, where L is the size of the longest preference list, i.e., $max(|R|, |H|)$. Still, our work could be extended by measuring the effect of considering ties on the matching computed by RAVEN. Our experimental results showed that RAVEN can compute accurate link specifications (F-score between 90% and 100%) by asking the user to classify a very small number of positive and negative examples (between 4 and 12 for a learning rate of 0.1). Our results also showed that our approach can be used in an interactive scenario because of LIMES' time efficiency, which allowed to compute new links in less than 1.5 seconds in the evaluation tasks. The advantages of this interactive approach can increase the quality of generated links while reducing the effort to create them.

In future work, we will explore how to detect optimal values for the threshold factor κ automatically, for example, by using clustering approaches. In addition, we will investigate the automatic detection of domain-specific metrics that can model the idiosyncrasies of the dataset at hand. Another promising extension to RAVEN is the automatic detection of the target knowledge base to even further simplify the linking process, since users often might not even be aware of appropriate linking targets (see [6] for research in this area). By these means, we aim to provide the first zero-configuration approach to Link Discovery.

Acknowledgement

This work was supported by the Eurostars grant SCMS E!4604, the EU FP7 grant LOD2 (GA no. 257943) and a fellowship grant of the University of Mainz.

References

1. A. Arasu, M. Götz, and R. Kaushik. On active learning of record matching packages. In *SIGMOD*, pages 783–794, 2010.
2. C. Bizer, T. Heath, and T. Berners-Lee. Linked data - the story so far. *International Journal on Semantic Web and Information Systems*, 2009.
3. J. Bleiholder and F. Naumann. Data fusion. *ACM Comput. Surv.*, 41(1):1–41, 2008.
4. D. Gale and L. S. Shapley. College admissions and the stability of marriage. *The American Mathematical Monthly*, 69(1):9–15, 1962.
5. H. Glaser, I. C. Millard, W.-K. Sung, S. Lee, P. Kim, and B.-J. You. Research on linked data and co-reference resolution. Technical report, University of Southampton, 2009.
6. C. Guéret, P. Groth, F. van Harmelen, and S. Schlobach. Finding the achilles heel of the web of data: Using network analysis for link-recommendation. In *ISWC*, pages 289–304, 2010.
7. A. Hogan, A. Polleres, J. Umbrich, and A. Zimmermann. Some entities are more equal than others: statistical methods to consolidate linked data. In *NeFoRS*, 2010.
8. H. Köpcke, A. Thor, and E. Rahm. Comparative evaluation of entity resolution approaches with fever. *Proc. VLDB Endow.*, 2(2):1574–1577, 2009.
9. V. I. Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. Technical Report 8, 1966.
10. A.-C. Ngonga Ngomo and S. Auer. Limes - a time-efficient approach for large-scale link discovery on the web of data. In *Proceedings of IJCAI*, 2011.
11. G. Papadakis, E. Ioannou, C. Niedere, T. Palpanasz, and W. Nejdl. Eliminating the redundancy in blocking-based entity resolution methods. In *JCDL*, 2011.
12. Y. Raimond, C. Sutton, and M. Sandler. Automatic interlinking of music datasets on the semantic web. In *1st Workshop about Linked Data on the Web*, 2008.
13. S. Sarawagi and A. Bhamidipaty. Interactive deduplication using active learning. In *KDD*, pages 269–278, 2002.
14. S. Sarawagi, A. Bhamidipaty, A. Kirpal, and C. Mouli. Alias: An active learning led interactive deduplication system. In *VLDB*, pages 1103–1106, 2002.
15. F. Scharffe, Y. Liu, and C. Zhou. Rdf-ai: an architecture for rdf datasets matching, fusion and interlink. In *Proc. IJCAI 2009 IR-KR Workshop*, 2009.
16. B. Settles. Active learning literature survey. Technical Report 1648, University of Wisconsin-Madison, 2009.
17. J. Sleeman and T. Finin. Computing foaf co-reference relations with rules and machine learning. In *Proceedings of SDoW*, 2010.
18. J. Volz, C. Bizer, M. Gaedke, and G. Kobilarov. Discovering and maintaining links on the web of data. In *ISWC 2009*, pages 650–665. Springer, 2009.

Towards an Automatic Parameterization of Ontology Matching Tools based on Example Mappings

Dominique Ritze[1] and Heiko Paulheim[2]

[1] Mannheim University Library
dominique.ritze@bib.uni-mannheim.de
[2] Technische Universität Darmstadt
Knowledge Engineering Group
paulheim@ke.tu-darmstadt.de

Abstract. With a growing number of ontologies and datasets using those ontologies, ontology mappings become an essential building block of the Semantic Web. In the last years, a larger number of sophisticated ontology matching tools for generating such mappings has been developed. The quality of the mappings provided by those tools typically depends on the settings of the tools' parameters. As this is a non-trivial task for an end user, we propose the *ECOMatch* approach, which asks the user to provide example mappings instead of parameter settings, and automatically determines a suitable parameter setting based on those examples. We show how the preliminary result quality of ontology mappings can be improved by applying automatic, example-based configuration of ontology matching tools.

1 Introduction

Ontologies formally describe the concepts used in a domain. While in an ideal scenario, there is one ontology which is shared throughout a whole domain, reality often faces the parallel use of different ontologies, which have been developed independently from each other. *Ontology matching* [8] is used for creating mappings between ontologies.

During the past years, a lot of research has been devoted to developing highly sophisticated tools for performing ontology matching automatically [6, 7]. Those tools are able to produce high-quality mappings between ontologies, given that their parameters (such as weights and thresholds used to compute the mappings) are tuned well. Such a tuning, however, is often complicated, since it involves the setting of many parameters and requires a lot of detail knowledge about the underlying algorithms and implementations. For example, the state of the art matching tool *Falcon-AO* [12] has 33 different parameters that can be manipulated, which makes it hard to guess an optimal parameter set without a planned approach. Furthermore, there are often no universally optimal settings: a configuration that performs well on one pair of ontologies may produce bad results on another one. Therefore, an automatic configuration of matching tools has been named as one of the top ten challenges for ontology matching [22].

In this paper, we introduce the *ECOMatch*[3] approach for automatic configuration of ontology matching tools. Instead of letting the user directly manipulate the parameters

[3] **E**xample-based **C**onfiguration of **O**ntology **Match**ing tools

(which he often does not understand), we ask her to provide a set of example mappings (a task which can be done by a domain expert in a reasonable amount of time). We use those example mappings to test a number of different configurations and determine a good or even optimal parameter setting. That setting is then used to match the input ontologies and provides the final mapping.

The rest of this paper is structured as follows. In Sect. 2, we lay out the theoretical considerations for our work, and in Sect. 3, we discuss the implementation of ECO-Match. This implementation is the basis of various experimental evaluations, which are discussed in Sect. 4. We conclude with a summary and an outlook on future work.

2 Approach

A mapping between two ontologies which has been created by a domain expert user is called a *reference alignment*[4]. The goal of ontology matching tools is to produce a mapping which gets as close to a reference alignment as possible, i.e., which is as good as if a human expert would have created the mapping manually to achieve semantic interoperability.

For automatic tuning of ontology matching tools, we assume that the user is able to provide a set of example mappings. We call that set of examples a *partial reference alignment* between the target ontologies. We use this partial reference alignment to evaluate several configurations of the target matching tool. The configuration which has been evaluated best based on the partial reference alignment is then used to produce the final mapping.

To determine which of the tested configurations is the best one, we use the output produced by the matching tool when applying the respective configuration, and compute the result quality on the partial reference alignment, i.e., how well the partial reference alignment is reproduced by the matcher. Our assumption is that a configuration which reproduces the partial reference alignment well will also produce a high-quality overall ontology mapping.

For computing the result quality, we introduce the following measures for computing recall, precision, and f-measure on a partial mapping. Following Euzenat and Shvaiko [8], a mapping between two ontologies O_1 and O_2 can be defined as a set of 5-tuples of the form $\langle id, e_1, e_2, r, n\rangle$, where $e_1 \in O_1$ and $e_2 \in O_2$, and where r defines a type of relation (such as equality, subclass, etc.), and n depicts a confidence level provided by a matching tool. Therefore, given a reference alignment R, a partial reference alignment $R' \subseteq R$, and an alignment A computed by a matching tool, we define the partial alignment $A' \subseteq A$ as the subset of A which contains all elements in A which share at least one entity with an element in R':

Definition 1 (Partial Alignment).

$$\begin{aligned} A' := \{&\langle id, e_1, e_2, r, n\rangle \in A | \exists id', e_1', n' : \langle id', e_1', e_2, r, n'\rangle \in R'\} \\ \cup\ \{&\langle id, e_1, e_2, r, n\rangle \in A | \exists id', e_2', n' : \langle id', e_1, e_2', r, n'\rangle \in R'\} \end{aligned}$$

[4] The terms *mapping* and *alignment* are often used synonymously.

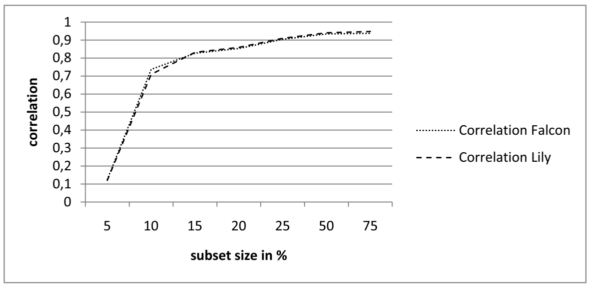


Fig. 1. Correlation between f-measure values of partial reference alignments with the f-measure value of the full reference alignment

	5	10	15	20	25	50	75
Falcon-AO	0.006	0.031	0.024	0.012	0.007	0.003	0.003
Lily	0.060	0.094	0.042	0.019	0.028	0.000	0.000

Table 1. T-test on correlations between f-measure values of partial and full reference alignments

Based on that definition, we can define a set of quality measures for an alignment A produced by a matching tool, which can be evaluated using a partial reference alignment:

Definition 2 (Precision on Partial Reference Alignment).

$$P'(A, R') := P(A', R') = \frac{|R' \cap A'|}{|A'|} \in [0, 1]$$

Definition 3 (Recall on Partial Reference Alignment).

$$R'(A, R') := R(A', R') = \frac{|R' \cap A'|}{|R'|} \in [0, 1]$$

Definition 4 (F-measure on Partial Reference Alignment).

$$F'(A, R') := M_{0.5}(A', R') = \frac{2 * P'(A,R') * R'(A,R')}{P'(A,R') + R'(A,R')} \in [0, 1]$$

It is particularly noteworthy that P', R', and F' can be computed only from the output of a matching tool and the partial reference alignment, without the need to know the complete reference alignment. To show that those measures are valid for assessing the result quality of a matching tool, we have analyzed the correlation of F and F', i.e., the f-measure computed on the full and on the partial reference alignment. This correlation is an indicator for the precision of the prediction of F by the means of F'.

To that end, we have used the two matching tools that we also used later on in our prototype (see Sect. 3), Falcon-AO and Lily, and 21 pairs of ontologies with reference alignments. We have run each tool with 100 random configurations and determined the f-measures F and F' for different sizes of partial reference alignments. The results are shown in Fig. 1. The key observation is that the results do not differ much between the individual matchers, and that there is a positive correlation, which becomes considerably high for partial reference alignments covering 10% or more of the reference alignment. Table 1 shows the confidence levels of the correlation, using a two-sample paired t-test. It reveals that for Falcon-AO, all the results are statistically significant, while for Lily, the correlation is only significant for partial reference alignment sizes of at least 15%.

These figures show that using partial reference alignments to score the configurations is reasonable: since a matcher configuration achieves a high f-measure value on a partial reference alignment it will most likely achieve a high f-measure value on the full reference alignment as well. With those considerations in mind, we have implemented a prototype to further analyze the potential of improving ontology matching by automatically configuring matchers based on example mappings.

3 Implementation

We have implemented our approach using a variety of algorithms for parameter optimization, as well as different matching tools. Fig. 2 shows the basic architecture of the system, based on the "classic" matching architecture proposed by Euzenat and Shvaiko [8]. The system takes as input two ontologies and a partial reference alignment, i.e., a set of known mappings.

The ECOMatch system itself treats the matcher it uses as a black box. For creating an optimized mapping, it performs the following steps:

1. Create an initial configuration for the matcher.
2. Run the matcher with that configuration on the pair of input ontologies and observe the mapping created.
3. Compute the f-measure F' based on the partial reference alignment.
4. Based on that resulting F', decide for a new configuration to try out.

Steps 2 to 4 are repeated until a stopping criterion – e.g., a fixed number of matcher runs or a time limit – is reached, or until the configuration generator decides not to proceed any further, e.g., because a search algorithm has reached a local optimum, or the whole search space has been covered. At that point, the mapping achieved with the configuration yielding the maximum F' is returned.

3.1 Configuration Generators

Besides a baseline algorithm which generates random configurations, we have included five metaheuristic algorithms for determining configurations from the large variety of parameter tuning approaches (see, e.g., [2] for a survey). As parameter optimization can also be regarded as a machine learning problem, we have furthermore implemented two machine learning algorithms.

Metaheuristics Metaheuristics are a family of stochastic optimization algorithms. They are especially practicable for problems for which it is hard to describe how to find an optimal solution, but which allow for easily assessing the quality of a given solution [17]. Metaheuristics subsequently create solutions and rate them according to a quality

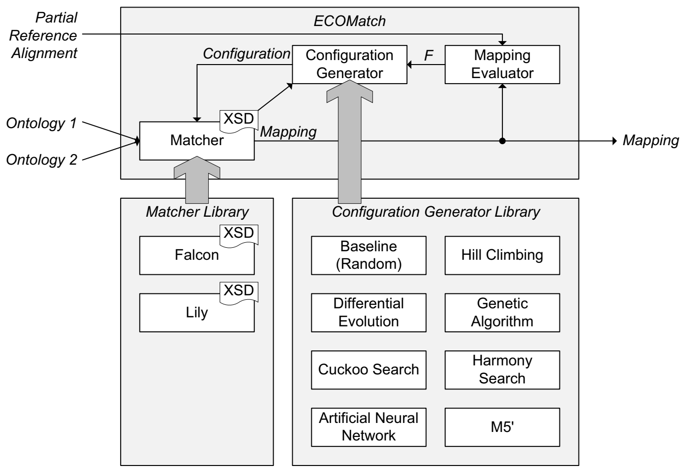


Fig. 2. Prototype architecture

function (in our case: the f-measure F' computed on the partial reference alignment). The rating determines which solution to examine next. Given that not every possible solution is examined, the optimal solution may not be found because it has not been examined at all. Thus, metaheuristics are usually used to solve optimization problem with a huge search space where it is hardly feasible to examine every possible solution [21]. In the prototype of *ECOMatch*, we have implemented the following metaheuristics: hill climbing [21], genetic algorithm [11], differential evolution [23], harmony search [10] and cuckoo search [26].

Machine Learning Techniques Machine learning methods are very widespread in various fields, e.g. in search engines, natural language processing or recommendation systems. While metaheuristics only examine a fixed set of candidate solutions, machine learning techniques can be used to train an explicit model, e.g., a decision tree or an artificial neural network, of the parameter space. This model can be used to predict the performance of other parameter combinations, which have not examined before. In *ECOMatch*, a small set of random parameter configurations is created and serves as training data. The employed methods M5', based on modeltrees [20], and artificial neural networks [9] each build a model which is used to predict the f-measure values F', called $F'_{predicted}$, for unseen random examples. For the configurations with best $F'_{predicted}$, the matching tool is run to determine the exact value F' and in turn the configuration with best F' is used to create the final mapping.

While determining the exact value F' using the matching tool is a costly operation, which takes minutes or even up to hours, calculating the value $F'_{predicted}$ using a learned model can be performed within milliseconds. Thus, a much larger amount of configurations can be examined using the trained model. To avoid negative effects due to wrong predictions, the predicted best configurations are double-checked using the matcher before taking them for producing a mapping.

3.2 Matching Systems

Besides different algorithms for parameter configuration, matching tools may be plugged into the *ECOMatch* framework. We have tested the framework with two matchers that performed very well in the last OAEI evaluations: Falcon-AO[5] [12] and Lily[6] [25]. While it is possible to run *ECOMatch* with every matching tool that provides a means to be run from the command line in batch mode, we have chosen those matchers because of their popularity and their performance in evaluations, as well as the provision of a sufficient size of the parameter set. Falcon-AO has 33 parameters in total, while Lily has eight. For our experiments with Falcon-AO, we have manually reduced the size of the parameter set to 26, discarding all parameters that do not have an effect on the result quality (e.g., parameters that only have an effect when processing large scale ontologies).

For each matching tool, we store a configuration description as an XML Schema Definition (XSD) file, which contains a list of the tool's parameters, as well as their respective types and ranges of values. This definition file represents the parameter space of the matching tool and is used by the configuration generator for creating the configurations.

4 Evaluation

To examine the impact of automatic parameter configuration on the result quality of ontology matching tools, we have conducted experiments on different data sets.

4.1 Evaluation Setup

For our evaluation, we have used all possible combinations of configuration generation algorithms and matchers, as described in Section 2.

We have used three different datasets for our evaluation:

- A subset of the OAEI benchmark dataset[7], which consists of the four non-artificial ontologies with reference alignments to one common ontology
- The OAEI conference dataset[8], consisting of seven ontologies and 21 reference alignments
- The six pairs of ontologies delivered with the FOAM ontology matching tool[9]

Altogether, we have tested the tool with 31 different pairs of ontologies and corresponding reference alignments.

To allow for comparison between the parameter optimization algorithms, we let each algorithm run the matching tool a certain amount of times depending on their

[5] Using the 2008 version, which can be downloaded from http://ws.nju.edu.cn/falcon-ao/index.jsp
[6] http://code.google.com/p/lilyontmap/
[7] http://oaei.ontologymatching.org/2010/benchmarks/index.html
[8] http://oaei.ontologymatching.org/2010/conference/index.html
[9] http://people.aifb.kit.edu/meh/foam/ontologies.htm

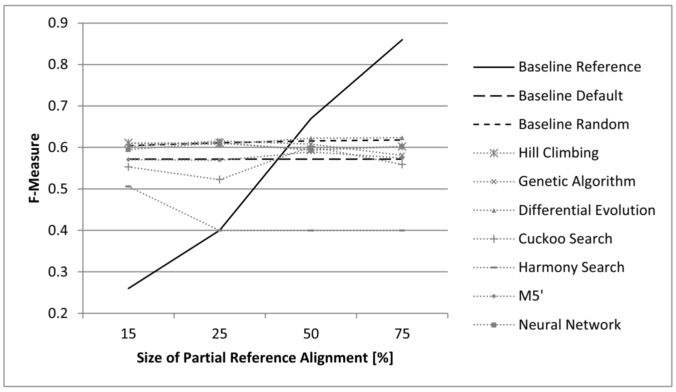


Fig. 3. Evaluations with Falcon-AO on the FOAM datasets

runtime, i.e. Falcon-AO 250 times and Lily 50 times (since running the matching tool is the most time consuming step). The machine learning approaches used 200 (Falcon-AO) resp. 40 (Lily) random configurations for training the respective models, 10,000 randomly generated configurations were rated by the trained model. Out of those, the 50 resp. 10 best-rated configurations were re-evaluated by running the matching tool with them. For each experiment, the average of three runs has been taken into the evaluation to reduce the impact of random outliers.

In order to show the value of our approach, we have tested against three baselines. The first baseline is the result quality achieved with default configuration of each tool. The second baseline is to choose the best out of 250 resp. 50 random configurations (since we let each algorithm perform 250 resp. 50 runs of the matcher). The third baseline is the result quality which would be achieved if the partial reference alignment would be returned as is, without running any matcher. For showing that parameter optimization based on example mappings has any value, it should at least perform better than the default configuration and the use of the partial reference alignment as is. A sophisticated approach for parameter optimization should outperform all three baselines, including the use of randomly generated configurations.

4.2 Evaluation Results

Running the evaluations leads to mixed results. For Falcon-AO, the default configuration proves to be rather robust and yield better results on most data sets than configurations found with any other approach. Figure 3 shows the average results for the FOAM datasets, which were the only ones where an optimized configuration could perform better than the default configuration.

The results reveal that most approaches, except for differential evolution, perform slightly worse than the random baseline. This is most likely due to the large size and dimensionality of the search space, combined with the restriction of the number of runs of the matching tool, which in the end make the search algorithms terminate early (and possibly too early to search the parameter space far enough for yielding good results).

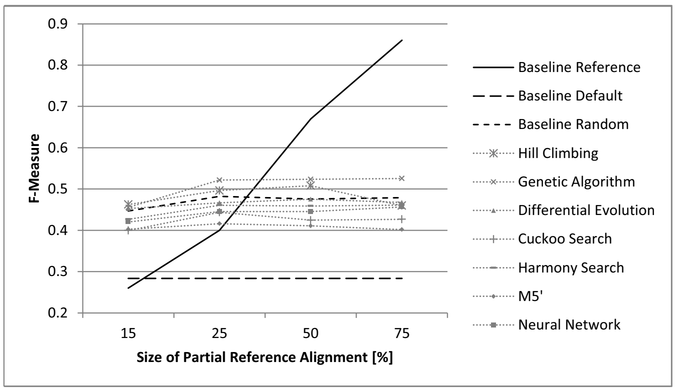


Fig. 4. Evaluations with Lily on the conference datasets

With Lily, the improvements that could be achieved over the standard configuration were more significant. Figure 4 depicts the average results achieved on the conference dataset. Again, a very good performance of the random baseline can be observed, with evolutionary algorithms performing slightly better. As for Falcon-AO, a possible explanation is the large search space which has to be explored with a low number of matcher runs for the optimization.

In all cases, the results achieved with partial reference alignments of up to 25% outperformed the baseline of using the partial reference alignment as is. Thus, the value gained from the partial reference alignment is higher than the effort that has to be used for producing that alignment. Partial reference alignments larger than 25% serve better as final mappings, rather than using them as input for parameter optimization.

The results achieved with both matchers also reveal that the size of the partial reference alignment does not significantly influence the result quality. This is a very encouraging result, as the provision of example mappings may be a crucial bottleneck for the *ECOMatch* approach. The observation that the result quality already significantly increases with a partial reference alignment of 15% (and probably even with smaller sizes) demonstrates that the *ECOMatch* approach is not only feasible in theory, but may also be practically applied.

When looking more closely at the parameter configurations that are created by the different approaches, there are several observations that can be made. We have compared those configurations both with each other as well as with the respective tool's default configuration. The first observation is that approaches perform better if they create more different configurations. This also explains the good performance of the random baseline, which ensures a maximum entropy of configurations. The second observation is that the best configurations that are found by the algorithms are rather close (although not identical) to the default configuration.

In summary, some optimization approaches, such as cuckoo search or harmony search, do not perform very well on that problem, while significant improvements of the resulting mapping quality are possible with a suitable approach, even when the set of examples is not too large.

5 Related Work

Despite the large body of work in the field of ontology matching and its predecessor, schema matching, there is little work done which is focused on automatic or semi-automatic parameterization of matching tools.

eTuner [16] is a tool which is directed at the parametrization for schema matching techniques and their combination. For evaluating different settings, eTuner generates synthetic pairs of schemes from the input schemes using predefined rules (similar to our approach using the result quality on a partial reference alignment as an approximation).

A problem closely related to parameter tuning of matching tools is selecting a suitable matcher – or a suitable combination of matchers – for a specific matching task. Such a combination can be described by a set of weights for each matcher, as done, e.g., with the matching framework *CROSI* [14] and can be seen as a special case of the parameter tuning problem. One work uses meta-level learning in order to find the best ensemble of matchers, as discussed by Eckert et al. [3]. Other strategies are based on Bayesian Networks [24] or Support Vector Machines [13]. Mochol et al. [18] propose an approach to find the best matcher to match two specified ontologies from a set of matchers.

Many matching tools like the systems QOM [5] or PROMPT [19] support semi-automatic matching processes, e.g. by presenting the user suggestions or potential problems. They typically use the examples provided by the user as anchors for searching for new mappings, rather than for tuning the parameters of the underlying matching algorithms. A very related idea which also takes user support into account is *APFEL* [4]. It tries to improve the alignment by involving the user into the matching process. It first creates potential correspondences based on initial settings, presents them to the user, the user marks the correct ones which serve as training set for the machine learning approach. The whole process is repeated to gradually improve the alignment.

In our approach, we have used partial reference alignments provided by a domain expert to optimize different matching tools. The matching tool *SAMBO* [15] also uses partial reference alignments for various purposes. They are used as anchors to give hints for partitioning larger ontologies in a preprocessing step, as well as for filtering potential incorrect results in a post-processing step. A schema matching system exploiting examples and applying machine learning methods is *LSD* [1]. The approach defines the matching problem as a classification problem, where the class to predict for an element of a source schema is the corresponding target schema element. Thus, a machine learning algorithm is trained using correspondences provided by the user.

The key differences between those approaches and *ECOMatch* are that existing approaches typically work in dialog mode and require constant user attention, while our approach works in batch mode once the user examples are given. Futhermore, *ECOMatch* is not restricted to a particular matching algorithm, but can be used with any existing matching tool.

6 Conclusion and Outlook

In this paper, we have presented the *ECOMatch* approach for automatically parameterizing ontology matching tools based on example mappings. Since we could observe a significant correlation of the result quality, it is possible to test different matcher configurations against a set of example mappings, and thereby find a good or even an optimal matcher configuration. ECOMatch treats the matcher it uses as a black box; thus, our approach can be easily adopted by developers of other matching tools or used as a generic framework for optimizing ontology mappings.

Since it is often not feasible to test every possible combination of parameter settings, we have used a number of heuristics for determining good, near-optimal parameter settings. We have tested our approach with two state-of-the-art matching tools and a set of benchmark ontologies. The results show that parameter optimization based on example mappings can help providing significant better results than using the default configuration of matching tools, and is a good alternative to manually trying to find a good configuration, since no understanding of the parameters is required.

The two main bottlenecks of our approach are the provisioning of examples (which typically means manual work for a domain expert), and the running of matching tools (which is a costly operation that drastically increases the overall processing time). These two bottlenecks point at the main opportunities for improving the ECOMatch system: reducing the amount of examples required, and reducing the number of required matcher runs.

In our experiments, we have used a random subset of a gold standard mapping for emulating a domain expert creating example mappings. This may not be entirely correct, as such an expert may first determine a set of *obvious* mappings, which may not be a representative random sample of the full mapping. Thus, we want to further evaluate how the selection of the mapping subset influences our approach. To reduce the workload for the domain expert providing the partial mapping, we would also like to explore how the size of the partial mapping can be minimized, e.g., in an interactive mode where the tool interrogates a domain expert and tries to determine mappings which efficiently divide the search space. As negative examples may be retrieved with less work than positive ones, we also want to explore how a combination of positive and negative examples can be used to find an optimal mapping.

Most matchers are tuned for achieving a good f-measure, i.e., a good trade-off between precision and recall. However, it is possible to develop matchers with high precision at the cost of worse recall. Such a matcher could be used as a generator for the example mappings, so that the whole tuning process of the target matcher could even be fully automatized in the future. In the context of ontology matching for linked open data, other mechanisms for obtaining the example mappings are also possible, such as a community creating a set of example mappings (the same as they are creating links between instances) [27], or guessing example mappings for classes of different ontologies that share a lot of common instances.

So far, we have only investigated the quality of the generated configurations *after* a fixed number of runs of the matching tool. In order to speed up the whole process, it would be interesting to look at the gradient, i.e., the time it takes for a certain optimization approach to find a good configuration. This would allow for a more sophisticated comparison of the individual strategies for finding an optimum configuration. Furthermore, our experiments have shown that most of the good configurations found by *ECOMatch* are similar, yet not identical, to the default configuration. This insight may also help improving the process of searching for a good configuration, e.g., when creating a starting population for a genetic algorithm.

In summary, the work presented in this paper has addressed one of the top ten challenges in ontology matching. We have proposed a solution that is suitable for domain experts with low skills in ontology matching, since we only rely on a set of example mappings provided by a domain expert. The results show that matcher configurations can be automatically improved based on example mappings, which makes this approach a promising research direction for future ontology matching tools.

Acknowledgements

The work presented in this paper has been partly funded by the German Federal Ministry of Education and Research under grant no. 01ISO7009. Dominqiue Ritze is supported by a Karl-Steinbuch scholarship of the MFG foundation Baden-Württemberg. The authors would like to thank Heiner Stuckenschmidt, Christian Meilicke, and Johannes Fürnkranz for their valuable feedback.

References

1. A. Doan, P. Domingos, and A. Halevy. Reconciling schemas of disparate data sources: A machine-learning approach. In *Proceedings of the 2001 ACM SIGMOD international conference on Management of data*, pages 509–520, 2001.
2. Felix Dobslaw. Automatic parameter tuning for metaheuristics, 2010. Accessed June 17th, 2011.
3. Kai Eckert, Christian Meilicke, and Heiner Stuckenschmidt. Improving ontology matching using meta-level learning. In *Proceedings of the 6th European Semantic Web Conference (ESWC)*, pages 158–172, 2009.
4. M. Ehrig, S. Staab, and Y. Sure. Bootstrapping ontology alignment methods with apfel. In *Proceedings of the 4th International Semantic Web Conference (ISWC)*, pages 186–200, 2005.
5. Marc Ehrig and Steffen Staab. Qom - quick ontology mapping. In *The Semantic Web - ISWC 2004*, volume 3298, pages 683–697. Springer, 2004.
6. Jérôme Euzenat, Alfio Ferrara, Laura Hollink, Antoine Isaac, Cliff Joslyn, Véronique Malaisé, Christian Meilicke, Andriy Nikolov, Juan Pane, Marta Sabou, François Scharffe, Pavel Shvaiko, Vassilis Spiliopoulos, Heiner Stuckenschmidt, Ondrej Sváb-Zamazal, Vojtech Svátek, Cássia Trojahn dos Santos, George A. Vouros, and Shenghui Wang. Results of the Ontology Alignment Evaluation Initiative 2009. In *Proceedings of the 4th International Workshop on Ontology Matching (OM-2009)*, volume 551 of *CEUR-WS*, 2009.
7. Jérôme Euzenat, Alfio Ferrara, Christian Meilicke, Juan Pane, François Scharffe, Pavel Shvaiko, Heiner Stuckenschmidt, Ondrej Sváb-Zamazal, Vojtech Svátek, and Cássia Trojahn. Results of the Ontology Alignment Evaluation Initiative 2010. In *Proceedings of the Fifth International Workshop on Ontology Matching (OM-2010)*, volume 689 of *CEUR-WS*, 2010.

8. Jérôme Euzenat and Pavel Shvaiko. *Ontology Matching*. Springer, Berlin, Heidelberg, New York, 2007.
9. M.W. Gardner and S.R. Dorling. Artificial neural networks (the multilayer perceptron) - a review of applications in the atmospheric sciences. *Atmospheric Environment*, 32(14–15):2627–2636, 1998.
10. Zong Woo Geem, Joong-Hoon Kim, and G. V. Loganathan. A new heuristic optimization algorithm: Harmony search. *Simulation*, 76(2):60–68, 2001.
11. John H. Holland. *Adaptation in Natural and Artificial Systems*. University of Michigan Press, 1975.
12. Wei Hu and Yuzhong Qu. Falcon-AO: A practical ontology matching system. *Journal of Web Semantics*, 6(3), 2008.
13. Ryutaro Ichise. Machine learning approach for ontology mapping using multiple concept similarity measures. In *Proceedings of the 7th International Conference on Computer and Information Science (ICIS)*, pages 340–346, 2008.
14. Yannis Kalfoglou and Bo Hu. CROSI Mapping System (CMS) - Result of the 2005 Ontology Alignment Contest. In *Integrating Ontologies '05, Proceedings of the K-CAP 2005 Workshop on Integrating Ontologies, Banff, Canada, October 2, 2005*, 2005.
15. Patrick Lambrix and Qiang Liu. Using partial reference alignments to align ontologies. In *Proceedings of the 6th European Semantic Web Conference (ESWC)*, pages 188–202, 2009.
16. Yoonkyong Lee, Mayssam Sayyadian, AnHai Doan, and Arnon S. Rosenthal. eTuner: tuning schema matching software using synthetic scenarios. *VLDB Journal: Very Large Data Bases*, 16(1):97–122, 2007.
17. Sean Luke. *Essentials of Metaheuristics*. Lulu, 2009.
18. Malgorzata Mochol, Anja Jentzsch, and Jérôme Euzenat. Applying an analytic method for matching approach selection. In *Proceedings of the 1st International Workshop on Ontology Matching (OM-2006)*, 2006.
19. Natalya F. Noy and Mark A. Musen. The PROMPT suite: interactive tools for ontology merging and mapping. *International Journal of Human-Computer Studies*, 59(6):983–1024, 2003.
20. J. R. Quinlan. Learning with continuous classes. In *Proceedings of the 5th Australian Joint Conference on Artificial Intelligence*, pages 343–348, 1992.
21. S. Russel and P. Norvig. *Artificial Intelligence: a Modern Approach*. Prentice-Hall, 1995.
22. Pavel Shvaiko and Jérôme Euzenat. Ten Challenges for Ontology Matching. In *On the Move to Meaningful Internet Systems: OTM 2008*, volume 5332 of *LNCS*, pages 1164–182. Springer, 2008.
23. R. Storn and K. Price. Differential evolution - a simple and efficient heuristic for global optimization over continuous spaces. *Journal of global optimization*, 11:341–359, 1997.
24. Ondrej Sváb and Vojtech Svátek. Combining ontology mapping methods using bayesian networks. In *Proceedings of the 1st International Workshop on Ontology Matching (OM)*, volume 225, pages 206–210, 2006.
25. Peng Wang and Baowen Xu. Lily: Ontology Alignment Results for OAEI 2009. In *Proceedings of the 4th International Workshop on Ontology Matching (OM-2009)*, 2009.
26. X.-S. Yang and S. Deb. Cuckoo search via lévy flights. In *Proceedings of World Congress on Nature and Biologically Inspired Computing (NaBIC 2009)*, pages 210–214, 2009.
27. Anna V. Zhdanova and Pavel Shvaiko. Community-driven ontology matching. In *Proceedings of the 3rd European Semantic Web Conference (ESWC)*, pages 34–49, 2006.

Evolution of the COMA Match System

Sabine Massmann, Salvatore Raunich, David Aumüller, Patrick Arnold, Erhard Rahm

WDI Lab, Institute of Computer Science
University of Leipzig, Leipzig, Germany
`{lastname}@informatik.uni-leipzig.de`
`http://wdilab.uni-leipzig.de`

Abstract. The schema and ontology matching systems COMA and COMA++ are widely used in the community as a basis for comparison of new match approaches. We give an overview of the evolution of COMA during the last decade. In particular we discuss lessons learned on strong points and remaining weaknesses. Furthermore, we outline the design and functionality of the upcoming COMA 3.0.

1 Introduction

Schema and ontology matching is the process of automatically deriving correspondences between the elements or concepts of two or more data models, such as XML schemas or formal ontologies. Computed correspondences typically need to be validated and corrected by users to achieve the correct match mappings. Match mappings are needed in many areas, in particular for data integration, data exchange, or to support schema and ontology evolution. Hence, match mappings are the input of many algorithms in these domains, e. g. to determine executable data transformation mappings or to perform ontology merging.

In the last decade, there has been a huge amount of research on schema and ontology matching and on mapping-based metadata management in general. Overviews of the current state-of-the-art are provided in two books [5, 13]. Many dozens of research prototypes have been developed and at least simple (linguistic) automatic match approaches found their way into commercial mapping tools [23]. COMA (Combining Matchers) is one of the first generic schema matching tools. Its development started about ten years ago at the University of Leipzig and is still going on. COMA and its successor COMA++ have been made available to other researchers and have been widely used as a comparison basis for new match approaches.

This paper reflects on the evolution of COMA during the last decade and reports on the major lessons learned. We also describe the design and functionality of a new version, dubbed COMA 3.0. In the next section, we give a short overview of the evolution of COMA and COMA++. We then discuss lessons learned in Section 3 by outlining strong points and remaining weaknesses. The new version of COMA is described in Section 4.

2 System Evolution

Key stations and publications concerning development and use of COMA are as follows:

2002 Initial release and publication of COMA paper at VLDB in Hong Kong [9]. Support for multi-matcher architecture and reuse of previous match mappings.

2003/04 Further evaluations [8] and initial design to support large XML schemas [25].

2005 Release and SIGMOD publication of COMA++ [2]. Support for ontology matching, GUI and fragment matching. Ph.D. Thesis of Hong Hai Do [7].

2006 Support for instance-based matching. Participation at OAEI contest [17].

2007/08 Further evaluations with larger and more diverse models, including web directories [10, 18]. Use of COMA++ within the QuickMig project [11]. Web edition.

2010/11 Redesign and development of COMA 3.0.

The development of COMA started in 2001 and was influenced by the findings and recommendations of the survey article [24] on schema matching. In contrast to most previous approaches at the time, COMA provides a generic approach to support matching of different kinds of schemas (in particular relational and XML schemas) and for different application domains. This is made possible by representing all kinds of schemas by a generic in-memory graph representation on which matching takes place. A key feature of COMA is the flexible support for multiple independently executable matchers that can be executed within a user-controlled match process or workflow. More than ten schema-based matchers were initially supported mainly based on the linguistic and structural similarity of elements. All matchers are evaluated on the Cartesian product of elements from the two input schemas, where each element is represented by a path to the schema root. Furthermore, different approaches to combine matcher results and select correspondences are provided. A unique feature of the initial COMA design was the support for reusing previous match mappings, especially the possibility to compose several existing mappings stored in a repository. An extensive evaluation on XML schemas showed that the combination of several matchers clearly outperforms single matchers.

The initial evaluation used relatively small schemas of less than 100 nodes. Hence, further evaluations and adjustments focused on larger schemas, especially for e-business schemas containing shared schema fragments (e. g. for address information) [25]. Several extensions to deal with large schemas were designed and integrated within the next major release of the prototype, called COMA++. COMA++ was introduced in 2005 [2] and represents a major re-implementation of the original COMA design to improve both performance and functionality. It provides a GUI to simplify the definition of match strategies and to correct computed match correspondences. It also supports matching of ontologies, especially OWL ontologies. COMA++ provides additional matchers and operators like merge and diff for post-processing of match mappings.

Several approaches in COMA++ facilitate the matching of larger schemas, in particular to avoid the evaluation of the Cartesian product of schema elements. First, fragment matching is supported that implements one of the first divide-and-conquer approaches where only similar schema fragments need to be matched with each other. Secondly, sequential execution of matchers (or mapping refinement) is supported so that a fast matcher can be executed first and more expensive matchers are only evaluated on more similar pairs of elements. In particular, a strategy called *FilteredContext* performs first matching only for nodes and restricts the evaluation of paths (i. e. node contexts) to the more similar node pairs. A detailed description and evaluation of these features can be found in [7, 10].

In 2006, two instance matchers were added to COMA++ to prepare the system for participation in the OAEI contest[1] that provides instances for its basic benchmark test cases. Sets of instances are associated to schema elements. One approach is to compare individual instance values with each other and aggregate the similarity values per element pair. Alternatively, all instances are combined within a virtual document and a TF/IDF-like document similarity is determined for element pairs (similar approaches are used in other match systems, e. g. RiMOM [16]). The instance-based matchers and their combination with metadata-based matchers were successfully evaluated on the ontologies of the OAEI benchmark [12] and on web directory taxonomies [18].

In a joint project with SAP, COMA++ was used as the basis for developing mappings for data migration [11]. Furthermore, we created a web edition of COMA++ to support matching without local installation of the tool and its underlying DBMS (MySQL). For evaluation, we also added to COMA++ some advanced approaches for aggregating similarity values of different matchers [22] as being used in other match systems such as Prior+ or OpenII Harmony. Since 2010, we partially redesigned and extended COMA++ as we will describe in Section 4.

Due to numerous requests, we made the binary version of COMA/COMA++ available for free to other researchers. Hundreds of researchers world-wide downloaded the prototype for use and comparison with their own matching approaches.

3 Lessons Learned

Working on and with a system with such a longevity as COMA, there are both positive and negative lessons learned. This section presents both sides.

3.1 Strong Points

In retrospect, we believe that many design decisions of COMA and COMA++ have been right. Several of them have also been adopted by later match systems. The positive points include the following:

Multi-matcher architecture Supporting many diverse matcher algorithms that can be combined within match workflows is key to obtain sufficient match quality for different match tasks. Almost all recent match systems follow such a design with support for linguistic, structural, and instance-based matchers.

Generic approach The generic representation of models as rooted, directed acyclic graphs allowed us to apply all matchers and combination approaches to diverse kinds of schemas and ontologies. By providing the respective parsers we could thus easily extend the scope of COMA to different domains and models. This flexibility also contributed to the popularity of COMA for other researchers.

[1] http://oaei.ontologymatching.org/

Effective default configuration COMA and COMA++ provide a default match configuration that can be used without manual tuning effort. The default configuration was determined for a set of XML schema match tasks and consists of four linguistic and structural matchers and a specific combination approach [10]. The default combination approach is based on taking the average matcher similarity per element pair and applying a so-called *Delta* selection returning the match candidates with the highest match similarity (above a threshold) as well as all further candidates within a small distance (delta) of the top candidate. For improved precision, a stable marriage-like selection is further applied (called *Both*) by default. The default strategy turned out to be surprisingly effective over a wide range of further match problems evaluated by other researchers, including for matching web directories [3], for n-way (holistic) schema matching [14] and even for matching of UML meta-models [15]. An evaluation of different methods to combine matcher results [22] showed that advanced approaches can perform well only in specific cases and are not as robust as simply determining the average matcher similarity applied by default in COMA++.

GUI The graphical user interface in COMA++ significantly improves the usability compared to the original COMA implementation, in particular for importing and visualizing schemas, configuring match strategies, and inspecting and correcting match mappings. Furthermore, it allows a simplified evaluation of different match strategies.

Customizability Even when using the default match strategy, it is possible to customize linguistic matching by providing domain-specific dictionaries, in particular synonym and abbreviation lists. These simple lists can significantly improve match quality and are an effective approach to leverage and reuse background knowledge. Particularly, they avoid the risk of many wrong match candidates (poor precision) when using general-purpose dictionaries such as Wordnet. There are many more possibilities to customize the match strategy, in particular the selection of promising matchers. For example, instance matching can be selected if instances are available for matching.

Advanced match strategies While not part of the default match strategy, COMA++ supports several advanced match strategies that are especially helpful to deal with large schemas and ontologies. These strategies include the reuse of previous match results, fragment matching, as well as mapping refinements such as in the *FilteredContext* strategy. While the approaches have been quite effective, there are still opportunities for improvement, e. g. for a more comprehensive reuse and for schema partitioning as discussed in [23].

Repository We store all imported schemas and ontologies as well as confirmed mappings in a repository for later inspection and reuse. The repository avoids the repeated import and matching of the same schemas and is a prerequisite for the reuse matchers.

3.2 Weak Points

Given the broad need for schema and ontology matching and increasing demands w. r. t. the size of match tasks and the use of mappings, we also encountered a set of difficulties with our match system asking for improvements. Most current match systems share similar limitations although some of the associated challenges have already been addressed in recent research [28].

Scalability issues The number of paths is generally much higher than the number of nodes per schema so that COMA's path-based matching leads to memory and runtime problems for large match tasks with millions of path pairs to evaluate in the Cartesian product. Memory problems are primarily caused by storing the similarity values for all matchers and path pairs in memory in addition to the schema graphs. These problems can be alleviated to some degree by applying fragment matching and node-based matching but a more general solution with reduced memory requirements is needed. Furthermore, additional performance techniques such as parallel matching on several processors are desirable.

Configuration effort While the default match strategy is often effective it cannot guarantee the best match quality and runtime performance for all match tasks. Finding better match strategies, however, is difficult even for experts due to the high flexibility that comes with the possibility to select from many matchers and combination options and having to find suitable parameter settings. Therefore, the system should help users to choose a match strategy based on an automatic analysis of the match task to solve.

Limited semantics of match mappings The system only determines match mappings consisting of simple equality correspondences between elements of schemas or ontologies. More expressive mappings are desirable supporting additional kinds of relationships such as containment or is-a relationships. Furthermore, applications such as data exchange or schema evolution need executable mappings that can be applied to instance data.

Limited accessibility COMA++ is a stand-alone tool designed for interactive use, not for use by programs. To improve the accessibility, the provision of an API, e. g. based on web services is desirable. The web edition of COMA++ is not widely used and a redesigned browser-based GUI would be attractive.

4 COMA 3.0

Since 2010 we have partially redesigned and extended COMA++ at the Web Data Integration Lab (WDI Lab) of the University of Leipzig. The mission of the WDI Lab is the development of advanced data integration tools and approaches that can be used for practical applications. Major work areas include support for schema and ontology integration, entity resolution as well as mashup-like data integration workflows.

In this section, we outline the new version of our match tool called COMA 3.0 that includes support for enriched mappings and ontology merging. We start with an overview of the architecture and functionality of COMA 3.0. We then discuss the mapping enrichment and ontology merging components and present preliminary evaluation results. COMA 3.0 will be made available in 2012. We plan to provide two versions: a community version as open source and a professional version to be distributed by a WDI Lab spinoff.

4.1 Overview

The revised architecture of COMA is shown in Figure 1. Several components are analogous to COMA++, including *Import* of schemas and ontologies, auxiliary information and mappings, an *Export* to output models and mappings, and a *Repository* where these information as well as match configurations are kept. Furthermore, there is an *Execution Engine* to execute defined match strategies using matchers and preprocessing and postprocessing steps from different libraries. The following components are largely new:

Configuration Engine which supports both a manual configuration (*expert mode*) as well as an initial kind of automatic configuration (*simple mode*) and validation of workflow configurations. The automatic configuration uses the default match strategy as a starting point and decides about which matchers to add or drop and how to change the approach for combining matcher results. For this purpose, the input schemas/ontologies are checked for the availability of instances, comments, and diverse data types to determine whether the matchers requiring these features can be used at all. Furthermore, we test whether the reuse of previously determined match mappings is applicable. Similar to other systems [16], we also calculate the degree of linguistic and structural similarity of the input models to decide about the use of linguistic and structural matchers. We also check the input sizes to decide about the use of fragment matching.

Enrichment Engine which supports a semi-automatic or manual enrichment of simple 1:1 correspondences into more complex mapping expressions including functions, e. g. to support data transformations. Furthermore, is-a and inverse is-a correspondences between ontologies can be determined as input for ontology merging. More details will be described in subsection 4.2.

Merge and Transformation Engines which support match-driven ontology merging (see subsection 4.3), as well as the generation of executable mappings (queries) for data transformation. The latter functionality is based on the approaches developed for the +Spicy mapping system [20].

User Interfaces. A new GUI as well as programmatic access to the COMA functionality is planned by offering an API and web service interfaces.

We retain virtually all matchers and advanced strategies (*reuse matching, fragment matching, FilteredContext*) of COMA++ and add some new ones. We implemented an additional kind of fragment matching for large schemas/ontologies where fragments are automatically identified by a clustering algorithm based on the structural similarity within schemas/ontologies. The approach is described and evaluated in [1]. For matching life science ontologies, we added a domain-specific matcher called `NameSyn` that exploits the frequent availability of name synonyms for concepts. It therefore considers two concepts to match if either their names or one of their synonyms are highly similar.

There are numerous implementation enhancements to improve performance and scalability. Since the matcher results (similarity matrices) can contain a very large number of similarity values we support their storage either in memory or in a database table. Both implementations support the same interface and are thus internally usable in the same way. Linguistic matching is now based on a set of optimized and more versatile string matchers (Trigram, Jaccard, Levensthein, TF/IDF, etc.) that have been devised in

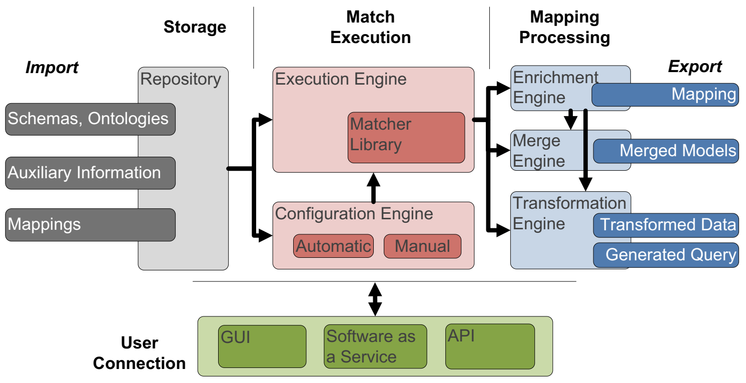


Fig. 1. Architecture of COMA 3.0

the WDI Lab for both entity resolution and schema/ontology matching. In particular the input and output of these matchers can either be in memory, in a file or in a database. Common optimizations of the string matchers include a preprocessing of input strings (stop word removal, tokenization, resolution of abbreviations/synonyms etc.) as well as an early pruning of highly unsimilar pairs of strings.

As mentioned, we plan to make the COMA functionality available as a web service (SaaS). The goal is to broaden the usability of COMA to diverse applications thereby reducing the need to repeatedly re-implement match functionality. Possible consumers include mashups or mobile applications as well as match-driven tools, e. g. for data migration or ontology merging. Furthermore, the match functionality can be utilized within new graphical user interfaces, e. g. a browser-based light-weight GUI for match processing. The core functionality to be provided as web service includes the possibility to load schemas and ontologies, to specify and execute a match strategy and to return computed match mappings. A challenge still to be addressed is support for multi-tenancy so that the data and execution of many users can be isolated from each other.

4.2 Enrichment Engine

The match process of COMA identifies a set of 1:1 equality match correspondences between elements of the input schemas or ontologies. While elements may participate in multiple such correspondences, there is no direct support for complex correspondences interrelating several elements per input schema. For example, there may be correspondences *Name–FirstName* and *Name–LastName* that should in fact be a complex correspondence *Name* – {*FirstName, LastName*}. One task of the enrichment engine is to semi-automatically determine such complex correspondences and extend them with data transformation functions, e. g. to specify that a split of *Name* should be applied to obtain the *FirstName* and *LastName* elements. The second task of the enrichment engine is the derivation of more semantic correspondences for ontologies, in particular is-a and their inverse is-a relationships that can be utilized for ontology merging.

We have extended the mapping structure of COMA to support complex correspondences with data transformation functions. We support similar functions as in commercial

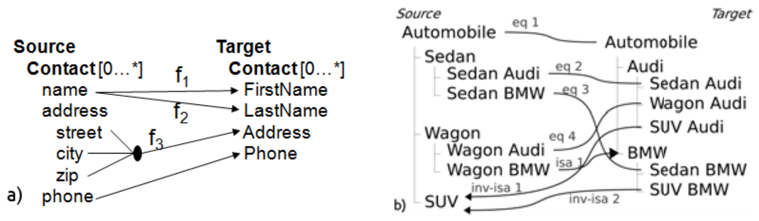


Fig. 2. a) A Mapping Scenario from STBenchmark – b) A Merging Scenario

mapping tools such as BizTalk Server[2], in particular numerical functions (e. g. sum, multiply), string functions (e. g. concat, lowercase, replace), date/time functions, conversion functions, logical functions, relational functions, etc. The semi-automatic determination of complex correspondences and mapping expressions is very complex and only few research approaches have been proposed so far, e. g. iMAP [6].

To find complex correspondences we analyze the set of 1:1 correspondences. In particular we check whether sets of correspondences involving the same source or target element should be combined within a complex correspondence, e. g. if they refer to elements in close proximity such as in the mentioned *Name* example. As usual, all suggestions are subject to user feedback for confirmation or correction.

Our approach to identify applicable transformation functions depends on instance data but also uses element names and structural information. For all correspondences we test and verify the applicability of the type-specific transformation functions. For illustration, consider the example from the STBenchmark[3] shown in Figure 2(a), where three functions are needed to correctly transform source instances into target instances. In particular, f_1 and f_2 describe how to split the source element *name* into the two target elements *FirstName* and *LastName*; similarly, the concatenation function f_3 defines how to combine the three elements *street, city, zip* into the target element *Address*. Our approach is able to find out, for example, the last complex correspondence by starting from the 1:1-correspondence *address–Address* automatically detected by the system and performing a structural analysis of these two elements. Since data is not stored in the inner node *address* but in its children, a complex correspondence will be created out of the subnodes. Furthermore, analyzing instance data is helpful to determine in which order the source elements should be concatenated by f_3.

Besides discovering complex matches, the enrichment engine supports the determination of semantic correspondences between ontologies, in particular is-a and inverse is-a relationships, that can be used for ontology merging. We provide a basic algorithm based on the analysis of labels that can detect textual containment relationships between concepts; this approach can be extended using external linguistic oracles (e. g. WordNet) providing semantic relationship indicators. For illustration, consider the simple example in Figure 2(b) where the catalog of a new online car shop (source) should be merged into the catalog of a price comparison portal (target). Analyzing the labels of the input con-

[2] http://www.microsoft.com/biztalk
[3] http://www.stbenchmark.org/

cepts, we find that the source label *Wagon BMW* “is contained” in the target label *BMW* and then derive that *Wagon BMW “is a” BMW*; similarly for concepts *SUV, SUV Audi* and *SUV BMW*, but in the opposite direction, identifying the inverse-is-a relationships shown in Figure 2(b).

A related approach for semantic matching is supported by S-Match [14] where different matchers are used to discover semantic relations, like equivalence, less general, more general and disjointness between concepts. The less and more general relationships correspond to is-a and inverse-is-a relationships in our approach.

4.3 Ontology Merging

A major extension of COMA 3.0 is the inclusion of an ontology merging component that consumes the match mapping as input and produces an integrated ontology as output, called merged ontology. The main approach is called AUTOMATIC TARGET-DRIVEN ONTOLOGY MERGING (ATOM) and is described in more detail in [27]. The current version is restricted to is-a taxonomies; multiple inheritance and instance data for leaf concepts are supported.

Ontology merging is a difficult problem since there often exists no ideal unique solution. Previous ontology merge approaches are largely user-controlled and do not clearly separate matching from merging resulting in complex approaches, [4], [21], [19], [29]. By utilizing a manually verified match mapping as input, our merge approach is largely automatic. We support two kinds of automatic ontology merging: a *symmetric* or *full merge* as well as an *asymmetric*, *target-driven* merge.

The symmetric approach fully preserves both input ontologies, combining equivalent concepts and maintaining all remaining concepts and relationships of both input ontologies. The main problem of such a full merge result is that maintaining different organizations of the same information can reduce its understandability and introduce multiple inheritance and semantic overlap.

As an alternative we therefore support the new asymmetric ATOM approach that merges the source (first) input ontology into the target (second) input ontology. It thus gives preference to the target ontology and preserves it fully while redundant source concepts and relationships might be dropped. We find that such an asymmetric merge is highly relevant in practice and allows us to incrementally extend the target ontology by additional source ontologies. For example, the product catalog of a merchant may have to be merged as a new source into the existing catalog of a price comparison portal.

Figure 3 shows a COMA 3.0 screenshot on the use of ATOM for merging the ontologies of Figure 2(b). The merge result is shown in the middle. The lines specify mappings between the input ontologies and the merged ontology that are also automatically determined and that can be used to migrate instances. If ATOM is provided with semantic correspondences as discussed in the previous Section, the merge result can be improved by finding a better placement of concepts. The screenshot in Figure 3 shows already the outcome when using the correspondences labeled isa_1, $inv\text{-}isa_1$ and $inv\text{-}isa_2$ in Figure 2(b). These correspondences could be used, for example, to place the concept *Wagon BMW* as a subclass of the *BMW* concept, which would not have been possible with equivalence correspondences alone. More details on our merging approach and its evaluation can be found in [26] and [27].

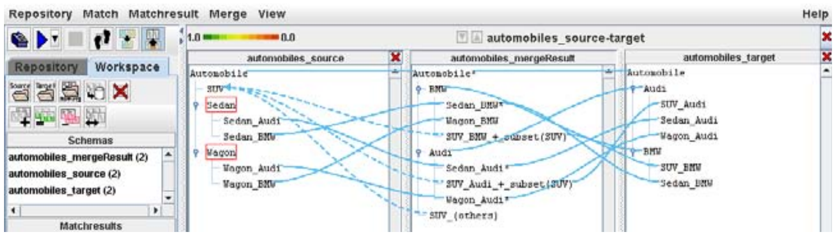


Fig. 3. A merging scenario in COMA 3.0

4.4 Preliminary Evaluation

We have just started to evaluate COMA 3.0 and can therefore discuss only some preliminary results. We report some results for the OAEI Anatomy match task. Some statistics of the two anatomy ontologies are shown in Table 1. The ontologies have around 3000 concepts resulting in about 9 million pairs to evaluate in the Cartesian product. The number of root paths is much higher due to multiple inheritance etc. resulting in about 470 million pairs to evaluate for the Cartesian product. COMA++ used to run into memory problems on the path-based evaluation of the anatomy match task and could thus only solve it with a node-based execution strategy.

With COMA 3.0 eliminating the memory bottleneck we are able to solve this match task for both node-based and path-based evaluation. Due to the high linguistic similarity of the input ontologies, we use a combination of mainly linguistic matchers: a simple `Name` matcher for concepts, the domain-specific `NameSyn` matcher as well as a `Path` matcher comparing the concatenated names of the concepts on the root path.

		Source	Target	Comparisons
OAEI Anatomy	Nodes	2,746	3,306	9 million
	Paths	12,362	39,001	468 million

Table 1. Statistics of the match task

	Source	Target
`Name`	18	18
`Path`	137	156

Table 2. Average string length

Figure 4(a) gives precision, recall, and F-measure results, evaluated against the gold standard containing about 1500 correspondences. The numbers in the combinations denote the weights of the individual matchers as used in the combined similarity value. The best F-measure value of 0.88 is achieved using the combination of `NameSyn`, `Path`, and `Parents` matchers. Figure 4(b) depicts execution times needed for the single matchers on an average workspace PC. The `Name` matcher itself is the fastest with an execution time of around 10 seconds, thereby still achieving 0.81 F-measure. The `Path` matcher requires in total over 41 minutes of which 32 are consumed by the evaluation of the 468 million pairs of paths, which are due to high number of path elements (cf. Table 2). The remaining time mainly is attributed to the combination of similarity values of the multiple paths per concept.

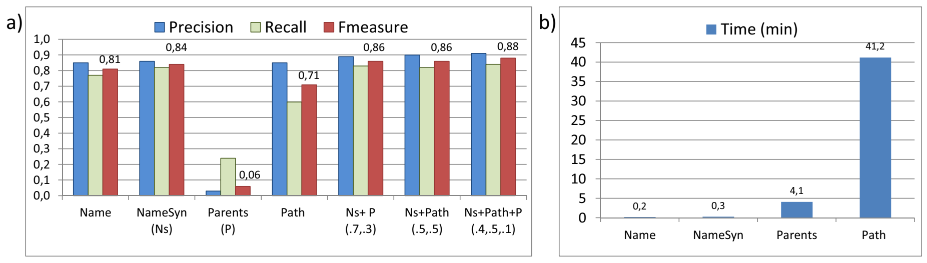


Fig. 4. Results for anatomy matching

5 Conclusions

We described the evolution of the COMA match systems during the last decade, discussed lessons learned, and sketched the upcoming version COMA 3.0. We find that many features and design decisions of the original COMA system are still valid today, and are being used in most other schema and ontology matching tools, such as the multi-matcher architecture. Further strong points include an effective default match configuration and advanced match strategies such as reuse of previous mappings and fragment matching. We also outlined current limitations of COMA++ and discussed how they are partially addressed in COMA 3.0. In particular, the new system provides improved scalability and initial support for self configuration. Furthermore, it supports the generation of enhanced mappings as well as automatic ontology merging. We are currently evaluating and completing the implementation of COMA 3.0. Further extensions are planned for future versions, such as parallel matching.

6 Acknowledgements

We thank Hong Hai Do for implementing COMA and helping in developing COMA++.

References

1. Alsayed Algergawy, Sabine Massmann, and Erhard Rahm. A Clustering-based Approach For Large-scale Ontology Matching. *Proc. ADBIS*, 2011.
2. David Aumüller, Hong-Hai Do, Sabine Massmann, and Erhard Rahm. Schema and Ontology Matching with COMA++. In *Proc. of ACM SIGMOD*, 2005.
3. Paolo Avesani, Fausto Giunchiglia, and Mikalai Yatskevich. A Large Scale Taxonomy Mapping Evaluation. In *Proc. Int. Conf. Semantic Web (ICSW), LNCS 3729*. Springer-Verlag, 2005.
4. Carlo Batini, Maurizio Lenzerini, and Shamkant B. Navathe. A Comparative Analysis of Methodologies for Database Schema Integration. *ACM Comp. Surv.*, 18(4), 1986.
5. Zohra Bellahsene, Angela Bonifati, and Erhard Rahm. *Schema Matching and Mapping*. Data-centric Systems and Applications. Springer, 2011.
6. Robin Dhamankar, Yoonkyong Lee, Anhai Doan, Alon Halevy, and Pedro Domingos. iMAP: Discovering Complex Semantic Matches between Database Schemas. In *Proc. of ACM SIGMOD*, 2004.

7. Hong Hai Do. *Schema Matching and Mapping-based Data Integration: Architecture, Approaches and Evaluation*. VDM Verlag, Saarbrücken, Germany, 2007.
8. Hong-Hai Do, Sergey Melnik, and Erhard Rahm. Comparison of Schema Matching Evaluations. In *Revised Papers from the NODe 2002 Web and Database-Related Workshops on Web, Web-Services, and Database Systems*. Springer-Verlag, 2003.
9. Hong-Hai Do and Erhard Rahm. COMA - A System for Flexible Combination of Schema Matching Approaches. In *Proc. of VLDB*, 2002.
10. Hong-Hai Do and Erhard Rahm. Matching Large Schemas: Approaches and Evaluation. *Inf. Syst.*, 32, September 2007.
11. Christian Drumm, Matthias Schmitt, Hong-Hai Do, and Erhard Rahm. QuickMig - Automatic Schema Matching for Data Migration Projects. In *Proc. of CIKM*, 2007.
12. Daniel Engmann and Sabine Massmann. Instance Matching with COMA++. In *BTW Workshops*, 2007.
13. Jérôme Euzenat and Pavel Shvaiko. *Ontology Matching*. Springer, 2007.
14. Fausto Giunchiglia, Aliaksandr Autayeu, and Juan Pane. S-Match: An Open Source Framework for Matching Lightweight Ontologies. *Semantic Web*, 2011.
15. Gerti Kappel, Horst Kargl, Gerhard Kramler, Andrea Schauerhuber, Martina Seidl, Michael Strommer, and Manuel Wimmer. Matching Metamodels with Semantic Systems - An Experience Report. In *BTW workshop on Model Management*, 2007.
16. Juanzi Li, Jie Tang, Yi Li, and Qiong Luo. RiMOM: A Dynamic Multistrategy Ontology Alignment Framework. *IEEE Trans. Knowl. Data Egineering*, 21(8), 2009.
17. Sabine Massmann, Daniel Engmann, and Erhard Rahm. COMA++: Results for the Ontology Alignment Contest OAEI 2006. *Int. Workshop on Ontology Matching*, 2006.
18. Sabine Massmann and Erhard Rahm. Evaluating Instance-based Matching of Web Directories. *11th International Workshop on the Web and Databases (WebDB)*, 2008.
19. Deborah L. McGuinness, Richard Fikes, James Rice, and Steve Wilder. An Environment for Merging and Testing Large Ontologies. In *KR*, 2000.
20. Giansalvatore Mecca, Paolo Papotti, Salvatore Raunich, and Marcello Buoncristiano. Concise and Expressive Mappings with +Spicy. *Proc. VLDB Endow.*, August 2009.
21. Natalya Fridman Noy and Mark A. Musen. PROMPT: Algorithm and Tool for Automated Ontology Merging and Alignment. In *AAAI/IAAI*, 2000.
22. Eric Peukert, Sabine Massmann, and Kathleen Koenig. Comparing Similarity Combination Methods for Schema Matching. In Klaus-Peter Faehnrich and Bogdan Franczyk, editors, *GI Jahrestagung (1)*, volume 175 of *LNI*. GI, 2010.
23. Erhard Rahm. Towards Large-scale Schema and Ontology Matching. In *Schema Matching and Mapping*. Springer, 2011.
24. Erhard Rahm and Philip A. Bernstein. A Survey of Approaches to Automatic Schema Matching. *VLDB J.*, 10, April 2001.
25. Erhard Rahm, Hong-Hai Do, and Sabine Massmann. Matching Large XML Schemas. *SIGMOD Record 33(4)*, 2004.
26. Salvatore Raunich and Erhard Rahm. Target-driven Merging of Taxonomies. Technical report, University of Leipzig, 2010.
27. Salvatore Raunich and Erhard Rahm. ATOM: Automatic Target-driven Ontology Merging. In *Proc. of ICDE*, 2011.
28. Pavel Shvaiko and Jérôme Euzenat. Ten Challenges for Ontology Matching. *Proc. OTM Conferences*, 2008.
29. Gerd Stumme and Alexander Maedche. FCA-MERGE: Bottom-Up Merging of Ontologies. In *IJCAI*, 2001.

Using Semantic Similarity in Ontology Alignment

Valerie Cross and Xueheng Hu

Computer Science and Software Engineering Department,
Miami University, Oxford, OH 45056
crossv@muohio.edu

Abstract. Many approaches to measure the similarity between concepts that exist in two different ontologies are used in the matchers of ontology alignment systems. These matchers belong to various categories depending on the context of the similarity measurement, such as lexical, structural, or extensional matchers. Although OA systems have used various forms of similarity measures along with some background knowledge sources, not many have incorporated the use of semantic similarity measures. This paper first reviews the use of semantic similarity in current OA systems, presents a unique application of such measures to assess the semantic alignment quality (SAQ) of OA systems and reports on the results of a study done using SAQ measures on the OAEI 2010 results from the anatomy track

Keywords: Semantic similarity, ontological similarity, ontology alignment, information content, semantic alignment quality.

1 Introduction

Ontology alignment (OA) research has typically concentrated on finding equivalence relationships between different concepts in different ontologies. The result of the OA process is typically a set of mappings between concepts from two different ontologies with a confidence value in [0, 1] for each mapping. OA techniques vary greatly depending on the features used to determine the mapping, i.e., the schema, its instances, etc. and the background knowledge sources used such as vocabularies or other ontologies, already existing alignments, free text, etc. Another term semantic matching has been used to describe the process when not only equivalence relations but also generalization and specialization relations are determined [1].

Early OA work focused on using string edit distances between the concept labels and the overall structure of the ontologies. Even in the same domain a wide variance in the terminology and the structuring of the concepts may still exist. Much research has worked on handling these wide variations in ontologies. GLUE [2] was one of the first to combine several different learners (similar to what is currently called a matcher) to establish mappings. A learner using instance information and one using a concept's complete list of ancestor concepts from the ontology root to the concept itself were combined to determine concept similarity between the two ontologies.

The basis for many matchers in OA systems can be found in [3] where Tversky's parameterized ratio model of similarity [4] is used with various features of concepts. Many of these similarity measures have been adapted for use in matchers in various categories depending on the context of the similarity measurement, such as lexical, structural, or extensional matchers [5]. General ontologies such as WordNet [6] have been used to find synonyms for differing concept string labels. The OLA system [7] calculates lexical similarity between two concepts by looking up their names in WordNet to find the synonyms for each concept. It does a string-based match between the pairs of synonyms and an aggregation on the resulting string similarities. RiMoM [8][9] incorporates the UMLS Metathesaurus [10] to align biomedical domain ontologies and general background knowledge sources such as Wiki to align common knowledge ontologies. More recent OA systems incorporate background knowledge sources to improve the OA process. AgreementMaker [11][12][13] extends its string-based matchers by integrating lexicons. The WordNet Lexicon is built to incorporate the synonym and definition annotations found in the ontologies themselves and then augments these with any non-duplicated synonyms and definitions existing in WordNet that correspond to those in the ontologies being aligned. The string-based matchers then work not only on the specific concept labels but also on the corresponding synonyms in the WordNet Lexicon. ASMOV [14] optionally permits a thesaurus to be used, either the UMLS Metathesaurus or WordNet, to calculate the lexical similarities between each pair of concepts, properties and individuals.

Although various forms of similarity measures are used in OA systems, only a few have incorporated semantic similarity in the OA process. This paper examines the use of semantic similarity for the evaluation of a mapping set produced by an OA system. Traditional OA evaluation strategies generally depend on a reference alignment considered to be a correct and complete set of mappings between the two ontologies and determined by a domain expert. Given a reference alignment, the quality of an OA system is evaluated with the three standard criteria: precision, recall, and f-measure. This evaluation approach has two obvious disadvantages. First, the reliability of the evaluation is directly determined by the quality of the reference alignment. For example, the reference alignment may only capture limited information of the related domain and be incomplete so OA system mappings might be correct but not found in the reference alignment. Second, in many practical cases, a reference alignment may not be available or requires too much effort to create.

This research proposes using semantic similarity measures for OA evaluation purposes, that is, a semantic alignment quality (SAQ) measure for use in addition to or in place of the standard three measures when a reference alignment is not available. The SAQ measure assesses the quality of a pair of mappings by comparing the semantic similarity between two concepts in the source ontology with the semantic similarity between the two target concepts they are mapped to. This process is performed on all pairs of mappings in the OA result to determine an overall SAQ.

First Section 2 reviews semantic similarity measures and provides examples of their use with background knowledge in current OA systems. Section 3 describes the SAQ measure. Section 4 presents the details and analysis of the experiments conducted using a wide variety of semantic similarity measures within the SAQ measure on the OAEI 2010 anatomy track ontologies. Section 5 summarizes the research and outlines plans for future research.

2 Semantic Similarity in OA

In ontology research, semantic similarity measurement is typically used to assess the similarity between concepts within an ontology. Cross-ontological similarity measures [15], i.e., ones that measure the similarity between concepts in different ontologies based on establishing association links between the concepts have been proposed. Another approach develops semantic similarity measures between concepts based on the description logic definition of the concepts. These approaches vary depending on what sets the similarity is measured such as instance sets [16], characteristic sets [17], or model sets [18]. Future research should investigate the usefulness of the cross-ontological and DL based semantic similarity measures in OA evaluations. The focus here, however, is semantic similarity measured within one ontology and using the subsumption relationship. Such semantic similarity measures are currently being used in OA systems with background knowledge sources. These semantic similarity measures were first divided into two main categories: path or distance-based and information content based. Later, set-based semantic similarity measures followed Tversky's parameterized ratio model of similarity [4]. A brief overview of these three categories, example measures, and references to some OA systems using such measures is provided [19].

The *path-based similarity measures* or edge-counting similarity measures rely on the distance between two concepts. This distance is a count of the number of edges on the path or a count of the number of nodes in the path linking the two concepts. Some approaches assign different weights to edges or use different conversions and normalizations of Rada's distance metric [20] into a similarity measure. For example, Leacock and Chodorow [21] converted Rada's distance metric into a path-based semantic similarity as follows:

$$sim_{LC} = -\log(min_p[len(p(c1,c2))]/2D)$$

where D is the depth of the ontology that contains c1 and c2. It basically normalizes Rada's distance measure *len(p(c1,c2))* using D and converts it to similarity by using the negative logarithm. An early OA system iMapper [22] uses a simple path based semantic distance between two terms *x* and *y* found in WordNet. If they belong to the same WordNet synset, then the path distance is 1. Otherwise, the path length is determined by first finding the paths from each sense of *x* to each sense of *y*, counting the number of nodes in each path between the two senses, and using the minimum count of nodes for the semantic distance. Note that path length is determined by the number of nodes rather than number of edges in the path.

The Wu and Palmer measure [23] calculates similarity using the distance from the root to the common subsumer of c1 and c2. The formula is:

$$sim_{WP}(c1, c2) = 2\times \frac{len(root,c3)}{len(c1,c3) + len(c2,c3) + 2\times len(root,c3)}$$

where c3 is the common subsumer of c1 and c2. In the case that c1 and c2 have multiple common subsumers, c3 is typically assumed to be the lowest, i.e., the one with the greatest distance from the root. For this research, c3 is selected as the one that minimizes the path distance between c1 and c2 since in a well-designed ontology, this c3 should also be the lowest one. OLA [7] uses a measure similar to the Wu-Palmer measure with the WordNet ontology. ASMOV [14] use the Wu-Palmer semantic similarity on the XML data type hierarchy for properties, when the ranges of two data type properties are being compared.

Information content (IC) based measures use a measure of how specific a concept is in a given ontology. The more specific a concept is the higher its IC. The more general a concept is the lower its IC. Originally, IC uses an external resource such as an associated corpus [24]. The corpus-based IC measure for concept *c* is given as

$$IC_{corpus}(c) = -log\, p(c)$$

where the value p(c) is the probability of the concept determined using the frequency count of the concept, i.e. the number of occurrences within the corpus of all words representing the concept and includes the total frequencies of all its children concepts.

The ontology-based IC [25] uses the ontology structure itself [25] and is defined as

$$IC_{ont}(c) = log\, \frac{(num\_desc(c)+1)}{max_{ont}} / log \frac{1}{max_{ont}} = 1 - \frac{log(num\_desc(c)+1)}{\log(max_{ont})}$$

where num_desc(c) is the number of descendants for concept *c* and max_{ont} is the maximum number of concepts in the ontology. This IC measure is normalized such that the information content values are in [0...1]. IC_{ont} has maximum value 1 for the leaf concepts and decreases until the value is 0 for the root concept of the ontology.

The first IC based ontological similarity measure was proposed by Resnik [24] as

$$sim_{RES}(c1,c2) = max_{S(c1,c2)}\, [IC_{corpus}(c)]$$

where S(*c1*,*c2*) is the set of concepts that subsume both *c1* and *c2*.

Lin [26] defined a measure that uses not only the shared information between the two concepts but also the separate information content of the two concepts:

$$sim_{Lin}(c1,c2) = \frac{2 \times IC(c3)}{IC(c1) + IC(c2)}$$

where *c3* is the subsuming concept with the most information content. ASMOV [14] uses the Lin measure to assess the semantic similarity between two labels in a thesaurus which is either WordNet or UMLS. UFOme [27] uses the Lin measure in its WordNet matcher to determine the semantic similarity between synsets found in WordNet when the concepts being mapped do not share the same synset in WordNet.

Jiang and Conrath [25] define another distance measure integrating path and information content based measures. The distance is based on totaling up their separate IC and subtracting out twice the IC of their most informative subsumer.

$$dist_{JC}(c1, c2) = IC(c1) + IC(c2) - 2 \times IC(c3)$$

so that the remaining IC indicates the distance between them. If no IC is left, i.e., 0, the two concepts are the same. This distance measure can be converted to similarity. Several approaches have been proposed. In [25], the following formula is used

$$sim_{JC}\ (c1, c2) = 1\text{-}\ (IC(c1) + IC(c2)\text{-}\ 2\times IC(c3))\times 0.5.$$

Set-based semantic similarity measures use Tversky's parameterized ratio model [4]:

$$S_{Tverksy}(X,\ Y) = \frac{f(X \cap Y)}{f(X \cap Y)\ +\ \alpha f(\mathrm{X}-\mathrm{Y})\ +\beta f(Y-X)}$$

where f is an evaluation measure on sets. The α and β permit variations on the similarity measure. Here f is defined as fuzzy set cardinality. Itbparallels set cardinality. The only difference is an element's degree of membership in the fuzzy set is added in instead of simply a 1 for the element. A concept's IC value is used as its membership degree. Fuzzy set cardinality of a set of concepts is the sum of each concept's IC in the set. A wide variety of fuzzy set similarity measures are based on the Tversky model [29]. One can view a concept in an ontology as an object with a set of features or a related set. If $\alpha = \beta = 1$, S becomes the fuzzy set Jaccard index:

$$\mathrm{S}_{\mathrm{Jaccard}}(\mathrm{c1},\ \mathrm{c2}) = \frac{\sum_{c\in(relatedSet(c1)\cap relatedSet(c2))} IC(c)}{\sum_{c\in(relatedSet(c1)\cup relatedSet(c2))} IC(c)}$$

If $\alpha = \beta = 0.5$, S becomes the Dice coefficient. If $\alpha = 1$, $\beta = 0$, S becomes the degree of inclusion of the related set for concept c1 within the related set for concept c2.

Many different sets can be a related set of a concept *c*. In this research the upset which is the ancestor set of c in addition to the concept c itself, the downset which is the descendant set of c in addition to the concept c itself, and the hourglass which is the union of the upset and downset of a concept [30] are used. Other feature sets for a concept are entirely possible such as the neighborhood set of a concept where neighbors can be based on other relationship types besides the is-a and a parameter may be used to determine how wide the neighborhood is from the concept [3].

3 Semantic Alignment Quality

The SAQ measure determines how well each pair of mappings, $(s_i,\ t_i)$ and $(s_j,\ t_j)$ maintains the same semantic similarity between the corresponding concepts in each ontology. A good pair of mappings should result in $|sim(s_i,s_j) - sim(t_i,t_j)|$ being close to 0. In [30] a similar approach is taken based on ordered concept lattice theory and proposes two new distance measures. The upper cardinality distance d_u and lower cardinality d_l between concepts a and b are defined as

$$d_u(a, b) = |upset(a)| + |upset(b)| - 2*max_{c\text{-}join}[\ |upset(c)|]$$
$$d_l(a, b) = |downset(a)| + |downset(b)| - 2*\ max_{c\text{-}meet}\ [|downset(c)|]$$

where c-join is the join concept, an ancestor concept shared between a and b in the lattice with no other concept less than it in the concept lattice. The lower cardinality distance between concepts a and b is defined similarly to upper except the upset is replaced by downset and the join is replaced by the meet, i.e., c-meet is a meet concept, a descendent concept shared between a and b in the lattice with no other concept greater than it in the concept. In [30], only results with d_l are reported using the OAEI 2009 anatomy track. The experiments reported here are performed with a wide variety of semantic similarity measures more familiar to the ontology research community than d_l. The d_l measure, however, is also used within SAQ for comparison purposes. The experimental results also show some considerations on using semantic similarity measures to evaluate OA results not examined in [30].

To more clearly explain the approach, assume the set of mappings M = $\{(s_i, t_i) \mid s_i \in O_s, t_i \in O_t$, and s_i maps to t_i in the OA result set$\}$. To measure the similarity difference for two mappings m_i and m_j, the following formula is used:

$$simDiff(m_i, m_j) = | sim(s_i, s_j) - sim(t_i, t_j) |$$

with s_i and s_j being source anchors and t_i and t_j being target anchors such that $m_i = (s_i, t_i)$ and $m_j = (s_j, t_j)$ in M. The overall difference of semantic similarity for source anchor pairs and target anchor pairs is calculated as

$$simDiff_{overall}(M) = \sum_{m_i, m_j \in M} simDiff(m_i, m_j)$$

and the average difference is calculated over all (m_i, m_j) pairs in M where $i \neq j$ as

$$simDiff_{average}(M) = \frac{simDiff_{overall}(M)}{C_2^N} \qquad \text{where } N = |M|.$$

The denominator is the number of combinations of N mappings taken two at a time.

The SAQ measure is 1- $simDiff_{average}$. The closer SAQ is to 1, then the smaller the semantic similarity difference is over all the pairs of mappings in the alignment. More specifically, the alignment results of high quality are expected to produce small values for the $simDiff_{average}$. Here ‘small values’ means being close to zero or no greater than a predefined threshold. This threshold can be derived through experimentation with existing reference alignments that are believed to have high quality.

Notice that the SAQ can have any semantic similarity measures substituted for *sim*. The experiments reported in the next section used the lower cardinality distance, the two path based measures, the three IC based measures and 9 variations of the set-based similarity measures resulting from the three standard Tverskey set-based similarity measures paired with the three different related sets, the downset, the upset and the hourglass for a concept. The experiments investigate the performance differences of these semantic similarity measures in SAQ and if the notion of SAQ corresponds with the standard performance measures used to evaluate OA results.

4 Experimenting with SAQ and the OAEI 2010 Results

The ontology alignment evaluation initiative (OAEI) [31] conducts yearly competitions that include the most up-to-date OA systems. These systems and their algorithms are evaluated using the same set of test cases so that performance comparisons can be made by those interested in using them. The OAEI 2010 campaign supported four tracks: anatomy, benchmark, conference, and directory.

Each track is specialized for different purposes. The experiments reported in this section focus on the anatomy track since the anatomy track uses two real-world ontologies from the biomedical domain, the NCIT human anatomy (HA) ontology and the mouse anatomy ontology (MA) which are considerably larger and also produce many more mappings than those of the other tracks. The reference alignment between the two ontologies is readily available and consists of 1520 mappings. The precision, recall and f-measure of each OA system that participated in the anatomy track are also available. Finally, the concept lattice lower distance measure research in [30] also used the anatomy track of the 2009 OAEI.

Table 1 lists the OA systems alphabetically along with the number of mappings produced and their performance measures on the anatomy track's first subtask which is to produce the best mappings possible emphasizing the f-measure.

Table 1. OA Systems OAEI 2010 Anatomy Track Precision (P), Recall (R), F-Measure (F)

OA Systems	**# of mappings**	**P**	**R**	**F**
AgrMaker	1436	0.903	0.853	0.877
Aroma	1347	0.770	0.682	0.723
ASMOV	1409	0.799	0.772	0.785
BLOOMS	1164	0.954	0.731	0.828
CODI	1023	0.968	0.651	0.779
Ef2Match	1243	0.955	0.781	0.859
GeRMeSMB	528	0.884	0.307	0.456
NBJLM	1327	0.920	0.803	0.858
SOBOM	1246	0.949	0.778	0.855
TaxoMap	1223	0.924	0.743	0.824

Tables 2 and 3 report the SAQ measure results for the various semantic similarity measures listed on the columns. The first number in parentheses after the semantic similarity label indicates the rank of that measure within the row of values for each OA system for only the measures in that table. The second number in parenthesis is the rank of that measure for both tables combined. For the most part each value for a semantic similarity measure had an identical rank across all rows. For example, the lower distance had the highest SAQ value (rank of 1) compared to all other semantic similarity measures across all OA systems in the Table 2. When compared with all measures in both tables, the lower distance was ranked 4th. The Wu-Palmer (WP) measure had the lowest SAQ value (rank of 6) compared to all other semantic similarity measures across all OA systems in Table 2. When compared with all measures in both tables 2 and 3, WP had the lowest SAQ value (rank of 15). If the ranking was not identical across all OA systems, a ranking was only one greater or one less than the most often reoccurring rank in the column. If more than half of one column's ranks had an identical rank value, that rank was used for the SAQ. Note that the first and second rows of the tables are the SAQ results on the partial and full reference alignments provided for the anatomy track.

The SAQ values for the lower distance measure seem to indicate that the alignment quality is extremely good for all these OA systems with it almost being perfect for CODI. There also is very little difference in the OA systems with a range of only

0.00259 between all the values for the SAQ result using the lower cardinality distance measure. The SAQ values for the Wu-Palmer measure seem to indicate that the alignment quality is not as high and has a wider range with a range of 0.01314. But notice all the other SAQ values are greater than 0.90. Another observation is the difference in SAQ results for the two path-based measures. The Leacock-Chodorow agrees more with the IC based results. This experiment indicates there is a substantial difference in SAQ measures depending on what semantic similarity measure is used.

To further investigate this issue, the average WP semantic similarity measure and the average Lin semantic similarity measure was calculated between all 1520*1519/2 pairs of concepts from both the MA and the HA. The WP averages for the MA and HA are 0.015 and 0.074 respectively. The Lin averages for the MA and HA are 0.018 and 0.315 respectively. For the MA, there is little difference in the WP and Lin measures but a substantial difference for the HA. A possible explanation is the MA ontology is not as deep as the HA (maximum depth of 7 vs. 13) and has a less complex structure than the HA (4% vs. 13% of the nodes with multiple parents).

Table 2. SAQ using lower distance, 2 path-based, 3 IC-based semantic similarity measures

OA Systems	**Lower dist 1-D(F) (1) (4)**	**WP (6) (15)**	**LC (4) (13)**	**Lin (3) (12)**	**Resnik (2) (11)**	**JC (5) (14)**
Partial ref	0.99934	0.70647	0.92652	0.94017	0.97274	0.93291
Full ref	0.99902	0.70179	0.92740	0.93633	0.93899	0.92662
AgrMaker	0.99906	0.70234	0.92706	0.93737	0.94009	0.92461
Aroma	0.99718	0.70469	0.92588	0.9385	0.94231	0.91968
ASMOV	0.99866	0.70202	0.92735	0.93836	0.94157	0.92352
BLOOMS	0.99846	0.70301	0.92721	0.94010	0.94296	0.92913
CODI	0.99977	0.70855	0.92660	0.94174	0.94339	0.93684
Ef2Match	0.99936	0.70595	0.92791	0.93816	0.94074	0.92839
GeRMeSMB	0.99936	0.69541	0.92872	0.93268	0.93330	0.92852
NBJLM	0.99907	0.70599	0.92728	0.93797	0.94061	0.92610
SOBOM	0.99921	0.70599	0.92789	0.93988	0.94254	0.93024
TaxoMap	0.99913	0.70816	0.92815	0.93881	0.94154	0.92614

Table 3 shows the results for the SAQ measure using the nine set based semantic similarity measures. The rankings indicate that the downset measures have extremely high SAQ values and the range over all SAQ values using downsets is 0.99996 – 0.99729 = 0.00267. The higher SAQ measure for the downset semantic similarity measures over the upset ones was a surprising result. Intuition suggests that the upset set semantic similarity measures should be better than the downset ones. The rationale is that for downsets, the descendents represent more specific concepts. For example if c is a descendent of a and b, then c inherits features from both A and B but those inherited features may be entirely different and for different purposes. They do not represent common features. But if a and b both have the common ancestor c, then both a and b share c’s features.

The unusually high SAQ values for the downset semantic similarity measures which ranked 1st, 2nd and 3rd overall caused further investigations which determined that the downset semantic similarity measures are not useful for the SAQ measure.

When downsets are used in the SAQ, many intersections between the two concepts' sets of descendents are empty. With an empty intersection, all the downset semantic similarity measures produce a 0. This situation is verified by counting the number of cases for the reference alignment where the result for |sim(a,b) – sim(a',b')| is |0 – 0| resulting in a 0 contribution to the *simDiff*$_{overall}$ total. Close to 50% of the *sumDiff* calculations were |0 – 0|. Not one s*umdiff* produced from any upset semantic similarity measures resulted in a |0 – 0| case. Since the hour set measures include both the upset and the downset, the hour measures also are affected by the extremely large number of cases where there is an empty intersection for the downset. The smaller semantic similarity values then produced smaller *sumdiff* values, thereby, reducing the *simDiff*$_{average}$. A small *simDiff*$_{average}$ using the downset measures is not an accurate reflection of the quality of the alignment.

Table 3. SAQ using the nine set-based semantic similarity measures

OA Systems	**Jacc Up (5) (6)**	**Jacc Down (1) (1)**	**Jacc Hr (4) (5)**	**Dice Up (8) (9)**	**Dice Down (2) (2)**	**Dice Hr (6) (7)**	**Inc Up (9) (10)**	**Inc Down (3) (3)**	**Inc Hr (7) (8)**
Partial ref	0.97962	0.99986	0.98311	0.96464	0.99980	0.97030	0.96343	0.99952	0.97962
Full ref	0.97862	0.99982	0.98211	0.96272	0.99974	0.96837	0.96117	0.99922	0.97862
AgrMaker	0.97895	0.99985	0.98243	0.96324	0.99976	0.96896	0.96206	0.99903	0.96363
Aroma	0.97958	0.99978	0.98410	0.96442	0.99966	0.97186	0.96350	0.99729	0.96456
ASMOV	0.97957	0.99971	0.98361	0.96436	0.99961	0.97098	0.96309	0.99870	0.96499
BLOOMS	0.97989	0.99984	0.98359	0.96498	0.99976	0.97105	0.96375	0.99868	0.96507
CODI	0.98020	0.99996	0.98253	0.96551	0.99994	0.96929	0.96384	0.99993	0.96508
Ef2Match	0.97941	0.99987	0.98279	0.96407	0.99980	0.96960	0.96281	0.99948	0.96467
GeRMeSMB	0.97934	0.99990	0.97994	0.96347	0.99985	0.96452	0.96137	0.99932	0.96144
NBJLM	0.97914	0.99984	0.98257	0.96366	0.99975	0.96924	0.96219	0.99908	0.96401
SOBOM	0.97966	0.99986	0.98320	0.96461	0.99978	0.97037	0.96336	0.99926	0.96514
TaxoMap	0.97923	0.99978	0.98284	0.96395	0.99971	0.96982	0.96276	0.99971	0.96463

A question also raised from this experiment is why the lower cardinality distance measure produces the 4th greatest SAQ values. It too uses the concepts' downsets to determine the distance between two concepts. In [30], d_l was chosen with the rationale that the ontologies are more strongly down-branching than up-branching so that down-sets are larger. Siblings deep in the hierarchy are closer together than siblings high in the hierarchy. The intuition behind this seems faulty. The lower cardinality distance suffers from the same problem that the downset set-based measures suffer from – what happens when there is no downset intersection. The SAQ using $|d_l(a,b) - d_l\ (a',b')|$ translates into the difference between the sum of the number of descendents for a and b and the sum of the number of descendents for a' and b'. Simply because pairs of concepts do not differ greatly in the total number of descendents within their respective ontologies does not mean the mapping is a good mapping. The concepts being mapped could all be leaf or close to leaf nodes but in totally different subtrees of the ontology. Further investigation on the reference alignment shows that the average number of descendents for source and target anchors is 3.2 and 2.8. These averages indicate a very small difference in the number of descendents, and therefore, a very small *simDiff*$_{average}$.

Table 4. Pearson Correlation with p-value for the SAQ and precision, recall, and f-measure.

SAQ	**Precision**		**Recall**		**F-measure**	
	Corr	**p-value**	**Corr**	**p-value**	**Corr**	**p-value**
Lower dist	0.7474974	0.01294	-0.1145901	0.7526	0.02893312	0.9368
WP	0.3600771	0.3068	0.6443854	0.2114	0.7366019	0.01511
LC	0.3710125	0.2912	-0.3711803	0.291	-0.3101822	0.3831
Lin	0.314404	0.3763	0.6277758	0.05198	0.7163154	0.01978
Res	0.131069	0.7182	0.7417546	0.01405	0.7808777	0.007669
JC	0.8058114	0.004886	-0.2055725	0.5688	-0.0002435	0.9995
Jacc Up	0.1846966	0.6095	-0.163976	0.6508	-0.05489762	0.8803
Jacc Down	0.6936438	0.0261	-0.3623896	0.3034	-0.1866459	0.6056
Jacc Hour	-0.2395826	0.505	0.7383087	0.01475	0.6876649	0.02797
Dice Up	0.2132382	0.5542	0.06231594	0.8642	0.171189	0.6363
Dice Down	0.719249	0.01905	-0.404883	0.2458	-0.2190521	0.5432
Dice Hour	-0.2128156	0.555	0.7432621	0.01376	0.6995049	0.02435
Inc Up	0.07231682	0.8426	0.3695583	0.2932	0.4315503	0.213
Inc Down	0.7811095	0.007638	-0.07342486	0.8402	0.08693707	0.8113
Inc Hour	0.1428098	0.6939	0.7285981	0.01685	0.7683169	0.009428

In [30] the Pearson correlation of the lower cardinality distance with the f-measure was given as -0.780. The Pearson correlation for this measure with precision in Table 4 is a 0.7474974. It is positive here since the SAQ is converted into a quality indicator by subtracting from 1. The correlation with f-measure is only 0.02893312. The difference in reported values for the f-measure is unclear unless the reported correlation value in [30] is actually for precision and not f-measure.

From Table 4, all the downset set-based measures had significant correlation with precision and yet, the investigation of the downset measures showed the problem with the huge number of |0 – 0| cases where the downset intersection for both pairs of concepts was empty. The Jiang-Conrath (JC) SAQ is the only other one that had significant correlation with the precision measure. More investigation needs to be done on the SAQ to validate its high correlation with precision.

5 Conclusions and Future Work

This research has investigated the use of semantic similarity measures to evaluate the quality of the mappings produced by OA systems and parallels the work in [30] which experimented with one distance measure between concepts. The research goal is to develop additional means of alignment evaluation that do not depend on a reference alignment. As the experimental results show there are some difficulties with this approach depending on the selected semantic similarity measure.

More research and experiments with SAQ should be undertaken to determine how useful SAQ is for assisting in ontology alignment evaluation especially with respect to precision. SAQ correlation with precision is more intuitive than with recall or f-measure since SAQ is based only on the produced mappings. SAQ has no knowledge of missed mappings.

Further investigation is needed to determine how much poor mappings affect the resulting SAQ and identify and eliminate these very poor mappings if the semantic similarity difference is above a specified threshold. The semantic similarity difference operation might be more useful in the alignment process itself than in the evaluation of the final mappings. The OA systems in this experiment specifically use semantic similarity with a knowledge source and not between concepts in the source and target ontologies in their matching algorithms. If a semantic similarity measure is used in an OA system's matching process, research is needed to see how the SAQ evaluation of its mapping result may be biased based on the selected measure.

Acknowledgments. The authors acknowledge the implementation of the SAQ measure was embedded in AgreementMaker and Cosmin Stroe served as a consultant.

References

1. Giunchiglia, F. Shvaiko, P. and Yatskevich, M.**:** S-Match: an algorithm and an implementation of semantic matching**.** Technical Report DIT-04-015, Department of Information Engineering and Computer Science, University of Trento. Proc. of the first European Semantic Web Symposium (ESWS) (2004)
2. Doan, A., Madhavan, J., Domingos, P., Halevy, A.: Ontology Matching: A Machine Learning Approach. In: Handbook on Ontologies in Information Systems. pp. 397—416. Springer (2003)
3. Rodriguez, M. A., Egenhofer, M. J.: Determining Semantic Similarity among Entity Classes from Different Ontologies. IEEE Transactions on Knowledge and Data Engineering, vol. 15, issue 2, pp, 442--456 (2003)
4. Tversky, A.: Features of Similarity. Psychological Rev., 84, pp. 327--352 (1977)
5. Sabou, M., d'Aquin, M., Motta, E.: Exploring the Semantic Web as Background Knowledge for Ontology Matching. J. Data Semantics 11, pp. 156--190 (2008)
6. Miller, G., Beckwith, R., Fellbaum, C., Gross, D., and Miller, K.J.: Introduction to Wordnet: An on-line Lexical Database. International Journal of Lexicography 3(4), pp. 235—244 (1990)
7. Euzenat, J., Valtchev, P.: An integrative proximity measure for ontology alignment. In: Proc. ISWC-2003 Workshop on semantic information integration, Sanibel Island (FL US), pp. 33--38 (2003)
8. Tang, J., Liang, B.Y., Li, Juanzi, Wang, Kehong: Risk Minimization based Ontology Mapping. 2004 Advanced Workshop on Content Computing (AWCC). LNCS, vol. 3309, pp. 469--480. Springer-Verlag (2004)
9. Li, Juanzi, Tang, Jie, Yi, Li, Luo, Qiong: RiMOM: A Dynamic Multistrategy Ontology Alignment Framework. IEEE Transactions on Knowledge and Data Engineering, vol. 21 issue. 8. pp. 1218--1232 (2009)
10. Unified Medical Language System (UMLS) http://umlsks.nlm.nih.gov
11. Cruz, I. F., Sunna, W.: Structural Alignment Methods with Applications to Geospatial Ontologies. Transactions in GIS, Special Issue on Semantic Similarity Measurement and Geospatial Applications vol. 12 no. 6, pp. 683—711 (2008)
12. Cruz, I F., Palandri Antonelli, F., Stroe, C.: AgreementMaker: Efficient Matching for Large Real-World Schemas and Ontologies. PVLDB,vol. 2, no. 2, pp. 1586--1589 (2009).
13. Cruz, I. F., Stroe, C., Caci, M., Caimi, F., Palmonari, M., Palandri Antonelli, F., Keles, U. C.: Using AgreementMaker to Align Ontologies for OAEI. In: ISWC International

Workshop on Ontology Matching (OM), ser. CEUR Workshop Proceedings vol. 689, pp. 118—125 (2010)
14. Jean-Mary, Y.R., Shironoshita, E. P., Kabuka, M. R.: Ontology matching with semantic verification. Web Semantics, vol 7 issue 3. pp. 235--251 (2009)
15. Posse, C., Sanfilippo, A., Gopalan, B., Riensche, R., Beagley, N., Baddeley, B.: Cross-Ontological Analytics: Combining Associative and Hierarchical Relations in the Gene Ontologies to Assess Gene Product Similarity. International Conference on Computational Science (2), pp. 871—878 (2006)
16. d'Amato , C., Fanizzi, N., Esposito, F.: A dissimilarity measure for ALC concept descriptions. In: Proc. ACM Symposium on Applied Computing (SAC), ACM. pp. 1695–1699 (2006)
17. Araujo, R., and Pinto, H. S. Towards semantics-based ontology similarity. In: Proc. Workshop on Ontology Matching (OM), International Semantic Web Conference (ISWC). (2007)
18. Janowicz, K., Wilkes, M.: SIM-DL_A: A Novel Semantic Similarity Measure for Description Logics Reducing Inter-Concept to Inter-Instance Similarity. In: The 6th Annual European Semantic Web Conference (ESWC2009). Lecture Notes in Computer Science 5554, Springer. pp. 353-367 (2009)
19. Cross, V. and Yu, Xinran: Investigating Ontological Similarity Theoretically with Fuzzy Set Theory, Information Content, and Tversky Similarity and Empirically with the Gene Ontology. In: Proc. of the 5th International Conference on Scalable Uncertainty Management, Dayton OH (2011)
20.Rada R, Mili H, Bicknell E, Blettner M: Development and Application of a Metric on Semantic Nets. In: IEEE Transaction on Systems, Man, and Cybernetics vol. 19, pp. 17–-30 (1989)
21. Leacock C. and Chodorow, M.: Combining local context and WordNet Similarity for Word Sense Identification. In: WordNet: An Electronic Lexical Database. Fellbaum, Ed. Cambridge, MA: MIT Press, pp. 265--283 (1998)
22. Su, Xiaomeng: Semantic Enrichment for Ontology Mapping, Ph.D. Thesis, Dept. of Computer and Information Science, Norwegian University of Science and Technology (2004)
23. Wu Z, Palmer M. S.: Verb Semantics and Lexical Selection. In: Proc. of the 32nd. Annual Meeting of the Association for Computational Linguistics, pp. 133--138. (1994)
24. Resnik, P.: Using Information Content to Evaluate Semantic Similarity in Taxonomy. In: Proc. of the 14th International Joint Conference on Artificial Intelligence, pp, 448--453 (1995)
25. Seco N, Veale T, Hayes J.: An Intrinsic Information Content Metric for Semantic Similarity in Wordnet. In: ECAI. pp. 1089--1090 (2004)
26. Lin D.: An Information-theoretic Definition of Similarity. In: Proc. of the 15th International Conference on Machine Learning. Morgan Kaufmann. pp. 296--304 (1998).
27. Giuseppe, P., Talia D.: UFOme: An Ontology Mapping System with Strategy Prediction Capabilities. Data Knowl.Eng. vol. 69 no. 5, pp. 444--71 (2010)
28. Jiang J, Conrath D: Semantic Similarity Based on Corpus Statistics and Lexical Taxonomy. In: Proc. of the 10th International Conference on Research on Computational Linguistics, Taiwan (1997)
29. Cross V., Sudkamp, T.: Similarity and Compatibility in Fuzzy Set Theory, Heildelberg: Physical-Verlag (2002)
30. Joslyn, Cliff A., Paulson, P., White, A.: Measuring the Structural Preservation of Semantic Hierarchy Alignment. In: ISWC International Workshop on Ontology Matching. CEUR-WS. (2009).
31. Euzenat, J.et al.: The Results of the Ontology Alignment Evaluation Initiative 2010. Ontology Matching Workshop, International Semantic Web Conference (2010)

Ontology matching benchmarks: generation and evaluation

Maria Roşoiu, Cássia Trojahn, and Jérôme Euzenat

INRIA & LIG, Grenoble, France
`Firstname.Lastname@inria.fr`

Abstract. The OAEI Benchmark data set has been used as a main reference to evaluate and compare matching systems. It requires matching an ontology with systematically modified versions of itself. However, it has two main drawbacks: it has not varied since 2004 and it has become a relatively easy task for matchers. In this paper, we present the design of a modular test generator that overcomes these drawbacks. Using this generator, we have reproduced Benchmark both with the original seed ontology and with other ontologies. Evaluating different matchers on these generated tests, we have observed that (a) the difficulties encountered by a matcher at a test are preserved across the seed ontology, (b) contrary to our expectations, we found no systematic positive bias towards the original data set which has been available for developers to test their systems, and (c) the generated data sets have consistent results across matchers and across seed ontologies. However, the discriminant power of the generated tests is still too low and more tests would be necessary to draw definitive conclusions.
Keywords: Ontology matching, Matching evaluation, Test generation, Semantic web.

1 Introduction

Evaluating ontology matching may be achieved in several ways. The most common one consists of providing matchers with two ontologies and comparing the returned alignment with a reference alignment [4]. However, this raises the issue of the choice of ontologies and the validity of the reference.

Since 2004, the Ontology Alignment Evaluation Initiative (OAEI)[1] makes available a collection of data sets for evaluating matching systems. One such data set is Benchmark. It is a well-defined set of tests in which each test is composed of two ontologies and a reference alignment. The tests are based on one particular ontology, from the bibliographic domain, and systematic alterations of this ontology, e.g., removing classes, renaming properties.

Benchmark was designed with the aim of covering the problem space, i.e., the various situations in which a matcher may be. However, this data set has various drawbacks: (a) lack of realism: tests are mechanically generated and cover

[1] `http://oaei.ontologymatching.org/`

a systematic alteration space, (b) lack of variability: it always uses the same seed ontology altered in the exact same way, and (c) lack of discriminability: the tests are not difficult enough to discriminate well matchers.

We are not particularly interested in Drawback (a) because it has been overcame by other data tests made available by OAEI. We focus on drawbacks (b) and (c). To that extent, we have developed a test generator that may be used with any seed ontology and allows for fine tuning the input parameters, as well as randomized modifications over the ontology entities. A byproduct of this generator is that it enables us to evaluate the relevance of the Benchmark dataset: by reproducing this dataset and using it to evaluate different matchers in the same conditions, we can assess how much the results obtained are dependent on the particular seed ontology or the particular matcher.

We run different matchers on the generated tests, which allows us to draw conclusions on the results obtained so far with Benchmark:

– The difficulties encountered by a matcher at a test are preserved across the seed ontology, hence, Benchmark is relevant.
– Matchers have, in general, no better results with the original Benchmark than with the new generated data sets, this goes counter our expectation that, because tests and results were available, matchers would perform better.
– Matcher results are generally consistent across seed ontologies and ontology results are generally consistent across matchers, but with low discrimination. This confirm that the lack of discriminability is due to Benchmark and not to the seed ontology.

The rest of the paper is structured as follows. In Section 2, we present the state-of-the-art in ontology matching test generation. In Section 3, we present the architecture of our test generator and the strategy we came with in order to reproduce the Benchmark dataset. In Section 4, we expose the results we have obtained with new generated datasets and their variability. Finally, the conclusions and future work are presented in Section 5.

2 Ontology matching evaluation and test generation

In this section, we briefly present the current setting of ontology matching evaluation (Section 2.1), the Benchmark data set (Section 2.2) and the state-of-the-art in alignment test generators (Section 2.3). The interested reader can find a broader overview of ontology matching evaluation in [4].

2.1 Evaluating ontology matching systems

Matching can be seen as an operation which takes as input two ontologies (o and o'), a set of parameters (p), a possibly empty partial alignment (A') and a set of resources (r) and outputs an alignment (A) between these ontologies (Fig. 1).

An alignment can be defined as a set of correspondences. A correspondence between two ontologies o and o' is a triple $\langle e, r, e' \rangle$, where e is an entity belonging to the first ontology, e' is an entity belonging to the second ontology, r is a relation, e.g., equivalence or subsumption, between them.

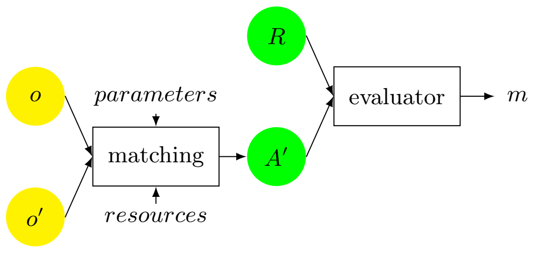


Fig. 1: Ontology matching process and evaluation (from [5]).

A matcher can be evaluated comparing its output alignment (A) with a reference alignment (R) using some measure (Fig. 1). Usually, such measures are precision, recall and F-measure [5]. Thus, in order to evaluate a matching system, one has to generate datasets in which a test is composed of two ontologies to be matched (o and o') and a reference alignment (R).

2.2 The Benchmark dataset

Benchmark aims at testing the strengths and the weaknesses of matching systems, depending on the availability of ontology features. This dataset has 111 tests, requiring to match an ontology written in OWL-DL to another one:

– Tests 1xx - compare the original ontology with itself, a random one and its generalization in OWL-Lite.
– Tests 2xx - compare the original ontology with the ontology obtained by applying the following set of modifications to it (Fig. 2):
  • names (naming conventions: synonyms, random strings, different generalization, translation into other language)
  • comments (no comments)
  • hierarchy (flattened hierarchy / expanded hierarchy / no specialization)
  • instances (no instance)
  • properties (no properties, no restrictions)
  • classes (flattened classes / expanded classes)
– Test 3xx - compare the original ontology with real ones found on the web.

Since 2004, Benchmark has been generated from the same seed ontology through the same set of XSLT stylesheets. This means, in particular, that no random modification is applied to these ontologies: the same 20% of classes are renamed and this renaming is always the same. This has advantages for studying the evolution of the field, because the test is always the same.

However, the Benchmark data set can be criticised on three main aspects:

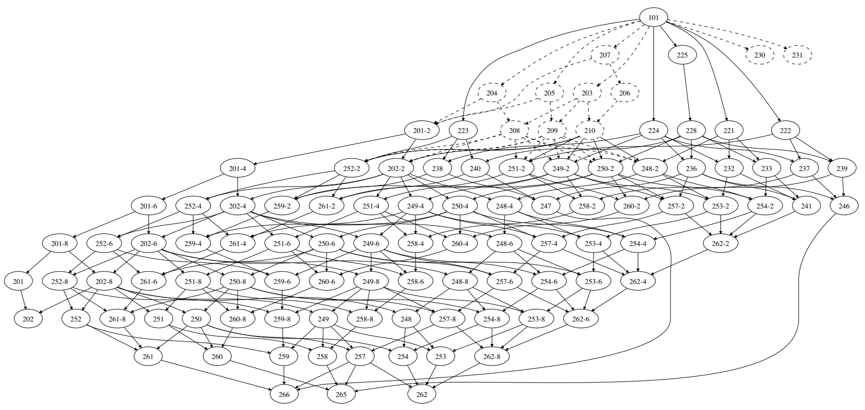


Fig. 2: The Benchmark lattice – the higher the test is in the hierarchy, the easier it is. Tests in dashed lines are not reproduced in the tests we used here (see §4).

Lack of realism Benchmark is not realistic because it covers a whole systematic space of mechanical alterations and in reality a matcher is not faced with such a space.

Lack in variability Benchmark always produces the same data set hence it is not variable. This covers three slightly different kinds of problems: (a) it can only be used with one seed ontology, (b) it always applies the same transformations (to the same entities), instead of applying them randomly, and (c) it is not flexible in the sense that it is not possible to produce an arbitrary test (such as 12% renaming, 64% discarding properties).

Lack of discriminability [7] Benchmark seems in general easy enough to OAEI participants so that they do not really allow them to make progress and to compare them. This is because, many of the proposed tests are easy and only a few are really difficult.

Our goal is to address variability and discriminability by producing a test generator (a) independent from the seed ontology, (b) with random modifications, and (c) which allows to fine tune parameters in order to cover the alteration space with any precision. With such a test generator it would be possible to generate different tests than Benchmark focusing on particular application profiles or particularly difficult cases.

We do not address the lack of realism because Benchmark has been designed to cover the problem space and not to offer one realistic profile[2]. Other initiatives, such as other tracks of OAEI and other generators, address this issue.

[2] One reviewer argues that we currently consider an alteration space, instead of a problem space, which assumes some realism, i.e., that these problems actually occurs. Moreover, (s)he write that we choose the kind of alteration and the granularity. This is right. But this alteration space is our attempt to cover, and not to represent, the problem space.

2.3 Other ontology alignment generators

So far, some alignment test generators have been developed.

An ontology generator inspired by Benchmark is the one developed in [2]. Its seed ontology is a random tree which is computed using a Gaussian distribution with average 4 and deviation 4 in order to determine the number of children per node. The second ontology is obtained from the first one by applying a set of alterations, similar to the ones used in Benchmark, such as label replacement, word addition or removal in labels, node deletion and node child addition and children shuffling. Then, these two generated ontologies are used to generate alignments between them. The aim of generating the original ontology is to perform realistic tests and to allow a wider coverage of variations in their structure.

The generator proposed in [10] satisfies two requirements: (a) to generate the structure and the instances of two taxonomies, and (b) to generate the mappings between these two generated taxonomies. Both taxonomies must have a fixed size and a Boltzmann sampler is used to achieve this. The probabilistic model used ensures an equal probability of appearance of a tree having a given size. Therefore, the input data is controlled using this sampler. The number of child nodes is controlled as well. Then, the mappings between the two taxonomies are generated, which must not be contradicted by the generated data. To achieve this goal, three constraints were enforced: the mappings must not introduce a cycle in the newly obtained graph (the mappings and the two given taxonomies), the mappings must not contradict the knowledge of the two taxonomies and they must not entail each other. In the end, instances are generated.

The TaxMe method [7] is build from existing directories and only approximates the reference alignment, it is not really a generator. In XML schema matching, STBenchmark [1] offers a way to generate one precise test (pair of schemas) by altering a source schema based on the combination of 11 base alterators. Their combination is defined through a set of input parameters. Swing [6] takes a similar approach as Benchmark and introduces a new interesting way of altering ontologies by using patterns. However, it is not an automatic generator and it is aimed at generating instance data: the same ontology is, in the end, used for all tests.

We decided to rewrite our own generator because we wanted to reproduce Benchmark first. Tournaire's generator was not suited because he was aiming at realism; Besana's generator would have been useful but was not available.

3 A modular benchmark test generator

We developed a test generator in Java based on the Jena API[3]. We present the principles of the generator (Section 3.1) and the testing strategy (Section 3.2).

[3] `http://jena.sourceforge.net/ontology/index.html`

3.1 Generator principles

Test generator architecture. The basic principles of the test generator is that, from one ontology, it can generate an altered one and an alignment between these two ontologies. The generator can as well accept a generated ontology, that is useful for generating scalability tests.

Because the alterations may be applied in sequence, we designed an alterator module taking as input an ontology and an alignment between this ontology and the seed ontology. This module outputs an altered ontology and an alignment between this ontology and the seed one (Fig. 3).

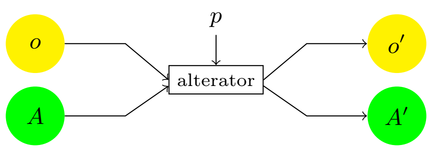


Fig. 3: Modular structure of test generators.

Test generator parameters. In order to assess the capability of matchers with respect to particular ontology features, we consider the following alterations: remove/add percentage of classes; remove/add percentage of properties; remove percentage of comments; remove percentage of restrictions; remove all classes from a level; rename percentage of classes; rename percentage of properties; add a number of classes to a specific level; flatten a level; remove individuals.

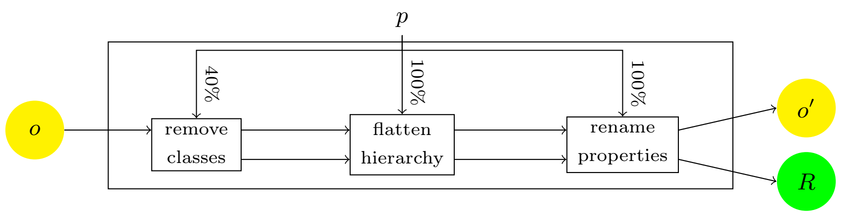


Fig. 4: Modular one-shot test generation.

Generating a dataset. For modifying an ontology according to a set of parameters we use the generator as illustrated in Fig. 4. It receives as input the seed ontology and the parameters which represent the alterations to be apply. The output is the modified ontology and the reference alignment. The program is implemented in a serial manner.

The test generator can be also used to reproduce data sets such as Benchmark. For that purpose, the program will either generate all the required tests independently by running in parallel the necessary composition of alterators

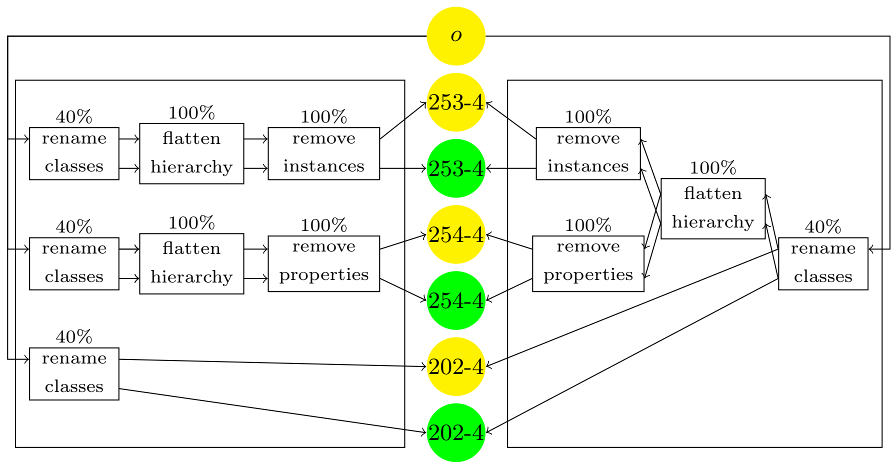


Fig. 5: Random (left) and continuous (left) test suite generation.

(Fig. 5, left) or generate them in sequence, as the initial Benchmark data set, i.e., by using a previous test and altering it further (Fig. 5, right). In the latter case, this corresponds to selecting paths in the lattice of Fig. 2 which cover the whole data set.

This approach may also be used to generate complete data sets covering the whole alteration space with a varying degree of precision (incrementing the alteration proportion by 50% or by 2%).

3.2 Preliminary experiments

Before evaluating matchers on the generated data sets, we have tested the generator through unit tests, checking if the percentage of alteration was indeed respected. Initially, the parameters were applied in a random order, using the bibliographic ontology as basis. From the results, we noticed a non expected matcher behaviour, that allowed us to improve the generation strategy.

Random vs. continuous policies. Contrary to expected, matchers did not had a continuous degradation of their performances as more alterations were applied. This made difficult to read one Benchmark test result, as developers would like to read them. This is the consequence of generating the dataset fully randomly (Fig. 5, left), where each test is generated independently from the others. In this modality, some tests with more alterations may be easier than other with less alterations by chance.

We validated this explanation by generating continuous tests (Fig. 5, right) as Benchmark were generated. In this case, new tests are generated from previous ones with the modular architecture of the generator. This behaviour is only observable locally, i.e., on one data set. When generating randomly several datasets, matcher behaviours are on average continuous. In results reported below, half of the tests were obtained with the random generation and half of them with continuous generation. Their results are the same (within 1 percentage point variation).

Modification dependencies. We observed that test difficulty may not be the same across tests supposed to have the same amount of alteration. This is explained by the dependency between alterations. Consider, for example, a scenario in which we would like to remove 60% of classes and to rename 20% of classes. According to these two parameters, three extreme cases may happen (as illustrated in Fig. 6):

- rename 20% of classes and then remove 60% of classes, including all renamed classes. In this situation, the test is easier than expected because all renamed classes have been removed;
- rename 20% of classes and then remove 60% of classes, including a part of renamed classes. In this situation, the test is as hard as expected because the required proportion of the renamed classes have been removed.
- rename 20% of classes and then remove 60% of classes, without removing a renamed class. In this situation, the test is harder than expected because none of the renamed classes has been removed.

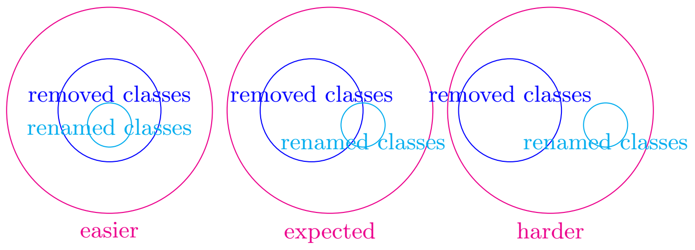


Fig. 6: Test dependency.

Hence, a random disposition of parameters might reduce the really hard cases. As can be seen from the example, the nominal expected case may be restored by removing 60% of the classes before renaming 20% of the remaining. Therefore, we established a relevant order for parameters: remove classes, remove properties, remove comments, remove restrictions, add classes, add properties, rename classes, rename properties. In this way, we obtained the expected results. This order helps determining the paths in Fig. 2 used for generating Benchmark. Such an order was not previously observed in the Benchmark tests because the value of parameters, except rename resources, was set to the value of 100%.

4 Benchmark validity

In order to test the validity of the Benchmark dataset principles, we used the test generator to reproduce them with different characteristics. Three modalities were used in the evaluation:

1. Regenerate the dataset using the same bibliographic ontology (biblio).
2. Generate datasets from two conference ontologies [11] (cmt and ekaw), of similar size and expressiveness as biblio.
3. Generate datasets from other ontologies of two other domains (tourism[4] and finance[5]).

These modalities were chosen because we assume that since participants have had the opportunity to test their systems with the original Benchmark, their results may be higher. The same holds for the conference dataset that these systems had the occasion to deal with, while the third group of tests is new for them. We thus expected them to be harder. We could evaluate the robustness of our generator, since the finance ontology has more than 300 classes and almost 2000 individuals.

In order to remove the possibility that the obtained results are an artifact of the generated test, we ran the tests five times for each method (continuous and random) and then we computed the average among the obtained results. Likewise, the tests are the same at each run (these tests are 201-202, 221-225, 228, 232-233, 236-241, 246-254, 257-262, 265-266). We decided not to reproduce the ones in which the labels are translated into another language or the ones in which the labels are replaced with their synonyms, because the corresponding alterators are not sufficiently good. The same algorithms with the same parameters have been used for all tests (the tests with the original Benchmark have been run again). In total, the results of this section are based on the execution of (1+(5 ontologies $\times$ 5 runs $\times$ 2 modalities) $\times$ 102 tests $\times$ 3 matchers =) 15606 matching tasks, i.e., a matcher has been run against a pair of ontologies and the result has been evaluated against a reference alignment.

4.1 Matchers

To test the generator, we used three different matchers participating in previous OAEI: Aroma, Anchor-Flood (Aflood) and Falcon-AO (Falcon). They are stable enough and generally available, yet sufficiently different.

Anchor-Flood tries to find alignments starting with some anchors and using the locality of a reference (super-concepts, sub-concepts, siblings, etc.) [8]. It is composed of two modules. The first one uses lexical and statistical information to extract the initial anchors. Then, starting with these anchors, it builds small blocks across ontologies and establishes the similarities between the two found blocks. Each similarity is stored in a partial alignment and, the process continues using as anchors the pairs found in the partial alignment. The process finishes when no more correspondences are found. We have used the preliminary OAEI 2010 version of Aflood.

Aroma [3] is divided in three stages. First, it builds a set of relevant terms for each entity, i.e. class or property. In order to achieve this, a single and binary

[4] http://www.bltk.ru/OWL/tourism.owl
[5] http://www.fadyart.com/ontologies/data/Finance.owl

term extractor applied to stemmed text is used to extract the vocabulary of a class or a property. In the second stage, the subsumption relations between entities are found using the implication intensity measure and an association rule model. Third, it establishes the best correspondence for each entity deducing first the equivalence relations, then suppressing the alignment graph cycles and the redundant correspondences. We have used Aroma 1.1.

Falcon-AO [9] is composed of two matchers: a linguistic (LMO) and a structural matcher (GMO). The linguistic matcher has two parts. The first one uses string comparisons and the second one uses virtual documents to describe each ontology entity. A virtual document is a “bag of words” containing the name, the comments, the labels and also neighbors names or labels of an entity. Vector space techniques are employed to measure the similarity between these virtual documents. The structural matcher represents an ontology as a bipartite graph and tries to find the similarity between the two input ontologies. In the end, if the result returned by the linguistic matcher is satisfying, it is returned. Otherwise, the structural matcher result is returned. We have used Falcon-AO 0.3.

4.2 Results

Table 1 provides the aggregated precision, recall and F-measure for each matcher and each data set. We discuss them only from the standpoint of F-measure because it allows for a more direct comparison.

	original			biblio			cmt			ekaw			tourism			finance			Σ
	P	F	R	P	F	R	P	F	R	P	F	R	P	F	R	P	F	R	F
Aflood	.99	**.87**	.78	.75	**.67**	.59	.95	**.72**	.58	.95	**.72**	.58	.94	**.76**	.62	.96	**.78**	.66	**.75**
Aroma	.79	**.63**	.53	.86	**.68**	.55	.92	**.65**	.50	.96	**.68**	.52	.85	**.74**	.64	.94	**.73**	.60	**.68**
Falcon	.83	**.76**	.70	.85	**.77**	.70	.82	**.69**	.59	.89	**.70**	.57	.83	**.73**	.65	.92	**.77**	.66	**.74**
Average		**.74**			**.71**			**.69**	.		**.70**			**.74**			**.76**		**72**

Table 1: Average results (on 5 random and 5 continuous runs) for the 2xx Benchmark series for the three matchers and six data sets.

First, we observed independently that the difficulties encountered by a matcher at a test are preserved across the seed ontology. Hence, Benchmark is useful for identifying weaknesses in matchers. What we mean here, is that by looking at the results, test by test, the relative performance of test for a matcher is preserved across seed ontologies. This is not visible in Table 1.

Then, for most of the matchers, the results obtained with the original Benchmark are not the highest. This goes counter our expectation of a positive bias in favour of the original Benchmark. In fact, the results are contrasted since Aflood has a significantly better score than with the reproduced Benchmark, Aroma has its worse score and Falcon has a very close score. It seems that the original

Benchmark is quite hard because for two matchers, results are better on the (randomised) reproduced Benchmark. Original Benchmark results, even if they do not show a systematic positive bias, seems to be the outlier.

The two other groups of tests, ekaw and cmt on one hand and tourism and finance on the other hand, have homogeneous results within the group and different results across groups. This indicates that the type of seed ontology has an influence on the results but for ontologies of the same type results are homogeneous. It seems that biblio is harder than conference which is harder than tourism-finance and this for all matchers (but Falcon).

But overall, results are found in the same range. If we exclude the results of Aflood and Aroma on the original Benchmark, the results of matchers vary of 11, 9 and 8 percentage points respectively.

Similarly, the order between matchers observed with the original Benchmark seems to be preserved in four out of six data sets. Surprisingly, the main outlier is the reproduced Benchmark. However, the few percentage point difference that is observed do not allow us to conclude, especially with respect to the 24 percentage points observed for the original Benchmark and the 10 percentage points in reproduced Benchmark.

What we observe is a relative homogeneity of these results: there is no more diversity across matchers than across data sets. In fact, the lack of discriminability observed in Benchmark is even reinforced in the other tests: the original Benchmark has 24 percentage points span while the reproduced biblio has only 10 points and the other data sets are lower. Hence, this is a property of the alterations generating Benchmark, thus another type of generation should be used.

5 Conclusion

In this paper we have looked for improving the tools available for evaluating ontology matchers. For that purpose, we have developed a test generator which follows a simple modular architecture and API. This generator does not depend on the seed ontology. It allows different modifications at each run of the program and the set of input parameters can be adjusted in order to cover the problem space with any precision. The generator can be extended by adding new ontology modifier modules and it can be used for generating individual tests with controlled characteristics as well as full data sets. Thus, we have largely improved the variability of generated tests.

The test generator was used to reproduce the Benchmark dataset not only for the bibliographic ontology, but for other ontologies with different structures. We observed that the obtained results are not better on original Benchmark than on new and different ontologies. This contradicts the assumption that there is a systematic positive bias towards Benchmark.

We observed that for the same type of seed ontology, each matcher has homogeneous results and that the order between matchers obtained on the original

Benchmark (Aflood, Falcon, Aroma) was preserved in four cases out of six. However, the difference between systems is too small to draw definitive conclusions. The new tests still lack discriminability. It is thus a feature of the Benchmark generation modalities.

We plan to improve these experiments by using other matching systems. This can be achieved using the SEALS platform and it is planned for the OAEI 2011 campaign for which the generator will be used. We may also use different more difficult generation modalities, in order to increase discriminability. Another perspective is to use the test generator for exploring the notion of test hardness, that could help to better approach the lack of discriminability.

Acknowledgements

This work has been partially supported by the SEALS project (IST-2009-238975).

References

1. Bogdan Alexe, Wang-Chiew Tan, and Yannis Velegrakis. STBenchmark: towards a benchmark for mapping systems. In *Proc. 34th Very Large Databases conference (VLDB), Auckland (NZ)*, pages 230–244, 2008.
2. Paolo Besana. *Predicting the content of peer-to-peer interactions.* PhD thesis, University of Edinburgh, 2009.
3. Jérôme David, Fabrice Guillet, and Henri Briand. Association rule ontology matching approach. *International Journal of Semantic Web and Information Systems*, 3(2):27–49, 2007.
4. Jérôme Euzenat, Christian Meilicke, Heiner Stuckenschmidt, Pavel Shvaiko, and Cássia Trojahn dos Santos. Ontology alignment evaluation initiative: Six years of experience. *Journal of Data Semantics*, XV:158–192, 2011.
5. Jérôme Euzenat and Pavel Shvaiko. *Ontology matching.* Springer-Verlag, Heidelberg (DE), 2007.
6. Alfio Ferrara, Stefano Montanelli, Jan Noessner, and Heiner Stuckenschmidt. Benchmarking matching applications on the semantic web. In *Proc. 8th Extended Semantic Web Conference (ESWC), Herssounisos (GR)*, number 6644 in Lecture notes in computer science, pages 108–122, 2011.
7. Fausto Giunchiglia, Mikalai Yatskevich, Paolo Avesani, and Pavel Shvaiko. A large scale dataset for the evaluation of ontology matching systems. *Knowledge engineering review*, 24(2):137–157, 2009.
8. Md. Seddiqui Hanif and Masaki Aono. Anchor-flood: Results for OAEI 2009. In *Proc. 4th ISWC workshop on ontology matching (OM), Washington (DC US)*, 2009.
9. Wei Hu, Yuzhong Qu, and Gong Cheng. Matching large ontologies: A divide-and-conquer approach. *Data and Knowledge Engineering*, 67(1):140–160, 2008.
10. Rémi Tournaire. *Découverte automatique de correspondances entre ontologies.* PhD thesis, Université de Grenoble, 2010.
11. Ondřej Šváb, Vojtěch Svátek, Petr Berka, Dušan Rak, and Petr Tomášek. Ontofarm: Towards an experimental collection of parallel ontologies. In *Poster Track of ISWC*, 2005.

Final results of the Ontology Alignment Evaluation Initiative 2011[⋆]

Jérôme Euzenat[1], Alfio Ferrara[2], Willem Robert van Hage[3], Laura Hollink[4], Christian Meilicke[5], Andriy Nikolov[6], François Scharffe[7], Pavel Shvaiko[8], Heiner Stuckenschmidt[5], Ondřej Šváb-Zamazal[9], and Cássia Trojahn[1]

[1] INRIA & LIG, Montbonnot, France
{jerome.euzenat,cassia.trojahn}@inria.fr
[2] Universita degli studi di Milano, Italy
ferrara@dico.unimi.it
[3] Vrije Universiteit Amsterdam, The Netherlands
W.R.van.Hage@vu.nl
[4] Delft University of Technology, The Netherlands
l.hollink@tudelft.nl
[5] University of Mannheim, Mannheim, Germany
{christian,heiner}@informatik.uni-mannheim.de
[6] The Open University, Milton Keynes, UK
A.Nikolov@open.ac.uk
[7] LIRMM, Montpellier, FR
francois.scharffe@lirmm.fr
[8] TasLab, Informatica Trentina, Trento, Italy
pavel.shvaiko@infotn.it
[9] University of Economics, Prague, Czech Republic
ondrej.zamazal@vse.cz

Abstract. Ontology matching consists of finding correspondences between semantically related entities of two ontologies. OAEI campaigns aim at comparing ontology matching systems on precisely defined test cases. These test cases can use ontologies of different nature (from simple directories to expressive OWL ontologies) and use different modalities, e.g., blind evaluation, open evaluation, consensus. OAEI-2011 builds over previous campaigns by having 4 tracks with 6 test cases followed by 18 participants. Since 2010, the campaign has been using a new evaluation modality which provides more automation to the evaluation. In particular, this year it allowed to compare run time across systems. This paper is an overall presentation of the OAEI 2011 campaign.

1 Introduction

The Ontology Alignment Evaluation Initiative[1] (OAEI) is a coordinated international initiative, which organizes the evaluation of the increasing number of ontology matching systems [10; 8; 15]. The main goal of OAEI is to compare systems and algorithms

[⋆] This paper improves on the "First results" initially published in the on-site proceedings of the ISWC workshop on Ontology Matching (OM-2011). The only official results of the campaign, however, are on the OAEI web site.

[1] http://oaei.ontologymatching.org

on the same basis and to allow anyone for drawing conclusions about the best matching strategies. Our ambition is that, from such evaluations, tool developers can improve their systems.

Two first events were organized in 2004: (i) the Information Interpretation and Integration Conference (I3CON) held at the NIST Performance Metrics for Intelligent Systems (PerMIS) workshop and (ii) the Ontology Alignment Contest held at the Evaluation of Ontology-based Tools (EON) workshop of the annual International Semantic Web Conference (ISWC) [17]. Then, a unique OAEI campaign occurred in 2005 at the workshop on Integrating Ontologies held in conjunction with the International Conference on Knowledge Capture (K-Cap) [1]. Starting from 2006 through 2010 the OAEI campaigns were held at the Ontology Matching workshops collocated with ISWC [9; 7; 2; 5; 6]. Finally in 2011, the OAEI results were presented again at the Ontology Matching workshop collocated with ISWC, in Bonn, Germany[2].

Since last year, we have been promoting an environment for automatically processing evaluations (§2.2), which were developed within the SEALS (Semantic Evaluation At Large Scale) project[3]. This project aims at providing a software infrastructure for automatically executing evaluations, and evaluation campaigns for typical semantic web tools, including ontology matching. Several OAEI data sets were evaluated under the SEALS modality. This provides a more uniform evaluation setting.

This paper serves as a synthesis to the 2011 evaluation campaign and as an introduction to the results provided in the papers of the participants. The remainder of the paper is organized as follows. In Section 2, we present the overall evaluation methodology that has been used. Sections 3-6 discuss the settings and the results of each of the test cases. Section 7 overviews lessons learned from the campaign. Finally, Section 8 outlines future plans and Section 9 concludes the paper.

2 General methodology

We first present the test cases proposed this year to the OAEI participants (§2.1). Then, we discuss the resources used by participants to test their systems and the execution environment used for running the tools (§2.2). Next, we describe the steps of the OAEI campaign (§2.3-2.5) and report on the general execution of the campaign (§2.6).

2.1 Tracks and test cases

This year's campaign consisted of 4 tracks gathering 6 data sets and different evaluation modalities:

The benchmark track (§3): Like in previous campaigns, a systematic benchmark series have been proposed. The goal of this benchmark series is to identify the areas in which each matching algorithm is strong or weak. The test is based on one particular ontology dedicated to the very narrow domain of bibliography and a number of alternative ontologies of the same domain for which reference alignments are

[2] `http://om2011.ontologymatching.org`
[3] `http://www.seals-project.eu`

provided. This year, we used new systematically generated benchmarks, based on other ontologies than the bibliographic one.

The expressive ontologies track offers real world ontologies using OWL modeling capabilities:

Anatomy (§4): The anatomy real world case is about matching the Adult Mouse Anatomy (2744 classes) and a part of the NCI Thesaurus (3304 classes) describing the human anatomy.

Conference (§5): The goal of the conference task is to find all correct correspondences within a collection of ontologies describing the domain of organizing conferences (the domain being well understandable for every researcher). Additionally, 'interesting correspondences' are also welcome. Results were evaluated automatically against reference alignments and by using logical reasoning techniques.

Oriented alignments: This track focused on the evaluation of alignments that contain other relations than equivalences. It provides two data sets of real ontologies taken from a) Academia (alterations of ontologies from the OAEI benchmark series), b) Course catalogs (alterations of ontologies concerning courses in the universities of Cornell and Washington). The alterations aim to introduce additional subsumption correspondences between classes that cannot be inferred via reasoning.

Model matching: This data set compares model matching tools from the Model-Driven Engineering (MDE) community on ontologies. The test cases are available in two formats: OWL and Ecore. The models to be matched have been automatically derived from a model-based repository.

Instance matching (§6): The goal of the instance matching track is to evaluate the performance of different tools on the task of matching RDF individuals which originate from different sources but describe the same real-world entity. Instance matching is organized in two sub-tasks:

Data interlinking (DI) This year the Data interlinking track focused on retrieving New York Times (NYT) interlinks with DBPedia, Freebase and Geonames. The NYT data set includes 4 data subsets: persons, locations, organizations and descriptors that should be matched to themselves to detect duplicates, and to DBPedia, Freebase and Geonames. Only Geonames has links to the Locations data set of NYT.

OWL data track (IIMB): The synthetic OWL data track is focused on (i) providing an evaluation data set for various kinds of data transformations, including value transformations, structural transformations, and logical transformations; (ii) covering a wide spectrum of possible techniques and tools. To this end, the IIMB benchmark is generated by starting from an initial OWL knowledge base that is transformed into a set of modified knowledge bases by applying several automatic transformations of data. Participants are requested to find the correct correspondences among individuals of the first knowledge base and individuals of the others.

Table 1 summarizes the variation in the results expected from the tests under consideration.

test	formalism	relations	confidence	modalities	language	SEALS
benchmarks	OWL	=	[0 1]	open	EN	√
anatomy	OWL	=	[0 1]	open	EN	√
conference	OWL-DL	=, <=	[0 1]	blind+open	EN	√
di	RDF	=	[0 1]	open	EN	
iimb	RDF	=	[0 1]	open	EN	

Table 1. Characteristics of the test cases (open evaluation is made with already published reference alignments and blind evaluation is made by organizers from reference alignments unknown to the participants).

This year we had to cancel the Oriented alignments and Model matching tracks which have not had enough participation. We preserved the IIMB track with only one participant, especially because the participant was not tied to the organizers and participated in the other tracks as well.

2.2 The SEALS platform

In 2010, participants of the Benchmark, Anatomy and Conference tracks were asked for the first time to use the SEALS evaluation services: they had to wrap their tools as web services and the tools were executed on the machines of the tool developers [18].

In 2011, tool developers had to implement a simple interface and to wrap their tools in a predefined way including all required libraries and resources. A tutorial for tool wrapping was provided to the participants. This tutorial described how to wrap a tool and how to use a simple client to run a full evaluation locally. After local tests had been conducted successfully, the wrapped tool was uploaded for a test on the SEALS portal[4]. Consequently it was executed on the SEALS platform by the organisers in a semi-automated way. This approach allowed to measure runtime and ensured the reproducibility of the results for the first time in the history of OAEI. As a side effect, this approach ensures also that a tool is executed with the same settings for all of the three tracks. This was already requested by the organizers in the past years. However, this rule was sometimes ignored by participants.

2.3 Preparatory phase

Ontologies to be matched and (where applicable) reference alignments have been provided in advance during the period between May 30^{th} and June 27^{th}, 2011. This gave potential participants the occasion to send observations, bug corrections, remarks and other test cases to the organizers. The goal of this preparatory period is to ensure that the delivered tests make sense to the participants. The final test base was released on July 6^{th}, 2011. The data sets did not evolve after that, except for the reference alignment of the Anatomy track to which minor changes have been applied to increase its quality.

[4] `http://www.seals-project.eu/join-the-community/`

2.4 Execution phase

During the execution phase, participants used their systems to automatically match the ontologies from the test cases. In most cases, ontologies are described in OWL-DL and serialized in the RDF/XML format [3]. Participants were asked to use one algorithm and the same set of parameters for all tests in all tracks. It is fair to select the set of parameters that provides the best results (for the tests where results are known). Beside parameters, the input of the algorithms must be the two ontologies to be matched and any general purpose resource available to everyone, i.e., no resource especially designed for the test. In particular, participants should not use the data (ontologies and reference alignments) from other test cases to help their algorithms.

Participants can self-evaluate their results either by comparing their output with reference alignments or by using the SEALS client to compute precision and recall.

2.5 Evaluation phase

Participants have been encouraged to provide (preliminary) results or to upload their wrapped tools on the SEALS portal by September 1^{st}, 2011. Organizers evaluated the results and gave feedback to the participants. For the SEALS modality, a full-fledged test on the platform has been conducted by the organizers and problems were reported to the tool developers, until finally a properly executable version of the tool has been uploaded on the SEALS portal. Participants were asked to send their final results or upload the final version of their tools by September 23^{th}, 2011. Participants also provided the papers that are published hereafter.

As soon as first results were available, these results were published on the respective web pages by the track organizers. The standard evaluation measures are precision and recall computed against the reference alignments. For the matter of aggregation of the measures, we used weighted harmonic means (weights being the size of the true positives). This clearly helps in the case of empty alignments. Another technique that was used is the computation of precision/recall graphs so it was advised that participants provide their results with a weight to each correspondence they found. New measures addressing some limitations of precision and recall have also been used for testing purposes as well as measures for compensating the lack of complete reference alignments. Additionally, we measured runtimes for all tracks conducted under the SEALS modality.

2.6 Comments on the execution

For a few years, the number of participating systems has remained roughly stable: 4 participants in 2004, 7 in 2005, 10 in 2006, 17 in 2007, 13 in 2008, 16 in 2009, 15 in 2010 and 18 in 2011. However, participating systems are now constantly changing.

The number of covered runs has increased more than observed last year: 48 in 2007, 50 in 2008, 53 in 2009, 37 in 2010, and 53 in 2011. This is, of course, due to the ability to run all systems participating in the SEALS modality in all tracks. However, not all tools participating in the SEALS modality could generate results for the anatomy track (see Section 4). This does not really contradict the conjecture we made last year that systems are more specialized. In fact, only two systems (AgreementMaker and CODI) participated also in the instance matching tasks, and CODI only participated in a task (IIMB) in which no other instance matching system entered.

This year we were able to run most of the matchers in a controlled evaluation environment, in order to test their portability and deployability. This allowed us comparing systems on the same execution basis. This is also a guarantee that the tested system can be executed out of their particular development environment.

The list of participants is summarized in Table 2.

System	AgrMaker	Aroma	CSA	CIDER	CODI	LDOA	Lily	LogMap	MaasMtch	MapEVO	MapPSO	MapSSS	OACAS	OMR	Optima	Serimi	YAM++	Zhishi	Total=18
Confidence	√	√	√	√		√	√	√	√	√	√		√	√		√	√	√	
benchmarks	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√		√		16
anatomy	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√		√		16
conference	√	√	√	√	√	√	√	√	√	√	√	√			√		√		14
di	√															√		√	3
iimb					√														1
Total	4	3	3	3	4	3	3	3	3	3	3	3	2	2	3	1	3	1	53

Table 2. Participants and the state of their submissions. Confidence stands for the type of result returned by a system: it is ticked when the confidence has been measured as non boolean value.

Two systems require a special remark. YAM++ used a setting that was learned from the reference alignments of the benchmark data set from OAEI 2009, which is highly similar to the corresponding benchmark in 2011. This affects the results of the traditional OAEI benchmark and no other tests. Moreover, we have run the benchmark in newly generated tests where YAM++ is indeed having weaker performances. Considering that indeed benchmarks was one of the few tests on which to train algorithms, we decided to keep YAM++ results with this warning.

AgreementMaker used machine learning techniques to choose automatically between one of three settings optimized for the benchmark, anatomy and conference data set. It used a subset of the available reference alignments as input to the training phase and clearly a specific tailored setting for passing these tests. This is typically prohibited by OAEI rules. However, at the same time, AgreementMaker has improved its results over last year so we found interesting to report them.

The summary of the results track by track is provided in the following sections.

3 Benchmark

The goal of the benchmark data set is to provide a stable and detailed picture of each algorithm. For that purpose, algorithms are run on systematically generated test cases.

3.1 Test data

The systematic benchmark test set is built around a seed ontology and many variations of it. The ontologies are described in OWL-DL and serialized in the RDF/XML format. The reference ontology is that of test #101. Participants have to match this reference ontology with the variations. Variations are focused on the characterization of the behavior of the tools rather than having them compete on real-life problems. They are organized in three groups:

Simple tests (1xx) such as comparing the reference ontology with itself, with another irrelevant ontology (the wine ontology used in the OWL primer) or the same ontology in its restriction to OWL-Lite;

Systematic tests (2xx) obtained by discarding features from a reference ontology. It aims at evaluating how an algorithm behaves when a particular type of information is lacking. The considered features were:

- *Name of entities* that can be replaced by random strings, synonyms, name with different conventions, strings in another language than English;
- *Comments* that can be suppressed or translated in another language;
- *Specialization hierarchy* that can be suppressed, expanded or flattened;
- *Instances* that can be suppressed;
- *Properties* that can be suppressed or having the restrictions on classes discarded;
- *Classes* that can be expanded, i.e., replaced by several classes or flattened.

Four real-life ontologies of bibliographic references (3xx) found on the web and left mostly untouched (there were added xmlns and xml:base attributes). This is only used for the initial benchmark.

This year, we departed from the usual bibliographic benchmark that have been used since 2004. We used a new test generator [14] in order to reproduce the structure of benchmark from different seed ontologies. We have generated three different benchmarks against which matchers have been evaluated:

benchmark (biblio) is the benchmark data set that has been used since 2004. It is used for participants to check that they can run the tests. It also allows for comparison with other systems since 2004. The seed ontology concerns bibliographic references and is inspired freely from BibTeX. It contains 33 named classes, 24 object properties, 40 data properties, 56 named individuals and 20 anonymous individuals. We have considered the original version of benchmark (referred as **original** in the subsections above) and a new automatically generated one (**biblio**).

benchmark2 (ekaw) The Ekaw ontology, one of the ontologies from the conference track (§5), was used as seed ontology for generating the Benchmark2 data set. It contains 74 classes and 33 object properties. The results with this new data set were provided to participants after the preliminary evaluation.

benchmark3 (finance) This data set is based on the Finance ontology[5], which contains 322 classes, 247 object properties, 64 data properties and 1113 named individuals. This ontology was not disclosed to the participants.

[5] http://www.fadyart.com/ontologies/data/Finance.owl

Having these three data sets allows us to better evaluate the dependency between the results and the seed ontology. The SEALS platform allows for evaluating matchers against these many data sets automatically.

For all data sets, the reference alignments are still limited: they only match named classes and properties and use the "=" relation with confidence of 1. Full description of these tests can be found on the OAEI web site.

3.2 Results

16 systems have participated in the benchmark track of this year's campaign (see Table 2). From the eleven participants last year, only four participated this year (AgreementMaker, Aroma, CODI and MapPSO). On the other hand, there are ten new participants, while two participants (CIDER and Lily) have been participating in the previous campaigns as well. In the following, we present the evaluation results, both in terms of runtime and compliance with relation to reference alignments.

Portability. 18 systems have been registered on the SEALS portal. One has abandoned due to requirements posed by the platform and another one abandoned silently. Thus, 16 systems bundled their tools into the SEALS format. From these 16 systems, we have not been able to run the final versions of OMR and OACAS (packaging error). CODI was not working on the operating system version under which we measured runtime. CODI runs under Windows and some versions of Linux, but has specific requirements not met on the Linux version that has been used for runnning the SEALS platform (Fedora 8). Some other systems still have (fixable) problems with the output they generate[6].

Runtime. This year we were able to measure the performance of matchers in terms of runtime. We used a 3GHz Xeon 5472 (4 cores) machine running Linux Fedora 8 with 8GB RAM. This is a very preliminary setting for mainly testing the deployability of tools into the SEALS platform.

Table 3 presents the time required by systems to complete the 94 tests in each data set[7]. These results are based on 3 runs of each matcher on each data sets. We also include the result of a simple edit distance algorithm on labels (edna). Unfortunately, we were not able to compare CODI's runtime with other systems'.

Considering all tasks but finance, there are systems which can run them within less than 15mn (Aroma, edna, LogMap, CSA, YAM++, MapEVO, AgreementMaker, MapSSS), there are systems performing the tasks within one hour (Cider, MaasMatch, Lily) and systems which need more time (MapPSO, LDOA, Optima). Figure 1 better illustrates the correlation between the number of elements in each seed ontology and the time taken by matchers for generating the 94 alignments. The faster matcher, independently from the seed ontology, is Aroma (even for finance), followed by LogMap,

[6] All evaluations have been performed with the Alignment API 4.2 [3] with the exception of LDOA for which we had to adapt the relaxed evaluators to obtain results.

[7] From the 111 tests in the original benchmark data set, 17 of them have not been automatically generated: 102–104, 203–210, 230–231, 301–304. For comparative purposes, they were discarded.

	original		**biblio**		**ekaw**		**finance**	
System	Runtime	Top-5	Runtime	Top-5	Runtime	Top-5	Runtime	Top-5
edna	1.07		1.06		1.00		33.70	
AgrMaker	12.42	√	—x		2.81	√	3.81h	√
Aroma	1.05		1.10	√	0.77		10.83	√
CSA	2.47	√	2.61	√	3.69	√	3.10h	√
CIDER	32.50		30.30		28.08		**46.15h**	√
CODI	—Error	√	—Error	√	—Error	√	—Error	
LDOA	28.94h		29.31h		17h		—T	
Lily	48.60		48.18	√	8.76		—T	
LogMap	2.45		2.47		2.16		—Error	
MaasMtch	28.32		36.06		35.87		29.23h	√
MapEVO	6.77		7.44		9.96		1.25h	
MapPSO	3.05h		3.09h		3.72h		**85.98h**	
MapSSS	8.84	√	—x		4.42	√	—x	
OACAS	—Error		—Error		—Error		—Error	
OMR	—Error		—Error		—Error		—Error	
Optima	3.15h		2.48h		**88.80h**		—T	
YAM++	6.51	√	6.68	√	8.02	√	—T	

Table 3. Runtime (in minutes) based on 3 runs, and the five best systems in terms of F-measure in each data set (top-5). 'Error' indicates that the tool could not run in the current setting; or their final version has some packaging error. 'T' indicates that tool could not process the single 101 test in less than 2 hours. 'x' indicates that the tool breaks when parsing some ontologies. Results in bold face are based on only 1 run.

CSA, YAM++, AgreementMaker (AgrMaker) and MapSSS. Furthermore, as detailed in the following, AgreementMaker, CSA, CODI and YAM++ are also the best systems for most of the different data sets.

For finance, we observed that many participants were not able to deal with large ontologies. This applies to the slowest systems of the other tasks, but other problems occur with AgreementMaker and MapSSS. Fast systems like LogMap could not process some of the test cases due to the inconsistency of the finance ontology (as CODI). Finally, other relatively fast systems such as YAM++ and Lily had to time out. We plan to work on these two issues in the next campaigns.

Compliance. Concerning compliance, we focus on the benchmark2 (ekaw) data set. Table 4 shows the results of participants as well as those given by edna (simple edit distance algorithm on labels). The full results are on the OAEI web site.

As shown in Table 4, two systems achieve top performance in terms of F-measure: MapSSS and YAM++, with CODI, CSA and AgreementMaker as close followers, respectively. Lily and CIDER had presented intermediary values of precision and recall. All systems achieve a high level of precision and relatively low values of recall. Only MapEVO had a significantly lower recall than edna (with LogMap and MaasMatch (MaasMtch) with slightly lower values), while no system had lower precision.

system	refalign			edna			AgrMaker			Aroma			CSA			CIDER			CODI			LDOA		
test	Prec.	FMeas.	Rec.	Prec.	FMeas.	Rec.	Prec.	FMeas.	Rec.	Prec.	FMeas.	Rec.	Prec.	FMeas.	Rec.	Prec.	FMeas.	Rec.	Prec.	FMeas.	Rec.	Prec.	FMeas.	Rec.
1xx	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	.96	1.00	.96	.25	.19	.15	.77	.86	.97
2xx	1.00	1.00	1.00	.50	.51	.51	.98	.71	.56	.93	.68	.53	.82	.72	.64	.89	.70	.58	.94	.74	.61	.51	.51	.51
H-mean	1.00	1.00	1.00	.50	.51	.52	.98	.71	.56	.93	.68	.53	.82	.73	.65	.89	.70	.58	.93	.73	.60	.51	.51	.51
Symmetric	1.00	1.00	1.00	.53	.53	.54	.98	.71	.56	.94	.68	.54	.83	.73	.66	.91	.71	.59	.94	.73	.61	.52	.52	.52
Effort	1.00	1.00	1.00	.53	.53	.55	.98	.71	.56	.94	.68	.54	.84	.73	.66	.90	.71	.59	.93	.72	.60	.52	.52	.52
P-oriented	1.00	1.00	1.00	.56	.56	.57	.98	.71	.56	.95	.68	.54	.84	.74	.67	.92	.72	.60	.95	.74	.61	.52	.52	.52
R-oriented	1.00	1.00	1.00	.56	.56	.57	.98	.71	.56	.95	.68	.54	.84	.74	.67	.92	.72	.60	.95	.74	.61	.52	.52	.52
Weighted	1.00	1.00	1.00	.71	.60	.52	.98	.71	.56	.95	.64	.49	.87	.58	.44	.91	.68	.54	.93	.73	.60	.57	.52	.48

	Lily			LogMap			MaasMtch			MapEVO			MapPSO			MapSSS			YAM++			Optima		
test	Prec.	FMeas.	Rec.	Prec.	FMeas.	Rec.	Prec.	FMeas.	Rec.	Prec.	FMeas.	Rec.	Prec.	FMeas.	Rec.	Prec.	FMeas.	Rec.	Prec.	FMeas.	Rec.	Prec.	FMeas.	Rec.
1xx	1.00	1.00	1.00	1.00	1.00	1.00	1.00	.99	.98	.99	1.00	1.00	.99	.96	.92	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
2xx	.93	.70	.57	.99	.66	.49	.99	.60	.43	.54	.31	.21	.63	.63	.62	.96	.77	.64	.97	.74	.60	.59	.55	.52
H-mean	.93	.70	.57	.99	.67	.50	.99	.61	.44	.55	.32	.22	.64	.63	.62	.96	.77	.64	.97	.74	.60	.60	.56	.53
Symmetric	.93	.71	.58	.99	.67	.50	.99	.61	.44	.63	.33	.22	.66	.64	.63	.97	.77	.64	.97	.74	.60	.60	.56	.53
Effort	.93	.71	.58	.99	.67	.50	.99	.61	.44	.63	.33	.23	.66	.64	.63	.97	.77	.64	.98	.74	.60	.61	.56	.53
P-oriented	.94	.71	.58	.99	.67	.50	.99	.61	.44	.64	.33	.23	.68	.66	.65	.97	.78	.65	.98	.74	.60	.61	.56	.54
R-oriented	.94	.71	.58	.99	.67	.50	.99	.61	.44	.64	.33	.23	.68	.66	.65	.97	.78	.65	.98	.74	.60	.61	.56	.54
Weighted	.95	.56	.39	.99	.55	.38	.99	.61	.44	.75	.34	.22	.76	.58	.47	.96	.77	.64	.99	.14	.07	.60	.56	.53

Table 4. Results obtained by participants on the Benchmark2 (ekaw) test case (harmonic means). Relaxed precision and recall correspond to the three measures of [4]: symmetric proximity, correction effort and oriented (precision and recall). Weighted precision and recall takes into account the confidence associated to correspondence by the matchers.

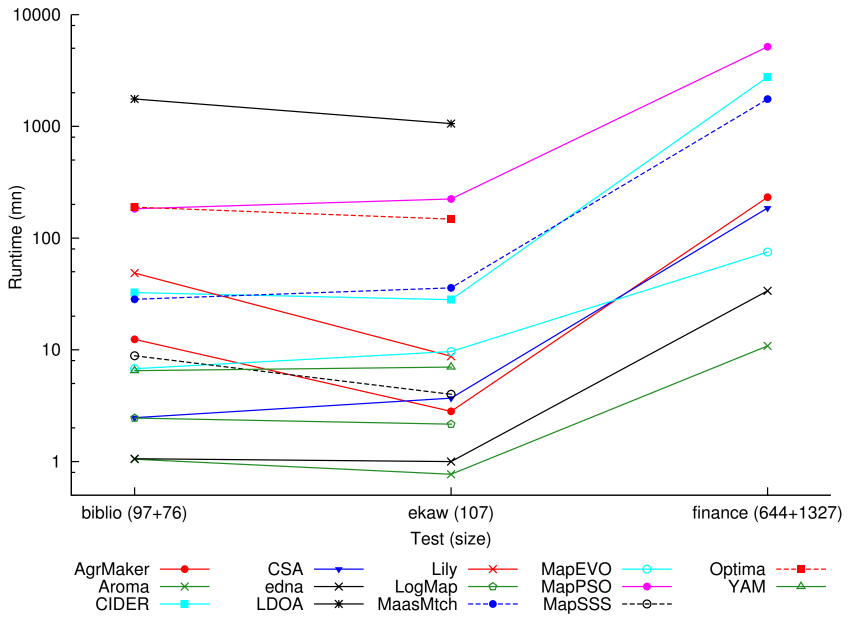


Fig. 1. Logarithmic plot of the time taken by matchers (averaged on 3 runs) to deal with different data sets: biblio, ekaw and finance

Looking at each group of tests, in simple tests (1xx) all systems have similar performance, excluding CODI. As noted in previous campaigns, the algorithms have their best score with the 1xx test series. This is because there are no modifications in the labels of classes and properties in these tests and basically all matchers are able to deal with the heterogeneity in labels. Considering that Benchmark2 has one single test in 1xx, the discriminant category is 2xx, with 101 tests. For this category, the top five systems in terms of F-measure (as stated above) are: MapSSS, YAM++, CODI, CSA and AgreementMaker, respectively (CIDER and Lily as followers).

Many algorithms have provided their results with confidence measures. It is thus possible to draw precision/recall graphs in order to compare them. Figure 2 shows the precision and recall graphs. These results are only relevant for the results of participants who provide confidence measures different from 1 or 0 (see Table 2). As last year, they show the real precision at n% recall and they stop when no more correspondences are available (then the end point corresponds to the precision and recall reported in Table 4).

The results have also been compared with the relaxed measures proposed in [4], namely symmetric proximity, correction effort and oriented measures ('Symmetric', 'Effort', 'P/R-oriented' in Table 4). Table 4 shows that these measures provide a uniform and limited improvement to most systems. As last year, the exception is MapEVO, which has a considerable improvement in precision. This could be explained by the fact this system misses the target, by not that far (the false negative correspondences found by the matcher are close to the correspondences in the reference alignment) so the gain provided by the relaxed measures has a considerable impact for this system. This may also be explained by the global optimization of the system which tends to be globally roughly correct as opposed to locally strictly correct as measured by precision and recall.

The same confidence-weighted precision and recall as last year have been computed. They reward systems able to provide accurate confidence measures (or penalizes less mistakes on correspondences with low confidence) [6]. These measures provide precision increasing for most of the systems, specially edna, MapEVO and MapPSO (which had possibly many incorrect correspondences with low confidence). This shows that the simple edit distance computed by edna is valuable as a confidence measure (the weighted precision and recall for edna could be taken as a decent baseline). It also provides recall decrease specially for CSA, Lily, LogMap, MapPSO and YAM++ (which had apparently many correct correspondences with low confidence). The variation for YAM++ is quite impressive: this is because YAM++ provides especially low confidence to correct correspondences. Some systems, such as AgreementMaker, CODI, MaasMatch and MapSSS, generate all correspondences with confidence = 1, so they have no change.

Comparison across data sets. Table 5 presents the average F-measure for 3 runs, for each data set (as Table 3, some of these results are based on only one run). These three runs are not necessary: even if matchers exhibit non deterministic behavior on a test case basis, their average F-measure on the whole data set remains the same [14]. This year, although most of the systems participating in 2010 have improved their algorithms, none of them could outperform ASMOV, the best system in the 2010 campaign.

With respect to the original benchmark data set and the new generated one (original and biblio in Table 5), we could observe a 1-2% constant and negative variation in F-measure, for most of the systems (except CODI and MapEVO). Furthermore, most of the systems perform better with the bibliographic ontology than with ekaw (a variation of 5-15%). The exceptions are LDOA, LogMap and MapPSO, followed by MaasMatch and CIDER with relatively stable F-measures. Although we have not enough results for a fair comparison with finance, we could observe that CSA and MaasMatch are the most stable matchers (with less variation than the others), followed by Aroma, CIDER and AgreementMaker, respectively.

Finally, the group of best systems in each data set remains relatively the same across the different seed ontologies. Disregarding finance, CSA, CODI and YAM++ are ahead as the best systems for all three data sets, with MapSSS (2 out of 3) and AgreementMaker, Aroma and Lily (1 out of 3) as followers.

3.3 Conclusions

For the first time, we could observe a high variation in the time matchers require to complete the alignment tasks (from some minutes to some days). We can also conclude that compliance is not proportional to runtime: the top systems in terms of F-measure were able to finish the alignment tasks in less than 15mn (with Aroma and LogMap as faster matchers, with intermediary levels of compliance). Regarding the capability of dealing with large ontologies, many of the participants were not able to process them, leaving room for further improvement on this issue.

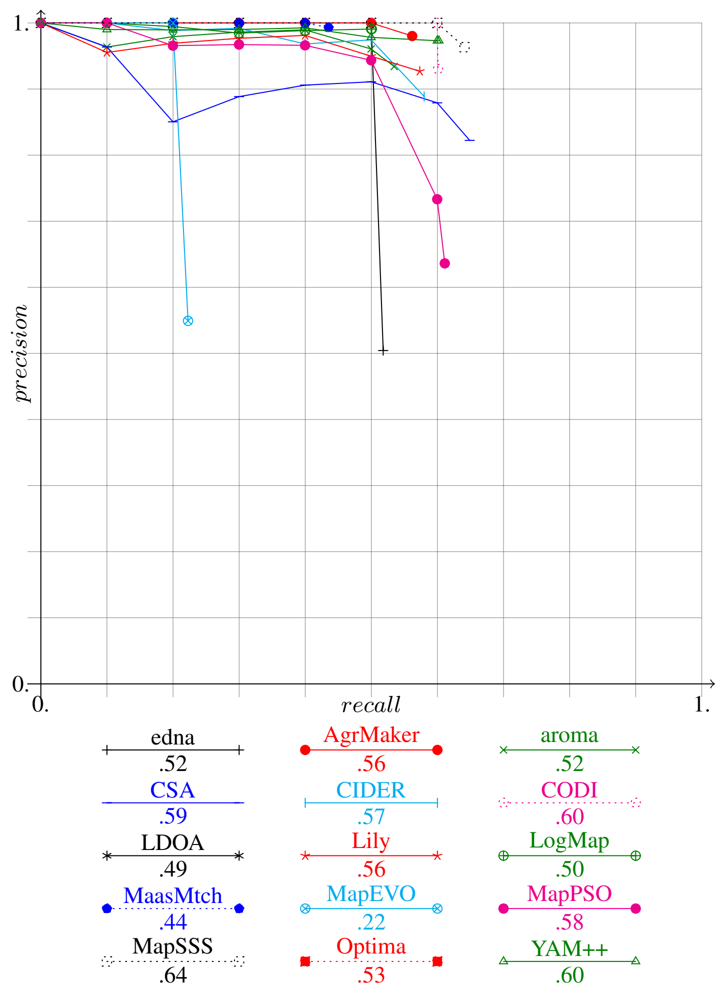


Fig. 2. Precision/recall graphs for benchmarks. The alignments generated by matchers are cut under a threshold necessary for achieving $n\%$ recall and the corresponding precision is computed. The numbers in the legend are the Mean Average Precision (MAP): the average precision for each correct retrieved correspondence. Systems for which these graphs are not meaningful (because they did not provide graded confidence values) are drawn in dashed lines.

	2010		**2011**							
	original		**original**		**biblio**		**ekaw**		**finance**	
System	Fmeas.	Top-5	Fmeas.	Top-5	Fmeas.	Top-5	Fmeas.	Top-5	Fmeas.	Top-5
ASMOV	.93	√								
AgrMaker	.89	√	.88	√	x		.71		.78	
Aroma	.59		.78		.76	√	.68		.70	√
CSA			.84	√	.83	√	.73	√	.79	√
CIDER			.76		.74		.70		.67	√
CODI	.55		.80	√	.75	√	.73	√	x	
edna	.51		.52		.51		.51		.50	
LDOA			.47		.46		.52		T	
Lily			.76		.77	√	.70		T	
LogMap			.60		.57		.66		x	
MaasMtch			.59		.58		.61		.61	√
MapEVO			.41		.37		.33		.20	
MapPSO	.61		.50		.48		.63		.14	
MapSSS			.84	√	x		.78	√	T	
Optima			.64		.65		.56		T	
YAM++			.87	√	.86	√	.75	√	T	

Table 5. Results obtained by participants on each data set (based on 94 tests), including the results from the participants in 2010, and the top-five F-measure (five better systems in each data set).

With respect to compliance, newcomers (CSA, YAM++ and MapSSS) have mostly outperformed other participants, for the new generated benchmarks. On the other hand, for the very known original benchmark data set, none of the systems was able to outperform the top-performer of the last year (ASMOV).

4 Anatomy

As in the previous years, the anatomy track confronts the existing matching technology with a specific type of ontologies from the biomedical domain. In this domain, many ontologies have been built covering different aspects of medical research. We focus on fragments of two biomedical ontologies which describe the human anatomy and the anatomy of the mouse. The data set of this track has been used since 2007. For a detailed description we refer the reader to the OAEI 2007 results paper [7].

4.1 Experimental setting

Contrary to the previous years, we distinguish only between two evaluation experiments. Subtask #1 is about applying a matcher with its standard setting to the matching task. In the previous years we have also asked for additional alignments that favor precision over recall and vice versa (subtask #2 and #3). These subtasks are not part of the anatomy track in 2011 due to the fact that the SEALS platform does not allow for running tools with different configurations. Furthermore, we have proposed a fourth subtask, in which a partial reference alignment has to be used as an additional input.

In our experiments we compare precision, recall, F-measure and recall+. We have introduced recall+ to measure the amount of detected non-trivial correspondences. From 2007 to 2009, we reported on runtimes measured by the participants themselves. This survey revealed large differences in runtimes. This year we can compare the runtimes of participants by executing them on our own on the same machine. We used a Windows 2007 machine with 2.4 GHz (2 cores) and 7GB RAM allocated to the matching systems.

For the 2011 evaluation, we improved again the reference alignment of the data set. We removed doubtful correspondences and included several correct correspondences that had not been included in the past. As a result, we measured for the alignments generated in 2010 a slightly better F-measure ($\approx$+1%) compared to the computation based on the old reference alignment. For that reason we have also included the top-3 systems of 2010 with recomputed precision/recall scores.

4.2 Results

In the following we analyze the robustness of the submitted systems and their runtimes. Further, we report on the quality of the generated alignment, mainly in terms of precision and recall.

Robustness and scalability. In 2011 there were 16 participants in the SEALS modality, while in 2010 we had only 9 participants for the anatomy track. However, this comparison is misleading. Some of these 16 systems are not really intended to match large biomedical ontologies. For that reason our first interest is related to the question, which systems generate a meaningful result in an acceptable time span. Results are shown in Table 6. First, we focused on the question whether systems finish the matching task in less than 24h. This is the case for a surprisingly low number of systems. The systems that do not finish in time can be separated in those systems that throw an exception related to insufficient memory after some time (marked with 'X'). The other group of systems were still running when we stopped the experiments after 24 hours (marked with 'T').[8]

Obviously, matching relatively large ontologies is a problem for five out of fourteen executable systems. The two systems MapPSO and MapEVO can cope with ontologies that contain more than 1000 concepts, but have problems with finding correct correspondences. Both systems generate comprehensive alignments, however, MapPSO finds only one correct corespondence and MapEVO finds none. This can be related to the way labels are encoded in the ontologies. The ontologies from the anatomy track differ from the ontologies of the benchmark and conference tracks in this respect.

Matcher	Runtime	Size	Precision	F-measure	Recall	Recall+
AgrMaker	634	1436	.943	.917	.892	.728
LogMap	24	1355	.948	.894	.846	.599
AgrMaker_{2010}	-	1436	.914	.890	.866	.658
CODI	1890	1298	.965	.889	.825	.564
NBJLM_{2010}	-	1327	.931	.870	.815	.592
Ef2Match_{2010}	-	1243	.965	.870	.792	.455
Lily	563	1368	.814	.772	.734	.511
StringEquiv	-	*934*	*.997*	*.766*	*.622*	*.000*
Aroma	39	1279	.742	.679	.625	.323
CSA	4685	2472	.465	.576	.757	.595
MaasMtch	66389	438	.995	.445	.287	.003
MapPSO	9041	2730	.000	.000	.001	.000
MapEVO	270	1079	.000	.000	.000	.000
Cider	T	0	-	-	-	-
LDOA	T	0	-	-	-	-
MapSSS	X	0	-	-	-	-
Optima	X	0	-	-	-	-
YAM++	X	0	-	-	-	-

Table 6. Comparison against the reference alignment, runtime is measured in seconds, the size column refers to the number of correspondences in the generated alignment.

For those systems that generate an acceptable result, we observe a high variance in measured runtimes. Clearly ahead is the system LogMap (24s), followed by Aroma (39s). Next are Lily and AgreementMaker (approx. 10mn), CODI (30mn), CSA (1h15), and finally MaasMatch (18h).

Results for subtask #1. The results of our experiments are also presented in Table 6. Since we have improved the reference alignment, we have also included recomputed precision/recall scores for the top-3 alignments submitted in 2010 (marked by subscript 2010). Keep in mind that in 2010 AgreementMaker (AgrMaker) submitted an alignment that was the best submission to the OAEI anatomy track compared to all previous

[8] We could not execute the two systems OACAS and OMR, not listed in the table, because the required interfaces have not been properly implemented.

submissions in terms of F-measure. Note that we also added the base-line *StringEquiv*, which refers to a matcher that compares the normalized labels of two concepts. If these labels are identical, a correspondence is generated. Recall+ is defined as recall, with the difference that the reference alignment is replaced by the set difference of $R \setminus A_{SE}$, where A_{SE} is defined as the alignment generated by *StringEquiv*.

This year we have three systems that generate very good results, namely AgreementMaker, LogMap and CODI. The results of LogMap and CODI are very similar. Both systems manage to generate an alignment with F-measure close to the 2010 submission of AgreementMaker. LogMap is slightly ahead. However, in 2011 the alignment generated by AgreementMaker is even better than in the previous year. In particular, AgreementMaker finds more correct correspondences, which can be seen in recall as well as in recall+ scores. At the same time, AgreementMaker can increase its precision. Also remarkable are the good results of LogMap, given the fact that the system finishes the matching task in less than half a minute. It is thus 25 times faster than AgreementMaker and more than 75 times faster than CODI.

Lily, Aroma, CSA, and MaasMatch (MaasMatch) have less good results than the three top matching systems, however, they have proved to be applicable to larger matching tasks and can generate acceptable results for a pair of ontologies from the biomedical domain. While these systems cannot (or barely) top the String-Equivalence baseline in terms of F-measure, they manage, nevertheless, to generate many correct non-trivial correspondences. A detailed analysis of the results revealed that they miss at the same time many trivial correspondences. This is an uncommon result, which might, for example, be related to some pruning operations performed during the comparison of matchable entities. An exception is the MaasMatch system. It generates results that are highly similar to a subset of the alignment generated by the *StringEquiv* baseline.

Using an input alignment. This specific task was known as subtask #4 in the previous OAEI campaigns. Originally, we planned to study the impact of different input alignments of varying size. The idea is that a partial input alignment, which might have been generated by a human expert, can help the matching system to find missing correspondences. However, taking into account only those systems that could generate a meaningful alignment in time, only AgreementMaker, implemented the required interface. Thus, a comparative evaluation is not possible. We may have to put more effort in advertising this specific subtask for the next OAEI.

Alignment coherence. This year we also evaluated alignment coherence. The anatomy data set contains only a small amount of disjointness statements, the ontologies under discussion are in $\mathcal{EL}$++. Thus, even simple techniques might have an impact on the coherence of the generated alignments. For the anatomy data set the systems LogMap, CODI, and MaasMatch generate coherent alignments. The first two systems put a focus on alignment coherence and apply special methods to ensure coherence. MaasMatch has generated a small, highly precise, and coherent alignment. The alignments generated by the other systems are incoherent. A more detailed analysis related to alignment coherence is conducted for the alignments of the conference data set in Section 5.

4.3 Conclusions

Less than half of the systems generate good or at least acceptable results for the matching task of the anatomy track. With respect to those systems that failed on anatomy, we can assume that this track was not in the focus of their developers. This means at the same time that many systems are particularly designed or configured for matching tasks that we find in the benchmark and conference tracks. Only few of them are robust "all-round" matching systems that are capable of solving different tasks without changing their settings or algorithms.

The positive results of 2011 are the top results of AgreementMaker and the runtime performance of LogMap. AgreementMaker generated a very good result by increasing precision and recall compared to its last years submissions, which was the best submission in 2010 already. LogMap clearly outperforms all other systems in terms of runtimes and still generates good results. We refer the reader to the OAEI papers of these two systems for details on the algorithms.

5 Conference

The conference test case introduces matching several moderately expressive ontologies. Within this track, participant results were evaluated using diverse evaluation methods. As last year, the evaluation has been supported by the SEALS platform.

5.1 Test data

The collection consists of sixteen ontologies in the domain of organizing conferences. Ontologies have been developed within the OntoFarm project[9].

The main features of this test case are:

– *Generally understandable domain.* Most ontology engineers are familiar with organizing conferences. Therefore, they can create their own ontologies as well as evaluate the alignments among their concepts with enough erudition.
– *Independence of ontologies.* Ontologies were developed independently and based on different resources, they thus capture the issues in organizing conferences from different points of view and with different terminologies.
– *Relative richness in axioms.* Most ontologies were equipped with OWL DL axioms of various kinds; this opens a way to use semantic matchers.

Ontologies differ in numbers of classes, of properties, in expressivity, but also in underlying resources. Ten ontologies are based *on tools* supporting the task of organizing conferences, two are based on experience of people with *personal participation* in conference organization, and three are based on *web pages* of concrete conferences.

Participants were asked to provide all correct correspondences (equivalence and/or subsumption correspondences) and/or 'interesting correspondences' within the conference data set.

[9] `http://nb.vse.cz/~svatek/ontofarm.html`

5.2 Results

This year, we provided results in terms of F_2-measure and $F_{0.5}$-measure, comparison with two baseline matchers and precision/recall triangular graph.

Evaluation based on the reference alignments. We evaluated the results of participants against reference alignments. They include all pairwise combinations between 7 different ontologies, i.e. 21 alignments.

Matcher	Prec.	$F_{0.5}$Meas.	Rec.	Prec.	F_1Meas.	Rec.	Prec.	F_2Meas.	Rec.
YAM++	.8	.73	.53	.78	**.65**	.56	.78	.59	.56
CODI	.74	.7	.57	.74	.64	.57	.74	.6	.57
LogMap	.85	**.75**	.5	.84	.63	.5	.84	.54	.5
AgrMaker	.8	.69	.44	.65	.62	.59	.58	**.61**	.62
$BaseLine_2$	**.79**	**.7**	**.47**	**.79**	**.59**	**.47**	**.79**	**.51**	**.47**
MaasMtch	.83	.69	.42	.83	.56	.42	.83	.47	.42
$BaseLine_1$	**.8**	**.68**	**.43**	**.8**	**.56**	**.43**	**.8**	**.47**	**.43**
CSA	.61	.58	.47	.5	.55	.6	.5	.58	.6
CIDER	.67	.61	.44	.64	.53	.45	.38	.48	.51
MapSSS	.55	.53	.47	.55	.51	.47	.55	.48	.47
Lily	.48	.42	.27	.36	.41	.47	.37	.45	.47
AROMA	.35	.37	.46	.35	.4	.46	.35	.43	.46
Optima	.25	.28	.57	.25	.35	.57	.25	.45	.57
MapPSO	.28	.25	.17	.21	.23	.25	.12	.26	.36
LDOA	.1	.12	.56	.1	.17	.56	.1	.29	.56
MapEVO	.27	.08	.02	.15	.04	.02	.02	.02	.02

Table 7. The highest average $F_{[0.5|1|2]}$-measure and their corresponding precision and recall for some threshold for each matcher.

For better comparison, we evaluated alignments with regard to three different average[10] F-measures independently. We used $F_{0.5}$-measure (where $\beta = 0.5$) which weights precision higher than recall, F_1-measure (the usual F-measure, where $\beta = 1$), which is the harmonic mean of precision and recall, and F_2-measure (for $\beta = 2$) which weights recall higher than precision. For each of these F-measures, we selected a global confidence threshold that provides the highest average $F_{[0.5|1|2]}$-measure. Results of these three independent evaluations[11] are provided in Table 7.

Matchers are ordered according to their highest average F_1-measure. Additionally, there are two simple string matchers as baselines. $Baseline_1$ is a string matcher based on string equality applied on local names of entities which were lowercased before. $Baseline_2$ enhances $baseline_1$ with three string operations: removing of dashes, underscores and “has” words from all local names. These two baselines divide matchers into four groups. Group 1 consists of best matchers (YAM++, CODI, LogMap and AgreementMaker) having better results than $baseline_2$ in terms of F_1-measure. Matchers which perform worse than $baseline_2$ in terms of F_1-measure but still better than

[10] Computed using the absolute scores, i.e. number of true positive examples.

[11] Precision and recall can be different in all three cases.

$baseline_1$ are in Group 2 (MaasMatch). Group 3 (CSA, CIDER and MapSSS) contains matchers which are better than $baseline_1$ at least in terms of F_2-measure. Other matchers (Lily, Aroma, Optima, MapPSO, LDOA and MapEVO) perform worse than $baseline_1$ (Group 4). Optima, MapSSS and CODI did not provide graded confidence values. Performance of matchers regarding F_1-measure is visualized in Figure 3.

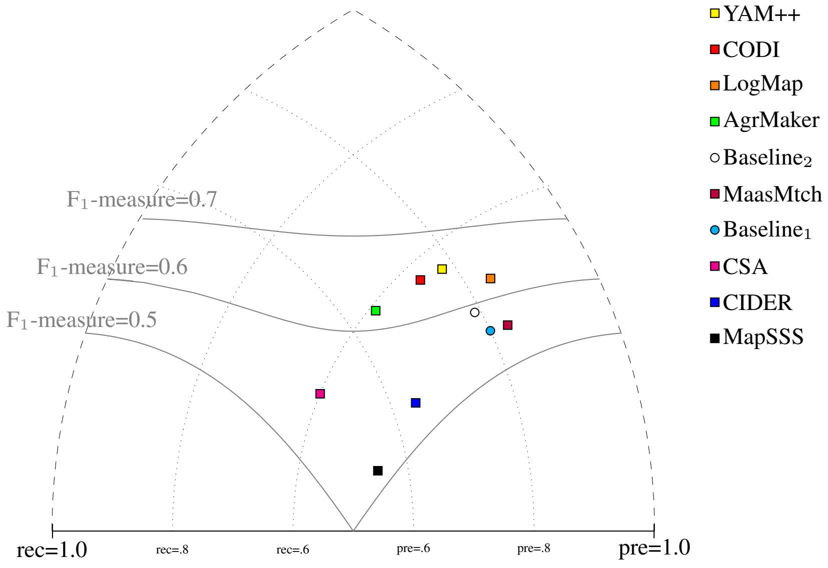


Fig. 3. Precision/recall triangular graph for conference. Matchers of participants from the first three groups are represented as squares. Baselines are represented as circles. Dotted lines depict level of precision/recall while values of F_1-measure are depicted by areas bordered by corresponding lines F_1-measure=0.[5|6|7].

In conclusion, all best matchers (group one) are very close to each other. However, the matcher with the highest average F_1-measure (.65) is YAM++, the highest average F_2-measure (.61) is AgreementMaker and the highest average $F_{0.5}$-measure (.75) is LogMap. In any case, we should take into account that this evaluation has been made over a subset of all possible alignments (one fifth).

Comparison with previous years. Three matchers also participated in the previous year. AgreementMaker improved its average F_1-measure from .58 to .62 by higher precision (from .53 to .65) and lower recall (from .62 to .59), CODI increased its average F_1-measure from .62 to .64 by higher recall (from .48 to .57) and lower precision (from .86 to .74). AROMA (with its AROMA- variant) slightly decreased its average F_1-measure from .42 to .40 by lower precision (from .36 to .35) and recall (from .49 to .46).

Evaluation based on alignment coherence. As in the previous years, we apply the Maximum Cardinality measure proposed in [13] to measure the degree of alignment incoherence. Details on the algorithms can be found in [12]. The reasoner underlying our implementation is Pellet [16].

The results of our experiments are depicted in Table 8. It shows the average for all test cases of the conference track, which covers more than the ontologies that are connected via reference alignments. We had to omit the test cases in which the ontologies `Confious` and `Linklings` are involved as source or target ontologies. These ontologies resulted in many cases in reasoning problems. Thus, we had 91 test cases for each matching system. However, we faced reasoning problems for some combinations of test cases and alignments. In this case we computed the average score by ignoring these test cases. These problems occurred mainly for highly incoherent alignments. The last row in Table 8 informs about the number of test cases that were excluded. Note that we did not analyze the results of those systems that generated alignments with precision less than .25.

Matcher	AgrMaker	AROMA	CIDER	CODI	CSA	Lily	LogMap	MaasMtch	MapSSS	Optima	YAM
Size	13.9	14.1	17.9	9.5	50.8	17	8	7.5	10	31.3	10.1
Inc. Alignments	49/90	58/88	69/88	0/91	69/69	70/90	8/91	21/91	51/90	73/84	41/91
Degree of Inc.	12%	16%	13%	0%	>29%	14%	2%	4%	9%	>31%	7%
Reasoning problems	1	3	3	0	22	1	0	0	1	7	0

Table 8. Average size of alignments, number of incoherent alignments, and average degree of incoherence. The prefix $>$ is added if the search algorithm stopped in one of the testcases due to a timeout of 10min prior to computing the degree of incoherence.

CODI is the only system that guarantees the coherence of the generated alignments. While last year some of the alignments were incoherent, all of the alignments generated in 2011 are coherent. LogMap, a system with special focus on alignment coherence and efficiency [11], generates in most cases coherent alignments. A closer look at the outliers reveals that all incoherent alignments occured for ontology pairs where the ontology `Cocus` was involved. This ontology suffers from a very specific modeling error based on the inappropriate use of universal quantification. At the third position we find MaasMatch. MaasMatch generates less incoherent alignments than the remaining systems. This might be related to the high precision of the system. Contrary to LogMap, incoherent alignments are generated for different combinations of ontologies and there is no specific pattern emerging.

It is not easy to interpret the results of the remaining matching systems due to the different sizes of the alignments that they have generated. The more correspondences are contained in an alignment, the higher is the probability that this results in a concepts unsatisfiability. It is not always clear whether a relatively low/high degree of incoherence is mainly caused by the small/large size of the alignments, or related to the use of a specific technique. Overall, we conclude that alignment coherence is not taken into account by these systems. However, in 2011 we have at least some systems that apply specific methods to ensure coherence for all or at least for a large subset of generated alignments. Compared to the previous years, this is a positive result of our analysis.

6 Instance matching

The goal of the instance matching track is to evaluate the performance of different matching tools on the task of matching RDF individuals which originate from different sources but describe the same real-world entity. Data interlinking is known under many names according to various research communities: equivalence mining, record linkage, object consolidation and coreference resolution to mention the most used ones. In each case, these terms are used for the task of finding equivalent entities in or across data sets. As the quantity of data sets published on the Web of data dramatically increases, the need for tools helping to interlink resources becomes more critical. It is particularly important to maximize the automation of the interlinking process in order to be able to follow this expansion.

Unlike the other tracks, the instance matching tests specifically focus on an ontology ABox. However, the problems which have to be resolved in order to correctly match instances can originate at the schema level (use of different properties and classification schemas) as well as at the data level, e.g., different formats of values. This year, the track included two tasks. The first task, data interlinking (DI), aims at testing the performance of tools on large-scale real-world data sets published according to the linked data principles. The second one (IIMB) uses a set of artificially generated and real test cases respectively. These are designed to illustrate all common cases of discrepancies between individual descriptions (different value formats, modified properties, different classification schemas). The list of participants to the instance matching track is shown in Table 9.

Dataset	**AgrMaker**	**SERIMI**	**Zhishi**	**CODI**
DI-nyt-dbpedia-locations	√	√	√	
DI-nyt-dbpedia-organizations	√	√	√	
DI-nyt-dbpedia-people	√	√	√	
DI-nyt-freebase-locations	√	√	√	
DI-nyt-freebase-organizations	√	√	√	
DI-nyt-freebase-people	√	√	√	
DI-nyt-geonames	√	√	√	
IIMB				√

Table 9. Participants in the instance matching track.

6.1 Data interlinking task (DI) – New York Times

This year the data interlinking task consists of matching the New York Times subject headings to DBpedia, Freebase and Geonames. The New York Times has developed over the past 150 years an authoritative vocabulary for annotating news items. The vocabulary contains about 30,000 subject headings, or tags. They are progressively published as linked open data and, by July 2010, over 10,000 of these subject headings, in the categories People, Organizations, Locations and Descriptors, have been published[12].

[12] http://data.nytimes.com/

The New York Times data set was used in OAEI 2010 track on very large crosslingual resources.

The reference alignments are extracted from the links provided and curated by The New-York Times. However, the set of reference links has been updated to reflect the changes made to the external data sets during the year. In particular, several missing links were added, links pointing to non-existing DBPedia instances were removed, and links to instances redirecting to others were updated. Moreover, the Descriptors facet has been removed from the evaluation, since there was not a clear identity criterion for its instances.

Facet	**# Concepts**	**Links to Freebase**	**Links to DBPedia**	**Links to Geonames**
People	4,979	4,979	4,977	0
Organizations	3,044	3,044	1,965	0
Locations	1,920	1,920	1,920	1,920

Table 10. Number of links between the New-York Times corpus and other data sources.

Subject heading facets are represented in SKOS. Each subject heading facet contains the label of the skos:Concept (skos:label), the facet it belongs to (skos:inScheme), and some specific properties: nyt:associated_article_count for the number of NYT articles the concept is associated with and nyt:topicPage pointing to the topic page (in HTML) gathering different information published on the subject. The Location facet also contains geo-coordinates. The concepts have links to DBpedia, Freebase and/or GeoNames.

	AgreementMaker			SERIMI			Zhishi.links		
Dataset	Prec.	FMeas.	Rec.	Prec.	FMeas.	Rec.	Prec.	FMeas.	Rec.
DI-nyt-dbpedia-loc.	.79	.69	.61	.69	.68	.67	.92	.92	.91
DI-nyt-dbpedia-org.	.84	.74	.67	.89	.88	.87	.90	.91	.93
DI-nyt-dbpedia-peo.	.98	.88	.80	.94	.94	.94	.97	.97	.97
DI-nyt-freebase-loc.	.88	.85	.81	.92	.91	.90	.90	.88	.86
DI-nyt-freebase-org.	.87	.80	.74	.92	.91	.89	.89	.87	.85
DI-nyt-freebase-peo.	.97	.96	.95	.93	.92	.91	.93	.93	.92
DI-nyt-geonames.	.90	.85	.80	.79	.80	.81	.94	.91	.88
H-mean.	.92	.85	.80	.89	.89	.88	.93	.92	.92

Table 11. Results of the DI subtrack.

DI results. An overview of the precision, recall and F_1-measure results per data set of the DI subtrack is shown in Table 11. A precision-recall graph visualization is shown in Figure 4. The results show a variation in both systems and data sets. Zhishi.links produces consistently high quality matches over all data sets, and obtains the highest overall scores. Matches to DBpedia locations (DI-nyt-dbpedia-loc.) appear to be difficult as AgreementMaker and SERIMI perform poorly on both precision and recall. This is not the case for Freebase locations (DI-nyt-freebase-loc.) and to a much lesser extent for Geonames (DI-nyt-geonames). We hypthesize that this is due to many locations not being present in DBPedia. Agreementmaker's scores considerably higher on People

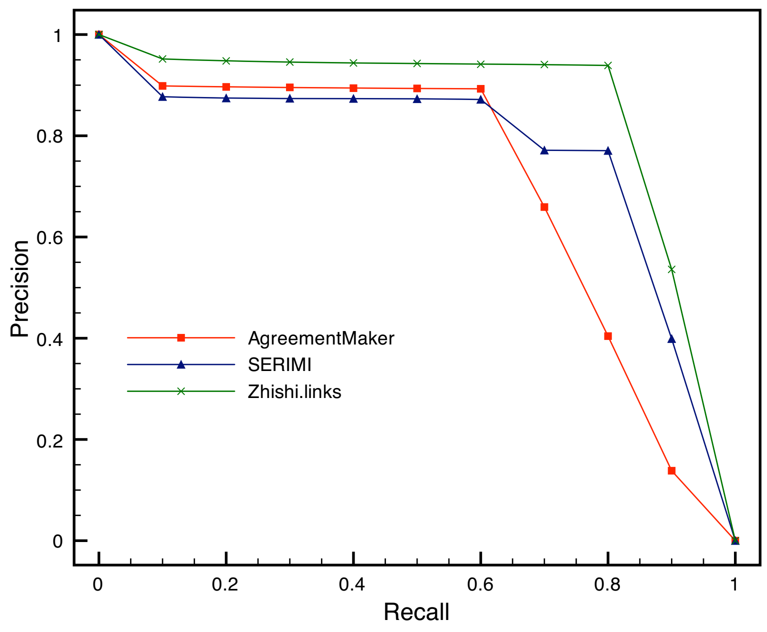


Fig. 4. Precision/recall of tools participating in the DI subtrack.

than on Locations and Organizations, which can be observed in both the DBpedia and the Freebase data set.

6.2 OWL data task (IIMB)

The OWL data task is focused on two main goals:

1. to provide an evaluation data set for various kinds of data transformations, including value transformations, structural transformations and logical transformations;
2. to cover a wide spectrum of possible techniques and tools.

To this end, we provided the ISLab Instance Matching Benchmark (IIMB). Participants were requested to find the correct correspondences among individuals of the first knowledge base and individuals of the other one. An important task here is that some of the transformations require automatic reasoning for finding the expected alignments.

IIMB is composed of a set of test cases, each one represented by a set of instances, i.e., an OWL ABox, built from an initial data set of real linked data extracted from the web. Then, the ABox is automatically 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 referring to the same objects are provided in different data sources; on the other side, we generate different data sets with a variable level of data quality and complexity. IIMB provides transformation techniques supporting modifications of data property values, modifications of number and type of properties used for the individual description, and modifications of the individuals classification. The first kind of transformations is called *data value transformation* and it aims at simulating the fact that data expressing the same real object in different data sources may be different because of data errors or because of the usage of different conventional patterns for data representation. The second kind of transformations is called *data structure transformation* and it aims at simulating the fact that the same real object may be described using different properties/attributes in different data sources. Finally, the third kind of transformations, called *data semantic transformation*, simulates the fact that the same real object may be classified in different ways in different data sources.

The 2011 edition of IIMB is created by extracting data from Freebase, 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[13]. IIMB2011 is a collection of OWL ontologies consisting of 29 concepts, 20 object properties, 12 data properties and more than 4000 individuals divided into 80 test cases.

Test cases from 0 to 20 contain changes in data format (misspelling, errors in text, etcetera); test cases 21 to 40 contain changes in structure (properties missing, RDF triples changed); 41 to 60 contain logical changes (class membership changed, logical errors); finally, test cases 61 to 80 contain a mix of the previous. One system, CODI, participated in this task. Its results (Table 5) show how precision drops moderately and recall drops dramatically as more errors are introduced.

IIMB results An overview of the precision, recall and F_1-measure results per set of tests of the IIMB subtrack is shown in Table 5. A precision-recall graph visualization is shown in Figure 6.

	codi		
test	Prec.	FMeas.	Rec.
001–010	.94	.84	.76
011–020	.94	.87	.81
021–030	.89	.79	.70
031–040	.83	.66	.55
041–050	.86	.72	.62
051–060	.83	.72	.64
061–070	.89	.59	.44
071–080	.73	.33	.21

Fig. 5. Results of the IIMB subtrack.

[13] `http://code.google.com/p/freebase-java/`

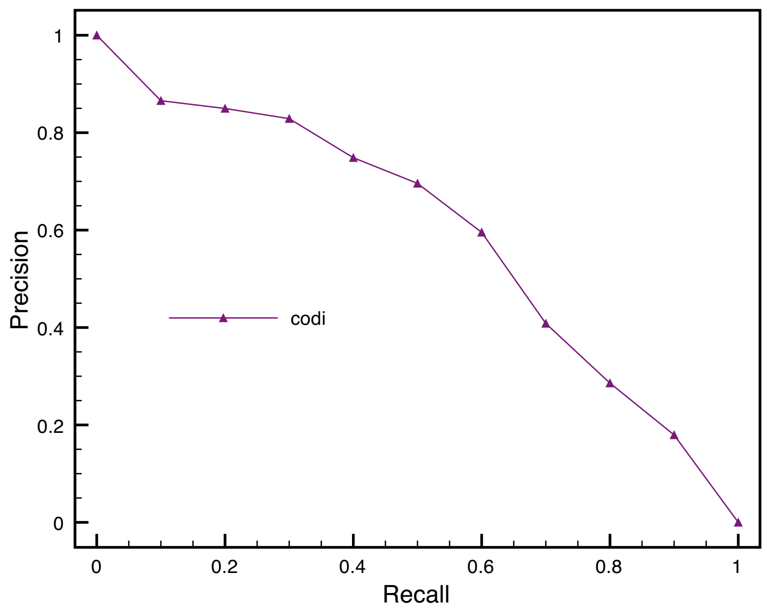


Fig. 6. Precision/recall of the CODI tool participating in the IIMB subtrack.

7 Lesson learned and suggestions

This year we implemented most of our 2010 future plans by providing a common platform on which evaluation could be performed. There still remains one lesson not really taken into account that we identify with an asterisk (*) and that we will tackle in the coming months. The main lessons from this year are:

A) This year again indicated that requiring participants to implement a minimal interface was not a strong obstacle to participation. The interface allows for comparing matchers on the same or similar hardware. It also allows for running more tests or reproducing results without running a new campaign.

B) By using the SEALS platform, we have eliminated the network issue that we had last year with web services and we can better testify the portability of tools.

C) The client available for testing and evaluating wrapped tools was intensively used by participants to test and improve their systems. So, interoperability and the ability to get immediate feedback was appreciated by the tool developers. Moreover, participants could use the client to generate preliminary results to be included in their papers.

D) There is a high variance in runtimes and there seems to be no correlation between runtime and quality of the generated results.

*E) The low number of systems that could generate results for the Anatomy track is an uncommon result. It seems that not many matching systems of this year are capable of matching large ontologies ($>$1000 entities). Even if we had introduced new benchmark generation facilities, we have not used it towards scalability benchmarks. We plan to address this in the next few months.

F) Last years we reported that there are not many new systems entering the competition. This year we had many new participants. Only a minority of systems participated in one of the previous years.

G) Two systems have not fully respected the OAEI rules. YAM++ used a setting learned from the reference alignments of the 2009 benchmark data set. Due to the fact that we run benchmarks also with newly generated tests, we decided to keep the YAM++ results with this warning. AgreementMaker used a specific setting to distinguish between Benchmarks, Anatomy and Conference. As AgreementMaker has improved its results over the last year, we decide to report on them as well. For the next campaigns we plan to be more attentive on these aspects.

H) In spite of claims that such evaluations were needed, we had to declare the model matching and oriented alignments tracks unfruitful. It is a pity. This confirms that setting up a data set is not sufficient for attracting participants.

I) More surprising, there are only a few matchers participating in the instance matching track. This is especially surprising given the high number of papers submitted and published on this topic nowadays. It seems that people involved in instance matching should cooperate to propose standard formats and evaluation modalities that everyone would use.

8 Future plans

In 2012, for logistic reasons, we plan to have an intermediate evaluation before OAEI-2012. This evaluation will concentrate on exploiting fully the SEALS platform and, in particular on:

- performing benchmark scalability tests by reducing randomly a large seed ontology;
- generating discriminating benchmarks by suppressing easy tests;
- adding new tasks, such as multilingual resources, on the SEALS platform.

We plan to run these tests within the next six months with the already registered tools that would like to be evaluated as well as with new tools willing to enter. These partial results will be integrated within the results of OAEI-2012.

9 Conclusions

The trend of the previous years, the number of systems and tracks they participate in, seem to stabilize. The average number of tracks entered by participants in 2011 (2.9) is above that of 2010 (2.6). This number is dominated by the use of the SEALS platform: each tool entering there can be evaluated on three tasks. It does not invalidate last year's remark that tools may be more specialized.

This year, systems did not deliver huge improvements in performance with respect to last year's performers which did not participate. However, AgreementMaker improved its results of last year to become one of the top performer. In addition, we have been able to test runtime and consistency of the tested matchers and noticed ample differences between systems. This may become a differentiating feature among matchers.

All participants have provided a description of their systems and their experience in the evaluation. These OAEI papers, like the present one, have not been peer reviewed. However, they are full contributions to this evaluation exercise and reflect the hard work and clever insight people put in the development of participating systems. Reading the papers of the participants should help people involved in ontology matching to find what makes these algorithms work and what could be improved. Sometimes participants offer alternate evaluation results.

The Ontology Alignment Evaluation Initiative will continue these tests by improving both test cases and testing methodology for being more accurate. Further information can be found at:

`http://oaei.ontologymatching.org`.

Acknowledgments

We warmly thank the participants of this campaign. We know that they have worked hard for having their matching tools executable in time and they provided insightful papers presenting their experience. The best way to learn about the results remains to read the following papers.

We are grateful to Dominique Ritze for participating in the extension of the reference alignments for the conference track.

We thank Jan Noessner for providing data in the process of constructing the IIMB data set. We are grateful to Martin Ringwald and Terry Hayamizu for providing the reference alignment for the anatomy ontologies and thank Elena Beisswanger for her thorough support on improving the quality of the data set.

We also thank the other members of the Ontology Alignment Evaluation Initiative steering committee: Yannis Kalfoglou (Ricoh laboratories, UK), Miklos Nagy (The Open University (UK), Natasha Noy (Stanford University, USA), Yuzhong Qu (Southeast University, CN), York Sure (Leibniz Gemeinschaft, DE), Jie Tang (Tsinghua University, CN), George Vouros (University of the Aegean, GR).

Jérôme Euzenat, Christian Meilicke, Heiner Stuckenschmidt and Cássia Trojahn dos Santos have been partially supported by the SEALS (IST-2009-238975) European project.

Ondřej Šváb-Zamazal has been supported by the CSF grant P202/10/0761.

References

1. Benhamin Ashpole, Marc Ehrig, Jérôme Euzenat, and Heiner Stuckenschmidt, editors. *Proc. of the K-Cap Workshop on Integrating Ontologies*, Banff (Canada), 2005.
2. Caterina Caracciolo, Jérôme Euzenat, Laura Hollink, Ryutaro Ichise, Antoine Isaac, Véronique Malaisé, Christian Meilicke, Juan Pane, Pavel Shvaiko, Heiner Stuckenschmidt, Ondrej Sváb-Zamazal, and Vojtech Svátek. Results of the ontology alignment evaluation initiative 2008. In *Proc. 3rd International Workshop on Ontology Matching (OM) collocated with ISWC*, pages 73–120, Karlsruhe (Germany), 2008.
3. Jérôme David, Jérôme Euzenat, François Scharffe, and Cássia Trojahn dos Santos. The alignment api 4.0. *Semantic web journal*, 2(1):3–10, 2011.
4. Marc Ehrig and Jérôme Euzenat. Relaxed precision and recall for ontology matching. In *Proc. of the K-Cap Workshop on Integrating Ontologies*, pages 25–32, Banff (Canada), 2005.

5. Jérôme Euzenat, Alfio Ferrara, Laura Hollink, Antoine Isaac, Cliff Joslyn, Véronique Malaisé, Christian Meilicke, Andriy Nikolov, Juan Pane, Marta Sabou, François Scharffe, Pavel Shvaiko, Vassilis Spiliopoulos, Heiner Stuckenschmidt, Ondrej Sváb-Zamazal, Vojtech Svátek, Cássia Trojahn dos Santos, George Vouros, and Shenghui Wang. Results of the ontology alignment evaluation initiative 2009. In *Proc. 4th Workshop on Ontology Matching (OM) collocated with ISWC*, pages 73–126, Chantilly (USA), 2009.
6. Jérôme Euzenat, Alfio Ferrara, Christian Meilicke, Andriy Nikolov, Juan Pane, François Scharffe, Pavel Shvaiko, Heiner Stuckenschmidt, Ondrej Sváb-Zamazal, Vojtech Svátek, and Cássia Trojahn dos Santos. Results of the ontology alignment evaluation initiative 2010. In Pavel Shvaiko, Jérôme Euzenat, Fausto Giunchiglia, Heiner Stuckenschmidt, Ming Mao, and Isabel Cruz, editors, *Proc. 5th ISWC workshop on ontology matching (OM) collocated with ISWC, Shanghai (China)*, pages 85–117, 2010.
7. Jérôme Euzenat, Antoine Isaac, Christian Meilicke, Pavel Shvaiko, Heiner Stuckenschmidt, Ondrej Svab, Vojtech Svatek, Willem Robert van Hage, and Mikalai Yatskevich. Results of the ontology alignment evaluation initiative 2007. In *Proc. 2nd International Workshop on Ontology Matching (OM) collocated with ISWC*, pages 96–132, Busan (Korea), 2007.
8. Jérôme Euzenat, Christian Meilicke, Pavel Shvaiko, Heiner Stuckenschmidt, and Cássia Trojahn dos Santos. Ontology alignment evaluation initiative: six years of experience. *Journal on Data Semantics*, XV:158–192, 2011.
9. Jérôme Euzenat, Malgorzata Mochol, Pavel Shvaiko, Heiner Stuckenschmidt, Ondrej Svab, Vojtech Svatek, Willem Robert van Hage, and Mikalai Yatskevich. Results of the ontology alignment evaluation initiative 2006. In *Proc. 1st International Workshop on Ontology Matching (OM) collocated with ISWC*, pages 73–95, Athens, Georgia (USA), 2006.
10. Jérôme Euzenat and Pavel Shvaiko. *Ontology Matching*. Springer, Heidelberg (DE), 2007.
11. Ernesto Jiménez-Ruiz and Bernardo Cuenca Grau. Logmap: Logic-based and scalable ontology matching. In *Proc. 10th International Semantic Web Conference (ISWC)*, pages 273–288, 2011.
12. Christian Meilicke. *Alignment Incoherence in Ontology Matching*. PhD thesis, University Mannheim, 2011.
13. Christian Meilicke and Heiner Stuckenschmidt. Incoherence as a basis for measuring the quality of ontology mappings. In *Proc. 3rd International Workshop on Ontology Matching (OM) collocated with ISWC*, pages 1–12, Karlsruhe (Germany), 2008.
14. Maria Roşoiu, Cássia Trojahn dos Santos, and Jérôme Euzenat. Ontology matching benchmarks: generation and evaluation. In Pavel Shvaiko, Isabel Cruz, Jérôme Euzenat, Tom Heath, Ming Mao, and Christoph Quix, editors, *Proc. 6th International Workshop on Ontology Matching (OM) collocated with ISWC, Bonn (Germany)*, 2011.
15. Pavel Shvaiko and Jérôme Euzenat. Ontology matching: state of the art and future challenges. *IEEE Transactions on Knowledge and Data Engineering*, 2012, to appear.
16. Evren Sirin, Bijan Parsia, Bernardo Cuenca Grau, Aditya Kalyanpur, and Yarden Katz. Pellet: a practical OWL-DL reasoner. *Journal of Web Semantics: Science, Services and Agents on the World Wide Web*, 5(2):51–53, 2007.
17. York Sure, Oscar Corcho, Jérôme Euzenat, and Todd Hughes, editors. *Proc. of the Workshop on Evaluation of Ontology-based Tools (EON) collocated with ISWC*, Hiroshima (Japan), 2004.
18. Cássia Trojahn dos Santos, Christian Meilicke, Jérôme Euzenat, and Heiner Stuckenschmidt. Automating OAEI campaigns (first report). In *Proc. International Workshop on Evaluation of Semantic Technologies (iWEST) collocated with ISWC*, Shanghai (China), 2010.

Grenoble, Milano, Amsterdam, Delft, Mannheim, Milton-Keynes, Montpellier, Trento, Prague, November 2011

Using AgreementMaker to Align Ontologies for OAEI 2011*

Isabel F. Cruz, Cosmin Stroe, Federico Caimi, Alessio Fabiani, Catia Pesquita, Francisco M. Couto, Matteo Palmonari

ADVIS Lab, Department of Computer Science, University of Illinois at Chicago
{ifc|cstroe1|fcaimi|fabiani|pmatteo}@cs.uic.edu

Facultade de Ciencias da Universitade de Lisboa, Portugal
cpesquita@xldb.di.fc.ul.pt, fcouto@di.fc.ul.pt

University of Milan-Bicocca, Italy
matteo.palmonari@disco.unimib.it

Abstract. The AgreementMaker system is unique in that it features a powerful user interface, a flexible and extensible architecture, an integrated evaluation engine that relies on inherent quality measures, and semi-automatic and automatic methods. This paper describes the participation of AgreementMaker in the 2011 OAEI competition in four tracks: benchmarks, anatomy, conference, and instance matching. After its successful participation in 2009 and 2010, the goal in this year's participation is to explore previously unused features of the ontologies in order to improve the matching results. Furthermore, the system should be able to automatically adapt to the matching task, choosing the best configuration for the given pair of ontologies. We believe that this year we have made considerable progress in both of these areas.

1 Presentation of the system

We have been developing the AgreementMaker system since 2001, with a focus on real-world applications [5,9] and in particular on geospatial applications [4,6,8,10,11,12,13,15]. However, the current version of AgreementMaker, whose development started in 2008, represents a whole new effort. The code base has more than doubled since then, with the AgreementMaker framework being expanded to accomodate many types of ontology matching techniques.

* Partially supported by NSF Awards IIS–0513553 and IIS-0812258 and by the Intelligence Advanced Research Projects Activity (IARPA) via Air Force Research Laboratory (AFRL) contract number FA8650-10-C-7061. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of IARPA, AFRL, or the U.S. Government.

1.1 Purpose and state of the system

The AgreementMaker system [1,2,3,7] is an extensible ontology matching framework that has been expanded to include many types of matching algorithms in order to handle many different matching scenarios. At the heart of the system is its ability to efficiently combine the results from several matching algorithms into one single and better result [2]. This capability allows us to focus on developing new matching algorithms and later combine them with our existing approach in order to improve our results.

2 Schema Matching Techniques Introduced in OAEI 2011

As compared to previous years, we have introduced several matching techniques in order to improve our matching algorithms.

2.1 Automatic Configuration Selection via Ontology Profiling Metrics

The AgreementMaker system can be run with different configurations that optimize the system accuracy and coverage depending on the specific ontologies to be aligned. Changing the composition of the matcher stack (e.g. an instance matcher is used only when instances are available) has a high impact on the system performance. We developed an approach to adaptively optimize the configuration of AgreementMaker depending on the ontologies to be aligned.

The approach we adopted can be sketched as follows: the ontologies to be aligned are profiled using several metrics proposed in the literature (e.g. relationship richness, inheritance richness, WorldNet coverage and so on [16]). The metric-based profiles are used to automatically classify the matching task into a configuration class with specific settings. The classification is based on a supervised machine learning framework trained with a subset of the OEAI dataset for which a reference alignment is available.

Our learning framework is very flexible: we can use many combinations of matchers and parameters, various types of classifiers (KStar, Naive Bayes, Multilayer Perceptron etc.) and new metrics. The experimental results show that the use of the automatic configuration methods improved the overall performance of AgreementMaker in the competition. In particular, in this paper we show the importance of this method for the significant improvements we achieved in the Benchmark and Conference tracks. The new AgreementMaker 's matching process follows the steps represented in Figure 1: the ontology pair to be matched is classified by the ontology-profiling algorithm; based on the classification, a run configuration is created, and an ontology matching algorithm is instantiated to create an alignment.

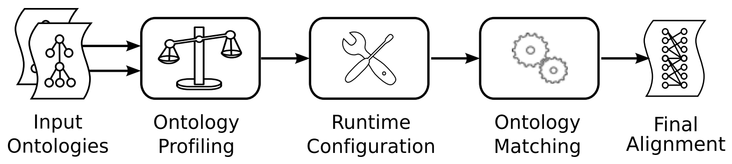


Fig. 1. AgreementMaker OAEI2011 Automatic configuration selection.

2.2 Lexicon Expansion via a Mediating Ontology

One approach to matching two domain specific ontologies is to use a third ontology from the same domain as a mediating ontology, with the mediating ontology to provide missing information relevant to the matching task. Shown in Figure 2, the source and target ontologies, O_S and O_T respectively, are first matched with the mediating ontology O_M. Mappings between the source and target ontologies are then created based on the distance between the concepts in the mediating ontology to which they have been mapped previously.

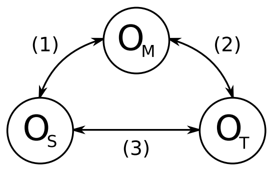


Fig. 2. Using a mediating ontology.

For the specific problem of matching the Mouse Anatomy ontology to the Human Anatomy ontology a successful approach has been to use the UBERON cross species anatomy ontology as a mediating ontology [14]. We have adapted this approach to our lexicon framework, using the BSM_{lex} to match the MA and HA ontologies with UBERON thereby making use of the extra synonyms defined in UBERON.

2.3 Extension of Synonyms

This strategy relies on synonyms defined in the OWL ontology itself, currently via the `hasRelatedSynonym` property, to generate a lexicon of *synonym terms* (single or multi-word terms). This is done by finding common terms between ontology synonyms to infer synonyms terms.

For example, in the Human Anatomy ontology, the concept NCI_C12275 ("Maxillary_Sinus") has the synonyms "Antrum, Maxillary" and "Sinus, Maxillary". Our algorithm infers that "sinus" and "antrum" are synonyms as well without any external reference. These synonym terms are then used to create novel synonyms, by interchanging terms in existing synonyms and labels with their synonymous term.

2.4 Alternate Hierarchy Support

In addition to the subclass hierarchy defined as part of the OWL ontologies of the Anatomy track, there is also a "part of" hierarchy defined using the `UNDEFINED_part_of` property. Taking into account this hierarchy in the VMM_{lex} increases the percision and recall of the matching algorithm.

3 Results

In this section, we present the results obtained by AgreementMaker in the OAEI 2011 competition. It participated in four tracks: benchmarks, anatomy, conference, and instance matching. Tests were carried out on a PC running Ubuntu Linux 10.04 with AMD Athlon^{TM} II X4 635 processor running at 2.9 Ghz and 8 GB RAM.

3.1 Link to the system and parameters file

The AgreementMaker system is available at `http://agreementmaker.org/`. The matching algorithm we used is implemented in the "OAEI 2011 Matcher" algorithm, in the "Hybrid" category. The alignment results obtained by AgreementMaker in the OAEI 2010 are available at `http://agreementmaker.org/oaei`.

3.2 Benchmarks Track

Benchmarks Track Results In this track, a source ontology is compared to 111 ontologies that describe the same domain which can be divided into 3 categories: concept tests cases (1xx cases), systematic tests cases (2xx cases), and real ontology test cases (3xx cases). The 2xx benchmarks test cases are subdivided into 3 groups: 1) 201 to 210, 2) 221 to 247 and 3) 248 to 266. As shown in the Table 1, our results are very good in all the tracks, due to the use of a combination of properties (lexical, structural, syntactic, etc.) to match the ontologies. We are able to perform well even if only one of these properties is available for comparison.

	101-104	201-210	221-247	248-266	301-304	H-Mean
Precision	1.00	1.00	0.98	0.96	0.90	0.97
Recall	1.00	0.95	0.99	0.66	0.85	0.87
F-Measure	1.00	0.97	0.99	0.76	0.87	0.91

Table 1. Results of AgreementMaker in the Benchmarks track of the OAEI 2011 competition.

	101-104	201-210	221-247	248-266	301-304	H-Mean
Precision 2010	0.98	0.97	0.95	0.96	0.88	0.95
Precision 2011	1.00	1.00	0.98	0.96	0.90	0.97
Recall 2010	1.00	0.90	0.99	0.74	0.53	0.79
Recall 2011	1.00	0.95	0.99	0.66	0.85	0.87
F-Measure 2010	0.99	0.94	0.97	0.82	0.61	0.84
F-Measure 2011	1.00	0.97	0.99	0.76	0.87	0.91

Table 2. Comparison of the results in the 2010 and 2011 OAEI Benchmarks track.

Benchmarks Track Comments Although we improved our results with respect last year in almost all the tasks, we are particularly satisfied of our improvements on the third group (301-304). This is because the goal of our system and of the matching methods we are using is to improve the performance on real ontology test cases. A detailed comparison between the results achieved in the 2010 and 2011 competitions is shown in Table 2. One of the main reasons for our improvements is the new automatic configuration method we introduced. In fact, the matching tasks of the Benchmark track are very diverse in order to test several aspects of automatic matching methods; therefore, when the user has to manually select only one configuration, she selects the configuration that obtains the best average results on the whole set of matching tasks, but such a configuration cannot be assumed to be the best one for each individual task. Instead, thanks to the new automatic configuration method, the system automatically select the best configuration for each individual matching task.

3.3 Anatomy Track

Anatomy Track Results This track consists of two real world ontologies to be matched, the source ontology describing the Adult Mouse Anatomy (with 2744 classes) and the target ontology is the NCI Thesaurus describing the Human Anatomy (with 3304 classes). This year, the reference alignment is available for this track, which allowed us to have instant evaluation of our improvements, greatly reducing our development time. As shown in Table 3, we have been able to consistently make improvements to our matching algorithms. A large part of these improvements has been increasing the recall by leveraging external sources, including WordNet and other anatomy ontologies. We have also been able to improve precision by managing a finer grained control of our combination algorithms.

Anatomy Track	Runtime	Precision	Recall	F-Measure
2009	≈23 min	86.5%	79.8%	83.1%
2010	≈5 min	90.3%	85.3%	87.7%
2011[1]	≈7 min	95.4%	88.4%	91.8%

Table 3. Comparison of previous results with this year's results.

Anatomy Track Comments This year we have been able to further increase precision and recall by using the UBERON multi-species anatomy ontology as a mediating ontology, an approach demonstrated by others at the International Conference for Biomedical Ontology [14], by extending our lexicon synonyms using *synonym terms*, and by using the part-of hierarchy in our matching algorithms . Improvment of our algorithms capability to correctly discern relevant concept information allowed us to increase precision by over 5% and was achieved by combining more similar matching algorithms first and using those combined results for the final combination.

3.4 Conference Track

	Manual Config. 2010	Manual Config. 2011	Automatic Config. 2011
Precision	0.53	0.83	0.71
Recall	0.62	0.45	0.62
F-Measure	0.58	0.56	0.64

Table 4. Results achieved by AgreementMaker in the 2011 OAEI conference track.

Conference Results The conference track consists of 15 ontologies from the conference organization domain and each ontology must be matched against every other ontology. We ran our algorithms on all the matching tasks and evaluated precision, recall and F-measure using the 21 provided reference alignments. We then computed an average of these measures, summarized in Table 4. Precision, recall and F-measure obtained using the automatic configuration method introduced in section 2.1 are compared with the results achieved with the manual configuration of the system used in OAEI 2010 and in OAEI 2011. AgreementMaker significantly improved on F-measure by optimizing the trade-off between precision and recall.

Conference Track Comments Some of the matching algorithms that we used in the OAEI 2010 competition underwent minor changes and improvements this year. As a consequence, when we manually defined a configuration of the system, we were able to achieve a significant gain in precision in this particular track but at the cost of a lower recall (which explains why F-measure decreases in the Conference 2011 track when a specific configuration is manually selected). However, the capability to automatically configure the system depending on the input ontologies, allows to achieve a very good trade-off between precision and recall, with a sizable gain of 6% on the average F-measure with respect to the best results we obtained last year.

4 Instance Matching

Differently from our 2009 and 2010 participations, we also entered the Instance Matching track at OAEI 2011. While our matchers were previously developed specifically for working at the schema level, we adapted our system to deal with instances. We decided to focus on the Data Interlinking sub-track, which consists of recreating the links from New York Times data to Freebase, DBPedia, and GeoNames.

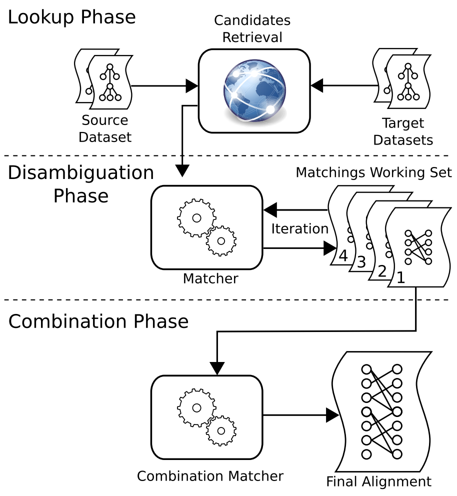


Fig. 3. Instance matching configuration.

We found this track particularly interesting and challenging, since it entails the following new problems: *a*) datasets are very large and not easy to wholly retrieve and work with, *b*) endpoints and APIs have to be queried online in order to get up-to-date information, *c*) these services provide data in different formats, and *d*) the source datasets (New York Times) do not have a schema associated with them so we cannot rely on traditional ontology matching to create schema level mappings.

All of the tasks of this track of the competition are characterized by the presence of a large amount of data. Therefore every instance in the source cannot be compared with every other in the target. We faced the problem of deciding how to reduce the number of comparisons, trying to minimize the loss in recall. Our solution consists of doing a lookup using the label of the instance, the type (when provided), and querying against an index which returns a reasonable number of candidate target instances.

We think this choice is very appropriate for several reasons: *a*) many SPARQL endpoints and APIs implement indexing which permits to get very fast answers to keyword lookups, *b*) the online version of these Knowledge Bases is always richer and more up to date with respect to downloadable versions, and *c*) we can query multiple Knowledge Bases at the same time in a parallel fashion.

Once we query the online service and obtain the results, we compute a similarity between the source instance and the candidate instances. These values are then used to rank the candidates and eventually select the best one. Reusing some of the techniques we have implemented for the other ontology matching tracks, we use different matchers

Source	Target	Types	Precision	Recall	F-Measure
NYT	Freebase	People	0.966	0.950	0.958
NYT	Freebase	Locations	0.884	0.811	0.846
NYT	Freebase	Organizations	0.873	0.735	0.798
NYT	GeoNames	Locations	0.902	0.797	0.846
NYT	DBPedia	People	0.977	0.801	0.881
NYT	DBPedia	Locations	0.790	0.612	0.690
NYT	DBPedia	Organizations	0.840	0.667	0.744
Average			0.890	0.768	0.823
Harmonic Mean			0.886	0.754	0.815

Table 5. Results of AgreementMaker in the Instance Matching track.

that compare several features about the instances, and then combine their outputs in order to give a final alignment.

The main features we use for comparison are: *a*) labels using a substring similarity, *b*) comments and other literals using a Vector Space Model approach, *c*) RDF Statements considering property-value pairs, and *d*) the score values returned by the lookup services (e.g. Freebase API, Apache Lucene score).

4.1 Instance Matching Results

The Data Interlinking sub-track of the Instance Matching at OAEI 2011 competition is composed of seven tasks. The source dataset is always the New York Times, while there are 3 different targets: Freebase, GeoNames, and DBPedia. The results obtained by AgreementMaker in the Instance Matching track are summarized in Table 5, showing precision, recall and F-measure for every matching task.

4.2 Comments about the Instance Matching results

The results are very good and encouraging. Both the average and the H-mean F-measure are over 81%. Our system performs slightly better in Freebase than in DBPedia matching, because the lookup service of the former returns fewer and more precise candidates. Therefore, the disambiguation task is easier when working with Freebase data. On GeoNames the result is very good thanks to the use of some shared properties (*geo:long*, *geo:lat*) between the datasets.

5 Conclusions

In this paper we presented the results of the AgreementMaker system for aligning ontologies in the OAEI 2011 competition in the four tracks in which it participated: benchmarks, anatomy, conference, and the data interlinking sub-track. We believe that while our results are very good already, ongoing research can lead to further improvements. The tracks of the OAEI have always been a challenge and have been a relevant measure of quality among matching systems. In order to uphold this standard, we believe that current matching tasks should be expanded to encompass the changing nature of the ontologies being used on the Semantic Web. More specifically, matching large ontologies (more than 50,000 concepts) and focusing on more linked open datasets are important directions to explore in the near future.

References

1. Isabel F. Cruz, Flavio Palandri Antonelli, and Cosmin Stroe. AgreementMaker: Efficient Matching for Large Real-World Schemas and Ontologies. *PVLDB*, 2(2):1586–1589, 2009.
2. Isabel F. Cruz, Flavio Palandri Antonelli, and Cosmin Stroe. Efficient Selection of Mappings and Automatic Quality-driven Combination of Matching Methods. In *ISWC International Workshop on Ontology Matching*. CEUR-WS, 2009.
3. Isabel F. Cruz, Flavio Palandri Antonelli, and Cosmin Stroe. Integrated Ontology Matching and Evaluation. In *International Semantic Web Conference (Posters & Demos)*, 2009.
4. Isabel F. Cruz and Afsheen Rajendran. Exploring a New Approach to the Alignment of Ontologies. In *ISWC Workshop on Semantic Web Technologies for Searching and Retrieving Scientific Data*, volume 83 of *CEUR-WS*, pages 7–12, 2003.
5. Isabel F. Cruz and Afsheen Rajendran. Semantic Data Integration in Hierarchical Domains. *IEEE Intelligent Systems*, March-April:66–73, 2003.
6. Isabel F. Cruz, Afsheen Rajendran, William Sunna, and Nancy Wiegand. Handling Semantic Heterogeneities Using Declarative Agreements. In *ACM Symposium on Advances in Geographic Information Systems (ACM GIS)*, pages 168–174, 2002.
7. Isabel F. Cruz, Cosmin Stroe, Michele Caci, Federico Caimi, Matteo Palmonari, Flavio Palandri Antonelli, and Ulas C. Keles. Using AgreementMaker to Align Ontologies for OAEI 2010. In *Proceedings of the International Conference on Biomedical Ontology*, volume 689. CEUR-WS, 2010.
8. Isabel F. Cruz and William Sunna. Structural Alignment Methods with Applications to Geospatial Ontologies. *Transactions in GIS, special issue on Semantic Similarity Measurement and Geospatial Applications*, 12(6):683–711, 2008.
9. Isabel F. Cruz, William Sunna, and Anjli Chaudhry. Ontology Alignment for Real-World Applications. In *National Conference on Digital Government Research (dg.o)*, pages 393–394, 2004.
10. Isabel F. Cruz, William Sunna, and Anjli Chaudhry. Semi-Automatic Ontology Alignment for Geospatial Data Integration. In *International Conference on Geographic Information Science (GIScience)*, volume 3234 of *Lecture Notes in Computer Science*, pages 51–66. Springer, 2004.
11. Isabel F. Cruz, William Sunna, Nalin Makar, and Sujan Bathala. A Visual Tool for Ontology Alignment to Enable Geospatial Interoperability. *Journal of Visual Languages and Computing*, 18(3):230–254, 2007.
12. Isabel F. Cruz, William G. Sunna, and Kalyan Ayloo. Concept Level Matching of Geospatial Ontologies. In *GIS Planet International Conference and Exhibition on Geographic Information*, 2005.
13. Isabel F. Cruz and Huiyong Xiao. Data Integration for Querying Geospatial Sources. In John Sample, Kevin Shaw, Shengru Tu, and Mahdi Abdelguerfi, editors, *Geospatial Services and Applications for the Internet*, pages 113–137. Springer, 2008.
14. Anika Gross, Michael Hartung, Toralf Kirsten, and Erhard Rahm. Mapping Composition for Matching Large Life Science Ontologies. In *Proceedings of the International Conference on Biomedical Ontology*, pages 109–116, 2011.
15. William Sunna and Isabel F. Cruz. Structure-Based Methods to Enhance Geospatial Ontology Alignment. In *International Conference on GeoSpatial Semantics (GeoS)*, volume 4853 of *Lecture Notes in Computer Science*, pages 82–97. Springer, 2007.
16. Samir Tartir, I. Budak Arpinar, Michael Moore, Amit P. Sheth, and Boanerges Aleman-Meza. OntoQA: Metric-Based Ontology Quality Analysis. In *IEEE Workshop on Knowledge Acquisition from Distributed, Autonomous, Semantically Heterogeneous Data and Knowledge Sources*, volume 9, 2005.

AROMA results for OAEI 2011

Jérôme David[1]

Université Pierre-Mendès-France, Grenoble
Laboratoire d'Informatique de Grenoble
INRIA Rhône-Alpes, Montbonnot Saint-Martin,
France
Jerome.David-at-inrialpes.fr

Abstract. This paper presents the results obtained by AROMA for its participation to OAEI. AROMA is an ontology alignment method that makes use of the association paradigm and a statistical interestingness measure, the implication intensity. AROMA performs a post-processing step that includes a terminological matcher. This year we do not modify this matcher.

1 Presentation of AROMA

1.1 State, purpose, general statement

AROMA is an hybrid, extensional and asymmetric matching approach designed to find out relations of equivalence and subsumption between entities, i.e. classes and properties, issued from two textual taxonomies (web directories or OWL ontologies). Our approach makes use of the association rule paradigm [Agrawal *et al.*, 1993], and a statistical interestingness measure. AROMA relies on the following assumption: *An entity A will be more specific than or equivalent to an entity B if the vocabulary (i.e. terms and also data) used to describe A, its descendants, and its instances tends to be included in that of B.*

1.2 Specific techniques used

AROMA is divided into three successive main stages: (1) The pre processing stage represents each entity, i.e. classes and properties, by a set of terms, (2) the second stage consists of the discovery of association rules between entities, and finally (3) the post processing stage aims at cleaning and enhancing the resulting alignment.

The first stage constructs a set of relevant terms and/or datavalues for each class and property. To do this, we extract the vocabulary of class and property from their annotations and individual values with the help of single and binary term extractor applied to stemmed text. In order to keep a morphism between the partial orders of class and property subsumption hierarchies in one hand and the inclusion of sets of term in the other hand, the terms associated with a class or a property are also associated with its ancestors.

The second stage of AROMA discovers the subsumption relations by using the association rule model and the implication intensity measure [Gras *et al.*, 2008]. In the context of AROMA, an association rule $a \rightarrow b$ represents a quasi-implication (i.e. an implication allowing some counter-examples) from the vocabulary of entity a into the vocabulary of the entity b. Such a rule could be interpreted as a subsumption relation from the antecedent entity toward the consequent one. For example, the binary rule $car \rightarrow vehicle$ means: "The concept car is more specific than the concept $vehicle$". The rule extraction algorithm takes advantage of the partial order structure provided by the subsumption relation, and a property of the implication intensity for pruning the search space.

The last stage concerns the post processing of the association rules set. It performs the following tasks:

- deduction of equivalence relations,
- suppression of cycles in the alignment graph,
- suppression of redundant correspondences,
- selection of the best correspondence for each entity (the alignment is an injective function),
- the enhancement of the alignment by using equality and a string similarity -based matcher.

The equality -based matche considers that two entities are equivalent if they share at least one annotation. This matcher is only applied on unaligned pairs of entities.

The string similarity based matcher still makes use of Jaro-Winkler similarity but relies on a different weighting scheme. As an ontology entity is associated to a set of annotations, i.e. local name, labels and comments, we use a collection measure for aggregating the similarity values between all entity pairs.

In order to favour the measure values of most similar annotations pairs, we choose to use the following collection measure:

$$\Delta_{mw}(e,e') = \begin{cases} \frac{\sum_{a \in T(e)} \arg\max_{a' \in T(e')} sim_{jw}(a,a')^2}{\sum_{a \in T(e)} \arg\max_{a' \in T(e')} sim_{jw}(a,a')} & \text{if } |T(e)| \leq |T(e')| \\ \Delta_{mw}(e',e) & \text{otherwise} \end{cases}$$

where $T(e)$ is the set which contains the annotations and the local name of e, and sim_{jw} is the Jaro-Winkler similarity. For all OAEI tracks, we choose a threshold value of 0.8.

For more details about AROMA, the reader should refer to [David *et al.*, 2007; David, 2007].

1.3 Link to the system and parameters file

The version 1.1 of AROMA has been used for OAEI2022. This version can be downloaded at : `http://gforge.inria.fr/frs/download.php/23649/AROMA-1.1.zip`.

The command line used for aligning two ontologies is:

```
java -jar aroma.jar onto1.owl onto2.owl [alignfile.rdf]
```

The resulting alignment is provided in the alignment format.

1.4 Link to the set of provided alignments (in align format)

http://www.inrialpes.fr/exmo/people/jdavid/oaei2009/results_AROMA_oaei2009.zip

2 Results

We participated to the benchmark, anatomy and conference tracks. We used the same configuration of AROMA for all tracks. We did not experience scaling problem. Since AROMA relies on syntactical data without using any multilingual resources, it is not able to find alignment on the multilingual library track. Finally, we also did not participate either to the instance matching track since AROMA is not designed for such a task.

2.1 Benchmark

Since AROMA mainly relies on textual information, it obtains bad recall values when the alterations affect all text annotations both in the class/property descriptions and in their individual/property values. AROMA does not seem to be influenced by structural alterations (222-247). On these tests, AROMA favours high precision values in comparison to recall values.

2.2 Anatomy

On anatomy test, we do not use any particular knowledge about biomedical domain. AROMA runs quite fast since it takes benefits of the subsumption relation for pruning the search space. ROMA needs around 1 min. to compute the alignment. This pruning feature used by AROMA partially explained the low recall values obtained last year. We enhanced the recall by using also an string equality based matcher before using the lexical similarity based matcher. Since AROMA returns not only equivalence correspondences but also subsumption correspondences, its precision value is negatively influenced. It could be interesting to evaluate results by using semantic precision and recall.

3 General comments

3.1 Comments on the OAEI test cases

In this section, we give some comments on the directory and oriented matching tracks of OAEI.

Directory The two large directories, that were given in previous editions of OAEI, are divided into very small sub directories. AROMA cannot align such very small directories because our method is based on a statistical measure and then it needs some large amount of textual data. However, AROMA discovers correspondences when it is applied to the complete directories. It would be interesting to reintroduce these large taxonomies for the next editions.

Oriented matching We did not participate to this track because we think that it is not well designed. Indeed, the proposed reference alignments are not complete.

For example in the 303 test, the reference alignment contains:

- 101#MastersThesis $\leq$ 103#Academic
- 103#MastersThesis $\leq$ 101#Academic

Obviously, no reliable matching algorithm would return these two correspondences but rather:

- 101#MastersThesis $\equiv$103#MastersThesis
- 101#Academic $\equiv$ 103#Academic

In addition, from these two last correspondences, we could easily deduce the two first ones.

Our suggestion for designing a better oriented matching track would be to remove some classes and properties in the target ontologies so as to obtain complete reference alignments with some subsumption relations. For example, it would be more accurate to remove the concept MasterThesis from the ontology 103 in order to naturally change 101#MastersThesis $\equiv$103#MastersThesis by 101#MastersThesis $\leq$ 103#Academic in the reference alignment.

4 Conclusion

AROMA is a time efficient matcher but suffers of its prunning strategy: it has lower support than other matchers. On anatomy track the precision is also degradated due to the subomption correspondences it returns. For the next editions, we should improve AROMA in terms of recall and precision. One way for doing that is to tune the parameters and also to had some strucural matcher.

References

[Agrawal *et al.*, 1993] Rakesh Agrawal, Tomasz Imielinski, and Arun Swami. Mining association rules between sets of items in large databases. In *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data*, pages 207–216. ACM Press, 1993.

[David and Euzenat, 2008] Jérôme David and Jérôme Euzenat. Comparison between ontology distances (preliminary results). In *Proceedings of the 7th International Semantic Web Conference*, volume 5318 of *Lecture Notes in Computer Science*, pages 245–260. Springer, 2008.

[David *et al.*, 2007] Jérôme David, Fabrice Guillet, and Henri Briand. Association rule ontology matching approach. *International Journal on Semantic Web and Information Systems*, 3(2):27–49, 2007.

[David, 2007] Jérôme David. *AROMA : une méthode pour la découverte d'alignements orientés entre ontologies à partir de règles d'association.* PhD thesis, Université de Nantes, 2007.

[Gras *et al.*, 2008] Régis Gras, Einoshin Suzuki, Fabrice Guillet, and Filippo Spagnolo, editors. *Statistical Implicative Analysis, Theory and Applications*, volume 127 of *Studies in Computational Intelligence.* Springer, 2008.

Ontology Matching with CIDER: evaluation report for OAEI 2011

Jorge Gracia[1], Jordi Bernad[2], and Eduardo Mena[2]

[1] Ontology Engineering Group, Universidad Politécnica de Madrid, Spain
jgracia@fi.upm.es
[2] IIS Department, University of Zaragoza, Spain
{jbernad,emena}@unizar.es

Abstract. CIDER is a schema-based ontology alignment system. Its algorithm compares each pair of ontology terms by, firstly, extracting their ontological contexts up to a certain depth (enriched by using lightweight inference) and, secondly, combining different elementary ontology matching techniques. In its current version, CIDER uses artificial neural networks in order to combine such elementary matchers.
In this paper we briefly describe CIDER and comment on its results at the Ontology Alignment Evaluation Initiative 2011 campaign (OAEI'11). In this new approach, the burden of manual selection of weights has been definitely eliminated, while preserving the performance with respect to CIDER's previous participation in the benchmark track (at OAEI'08).

1 Presentation of the system

CIDER (Context and Inference baseD alignER) is a system for ontology alignment that performs semantic similarity computations among terms of two given ontologies. It extracts the ontological context of the compared terms and enriches it by applying lightweight inference rules. Elementary similarity comparisons are performed to compare different features of the extracted ontological contexts. Such elementary comparisons are combined by means of artificial neural networks (ANNs).

CIDER was initially created in the context of a system [9] for discovering the semantics of user keywords and already participated in the OAEI'08 campaign, leading to good results [5]. In CIDER's previous version, the elementary comparisons performed during the similarity computation were combined linearly. The weights of this linear combination were manually tuned after experimentation. This was a major limitation of the approach, which hampered the flexibility of the method and the capacity for quickly adapting it into different domains. This has been solved in the current version by the addition of ANNs.

We expect to confirm that this contribution will not have a negative impact on the initial algorithm. We also expect to discover areas of potential improvement that guide us in our future exploration of this research path. For OAEI'11 campaign, CIDER has participated in the Seals-based tracks[3], i.e., *benchmark, anatomy*, and *conference* tracks.

[3] http://oaei.ontologymatching.org/2011/seals-eval.html

1.1 State, purpose, general statement

According to the high level classification given in [3], our method is a *schema-based* system (opposite to others which are instance-based, or mixed), because it relies mostly on schema-level input information for performing ontology matching. CIDER admits any two OWL ontologies and a threshold value as input. Comparisons among all pairs of ontology terms are established, producing as output an RDF document with the obtained alignments. In its current version, the process is enhanced with the use of ANNs. The type of alignments that CIDER obtains is *semantic equivalences*.

1.2 Specific techniques used

Our alignment process takes as basis the *semantic similarity measure* described in [9], with the improvements introduced in [5]. Briefly explained, the similarity computation is as follows (see Figure 1):

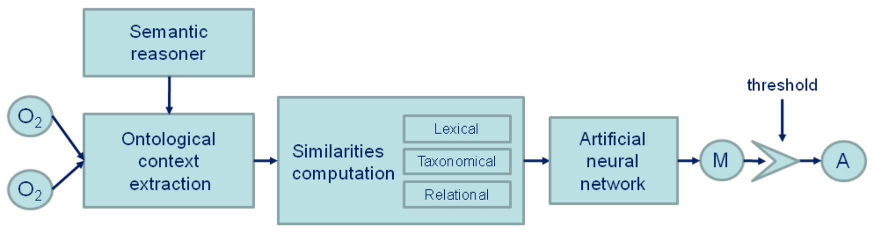


Fig. 1. Scheme of the matching process.

1. Firstly, the ontological context of each ontology term is extracted, up to a certain depth. That is (depending on the type of term), their synonyms, textual descriptions, hypernyms, hyponyms, properties, domains, roles, associated concepts, etc. This process is enriched by applying a lightweight inference mechanism[4], in order to add more semantic information that is not explicit in the asserted ontologies.
2. Secondly, several similarities between each pair of compared terms are computed: Linguistic similarity between the labels of terms (based on Levenhstein [6] similarity) as well as structural similarities, by exploiting the ontological context of the terms and using vector space modelling [7] in the comparisons. This comprises comparison of taxonomies and relationships among terms (e.g., properties of concepts).

[4] Typically transitive inference, although RDFS or more complex rules can be also applied, at the cost of processing time.

3. Then, the different similarities are combined within an ANN to provide a final similarity degree. ANNs constitute an adaptive type of systems composed of interconnected artificial neurons, which change the structure based on external or internal information that flows through the network during a learning phase [8]. CIDER uses two different neural networks for computing similarities between classes and properties, respectively[5].
4. Finally, a matrix (M in Figure 1) with all similarities is obtained. The final alignment (A) is then extracted from this matrix, finding the highest rated one-to-one relationships among terms, and filtering out the ones that are below the given threshold.

Figure 2 shows, as an example, the structure of the neural network for computing similarity between classes (the other one for properties follows an equivalent pattern). Without entering into the details, this corresponds to a *multilayer perceptron*, which consists of multiple layers of nodes in a directed graph, each layer fully connected to the next one. Each connection (synapse) has an associated weight. In our particular situation, the network is composed of three layers: input, hidden, and output layer (with five, three, and one neurons respectively; additionally two bias neurons are used in the input and hidden layer respectively). Each neuron in the input layer receives the value of an elementary similarity measure. Each intermediate neuron uses a sigmoid function to combine the inputs. Finally, the resultant similarity value is given by the neuron in the output layer.

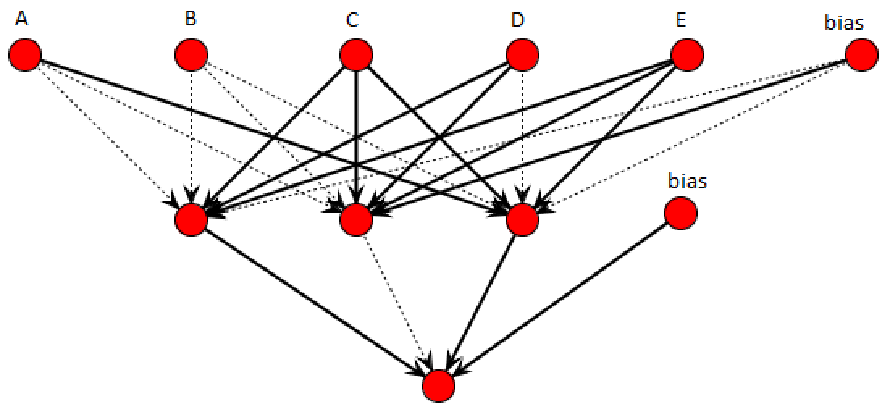


Fig. 2. Scheme of the neural network for computing similarity between classes. Highlighted connexions correspond to higher weights.

[5] Similarity between individuals follows the approach of the previous versions, although the addition of a new ANN for that is planned as future work.

The inputs for the neural network that computes class similarity (labelled A - E in the figure) are: lexical similarity between labels, similarity of textual descriptions (e.g., rdfs:comment), similarity between hypernyms, similarity between hyponyms, and similarity between associated properties.

In terms of implementation, the CIDER prototype has been developed in Java, extending the Alignment API [1]. To create and manipulate neural networks we use Neuroph Studio library[6]. The input to CIDER are OWL ontologies and the output is served as a file expressed in the *alignment format* [1], although it can be easily translated into another formats.

1.3 Adaptations made for the evaluation

According to the conditions of the competition[7] "it is fair to select the set of parameters that provide the best results (for the tests where results are known)". Thus, we chose a subset of the OAEI'08 benchmark to train the neural networks and find suitable weights for combining the elementary matchers that CIDER uses. We used the 2008 benchmark dataset but excluding cases 202 and 248-266 (which present a total absence or randomization of labels and comments). The weights and the configuration of the neural network remained constant for the whole evaluation.

Furthermore, as the Seals-based tracks of the competition do not consider mappings between instances, we have disabled instance comparison. Finally, some minor technical adaptations were required for integrating the system into the Seals platform, such as updating some libraries (e.g., Alignment API) or changing the way some parameters are communicated.

1.4 Link to the system and parameters file

The version of CIDER used for this evaluation (v0.4c) can be found at Seals platform: http://www.seals-project.eu/ . More information can be found at
http://sid.cps.unizar.es/SEMANTICWEB/ALIGNMENT/

1.5 Link to the set of provided alignments (in align format)

The resultant alignments will be provided by the Seals platform: http://www.seals-project.eu/

2 Results

At the time of writing this, the preliminary results of this year competition are available at [2], although the organisers will make public additional results in the future. From the tracks in which CIDER participated, CIDER was not able to produce results on the *anatomy* tests. The reason is that CIDER's current implementation is not optimised

[6] http://neuroph.sourceforge.net/
[7] http://oaei.ontologymatching.org/2011/

to be used with large ontologies, and the execution of the test gave a timeout before completion. Actually only six, out of the 16 participants in the anatomy track, were able to complete the evaluation. We plan to improve CIDER's performance for large ontologies in a near future.

In the following paragraphs, we summarize the results obtained in the benchmark and conference tracks (the detailed results can be found in [2]), as well as some complementary experiments that we run locally.

2.1 Benchmark

The target of this experiment is the alignment of bibliographic ontologies. A reference ontology is proposed, and many comparisons with other ontologies of the same domain are performed. The tests are systematically generated, modifying differently the reference ontology in order to evaluate how the algorithm behaves when the aligned ontologies differ in some particular aspects. A total of 111 test cases have to be evaluated. They are grouped in three sets:

1. Concept test (cases *1xx*: 101, 102, ...), that explore comparisons between the reference ontology and itself, described with different expressivity levels.
2. Systematic (cases *2xx*). It alters systematically the reference ontology to compare different modifications or different missing information.
3. Real ontology (cases *3xx*), where comparisons with other "real world" bibliographic ontologies are explored.

As in our previous participation, we point out that our system is not intended to deal with ontologies in which syntax is not significant at all, as it is the case for benchmark cases 202 and 248-266 (these cases present a total absence or randomization of labels and comments). Consequently, we expect a result with a low recall in this experiment, as these benchmark tests do not favour methods that are not based on graph structure analysis or similar techniques.

In addition to the traditional test data, a new benchmark data set (Benchmark2) has been provided this year. This uses the EKAW conference ontology[8] as basis and, same as in the Benchmark data set, different systematic variations are explored, resulting in 103 test cases.

Finally, there was a "blind" benchmark data set that was used for providing the final results in the competition. This test, consisting of 102 alignments, resulted in 0.89 precision, 0.58 recall and 0.70 F-Measure for CIDER, which is *above the average results in the competition* (0.66 F-Measure, ranging from 0.32 to 0.77). Besides the usual precision/recall, other extended metrics were considered for this test in this year's competition [2]. For instance, *weighted precision/recall* were computed taking into account the score that the tool assigned to each correspondence. For this weighted metrics CIDER obtained: 0.91 precision, 0.54 recall, and 0.68 F-measure (being the *4th best in the competition*, out of 16 participants).

In Table 1 we show the results of evaluating CIDER with the different benchmark tracks: Benchmark (2010), Benchmark2 (2011), and "blind" Benchmark (2011). Additionally, and for comparison purposes, we also run the 2008 benchmark data with the

[8] http://nb.vse.cz/ svabo/oaei2011/data/ekaw.owl

version of CIDER submitted for evaluation (v0.4) and compared it to the results obtained by CIDER at OAEI08 (v0.1). Table 2 shows the results. Baseline results (edit distance) are also included.

	Benchmark	**Benchmark2**	**Benchmark(blind)**
Precision	0.87	0.74	0.89
Recall	0.66	0.58	0.58
F-Measure	**0.75**	**0.65**	**0.70**

Table 1. Averaged results of CIDER for the benchmark datasets.

	baseline(edna)	**CIDER v0.1**	**CIDER v0.4**
Precision	0.56	0.97	0.88
Recall	0.60	0.62	0.69
F-Measure	**0.58**	**0.76**	**0.77**

Table 2. H-mean results for the OAEI08 benchmark dataset.

2.2 Conference

This track consisted of aligning several ontologies from the conference domain, which resulted in 21 alignments. To evaluate the resultant alignments, various F-measures were computed: F1 (harmonic mean between precision and recall), F2 (promotes recall), and F0.5 (promotes precision).

The results given by CIDER were: F1-Measure = 0.53, F2-Measure = 0.48, and F0.5-Measure = 0.61, which place our system in intermediate positions in this track. We expect that the use of more "real world" training data in CIDER will improve the results in this track in future contests. The effect of the threshold selection in these results has been nicely illustrated in the figures provided by the organisers[9], and shows that performance can be dramatically improved with a suitable selection of the threshold.

3 General comments

The following subsections contain some remarks and comments about the results obtained and the evaluation process.

[9] See http://nb.vse.cz/∼svabo/oaei2011/eval.html#reference)

3.1 Comments on the results

As it is shown in Table 2, a direct comparison between the current and previous version of CIDER shows that the addition of ANNs does not has a negative effect on the algorithm but, on the contrary, leads to slightly better results. Such results indicate also that the new approach leads to a better recall, at the cost of precision.

The results for benchmark tracks (Table 1), although acceptable, are hampered by the presence of test cases with ontologies lacking lexical information or that has been randomly generated.

With respect to the conference track, the results are influenced by the fact that our ANNs only used open data from the benchmark track for training. More reference alignments from "real world" ontologies will be used in the future for training the ANNs, in order to cover different domains and different types of ontologies.

3.2 Discussions on the way to improve the proposed system

Although the obtained results are acceptable, we consider that there is still room for further improvements. In fact, the addition of ANNs for similarity computation in CIDER is in a preliminary stage and has to be further studied. We have to use more "real" data for training, and alternative configurations for our multilayer perceptrons has to be studied. On the other hand, time response in CIDER is still an issue and has to be further improved. Also, CIDER works well with small and medium sized ontologies but not with large ones. Partitioning and other related techniques will be explored in order to solve this.

3.3 Comments on the OAEI 2011 test cases

We have found the benchmark test very useful as a guideline for our internal improvements of the method, as well as to establish a certain degree of comparisons with other existing methods. On the other hand, we have missed some important issues that are not taken into account in the systematic benchmark series. Some of them coincide with the ones we already reported in 2008:

1. Benchmark tests only consider positive matchings, not measuring the ability of different methods to avoid links among barely related ontologies.
2. For our purposes, we try to emulate the human behaviour when mapping ontological terms. As human experts cannot properly identify mappings between ontologies with scrambled texts, neither does our system. However, reference alignments provided in the benchmark evaluation for cases 202 and 248-266, do not follow this intuition. We hope this bias will be reduced in future contests.
3. Related to the latter, cases in which equal topologies, but describing different things, lead to false positives, are not explicitly taken into account in the benchmark.
4. How ambiguities can affect the method is not considered either in the test cases. It is a consequence of using ontologies belonging to the same domain. For example, it would be interesting to evaluate whether "film" in an ontology about movies is mapped to "film" as a "thin layer" in another ontology. Therefore it is difficult to evaluate the benefits of including certain disambiguation techniques in ontology matching [4].

4 Conclusion

CIDER is a schema-based alignment system that compares the ontological contexts (enriched with transitive inference) of each pair of terms in the aligned ontologies. Several elementary ontology matching techniques are computed and combined by means of artificial neural networks. We have presented here some results of the participation of CIDER at OAEI'11 contest, particularly in the Seals-based tracks (benchmark, anatomy, and conference). The results on the benchmark track are good and constitute our starting point for testing future improvements. We confirmed that the addition of artificial neural networks keeps the performance and, furthermore, eliminates the necessity of tuning the weights manually.

Acknowledgments. This work is supported by the CICYT project TIN2010-21387-C02-02, the Spanish Ministry of Science and Innovation within the Juan de la Cierva program, and the EC within the FP7 project DynaLearn (no. 231526).

References

1. J. Euzenat. An API for ontology alignment. In *3rd International Semantic Web Conference (ISWC'04), Hiroshima (Japan)*. Springer, November 2004.
2. J. Euzenat, A. Ferrara, W. R. van Hage, L. Hollink, C. Meilicke, A. Nikolov, D. Ritze, F. Scharffe, P. Shvaiko, H. Stuckenschmidt, O. Šváb-Zamaza, and C. Trojahn. First results of the ontology alignment evaluation initiative 2011. In *In Proc. of 6th Ontology Matching Workshop (OM11), at International Semantic Web Conference (ISWC11), Bonn, Germany*, 2011.
3. J. Euzenat and P. Shvaiko. *Ontology matching*. Springer-Verlag, 2007.
4. J. Gracia, V. López, M. d'Aquin, M. Sabou, E. Motta, and E. Mena. Solving semantic ambiguity to improve semantic web based ontology matching. In *Proc. of 2nd Ontology Matching Workshop (OM'07), at 6th International Semantic Web Conference (ISWC'07), Busan (Korea)*, November 2007.
5. J. Gracia and E. Mena. Ontology matching with CIDER: Evaluation report for the OAEI 2008. In *Proc. of 3rd Ontology Matching Workshop (OM'08), at 7th International Semantic Web Conference (ISWC'08), Karlsruhe, Germany*, volume 431, pages 140–146. CEUR-WS, October 2008.
6. V. Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. *Cybernetics and Control Theory*, 10(8):707–710, 1966. Original in *Doklady Akademii Nauk SSSR* 163(4): 845–848 (1965).
7. V. V. Raghavan and M. S. K. Wong. A critical analysis of vector space model for information retrieval. *Journal of the American Society for Information Science*, 37(5):279–287, 1986.
8. M. Smith. *Neural Networks for Statistical Modeling*. John Wiley & Sons, Inc., New York, NY, USA, 1993.
9. R. Trillo, J. Gracia, M. Espinoza, and E. Mena. Discovering the semantics of user keywords. *Journal on Universal Computer Science. Special Issue: Ontologies and their Applications*, November 2007.

CODI: Combinatorial Optimization for Data Integration – Results for OAEI 2011

Jakob Huber, Timo Sztyler, Jan Noessner, and Christian Meilicke

KR & KM Research Group
University of Mannheim, Germany
{jahuber, tsztyler}@mail.uni-mannheim.de
{jan, christian}@informatik.uni-mannheim.de

Abstract. In this paper, we describe our probabilistic-logical alignment system CODI (Combinatorial Optimization for Data Integration). The system provides a declarative framework for the alignment of individuals, concepts, and properties of two heterogeneous ontologies. CODI leverages both logical schema information and lexical similarity measures with a well-defined semantics for A-Box and T-Box matching. The alignments are computed by solving corresponding combinatorial optimization problems.

1 Presentation of the system

1.1 State, purpose, general statement

CODI (**C**ombinatorial **O**ptimization for **D**ata **I**ntegration) leverages terminological structure for ontology matching. The current implementation produces mappings between concepts, properties, and individuals. The system combines lexical similarity measures with schema information to completely avoid *incoherence* and *inconsistency* during the alignment process. CODI participates in 2011 for the second time in an OAEI campaign. Thus, we put a special focus on differences compared to the previous 2010 version of CODI.

1.2 Specific techniques used

CODI is based on the syntax and semantics of Markov logic [2] and transforms the alignment problem to a maximum-a-posteriori optimization problem. This problem needs a-priori confidence values for each matching hypotheses as input. Therefore, we implemented an aggregation method of different similarity measures. Another new feature of CODI is the recognition of ontology pairs belonging to different versions of the same ontology. In instance matching CODI does not compute lexical similarities for all existing pairs of instances but utilizes object-property assertions for reducing the necessary comparisons.

Markov Logic Framework Markov logic combines first-order logic and undirected probabilistic graphical models [11]. A Markov logic network (MLN) is a set of first-order formulae with weights. Intuitively, the more evidence there is that a formula is true the higher the weight of this formula. It has been proposed as a possible approach to several problems occurring in the context of the semantic web [2]. We have shown that Markov logic provides a suitable framework for ontology matching as it captures both *hard* logical axioms and *soft* uncertain statements about potential correspondences between entities. The probabilistic-logical framework we propose for ontology matching essentially adapts the syntax and semantics of Markov logic. However, we always *type* predicates and we require a strict distinction between *hard* and *soft* formulae as well as *hidden* and *observable* predicates. Given a set of constants (the classes and object properties of the ontologies), a set of formulae (the axioms holding between the objects and classes), and confidence values for correspondences, a Markov logic network defines a probability distribution over possible alignments. We refer the reader to [8, 7] for an in-depth discussion of the approach and some computational challenges. For generating the Marcov logic networks we used the approach described in [12]. Our OAEI paper from last year contains a more technical description of the framework [9].

Cardinality Constraints A method often applied in real-world scenarios is the selection of a functional one-to-one alignment [1]. Within the ML framework, we can include a set of hard cardinality constraints, restricting the alignment to be functional and one-to-one.

Coherence Constraints Incoherence occurs when axioms in ontologies lead to logical contradictions. Clearly, it is desirable to avoid incoherence during the alignment process. All existing approaches that put a focus on alignment coherence remove correspondences after computing the alignment. Within the ML framework we can incorporate incoherence reducing constraints *during* the alignment process.

Stability Constraints Several approaches to ontology matching propagate alignment evidence derived from structural relationships between concepts and properties. These methods leverage the fact that existing evidence for the equivalence of concepts C and D also makes it more likely that, for example, child concepts of C and D are equivalent. One such approach to evidence propagation is *similarity flooding* [6]. As a reciprocal idea, the general notion of stability was introduced, expressing that an alignment should not introduce new structural knowledge [5].

Combination of Different Similarity Measures Compared to last year we improved our lexical string similarity measures significantly. In a first step we collect and standardize all string information like ids, labels and annotations from the entities. During the standardization process we split tokens into separate words if necessary (e.g. *hasAuthor* is transformed to *has Author*), replace special characters with spaces, and remove few words like *a* or *the* according to a stop-words list.

Furthermore, the functionality of computing string similarities has been improved. CODI is able to combine several string similarity measures by taking the average, the maximum or by weighting each measure with a specific predefined weight. These weights could be learned with machine learning algorithms. In the standard configuration CODI combines the Cosine, Levenshtein, Jaro Winkler, Simth Waterman Goto, Overlap coefficient, and Jaccard similarity measures[1] with specific weights.

Matching different Ontology Versions A specific task in ontology matching is the alignment of different versions of the same ontology. The test cases of the benchmark track can be seen as an example for this kind of task. In the following we argue that (a) matching versions requires a different approach compared to a standard matching task, and (b) that, therefore, it is required to detect automatically that two ontologies are different versions of the same ontology.

(a) Suppose that $\mathcal{O}$ and $\mathcal{O}'$ are versions of the same ontology. Further, let $\mathcal{O}$ contain less concepts and properties than $\mathcal{O}'$. Then it is highly probable that many or nearly all entities in $\mathcal{O}$ have a counterpart in $\mathcal{O}'$. A good one-to-one alignment will have, thus, as many correspondences as there are entities in $\mathcal{O}$. Based on this assumption it makes sense to lower the threshold or to use a structural measure in addition to the computation of string-based similarities. In particular, we apply the following measure.

We first calculate the number of subclasses $\#sub$, superclasses $\#sup$, disjoint classes $\#dis$, and domain- and range-restrictions ($\#dom$ and $\#ran$) for a specific concept C. These results are then used to calculate a similarity. For example, given $C \in \mathcal{O}$ and $D \in \mathcal{O}'$ we have $sim_{\#sub}(C, D) = (1 + min(\#sub(C), \#sub(D)))/(1 + max(\#sub(C), \#sub(D)))$. The overall similarity $sim(C, D)$ is then computed as weighted average over all different similarity values for each of $\#sub$, $\#sup$, $\#dis$, $\#dom$, $\#ran$.

The resulting similarity measure is highly imprecise, but has a high recall if we apply it to two ontologies with high structural similarity. Whenever there is a high probability that the two input ontologies are versions of the same ontology, we add for each concept C the top-k counterparts D with respect to $sim(C, D)$ as matching hypotheses with low confidence to the optimization problem (same for properties). This approach sounds quite drastic, but keep in mind that there are anchor-correspondences generated by our string-based measures and constraints that interact and result in a meaningful final alignment.

(b) In order to determine whether two ontologies are versions of each other, we apply the Hungarian method on the input generated by our structural measure. The Hungarian method finds an optimal one-to-one alignment $\mathcal{A}_{opt}$. Now suppose that we match an ontology on itself. The number of correspondences in $\mathcal{A}_{opt}$ is then equal to the number of entities in the ontology, i.e., $\mathcal{A}_{opt}$ has a full coverage. Moreover, the total of confidences $\sum_{c \in \mathcal{A}_{opt}} conf(c)$ will be $|\mathcal{A}_{opt}|$. In general, we assume that $\sum_{c \in \mathcal{A}_{opt}} conf(c)$ divided by the size of the smaller ontology is close to 1 for versions of the same ontology. In particular, we treat each pair of ontologies as versions if the measured value is above 0.9.

[1] Implemented in http://sourceforge.net/projects/simmetrics/.

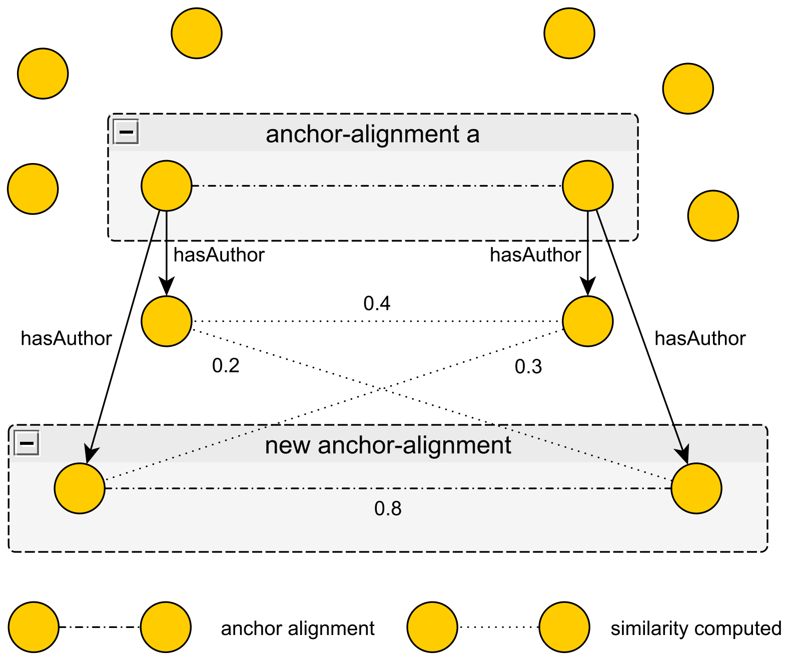


Fig. 1. Process of Selecting Individuals for Computing their Lexical Similarities with $thres = 0.7$.

Instance Matching In real-world instance matching tasks we are often faced with data sources containing a large amount of instances. Hence, it is obvious that computing the lexical similarity for every pair of these instances is not suitable. We implemented an approach which utilizes object-properties to determine the instances for which the similarity should be computed. Our approach assumes that we have one common TBox and two different ABoxes. Consequently, we assume that both TBoxes have been integrated beforehand.

In a first step we compute *anchor*-alignments. Therefore, we compare a small subset of all individuals with each other (e.g. all individuals which are asserted to a specific concept like $Film$), compute their lexical similarities $lexSim$, and add those to the anchor-alignments if their respective similarities are above a threshold $thres$. Then, we take the first anchor-alignment a. For all individuals which are connected with an object-property-assertion with one of the individuals in the alignment a we again compute the lexical similarity $lexSim$. We add them to the *end* of the anchor-alignments if $lexSim$ is higher than the threshold $thres$. Figure 1 visualizes this process. The anchor-alignments is a unique set, which means that only new alignments are added. We repeat this procedure for the second, third, and all following anchor-alignments until we went through the whole set.

The lexical similarity $lexSim$ is computed as described in [9]. However, we integrated coherence checks as proposed by [10] in order to avoid inconsistent alignments. Comparisons can be further reduced, by omitting those individual pairs which have no asserted inferred concept in common.

This basic idea is extended by some post-processing steps. For catching correspondences which are not connected with an object-property-assertion, we compare all remaining individuals which do not yet occur in the anchor-alignment and add them if their lexical similarity $lexSim$ is above $thres$. At the end, a greedy algorithm for computing a one-to-one alignment is applied.

These techniques reduce the runtime significantly on large instance-matching benchmarks.

1.3 Adaptations made for the evaluation

Prior to each matching task, CODI automatically analyzes the input ontologies and adapts itself to the matching task. The first distinction is based on the use of OBO constructs. If this is the case CODI automatically switches to a setting optimized for matching biomedical terms. The main difference in this setting is the use of a different similarity measure which exploits the fact that in medical domains the order of words is often transposed. The measure basically splits the two strings in two sets of words and computes the largest common subset of these sets relative to the smaller one.

If this is not the case CODI checks if the ontologies might be versions of the same ontology. This test does not always correctly discriminate and we sometimes do not detect that two ontologies are different version of the same ontology resulting in poor performance for some of the benchmark test cases.

1.4 Link to the System

CODI can be downloaded from the SEALS portal via `http://www.seals-project.eu/tool-services/browse-tools`. Further information, an executable jar file, and the source code are available at `http://code.google.com/p/codi-matcher/`.

1.5 Link to the Set of Provided Alignments

The alignments for the tracks *Benchmark*, *Conference*, and *Anatomy* has been created on top of the SEALS platform. For *IIMB* the alignments can be found at `http://code.google.com/p/codi-matcher/downloads/list`

2 Results

Benchmark Track The benchmark track is constructed by applying controlled transformations on one source ontology. Thus, all test-cases consist of different versions of the same ontology. However, our *adaptive* method for detecting these ontologies only categorize about 50 % beeing different versions of each other. Especially if their semantic structure is heavily changed (e.g. deleting class hierarchy, etc.) our algorithm fails. Nevertheless, with our adaptive method we were able to improve our F_1 score from 0.51 to 0.75 compared to last year. If all test-cases would have been *correctly* categorized as different versions CODI's F_1 score would have been 0.83 which is 32 % higher than last year. For the newly introduced dataset 2 our adaptive setting even produces a slightly higher F_1 score of 0.70 compared to the correct assignments. Thus, the structure of some test cases differs so much that it is beneficial to consider them *not* as ontologies of the same version (even if they are). The results are shown in Table 1.

Table 1. Benchmark results

	Dataset 1			Dataset 2	
	2011		2010	2011	
	adaptive	correct		adaptive	correct
Precision	0.88	0.90	0.72	0.86	0.80
Recall	0.65	0.77	0.44	0.59	0.61
F_1 score	0.75	0.83	0.51	0.70	0.69

Conference Track Since the conference dataset contains many trivial correspondences matchers can easily reach a high precision. The challenge of this dataset consists in finding the non-trivial correspondences. Concentrating on these non-trivial correspondences we were able to increase our recall from 0.51 to 0.61 compared to the results of last year and gained 2 % additional F_1 score. In the conference track CODI was able to detect that all ontology pairs are not versions of the same ontology. Consequently, the adaptive and the correctly assigned results are similar (see Table 2). We also made some experiments where we matched the Conference ontologies with the fixed version-setting. We observed a significant loss in precision. This illustrates the importance of an adaptive approach.

Table 2. Conference results

	2011		2010
	adaptive	correct	
Precision	0.75	0.75	0.87
Recall	0.61	0.61	0.51
F_1 score	0.66	0.66	0.64

Anatomy Track Due to our special lexical similarity measure for medical ontologies, we were able to improve our F_1 score of last year from 0.794 to 0.879. Currently, our results are better than the best participating system of the OAEI 2010. CODI requires approximately 35min to finish this matching task on a 2.3GHz dual core machine with 8G RAM.

Table 3. Anatomy results

	2011	2010
Precision	0.955	0.954
Recall	0.815	0.680
F_1 score	0.879	0.794

IIMB Track The IIMB benchmark is created by applying lexical, semantical, and structural transformation techniques on real data extracted from freebase [3]. The transformations are divided into four transformation categories containing 20 transformations each. The size of the IIMB track heavily increased compared to last year. Each of the 80 existing transformations consist of ontology files with sizes larger than 20 MB. For computing a very basic string similarity for every pair of individuals the runtime explodes to over one hour per test case. With our new instance matching method which only compares related individuals we were able to reduce the runtime to 34 minutes per test-case in average. This runtime includes the time for consistency checking, for computing a functional one-to-one alignment, and for calculating a more sophisticated lexical similarity.

Beside the increase in size, the transformations have been made much harder. Thus, comparisons to last year results are not expedient. Table 4 summarizes the different results of the CODI system for each of the 4 transformation categories[2].

Table 4. IIMB results

Transformations	0-20	21-40	41-60	61-80	overall
Precision	0.93	0.83	0.73	0.66	0.79
Recall	0.78	0.59	0.67	0.28	0.63
F_1 score	0.84	0.68	0.64	0.36	0.66

3 General comments

3.1 Discussions on the way to improve the proposed system

Improvements in usability by designing a suitable user interface are future steps that have to be taken. Although we focussed this year on the implementation and evaluation of a combination of more sophisticated lexical similarity measures, we think that we still have not exploit CODIs full potential regarding this issue. Last but not least improvements in matching different ontology versions will be subject of next years participation.

3.2 Comments on the OAEI 2011 procedure

The SEALS evaluation campaign is very beneficial since it is the first time that the matchers are publically available for download implementing a common interface.

3.3 Comments on the OAEI 2011 measures

We encourage the organizers to use semantic precision and recall measures as described in [4].

[2] In several test cases every supplementary information for individuals has been deleted. These test cases will not be considered in the official OAEI evaluation and, thus, are omitted here.

4 Conclusion

This year we improved the lexical similarity measures and developed a methodology for automatically choosing between different settings. Combining these improvements with our Markov logic system from last year, we were able to improve our results for the anatomy, conference, and benchmark track significantly. Furthermore, we developed a new instance matching algorithm, which only computes the similarity of promising instances. With this technique we were able to reduce the runtime of the large instance matching benchmark.

The strength of the CODI system is the combination of lexical and structural information and the declarative nature that allows easy experimentation. We will continue the development of the CODI system and hope that our approach inspires other researchers to leverage terminological structure and logical reasoning for ontology matching.

References

1. I. Cruz, F. Palandri, Antonelli, and C. Stroe. Efficient selection of mappings and automatic quality-driven combination of matching methods. In *Proceedings of the ISWC 2009 Workshop on Ontology Matching*, 2009.
2. P. Domingos, D. Lowd, S. Kok, H. Poon, M. Richardson, and P. Singla. Just add weights: Markov logic for the semantic web. In *Proceedings of the Workshop on Uncertain Reasoning for the Semantic Web*, pages 1–25, 2008.
3. A. Ferrara, S. Montanelli, J. Noessner, and H. Stuckenschmidt. Benchmarking matching applications on the semantic web. In *The Semanic Web: Research and Applications - 8th Extended Semantic Web Conference, ESWC 2011*, Lecture Notes in Computer Science, pages 108–122, Heraklion, Crete, Greece, 2011. Springer.
4. D. Fleischhacker and H. Stuckenschmidt. A Practical Implementation of Semantic Precision and Recall. In *2010 International Conference on Complex, Intelligent and Software Intensive Systems*, pages 986–991. IEEE, 2010.
5. C. Meilicke and H. Stuckenschmidt. Analyzing mapping extraction approaches. In *Proceedings of the Workshop on Ontology Matching*, Busan, Korea, 2007.
6. S. Melnik, H. Garcia-Molina, and E. Rahm. Similarity flooding: A versatile graph matching algorithm and its application to schema matching. In *Proceedings of ICDE*, pages 117–128, 2002.
7. M. Niepert. A Delayed Column Generation Strategy for Exact k-Bounded MAP Inference in Markov Logic Networks. In *Proceedings of the 25th Conference on Uncertainty in Artificial Intelligence*, 2010.
8. M. Niepert, C. Meilicke, and H. Stuckenschmidt. A Probabilistic-Logical Framework for Ontology Matching. In *Proceedings of the 24th AAAI Conference on Artificial Intelligence*, 2010.
9. J. Noessner and M. Niepert. Codi: Combinatorial optimization for data integration–results for oaei 2010. *Ontology Matching*, page 142, 2010.
10. J. Noessner, M. Niepert, C. Meilicke, and H. Stuckenschmidt. Leveraging Terminological Structure for Object Reconciliation. *The Semantic Web: Research and Applications*, pages 334–348, 2010.
11. M. Richardson and P. Domingos. Markov logic networks. *Machine Learning*, 62(1-2):107–136, 2006.
12. S. Riedel. Improving the accuracy and efficiency of map inference for markov logic. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, pages 468–475, 2008.

Cluster-based Similarity Aggregation for Ontology Matching

Quang-Vinh Tran[1], Ryutaro Ichise[2], and Bao-Quoc Ho[1]

[1] Faculty of Information Technology, Ho Chi Minh University of Science, Vietnam
{tqvinh,hbquoc}@fit.hcmus.edu.vn
[2] Principles of Informatics Research Division, National Institute of Informatics, Tokyo, Japan
ichise@nii.ac.jp

Abstract. Cluster-based similarity aggregation (CSA) is an automatic similarity aggregating system for ontology matching. The system have two main part. The first is calculation and combination of different similarity measures. The second is extracting alignment. The system first calculates five different basic measures to create five similarity matrixes, i.e, string-based similarity measure, WordNet-based similarity measure... Furthermore, it exploits the advantage of each measure through a weight estimation process. These similarity matrixes are combined into a final similarity matrix. After that, the pre-alignment is extracted from this matrix. Finally, to increase the accuracy of the system, the pruning process is applied.

1 Presentation of the system

In the Internet, ontologies are widely used to provide semantic to data. Since they are created by different users for different purposes, we need to develop a method to match multiple ontologies for integrating data from different resources [2].

1.1 State, purpose, general statement

CSA (**C**luster-based **S**imilarity **A**ggregation) is the automatic weight aggregating system for ontology alignment. The system is designed to search for semantic correspondence between heterogeneous data sources from different ontologies. The current implementation only support one-to-one alignment between concepts and properties (including object properties and data properties). The core of CSA is utilizing the advantage of each basic strategy for the alignment process. For example, the string-based similarity measure works well when the two entities are similar linguistically while the structure-based similarity measure is effective when the two entities are similar in their local structure. The system automatically combines many similarity measurements based on the analysis of their similarity matrix. Details of the system are described in the following parts.

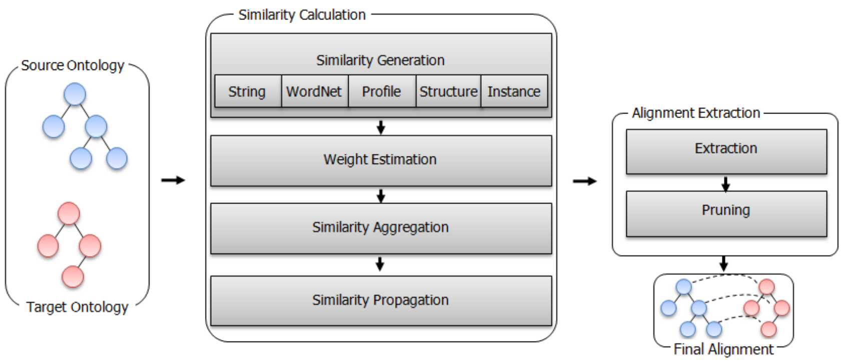


Fig. 1. The main process of CSA

1.2 Specific techniques used

The process of the system is illustrated in Figure 1. First, we calculate five basic similarity measures. These similarities are String edit distance, WordNet based, Profile, Structure, and Instance based. Second, the weight for each similarity is estimated through a weight estimation process. We then aggregate these similarities based on their weights. After that, we propagate the similarity to get the final similarity matrix. The pre-alignment is then extracted from this matrix. Finally, we apply the pruning process to get the final alignment.

Similarity Generation The similarity between entities in the two ontologies is computed by five basic measures. The String edit distance measures the lexical feature of the entity's name. The WordNet [3] exploits the similarity between words occur in the entity's name. We use the method of Wu and Palmer for calculating the WordNet similarity [8]. The profile similarity makes use of the id, label, and comments information contained in an entity. The profile for a class takes their properties, instances into account. The profile for a property includes their domains and their ranges. We then construct the weight feature vector using tf-idf. The similarity is then calculated by the cosine similarity of the two vectors. The structure similarity is calculated for class only. This similarity measures the difference in the local structure of an entity. We implement the method introduced in [7] for the structure measure. This calculation is based on the difference of number of class's children, number of a class's siblings, the normalized depth from the root and the number of properties restricted to this class. The instance-based measure is similar to the profile except that we only utilize the content of instances that belong to classes and the properties appear in these instances.

Weight estimation Weight estimation is the core of CSA. In this step, we analyze each similarity matrix of each basic measure to find which one is actually effective for the alignment. This process is based on two sources of information. First, for each single method, the process of finding a threshold for distinguishing matching pairs from non matching pairs can be viewed as a binary classification problem [5]. The positive class contains matching pairs and the negative class contains non matching ones. Second, in one-to-one ontology alignment, the maximum number of matching pairs is equal to the minimum number of entities in the two ontologies. If a single method is effective, its correspondent similarity matrix must have the two criteria: The matrix that can distinguish matching from non matching pairs and the number of matching pairs must approximate the minimum number of entities in the two ontologies.

On the basis of these criteria, we model the weight estimation process for concept as follows: First, for each similarity matrix we use the K-means algorithm to cluster the similarity values into two different classes (k = 2). The feature is the similarity value of each pair of classes. The cluster with higher mean represents the matching set, and the lower one represents the non matching set. We filter out all the values that belong to the non matching set. What remains is the similarity matrix with the higher values. We then we calculate the number of row that has value in the matrix. These row represent the possible matching pairs. Because in our case we consider the one-to-one matching, one concept from source ontology is only matched up to one concept from target ontology. Finally, the weight is estimated by the ratio of the number of rows over the number of matched values in the filtered matrix.

$$weight = \frac{|\textit{number of row that has value}|}{|\textit{number of value in matching set}|} \tag{1}$$

The weight estimation for property similarity matrix is calculated in the same manner.

Similarity Aggregation The similarity combination can be defined as the weight average of the five basic measures. The weight for each measure is estimated in the previous step.

$$Sim_{combine}(e_1, e_2) = \frac{\sum_{i=1}^{n} weight_i \times Sim_i(e_1, e_2)}{\sum_{i=1}^{n} weight_i} \tag{2}$$

Similarity Propagation This step considers the impact of structural information on the similarity between each entity pair in the aggregated matrix. The intuition is that the more similar in structure two entities are, the more similar they are. To exploit the structure information, we use Descendant Similarity Inheritance [1].

Extraction In our system, only one-to-one matching is allowed. The final similarity matrix can be viewed as a bipartite graph with the first set of vertices are entities from source ontology and the second set of vertices are entities from target ontology. Thus, the alignment extraction can be modeled as the process of finding the mapping from the bipartite graph. To solve this, we apply the stable marriage problem algorithm [4]. We model the two set of entities as sets of men and women. For each man and each woman, in the correspondence set, a list of priority of men and women is created based on their similarity value. The stable marriage algorithm is then applied to find the stable mapping between two sets. The result is the pre-alignment.

Table 1. Performance of CSA on benchmark track

Test	Prec.	Rec.
101	1.0	1.0
201-202	0.83	0.73
221-247	0.98	1.0
248-252	0.79	0.61
253-259	0.81	0.55
260-266	0.70	0.49
H-mean	0.82	0.65

Pruning This is the final step of our system. In this step we filter out a proportion of entities pair that have low confidence to increase the precision of our system. For the threshold, we set it manually. The result is the final alignment of the two ontologies.

1.3 Adaptations made for the evaluation

We do not make any specific adaptation for the OAEI 2011 campaign. The three track are run in the same set of parameter.

1.4 Link to the system and parameters file

The CSA system can be downloaded from seal-project at `http://www.seals-project.eu/`.

1.5 Link to the set of provided alignments (in align format)

The result of CSA system can be downloaded from seal-project at `http://www.seals-project.eu/`.

2 Results

In this section, we present the results of the CSA system. We participate in the three tracks of benchmarks, anatomy, and conference. The result is in the following part.

2.1 Benchmarks

On the benchmarks of 2011, the reference ontology are different that the previous year. Since the descriptions, restrictions and instances are limited, it affects our algorithm very much. The result is shown at Table 1. We group the test into six groups based on their difficulty. The result shows the harmonic means precision and recall for each group.

Table 2. Performance of CSA on anatomy track

Precision	Recall	F-measure
0.465	0.757	0.576

Table 3. Performance of CSA on conference track

Prec.	F_1Meas.	Rec.
0.5	0.55	0.6
Prec.	F_2Meas.	Rec.
0.5	0.58	0.6
Prec.	$F_{0.5}$Meas.	Rec.
0.61	0.58	0.47

2.2 Anatomy

The anatomy dataset consists of two large ontologies of adult mouse anatomy with 2744 classes and a part of NCI Thesaurus for describing human anatomy with 3304 classes. The CSA result is shown in Table 2. Because of the high cost of computation, the execution time is quite high (4685s). In this track our system is high in recall (0.76) but the precision is quite low (0.47).

2.3 Conference

The results of conference track are shown in Table 3. It is difficult to archive the good results since ontologies from this track are real and developed by different organizations for different purposes.

3 General comments

This is the first time CSA has participated in the OAEI tracks, and our systems new to the seals platform. Further, the same set of parameter for all test in all tracks are difficult, because for each track the ontologies have a different characteristics to be processed. Thus, for any given dataset we need a different method for defining the threshold to extract the final alignment.

3.1 Comments on the results

Strengths CSA can be used to automatically combine different similarity measures. Our system does not need any external resources or training data for estimating the weight in the aggregation step.

Weaknesses The structure based similarity included in CSA is not strong enough to distinguish the different between matching and non-matching pairs. There are no structure similarity for properties. Further, we have not yet integrated any semantic verification or constraints in our system.

3.2 Discussions on the way to improve the proposed system

Our system is new and there are many opportunities to improve our method. First, we can integrate more basic similarity measures for aggregating. Second, for the pruning step, we can find the way for automatic defining a threshold rather than manually tuning. Finally, we can use some semantic verification as in [6] to pruning the low confidence matching pairs.

4 Conclusion

This is the first time the CSA has participated in OAEI campaign. In this year, we have participated in three tracks of benchmarks, anatomy and conference. We have introduced a new method for aggregating different similarity measures. The results show that our method is promising.

References

1. Isabel F. Cruz and William Sunna. Structural alignment methods with applications to geospatial ontologies. *Transactions in GIS*, 12(6):683–711, 2008.
2. Jérôme Euzenat and Pavel Shvaiko. *Ontology matching*. Springer-Verlag, Heidelberg (DE), 2007.
3. Christiane Fellbaum, editor. *WordNet: an electronic lexical database*. MIT Press, 1998.
4. David Gale and Lloyd S Shapley. College admissions and the stability of marriage. *American Mathematical Monthly*, 69(1):9–15, 1962.
5. Ryutaro Ichise. Machine learning approach for ontology mapping using multiple concept similarity measures. In *Proceedings of the Seventh IEEE/ACIS International Conference on Computer and Information Science*, pages 340–346, Washington, DC, USA, 2008. IEEE Computer Society.
6. Yves R. Jean-Mary, E. Patrick Shironoshita, and Mansur R. Kabuka. Ontology matching with semantic verification. *Web Semantics: Science, Services and Agents on the World Wide Web*, 7:235–251, September 2009.
7. Ming Mao, Yefei Peng, and Michael Spring. An adaptive ontology mapping approach with neural network based constraint satisfaction. *Web Semantics: Science, Services and Agents on the World Wide Web*, 8:14–25, March 2010.
8. Zhibiao Wu and Martha Palmer. Verbs semantics and lexical selection. In *Proceedings of the 32nd annual meeting on Association for Computational Linguistics*, ACL '94, pages 133–138, Stroudsburg, PA, USA, 1994. Association for Computational Linguistics.

LDOA Results for OAEI 2011

Marouen Kachroudi, Essia Ben Moussa, Sami Zghal, and Sadok Ben Yahia

University of Tunis El Manar
Computer Science Department, Faculty of Sciences of Tunis, Tunisia
Campus Universitaire, 1060 Tunis, Tunisia
{marouen.kachroudi,sadok.benyahia}@fst.rnu.tn
essia.ben_moussa@etu.upmc.fr
sami.zghal@planet.tn

Abstract. This paper presents and discusses the results produced by the Ldoa system for the 2011 Ontology Alignment Evaluation Initiative (OAEI). This method is based on the exploitation of an external resource through Linked Data. These data represent a wealth at the level of the Web. Indeed, it brings more semantics through relations that they maintain. The proposed alignment method Ldoa exploits terminological measures for concepts matching, topological measure for the exploration of structures as well as a semantic approach based on Linked Data.

1 Presentation of the system

Ontology alignment is a major process which contributes to the foundation of semantic Web, by facilitating the reconciliation of resources described by different ontologies. It can be defined as a production of a set of correspondences between the entities of two given ontologies. This process can be seen as a solution of the data heterogeneousness in the semantic Web, by allowing their interoperability. Indeed, a multitude of alignment methods appeared. These methods can be classified according to their approaches and strategies. Certain methods are based on lexical and linguistic treatments [1]. While other methods, qualified as hybrids, besides the lexical treatments, they rely the structural study of the ontologies to be aligned [2]. Nevertheless, this operation uses in certain cases external resources [3]. They serve to complete the classic techniques of matching, which exploit the structure or the wealth of the ontologies representative language. With the emergence of Linked Data [4], Web of Data is growing and realizes an important development. In classic Web, connections are anchors of relations linking HTML documents. On the other hand, Linked Data establish links between arbitrary objects. These connections exceed the borders of the HTML documents, by gathering and describing all the data of the Web according to the RDF (Resource Description Framework) formalism. Indeed, it is about another pillar of semantic Web, which aims at favoring data sharing and reuse. In this context, we introduce a new alignment method for OWL-DL ontologies using external resource. An intuitive way of connecting data on Web is the use of the *owl:sameAs* primitive, which is used to express links of identity.

1.1 State, purpose, general statement

The proposed method, LDOA (Linked Data for Ontology Alignment), presents an originality by the fact that it exploits besides the classic techniques (the terminological and structural measures of similarity) an external resource by using Linked Data. These data bring complementary information on the ontological entities to be aligned. This complementary information can increase in a considerable way the interpretation and consequently semantics. The method LDOA implements an alignment strategy which aims at exploiting all the wealth of the used ontologies. Indeed, it operates on three successive levels: terminological, topological and semantic.

1.2 Specific techniques used

The introduced LDOA method, as shown in figure 1, consists of two modules: a pretreatment module and an alignment module. The pretreatment module allows the transformation of the considered ontologies into two graphs. The alignment module exploits the obtained graphs with the aim of establishing correspondences between the various constituents of both ontologies to be aligned.

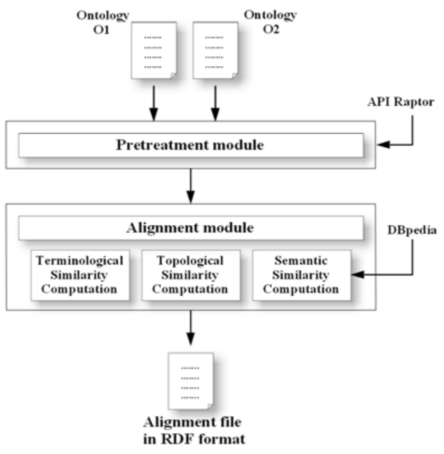

Ontology O1
Ontology O2
API Raptor
Pretreatment module
Alignment module
DBpedia
Terminological Similarity Computation
Topological Similarity Computation
Semantic Similarity Computation
Alignment file in RDF format

Fig. 1. Sketch of architecture for LDOA method

Pretreatment module In the stage of pretreatment, both considered ontologies in entry are transformed into a structure of graph. For the LDOA method, parsing is realized through the RAPTOR API[1]. Indeed, all the informative wealth

[1] http://librdf.org/raptor/

of every ontology is described by a corresponding graph, *i.e.*, classes, relations and instances. Nodes of each graph are classes and instances, whereas arcs represent links between the ontological entities. Each entity of an ontology is expressed with the RDF formalism : $< subject, predicate, Object >$ [5] and described thanks to OWL-DL constructors.

Alignment module The alignment module contains three complementary constituents. The terminological similarity computation Tsc allows the calculation of a compound terminological similarity between the descriptors of the ontological entities to be aligned. The topological similarity computation Tpsc exploits the internal structure of the ontologies by considering their hierarchies. The semantic similarity computation Ssc uses *Linked Data* to look for a certain complementarity between the entities.

- **Terminological Similarity Computation (Tsc)**
  Terminological constituent of the Ldoa method rests on the exploitation of three similarity measures based on strings treatment. These measures are applied to three descriptors of each entity to be aligned. Each ontological entity is described by three different descriptors : names, labels and comments. The used similarity measures are adapted to the various descriptors [6]. Levenshtein measure [2] [7] is used to calculate the similarity between the names of the ontological entities. Jaro-Winkler measure [3] [4] [8]computes similarity between labels. SoftJaccard measure [9] is dedicated for the computation of the similarity between comments.
- **Topological Similarity Computation (Tpsc)**
  The Topological Similarity Computation Tpsc recovers from all the techniques of alignment based on the study of the relational structures in the morphology of an ontology. It is about the relations that an ontological entity can maintain with its neighbors within. The hierarchy of the ontology [10]. Indeed, the Ldoa method exploits the taxonomic structure of ontological classes to estimate their degree of similarity. This technique emphasizes on the relational primitive OWL-DL *SubClassOf*, which endows an ontology of a hierarchical shape comparable to a graph.
  In Ldoa method, Wu-Palmer [11] similarity is used. It is a measure of similarity between the concepts of ontologies. Resnik [12] defines the similarity between two concepts by the quantity of information which they share. This shared information is equal to the informative contents of the smallest generalizing, *i.e.*, the most specific concept which subsumes both concepts in the ontology. Indeed, in a domain of concepts, the similarity is defined with regard to the distance which separates two concepts in the hierarchy and also by their position with regard to the root.

[2] $\text{Levenshtein}(s,t) = max(0, \frac{(min(|s|,|t|-\delta(s,t))}{min(|s|,|t|)})$
[3] $\text{J-W}(s,t) = \sigma_{\mathrm{J}}(s,t) + P \times \frac{(1-\sigma_{\mathrm{J}}(s,t))}{10}$
[4] $\mathrm{J}(s,t) = \frac{1}{3}(\frac{|c(s,t)|}{|s|} + \frac{|c(s,t)|}{|t|} + \frac{|c(s,t)|-|tr(s,t)|}{|c(s,t)|})$

– **Semantic Similarity Computation (Ssc)**
The (Ssc) uses DBpedia[5] as an external resource. This resource brings more semantics at the level of the terms to be aligned. Indeed, for each visited node of an ontology graph, a consultation of several data sets is launched. This consultation is performed for the various descriptors of the ontological entities to be aligned by exploiting OWL primitives, namely: *sameAs* and *seeAlso*. This task allows to collect for the three various nodes descriptors three sets of semantic equivalents (E_N for names, E_L for labels and E_C for comments). Whenever the descriptors belong to equivalent semantic sets, the value of the semantic similarity is equal 1. Otherwise, the value of this similarity is set to 0. The semantic similarity measure is computed as follow:

$$\textsc{Ssc}(E_1, E_2) = \begin{cases} 0 & if\ \{\mathcal{O}_2.name \cup \mathcal{O}_2.comment \\ & \cup\ \mathcal{O}_2.label\} \notin \{E_N \cup E_C \cup E_E\} \\ 1 & otherwise. \end{cases}$$

The process of alignment ends with the computation of the correspondences by aggregating the various stemming values of the three similarity constituents: terminological, topological and semantic. The aggregation is realized through a fair weighty combinaison in the various modules. The value of the correspondence, V_C, is computed as follows: $V_C(E_1, E_2) = \Pi_{\textsc{Tsc}} \times \textsc{Tsc}(E_1, E2) + \Pi_{\textsc{Tpsc}} \times \textsc{Tpsc}(E_1, E_2) + \Pi_{\textsc{Ssc}} \times \textsc{Ssc}(E_1, E_2)$, with the normalized sum of various weights which is equal to 1 ($\Pi_{\textsc{Tsc}} + \Pi_{\textsc{Tpsc}} + \Pi_{\textsc{Ssc}} = 1$). Indeed, the sum various levelheadednesses equal to 1 allows to obtain a value of correspondence which is equal to 1. This facilitates then the process of comparison of the obtained results with the other methods in the experimental study.

1.3 Adaptation made for the evaluation

The Ldoa method deals with the three test suites used in the Ontology Alignment Evaluation Initiative, *i.e*, Benchmark, Conference, and Anatomy. For this reason, our method was wrapped in a certain folder structure to be evaluated locally after being integrated in the SEALS platform. The package contains all the libs files required by the method and a zipped `.jar` file that acts as a bridge between the signature of the Ldoa method and the signature expected by the SEALS platform. All the package content is described in an XML file, namely `descriptor.xml`. The evaluation process can be launched through the command-line interface by indicating the name of the test track.

1.4 Links to the system, parameters file and the set of provided alignments

The release of the Ldoa method and the parameter file used for OAEI 2011 are located at `http://sourceforge.net/projects/the-ldoa-method/`. The alignments RDF files of the OAEI 2011 provided by the Ldoa method are located at `http://sourceforge.net/projects/ldoaresults2011/`.

[5] http://wiki.dbpedia.org/

2 Results

In this section, we describe the results of the LDOA method against the three test tracks (Benchmark, Anatomy, Conference) correspondingly to the SEALS platform evaluation modalities for OAEI 2011.

2.1 Benchmark

The metrics of Precision and Recall, recapitulated in Table 1, are grouped by family of tests. The values corresponding to the family 10x show that the LDOA method supplies good values. For the family 20x, values shows a degradation. Those low values are explained by the fact that ontological entities of this family of tests are marked by the absence of concepts names and comments. Also, in two test cases those names are either translated nor replaced by their synonyms. Indeed, the LDOA method, based on terminological measures, syntactical and semantic treatments, shows a degradation. For the two family tests 22x and 23x LDOA provides good values of recall but low values of precision. This is due to the important number of similar pairs of entities detected by the method that exceeds the number of pairs provided by the reference alignment. In addition, for test cases 24x, 25x and 26x we marked low values for both metrics of precision and recall. In those test cases, we note the absence of certain entities descriptors, *i.e.*, scrambled labels, no comments, no instance, no property as well as a flattened hierarchy. This decreases the efficiency of the terminological and topological measures. For the real test cases, *i.e.*, 30x, results obtained by the LDOA method supplies average values because our method can deals only with equivalence alignment relations, contrary to the alignment result which contains some inclusion ($<$) alignment relations.

Tests	**Precision**	**Recall**
10x	0.71	1.00
20x	0.44	0.50
22x	0.64	1.00
23x	0.57	1.00
24x	0.40	0.57
25x	0.34	0.46
26x	0.17	0.42
30x	0.47	0.69

Table 1. Precision and Recall metrics from OAEI 2011 for Benchmark dataset

2.2 Conference

This dataset consists of several, relatively expressive ontologies that describe the domain of organizing conferences from different perspectives. Table 2 recapitulates the Precision and the Recall. The goal of this track is to find all correct correspondences within a collection of ontologies describing the domain of organizing conferences (the domain being well understandable for every researcher). Additionally, "*interesting correspondences*" are also welcome. Results were evaluated automatically against reference alignments and by data-mining and logical reasoning techniques.

Test	**Precision**	**Recall**
cmt-confOf	0.07	0.43
cmt-conference	0.04	0.31
cmt-edas	0.08	0.69
cmt-ekaw	0.04	0.45
cmt-iasted	0.03	1.00
cmt-sigkdd	0.11	0.83
confOf-edas	0.11	0.57
confOf-ekaw	0.12	0.50
confOf-iasted	0.04	0.44
confOf-sigkdd	0.05	0.57
conference-confOf	0.06	0.46
conference-edas	0.06	0.52
conference-ekaw	0.14	0.68
conference-iasted	0.03	0.35
conference-sigkdd	0.08	0.53
edas-ekaw	0.08	0.52
edas-iasted	0.05	0.52
edas-sigkdd	0.08	0.60
ekaw-iasted	0.04	0.60
ekaw-sigkdd	0.07	0.63
iasted-sigkdd	0.10	0.86

Table 2. Precision and Recall metrics from OAEI 2011 for Conference dataset

2.3 Anatomy

The anatomy real world case is to match the Adult Mouse Anatomy and the NCI Thesaurus describing the human anatomy. Mouse has 2,744 classes, while Human has 3,304 classes. Matching these ontologies is also challenging in terms of efficiency because these ontologies are relatively large. Our method shows problems when handling those two ontologies and can't supply correspondences.

3 General comments

In the following some general statements about the OAEI procedure, modalities, and results obtained are given.

3.1 Comments on the results

For this year, the reference alignments of the three SEALS tracks are only concerned with classes and properties. There is no coverage of instances, also called individuals. This can explain the low values we obtained, especially for the Precision metric. We are looking for the possibility of adding individuals for the reference alignments.

3.2 Discussions on the way to improve the proposed system

Besides, the determination of the adequate weights for the various constituents is our current priority. Thus, we work on the automatic detection of hierarchical trends in the considered ontologies, *e.g.*, a standard treatment of the flattened hierarchies strongly degraded the value of the measure of topological similarity. Besides, the idea to concretize the semantic aspect in the alignment process will bring us to conceive a purely semantic approach. This approach will have to asset the coverage of semantics of all the ontological entities as well as for relations which they maintain. In the scalability register, the LDOA method would be able to handle ontologies of real world having bigger sizes. So, using the WORDNET API [13] or dictionaries can be considered as a necessary step, to improve the values of the terminological similarity measures, in particular in the multilingual alignment task, (*e.g.*, tests of the family $2xx$).

3.3 Comments on the OAEI 2011 procedure

The SEALS evaluation campaign is very beneficial since it allows all alignment systems to use a standardized interface which could possibly be used by everyone. The evaluation procedure was full automatized through the use of the `Matcherbridge` class.

4 Conclusions

The LDOA method was briefly described. The results obtained for the OAEI 2011 tracks, cooresponding to the SEALS platform evaluation modalities. Several observations regarding these results were highlighted, in particular the impact of the elimination of any ontological resource on the similarity values. Also the effect of having a single configuration throughout all OAEI tracks were discussed. Future development for LDOA method will be targeted towards more interactivity and intelligence in dealing with weights assigned for every similarity value when some ontological resource are lost, i.e., terminologies, structures or semantics.

Acknowledgement

The authors would like to thank Aicha Ben Jrad for the valuable contributions and feedback about DBpedia access and exploitation.

References

1. Kortis, K., Vouros, G., Stergiou, K.: Towards automatic merging of domain ontologies: Approach the hcone-merge a. Journal of Web Semantics **4** (2006) 60–79
2. Xu, P., Tao, H., Zang, T., Wang, Y.: Alignment results of sobom for oaei 2009. In: Proceedings of the 4^{th} International Workshop on Ontology Matching (OM-2009), Washington, USA (2009) 216–223
3. Safar, B., Reynaud, C.: Alignement d'ontologies basé sur des ressources complémentaires illustration sur le système taxomap. Technique et Science Informatiques **28** (2009) 1211–1232
4. Parundekar, R., Knoblock, C., Ambite, J.: Linking and building ontologies of linked data. In: Proceedings of 9^{th} International Semantic Web Conference (ISWC 2010), Shanghai, China (2010) 598–614
5. Klyne, G., Carroll, J.J.: Resource Description Framework (RDF): Concepts and Abstract Syntax. Technical report, W3C: World Wide Web Consortium, http://www.w3.org/TR/2004/REC-rdf-concepts-20040210/ (05/24/2010) (2004)
6. Zghal, S., Kachroudi, M., Ben Yahia, S., Mephu Nguifo, E.: OACAS: Ontologies alignment using composition and aggregation of similarities. In: Proceedings of the 1^{st} International Conference on Knowledge Engineering and Ontology Development (KEOD 2009), Madeira, Portugal (2009) 233–238
7. Levenshtein, I.V.: Binary codes capables of corrections, deletions, insertions and reversals. Soviet Physics-Doklady **10** (1966) 707–710
8. Winkler, W.: The state of record linkage and current research problems. Technical report, Statistical Research Division, U.S. Bureau of the Census (1999)
9. Largeron, C., Kaddour, B., Fernandez, M.: SoftJaccard : une mesure de similarité entre ensembles de chaînes de caratères pour l'unification d'entités nommées. In: Actes des $9^{ème}$ Journées Francophones Extraction et Gestion des Connaissances (EGC'2009), Strasbourg, France (2009) 443–444
10. Ehrig, M.: Ontology alignment: bridging the semantic gap. Springer-Verlag, New-York (2007)
11. Wu, Z., Palmer, M.: Verb semantics and lexical selection. In: Proceeding of 32^{nd} Annual Meeting of the Association for Computational Linguistics (ACL 1994). (1994) 133–138
12. Resnik, P.: Using information content to evaluate semantic similarity in a taxonomy. In: Proceedings of the 14^{th} International Joint Conference on Artificial Intelligence (IJCAI 1995). (1995) 448–453
13. Miller, G.A.: WordNet : a Lexical Database for English. Communications of the ACM **38** (1995) 39–41

Lily Results on SEALS Platform for OAEI 2011

Peng Wang

School of Computer Science and Engineering, Southeast University, China
pwang@seu.edu.cn

Abstract. This paper presents the alignment results of Lily on SEALS platform for the ontology alignment contest OAEI 2011. Lily is an ontology matching system. In OAEI 2011, Lily submited the results for three matching tasks on the SEALS platform: benchmark, anatomy, conference. The specific techniques used by Lily are introduced. The matching results of Lily are also discussed.

Keywords: Ontology Matching, SEALS Platform, OAEI

1 Presentation of the system

Lily is an ontology matching system for solving the key issues related to heterogeneous ontologies, and it uses hybrid matching strategies to execute the ontology matching task. Lily can be used to discovery alignments for both normal size ontologies and large scale ontologies. In past ontology alignment contests, Lily showed it is a good ontology matching system and obtained good results in some tasks [1-3].

1.1 State, purpose, general statement

The core principle of the matching strategy in Lily is utilizing the useful information effectively and correctly. Lily combines several novel and efficient matching techniques to find alignments. Lily has three main matching functions: (1) Generic Ontology Matching (GOM) is used for common matching tasks with normal size ontologies. (2) Large scale Ontology Matching (LOM) is used for the matching tasks with large size ontologies. (3) Ontology mapping debugging is used to verify and improve the alignment results.

The matching process mainly contains three steps: (1) In preprocess, Lily parses ontologies and prepares the necessary information for subsequent steps. (2) In similarity computation step, Lily uses special methods to calculate the similarities between elements from different ontologies. (3) In post-process, alignments are extracted and refined by mapping debugging.

In OAEI2011, we redesign or modify some key algorithms, and we also make it can be run on the SEALS platform.

1.2 Specific techniques used

Lily aims to provide high quality 1:1 concept pair or property pair alignments. The main specific techniques used by Lily are as follows.

Semantic subgraph: An element may have heterogeneous semantic interpretations in different ontologies. Therefore, understanding the real local meanings of elements is very useful for similarity computation, which are the foundations for many applications including ontology matching. Therefore, before similarity computation, Lily first describes the meaning for each entity accurately. However, since different ontologies have different preferences to describe their elements, obtaining the semantic context of an element is an open problem. We proposed the semantic subgraph to capture the real meanings of ontology elements [4]. To extract the semantic subgraphs, a hybrid ontology graph is used to represent the semantic relations between elements. An extracting algorithm based on an electrical circuit model is then used with new conductivity calculation rules to improve the quality of the semantic subgraphs. We have showed that the semantic subgraphs can properly capture the local meanings of elements [4].

Based on the extracted semantic subgraphs, we can build more credible matching clues. Therefore it can reduce the negative affection of the matching uncertain.

Generic ontology matching method: The similarity computation is based on the semantic subgraphs, i.e. all the information used in the similarity computation is come from the semantic subgraphs. Lily combines the text matching and structure matching techniques.

Semantic Description Document (SDD) matcher measures the literal similarity between ontologies. A semantic description document of a concept contains the information about class hierarchies, related properties and instances. A semantic description document of a property contains the information about hierarchies, domains, ranges, restrictions and related instances. For the descriptions from different entities, we calculate the similarities of the corresponding parts. Finally, all separate similarities are combined with the experiential weights.

Matching weak informative ontologies: Most existing ontology matching methods are based on the linguistic information. However, some ontologies have not sufficient or regular linguistic information such as natural words and comments, so the linguistic-based methods cannot work. Structure-based methods are more practical for this situation. Similarity propagation is a feasible idea to realize the structure-based matching. But traditional propagation does not take into consideration the ontology features and will be faced with effectiveness and performance problems. We analyze the classical similarity propagation algorithm Similarity Flood and propose a new structure-based ontology matching method [5]. This method has two features: (1) It has more strict but reasonable propagation conditions which make matching process become more efficient and alignments become better. (2) A series of propagation strategies are used to improve the matching quality. We have demonstrated that this method performs well on the OAEI benchmark dataset [5].

However, the similarity propagation is not always perfect. When more alignments are discovered, more incorrect alignments would also be introduced by the similarity propagation. So Lily also uses a strategy to determine when to use the similarity propagation.

***Large scale ontology matching*:** Matching large ontologies is a challenge due to the high time complexity. We propose a new matching method for large ontologies based on reduction anchors [6]. This method has a distinct advantage over the divide-and-conquer methods because it does not need to partition large ontologies. In particular, two kinds of reduction anchors, positive and negative reduction anchors, are proposed to reduce the time complexity in matching. Positive reduction anchors use the concept hierarchy to predict the ignorable similarity calculations. Negative reduction anchors use the locality of matching to predict the ignorable similarity calculations. Our experimental results on the real world data sets show that the proposed method is efficient for matching large ontologies [6].

Ontology mapping debugging Lily uses a technique called ontology mapping debugging to improve the alignment results [7]. Different from existing methods, which focus on finding efficient and effective solutions for the ontology mapping problem, mapping debugging emphasis on analyzing the mapping result to detect/diagnose the mapping defects. We proposed a technique called debugging ontology mappings [7]. During debugging, some types of mapping errors, such as redundant and inconsistent mappings, can be detected. Some warnings, including imprecise mappings or abnormal mappings, are also locked by analyzing the features of mapping result. More importantly, some errors and warnings can be repaired automatically or can be presented to users with revising suggestions.

1.3 Adaptations made for the evaluation

Lily is fully automatic in OAEI2011, and there is no any parameter turning during the matching. Lily has a simple strategy to choose the right matching method.

1.4 Link to the system and the set of provided alignments

Lily system for OAEI 2011 is available at http://cse.seu.edu.cn/people/pwang/software/Lily/lily-package.zip

2 Results

2.1 benchmark

The Benchmark2010 dataset can be divided into five groups: 101-104, 201-210, 221-247, 248-266 and 301-304.

101-104 Lily plays well for these test cases.

201-210 Lily can produce good results for this test set. Even without right labels and comments information, Lily can find most correct alignments through making use of other information such as instances. Using few alignment results obtained by the basic methods as inputs, the similarity propagation strategy will generate more alignments.

221-247 Lily can find most correct alignments using the labels and comments information.

248-266 This group is the most difficult test set. Lily first uses the GOM method to find alignments, and then use matching weak informative method to discover more alignments.

301-304 This test set are the real ontologies. Lily only finds the equivalent alignment relations.

The following table shows the average performance of each group and the overall performance on the Benchmark2010 dataset.

Table 1. The results on the Benchmark2010

	101-104	201-210	221-247	248-266	301-304	Average
Precision	1.00	0.92	0.98	0.81	0.71	0.86
Recall	1.00	0.82	0.99	0.51	0.69	0.66
F1-Measure	1.00	0.85	0.97	0.59	0.70	0.71

The BenchmarkII2011 dataset can be divided into three groups: 101-102, 221-247 and 248-266. It seems that this task is more difficult than Benchmark2010. Our alignment results have small decrease in average.

The following table shows the average performance of each group and the overall performance on the BenchmarkII2011 dataset.

Table 2. The results on the BenchmarkII2011

	101-202	221-247	248-266	Average
Precision	0.92	0.96	0.74	0.79
Recall	0.65	0.98	0.55	0.63
F1-Measure	0.70	0.97	0.61	0.67

2.2 anatomy

The anatomy track consists of two real large-scale biological ontologies. Lily can handle such ontologies smoothly with LOM method. Task#1 means that the matching system has to be applied with standard settings to obtain a result that is as good as possible. Table 3 shows the performance of the task #1 on anatomy dataset.

Compared to the result in OAEI2009, our result has little increase. However, it has obvious gap to the results of other matching system in OAEI2010.

Table 3. The performance on the anatomy

	Runtime	Precision	Recall	F1-Measure
Task#1	20min	0.80	0.72	0.76

2.4 conference

This task contains 16 real world ontologies about conference. We only get part of alignments from SEALS platform, which is showed in Table 4. The heterogeneous character in this track is various. It is a challenge to generate good results for all ontology pairs in this test set.

Table 4. The performance on the conference based on reference mappings

Ontology Pair	Precision	Recall	F1-Measure
cmt-confOf	0.46	0.38	0.41
cmt-conference	0.21	0.25	0.23
cmt-edas	0.29	0.54	0.38
cmt-ekaw	0.25	0.45	0.32
cmt-iasted	0.18	0.50	0.27
cmt-sigkdd	0.27	0.25	0.26
confOf-edas	0.57	0.42	0.48
confOf-ekaw	0.72	0.65	0.68
confOf-iasted	0.38	0.67	0.48
confOf-sigkdd	0.09	0.14	0.11
conference-confOf	0.55	0.73	0.63
conference-edas	0.16	0.29	0.20
conference-ekaw	0.38	0.32	0.35
conference-iasted	0.39	0.50	0.44
conference-sigkdd	0.32	0.47	0.38
edas-ekaw	0.50	0.52	0.51
edas-iasted	0.31	0.47	0.37
edas-sigkdd	0.47	0.53	0.50
ekaw-iasted	0.30	0.70	0.42
ekaw-sigkdd	0.33	0.45	0.38
iasted-sigkdd	0.43	0.67	0.53

3. General comments

We redesign some key algorithms of Lily this year, but it does not produce better alignment results than previous versions. For example, we try to use mapping debugging technique to improve the precision of results. However, for the reason that the generated results are 1:1 equivalent mappings, the mapping debugging can only find few wrong alignments.

The SEALS platform is very important for ontology matching research. It provides a way to examine new matching method and compare to other matching systems.

4 Conclusion

We briefly introduce our ontology matching tool Lily. The matching process and the special techniques used by Lily are presented. The alignment results are carefully analyzed.

References

1. Peng Wang, Baowen Xu: Lily: ontology alignment results for OAEI 2009. In The 4th International Workshop on Ontology Matching, Washington Dc., USA (2009)
2. Peng Wang, Baowen Xu: Lily: Ontology Alignment Results for OAEI 2008. In The Third International Workshop on Ontology Matching, Karlsruhe, Germany (2008)

3. Peng Wang, Baowen Xu: LILY: the results for the ontology alignment contest OAEI 2007. In The Second International Workshop on Ontology Matching (OM2007), Busan, Korea (2007)
4. Peng Wang, Baowen Xu, Yuming Zhou: Extracting Semantic Subgraphs to Capture the Real Meanings of Ontology Elements. Journal of Tsinghua Science and Technology, vol. 15(6), pp. 724-733 (2010)
5. Peng Wang, Baowen Xu: An Effective Similarity Propagation Model for Matching Ontologies without Sufficient or Regular Linguistic Information, In The 4th Asian Semantic Web Conference (ASWC2009), Shanghai, China (2009)
6. Peng Wang, Yuming Zhou, Baowen Xu: Matching Large Ontologies Based on Reduction Anchors. In The Twenty-Second International Joint Conference on Artificial Intelligence (IJCAI 2011), Barcelona, Catalonia, Spain (2011)
7. Peng Wang, Baowen Xu: Debugging Ontology Mapping: A Static Approach. Computing and Informatics, vol. 27(1), pp. 21–36 (2008)

Appendix: Raw results

The final results of benchmark task are as follows.

Matrix of results

#	Comment	Prec.	Rec.	#	Comment	Prec.	Rec.
101	Reference alignment	1.00	1.00	251		1.00	0.08
103	Language generalization	1.00	1.00	251-2		0.93	0.83
104	Language restriction	1.00	1.00	251-4		0.91	0.74
201	No names	0.96	0.96	251-6		0.94	0.67
201-2		1.00	1.00	251-8		0.88	0.49
201-4		1.00	1.00	252		0.29	0.02
201-6		0.98	0.98	252-2		0.92	0.82
201-8		1.00	1.00	252-4		0.89	0.79
202	No names, no comment	0.43	0.03	252-6		0.90	0.80
202-2		0.95	0.86	252-8		0.92	0.82
202-4		0.94	0.76	253		0.33	0.02
202-6		0.94	0.67	253-2		0.94	0.80
202-8		0.92	0.51	253-4		0.92	0.71
203	Misspelling	0.98	0.98	253-6		0.95	0.64
204	Naming conventions	1.00	1.00	253-8		0.91	0.44
205	Synonyms	1.00	0.99	254		0.00	0.00
206	Translation	1.00	0.99	254-2		0.84	0.64
207		1.00	0.99	254-4		0.95	0.55
208		0.96	0.93	254-6		0.92	0.36
209		0.71	0.53	254-8		0.86	0.18
210		0.71	0.53	257		1.00	0.03
221	No specialisation	1.00	1.00	257-2		0.94	0.88
222	Flattened hierarchy	1.00	1.00	257-4		0.93	0.76
223	Expanded hierarchy	0.98	0.98	257-6		0.75	0.55

224	No instances	1.00	1.00	257-8		0.92	0.36
225	No restrictions	1.00	1.00	258		0.57	0.04
228	No properties	1.00	1.00	258-2		0.93	0.83
230	Flattened classes	0.94	1.00	258-4		0.92	0.75
231	Expanded classes	1.00	1.00	258-6		0.97	0.69
232		1.00	1.00	258-8		0.94	0.52
233		1.00	1.00	259		0.29	0.02
236		1.00	1.00	259-2		0.91	0.81
237		1.00	1.00	259-4		0.91	0.81
238		0.97	0.97	259-6		0.89	0.79
239		0.97	1.00	259-8		0.89	0.79
240		0.94	0.97	260		0.50	0.03
241		1.00	1.00	260-2		0.85	0.79
246		0.97	1.00	260-4		0.87	0.69
247		0.91	0.94	260-6		0.88	0.52
248		0.43	0.03	260-8		0.85	0.38
248-2		0.93	0.79	261		0.50	0.03
248-4		0.95	0.73	261-2		0.83	0.76
248-6		0.95	0.64	261-4		0.83	0.76
248-8		0.90	0.44	261-6		0.83	0.76
249		0.71	0.05	261-8		0.84	0.79
249-2		0.95	0.86	262		0.00	0.00
249-4		0.91	0.74	262-2		0.88	0.67
249-6		0.96	0.68	262-4		0.89	0.52
249-8		0.91	0.49	262-6		0.92	0.36
250		1.00	0.03	262-8		0.86	0.18
250-2		0.94	0.88	265		0.50	0.03
250-4		0.93	0.76	266		0.50	0.03
250-6		0.83	0.61	301	BibTeX/MIT	0.82	0.76
250-8		0.92	0.36	302	BibTeX/UMBC	0.54	0.42
				303	Karlsruhe	0.57	0.65
				304	INRIA	0.89	0.93

LogMap results for OAEI 2011

Ernesto Jiménez-Ruiz, Antón Morant, and Bernardo Cuenca Grau

Department of Computer Science, University of Oxford
{ernesto,anton.morant,berg}@cs.ox.ac.uk

Abstract. We present the results obtained by the ontology matching system LogMap within the OAEI 2011 campaign. This is the first participation of LogMap in the campaign, and the results have so far been quite promising.

1 Presentation of the System

The LogMap[1] project started in January 2011 with the objective of developing a scalable and logic-based ontology matching system.

Such system should be able to deal efficiently with large-scale ontologies; furthermore, it should exploit logic-based reasoning and diagnosis techniques to compute output mappings that do not lead to logical inconsistencies when integrated with the input ontologies [7]. Although the development of LogMap is relatively recent, the authors' experience in the field of ontology integration dates back to 2008 [9, 11].

1.1 Motivation and Problem Statement

Despite the impressive state of the art, large-scale biomedical ontologies still pose serious challenges to existing ontology matching tools [15, 6].

Insufficient scalability. Although existing matching tools can efficiently deal with moderately sized ontologies (e.g. those in the OAEI Anatomy track), large-scale ontologies such as FMA, SNOMED CT and NCI are still beyond their reach.

Logical inconsistencies. OWL ontologies have well-defined semantics based on first-order logic, and mappings are commonly represented as OWL class axioms. Many existing tools, however, disregard the semantics of the input ontologies; thus, they are unable to detect and repair inconsistencies that logically follow from the union of the input ontologies and the computed mappings. Although there is a growing interesting in applying reasoning techniques to ontology matching, reasoning is known to severely aggravate the scalability problem.

1.2 Technical Approach

LogMap is a highly scalable ontology matching system with 'built-in' reasoning and diagnosis capabilities, which aims at addressing the aforementioned challenges.

We next present a brief overview of LogMap and refer the reader to [7] for a comprehensive description. The main steps performed by LogMap are shown in Figure 1.

[1] http://www.cs.ox.ac.uk/isg/projects/LogMap/

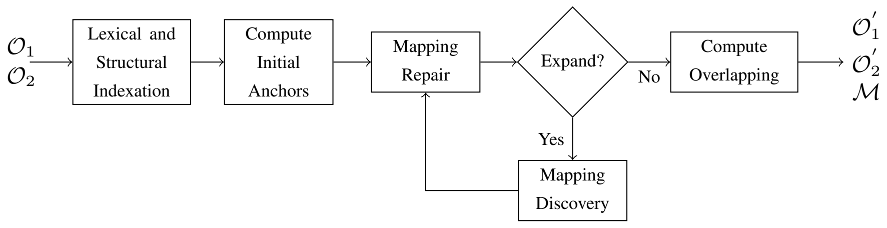


Fig. 1. LogMap in a nutshell.

Inverted index for NCI Anatomy labels		**Index for NCI Anatomy class URIs**	
Entry	***Cls ids***	***Cls id***	***URI***
external,ear	1	1	NCI_C12292 (external_ear)
atrial,auricle	1,392	392	NCI_C32165 (auricle)
auricle	1,392; 529	529	NCI_C12394 (ear)
ear	529		
Inverted index for Mouse Anatomy labels		**Index for Mouse Anatomy class URIs**	
Entry	***Cls ids***	***Cls id***	***URI***
auricle	214	214	MA_0000259 (auricle)
atrial,auricle	214	216	MA_0000258 (outer_ear)
ear,external	216		
outer,ear	216		

Table 1. Fragment of the lexical indices for NCI and Mouse anatomy ontologies

Lexical indexation. The first step after parsing the input ontologies is their lexical indexation. LogMap indexes the labels of the classes in each ontology as well as their lexical variations, and allows for the possibility of enriching these indices by using external sources (e.g., WordNet or UMLS-lexicon). LogMap constructs an 'inverted' lexical index (see Table 1) for each input ontology. In general, an entry in the index can be mapped to several classes (e.g., see 'auricle' in Table 1). This type of index, which is commonly used in information retrieval applications, will be exploited by LogMap to efficiently compute an initial set of candidate mappings, called *anchors*.

Structural indexation. LogMap exploits the information in the (extended) class hierarchy of the input ontologies in different steps of the matching process. Efficient access to the information in these hierarchies is critical to LogMap's scalability.

LogMap classifies the input ontologies using either incomplete structural heuristics, or an off-the-shelf complete DL reasoner. Then, the classified hierarchies are indexed using an interval labelling schema—an optimised data structure for storing DAGs and trees [1], which has been shown to significantly reduce the cost of computing typical queries over large class hierarchies [3, 14].

The class hierarchies computed by LogMap are *extended* since, apart from the typical classification output of DL reasoners, they also include those explicit axioms in

Entry	NCI ids	Mouse ids	Mappings
external,ear	1	216	NCI_C12292 $\equiv$ MA_0000258
atrial,auricle	1,392	214	NCI_C12292 $\equiv$ MA_0000259 NCI_C32165 $\equiv$ MA_0000259
auricle	1,392; 529	214	NCI_C12292 $\equiv$ MA_0000259 NCI_C32165 $\equiv$ MA_0000259 NCI_C12394 $\equiv$ MA_0000259

Table 2. Fragment of the intersection between the inverted indices for NCI and Mouse ontologies

the input ontologies that can be directly encoded in Horn propositional logic (e.g., *class disjointness* axioms, subsumption axioms between an intersection of named classes and a named class).

Computation of 'anchor mappings'. LogMap computes an initial set of *anchor mappings* by intersecting the inverted indices of the input ontologies (i.e., by checking whether two lexical entries in those indices contain exactly the same strings). Anchor computation can hence be implemented very efficiently. Table 2 shows the intersection of the inverted indices of Table 1, which yields four anchors.

Given an anchor $m = (C_1 \equiv C_2)$, LogMap uses the string matching tool ISUB [16] to match the neighbours of C_1 in the hierarchy of $\mathcal{O}_1$ to the neighbours of C_2 in the hierarchy of $\mathcal{O}_2$. LogMap then assigns a confidence value to m by computing the proportion of matching neighbours weighted by the ISUB similarity values. This technique is based on a *principle of locality*: if classes C_1 and C_2 are correctly mapped, then the classes semantically related to C_1 in $\mathcal{O}_1$ are likely to be mapped to those semantically related to C_2 in $\mathcal{O}_2$. Thus, if the hierarchy neighbours of the classes in an anchor mapping match with low confidence, then the anchor may be incorrect.

Mapping repair and discovery. The core of LogMap is an iterative process that alternates mapping repair and mapping discovery steps (see Figure 1).

Unsatisfiability checking and repair. LogMap uses a Horn propositional logic representation of the extended hierarchy of each ontology together with all existing mappings. Although such propositional Horn encoding is possibly incomplete, it is key to LogMap's scalability. Provably complete DL reasoners do not scale well when integrating large ontologies via mappings; the scalability problem is exacerbated by the number of unsatisfiable classes (more than 10,000 found by LogMap when integrating SNOMED and NCI using only anchors) and the large number of additional reasoner calls required for repairing each unsatisfiability.

For unsatisfiability checking, LogMap implements the highly scalable Dowling-Gallier algorithm [5] for propositional Horn satisfiability, and calls the Dowling-Gallier module once (in each repair step) for each class. Our implementation takes as input a class C (represented as a propositional variable) and determines the satisfiability of the propositional theory $\mathcal{P}_C$ consisting of

- the rule $(\mathsf{true} \rightarrow C)$;

– the propositional representations $\mathcal{P}_1$ and $\mathcal{P}_2$ of the extended hierarchies of the input ontologies $\mathcal{O}_1$ and $\mathcal{O}_2$; and
– the propositional representation $\mathcal{P}_M$ of the mappings computed thus far.

LogMap computes a *repair* for each unsatisfiable class in the input ontologies. Given an unsatisfiable class C and the propositional theory $\mathcal{P}_C$, a *repair* $\mathcal{R}$ of $\mathcal{P}_C$ is a minimal subset of the mappings in $\mathcal{P}_M$ such that $\mathcal{P}_C \setminus \mathcal{R}$ is satisfiable.

LogMap extends Dowling-Gallier's algorithm to record all *active mappings* ($\mathcal{P}_{act}$) that may be involved in each unsatisfiability. To improve scalability, repair computation is based on a 'greedy' algorithm. Given each unsatisfiable class C and the relevant active mappings $\mathcal{P}_{act}$ computed using Dowling-Gallier, the algorithm identifies subsets of $\mathcal{P}_{act}$ of increasing size until a repair is found. Thus, our algorithm is guaranteed to compute all repairs of smallest size. If more than one repair is found, LogMap selects the one with involving mappings with the lowest confidence values.

Mapping discovery. In order to *discover new mappings*, LogMap maintains two *contexts* (sets of 'semantically related' classes) for each anchor. Contexts for the same anchor are expanded in parallel using the class hierarchies of the input ontologies. New mappings can then be found by matching classes in the relevant contexts using ISUB. Matches with a similarity value exceeding a given *confidence* threshold are considered as candidate mappings.

LogMap continues the iteration of repair and discovery steps until no context is expanded. The output of this process is a set of mappings that is likely to be 'clean', in the sense that it will not lead to unsatisfiable classes when merged with the input ontologies.

Ontology overlapping estimation. In addition to the mappings, LogMap also returns two (hopefully small) fragments $\mathcal{O}'_1$ and $\mathcal{O}'_2$ of $\mathcal{O}_1$ and $\mathcal{O}_2$, respectively. Intuitively, $\mathcal{O}'_1$ and $\mathcal{O}'_2$ represent the 'overlapping' between $\mathcal{O}_1$ and $\mathcal{O}_2$, in the sense that each 'correct' mapping not found by LogMap is likely to involve only classes in these fragments. The computation of $\mathcal{O}'_1$ and $\mathcal{O}'_2$ is performed in two steps.

1. *Computation of 'weak' anchors*. LogMap computed the initial anchor mappings by checking whether two entries in the inverted index of $\mathcal{O}_1$ and $\mathcal{O}_2$ contained *exactly the same* set of strings. For the purpose of overlapping estimation, LogMap also computes new anchor mappings that are 'weak' in the sense that relevant index entries are only required to contain *some* common string. Thus, weak anchors represent correspondences between classes with a common lexical component.
2. *Module extraction*. The sets S_i of classes in $\mathcal{O}_i$ involved in either a weak anchor or a mapping computed by LogMap are then used as 'seed' signatures for module extraction. In particular, $\mathcal{O}'_1$ (resp. $\mathcal{O}'_2$) are computed by extracting a locality-based module [4] for S_1 in $\mathcal{O}_1$ (resp. for S_2 in $\mathcal{O}_2$).

1.3 Adaptations made for the evaluation

To participate in the OAEI 2011, LogMap has been extended with a property matching facility as well as with the ability to consider 'weak' anchors as candidate mappings.

Extended inverted index for NCI Anatomy		**Index for NCI Anatomy class URIs**	
Lexical entry	***Cls ids***	***Cls id***	***Cls name***
gallbladder,smooth,tissue,muscle	2061	2061	NCI_C49483 (gallbladder_smooth_muscle_tissue)
smooth,muscle	2061; 3214,...	3214	NCI_C49306 (trachea_smooth_muscle_tissue)
Extended inverted index for Mouse Anatomy		**Index for Mouse Anatomy class URIs**	
Lexical entry	***Cls ids***	***Cls id***	***Cls name***
gall,bladder,smooth,muscle	600	600	MA_0001635 (gall_bladder_smooth_muscle)
smooth,muscle	600; 2629; ...	2629	MA_0001741 (prostate_gland_smooth_muscle)

Entry	**NCI ids**	**Mouse ids**	**Mappings**
smooth,muscle	2061; 3214; ...	600; 2629; ...	NCI_C49483 $\equiv$ MA_0001635 NCI_C49483 $\equiv$ MA_0001741 NCI_C49306 $\equiv$ MA_0001635 NCI_C49306 $\equiv$ MA_0001741 ...

Table 3. Extend inverted indices an their intersection for NCI and Mouse Anatomy ontologies

Computation of property anchors. Similarly to the case of anchor mappings between classes, the computation of anchor mappings between (object or data) properties also relies on the intersection of inverted lexical indexes. These mappings, however, are currently not taken into account by LogMap's repair module.

In the current version of LogMap, a mapping between properties p_1 and p_2 is returned as output only if both their respective domains D_1, D_2 and ranges R_1, R_2 are 'compatible'— that is, if LogMap's repair module does not find inconsistencies when extending the final output class mappings with the mappings $D_1 \equiv D_2$ and $R_1 \equiv R_2$.

For example, in the OAEI conference track, LogMap identified an equivalence mapping between the properties cmt:writtenBy and confOf:writtenBy. This mapping, however, was discarded since the extension of LogMap's output class mappings with the mappings cmt:Reviewer $\equiv$ confOf:Author and cmt:Review $\equiv$ confOf:Contribution between the respective domains and ranges of these properties led to an inconsistency.

Inclusion of 'weak' anchors. Weak anchor mappings are well-suited for overlapping estimation purposes (see Section 1.2); however, it is dangerous to treat them as candidate output mappings since they are likely to introduce unmanageable levels of 'noise' during mapping repair.

The upper part of Table 3 shows an excerpt of the inverted indices for NCI and Mouse Anatomy ontologies extended with partial lexical entries. The intersection of these inverted indices includes the entry *'smooth,muscle'*, which appears in 19 concepts in Mouse Anatomy and in 9 concepts in NCI Anatomy; as a result, 171 weak anchor mappings can be obtained (see lower part of Table 3). Most of these mapping are obviously incorrect (e.g. NCI_C49483 $\equiv$ MA_0001741 or NCI_C49306 $\equiv$ MA_0001741), however valid mappings can still be discovered (e.g. NCI_C49483 $\equiv$ MA_0001635).

The current version of LogMap considers a weak anchor as a candidate output mapping (hence taking it into account for mapping repair) only if exceeds a given ISUB confidence threshold.

System	Precision	Recall	F-score	Time (s)
AgrMaker	0.943	0.892	0.917	634
LogMap	***0.948***	***0.846***	***0.894***	***24***
CODI	0.965	0.825	0.889	1890
Lily	0.814	0.734	0.772	563
AROMA	0.742	0.625	0.679	39
CSA	0.465	0.757	0.576	4685

Table 4. Comparing LogMap with the top 6 tools in the Anatomy Track of the OAEI 2011

2 Results

In this section, we present the official results obtained by LogMap in the OAEI 2011.

2.1 Anatomy 2011 Track

This track involves two biomedical ontologies: the Adult Mouse Anatomy ontology (2,744 classes) and a fragment of the NCI ontology describing human anatomy (3,304 classes). The reference alignment [2] has been manually curated, and it contains a significant number of non-trivial mappings.

Table 4 compares LogMap's results with the tools providing meaningful results in the Anatomy 2011 track. LogMap obtained the second best results; furthermore, LogMap was faster than the other tools: while LogMap matched these ontologies in 24 seconds, the best tool (AgrMaker) required more than 10 minutes. We also verified, using an off-the-shelf DL reasoner, that the integration of these ontologies with LogMap's output mappings did not lead to unsatisfiable classes. According to official results only LogMap and CODI provided mappings without incoherence [13].

2.2 Conference 2011 Track

The Conference 2011 Track contains 16 ontologies describing the domain of conference organisation.

Table 5 compares LogMap's results with the official results obtained by the top 6 tools. LogMap obtained the third best results in terms of F-score; furthermore, reasoning with the union of the input ontologies and LogMap's output mappings did not lead to unsatisfiable classes in the most of the cases. Only in the tests including the ontology *Cocus* we could not provided a clean output since the logical errors involved universal quantifications.

2.3 Benchmark 2011 Track

The goal of this track is to evaluate the tools' behaviour when the input ontologies are lacking important information. The test ontologies for this track have been obtained by performing certain synthetic transformations on realistic ontologies (e.g., suppressing entity labels, flattening the class hierarchy).

System	Precision	Recall	F-score	Incoherence
YAM++	0.78	0.56	0.65	0.07
CODI	0.74	0.57	0.64	0.00
LogMap	***0.84***	***0.5***	***0.63***	***0.02***
AgrMaker	0.65	0.59	0.62	0.12
MassMatch	0.83	0.42	0.56	0.04
CSA	0.5	0.6	0.55	0.29

Table 5. Comparing LogMap with the top 6 tools in the Conference Track of the OAEI 2011.

The computation of candidate mappings in LogMap heavily relies on the similarities between the lexicons of the input ontologies; hence, replacing entity names by random strings has a direct negative impact in the number of discovered mappings.

When taking into account only those tests for which LogMap was able to compute at least one mapping, we obtained an average precision of 0.992 and an average recall of 0.605. In 17 (out of 112) test cases, however, LogMap found no mappings. When taking into account also these cases, we obtained average precision and recall values of 0.827 and 0.504, respectively.

3 General Comments and Conclusions

Comments on the results. We find LogMap's results quite promising.

- In the most of cases, LogMap was able to compute a clean set of output mappings (i.e., not leading to unsatisfiable classes when merged with the input ontologies).
- LogMap was the fastest of all tools in the the Anatomy 2011 Track (computationally, the most challenging of all tracks).
- LogMap obtained very good results in terms of F-score for both the Anatomy 2011 and Conference 2011 tracks.

LogMap's main weakness is that the computation of candidate mappings relies on the similarities between the lexicons of the input ontologies. As already mentioned, LogMap could not find any mappings for 17 of the test cases in the Benchmark 2011 track, since class names were substituted by random strings.

Comments on the OAEI 2011 test cases. Ontology matching tools have significantly improved in the last few years, and there is a need for more challenging and realistic matching problems [15, 6]. To address this need, in [10, 8] we proposed the use of (clean subsets of) UMLS mappings as reference alignments between the large-scale biomedical ontologies FMA, SNOMED CT and NCI. The use in an OAEI track of these ontologies represents a significant leap in complexity w.r.t. the existing anatomy track; however, we take our positive experiences with LogMap as an indication that a new track based on these ontologies and their UMLS alignments would be feasible.

Future developments. We aim at creating a more scalable and robust LogMap; furthermore we also plan to develop the necessary infrastructure for domain experts to interactively contribute to the matching process [12].

Acknowledgements.

We would like to acknowledge the funding support of the Royal Society and the EPSRC project *LogMap*, and also thank the organizers of the OAEI campaign for providing test data and infrastructure.

References

1. Agrawal, R., Borgida, A., Jagadish, H.V.: Efficient management of transitive relationships in large data and knowledge bases. SIGMOD Rec. 18, 253–262 (1989)
2. Bodenreider, O., Hayamizu, T.F., et al.: Of mice and men: Aligning mouse and human anatomies. In: AMIA Annu Symp Proc. pp. 61–65 (2005)
3. Christophides, V., Plexousakis, D., Scholl, M., Tourtounis, S.: On labeling schemes for the Semantic Web. In: Proc. of WWW. pp. 544–555. ACM (2003)
4. Cuenca Grau, B., Horrocks, I., Kazakov, Y., Sattler, U.: Just the right amount: extracting modules from ontologies. In: Proc. of WWW. pp. 717–726 (2007)
5. Dowling, W.F., Gallier, J.H.: Linear-time algorithms for testing the satisfiability of propositional Horn formulae. J. Log. Program. pp. 267–284 (1984)
6. Euzenat, J., Meilicke, C., Stuckenschmidt, H., Shvaiko, P., Trojahn, C.: Ontology Alignment Evaluation Initiative: six years of experience. J Data Semantics (2011)
7. Jimenez-Ruiz, E., Cuenca Grau, B.: LogMap: Logic-based and Scalable Ontology Matching. In: et al., L.A. (ed.) The 10th International Semantic Web Conference (ISWC). LNCS, vol. 7031, pp. 273–288. Springer (2011)
8. Jimenez-Ruiz, E., Cuenca Grau, B.: Towards more challenging problems for ontology matching tools. In: Proc. of the 3th International Workshop on Ontology Matching (OM) (2011)
9. Jimenez-Ruiz, E., Cuenca Grau, B., Horrocks, I., Berlanga, R.: Ontology integration using mappings: Towards getting the right logical consequences. In: Proc. of European Semantic Web Conference (ESWC). pp. 173–187 (2009)
10. Jiménez-Ruiz, E., Cuenca Grau, B., Horrocks, I., Berlanga, R.: Towards a UMLS-based silver standard for matching biomedical ontologies. In: Proc. of the 5th International Workshop on Ontology Matching (2010)
11. Jiménez-Ruiz, E., Cuenca Grau, B., Horrocks, I., Berlanga, R.: Logic-based assessment of the compatibility of UMLS ontology sources. J Biomed. Sem. 2 (2011)
12. Jimenez-Ruiz, E., Cuenca Grau, B., Zhou, Y.: Logmap 2.0: towards logic-based, scalable and interactive ontology matching. In: Proc. of the 4th International Workshop on Semantic Web Applications and Tools for Life Sciences (SWAT4LS) (2011)
13. Meilicke, C., Stuckenschmidt, H.: Incoherence as a basis for measuring the quality of ontology mappings. In: Proc. of the 3rd International Workshop on Ontology Matching (2008)
14. Nebot, V., Berlanga, R.: Efficient retrieval of ontology fragments using an interval labeling scheme. Inf. Sci. 179(24), 4151–4173 (2009)
15. Shvaiko, P., Euzenat, J.: Ten challenges for ontology matching. In: On the Move to Meaningful Internet Systems (OTM Conferences) (2008)
16. Stoilos, G., Stamou, G.B., Kollias, S.D.: A string metric for ontology alignment. In: Proc. of the International Semantic Web Conference (ISWC). pp. 624–637 (2005)

MaasMatch results for OAEI 2011

Frederik C. Schadd, Nico Roos

Maastricht University, The Netherlands
{frederik.schadd, roos}@maastrichtuniversity.nl

Abstract. This paper summarizes the results of the first participation of MaasMatch in the Ontology Alignment Evaluation Initiative (OAEI) of 2011. We provide a brief description of the techniques that have been applied, with the emphasis being on the application of virtual documents and information retrieval techniques in order effectively utilize linguistic ontologies. Also, we discuss the results achieved in the tracks provided under the SEALS modality: benchmark, conference and anatomy.

1 Presentation of the system

1.1 State, purpose, general statement

Sharing and reusing knowledge is an important aspect in modern information systems. Since multiple decades, researchers have been investigating methods that facilitate knowledge sharing in the corporate domain, allowing for instance the integration of external data into a company's own knowledge system. Ontologies are at the center of this research, allowing the explicit definition of a knowledge domain. With the steady development of ontology languages, such as the current OWL language [3], knowledge domains can be modeled with an increasing amount of detail.

Unfortunately, since ontologies of the same knowledge domain are commonly developed separately or for different purposes, transferring information across different sources becomes challenging as the heterogeneities between the ontologies need to be resolved. Several types of heterogeneities can emerge between two ontologies, commonly divided into syntactic, terminological, semantic and semiotic heterogeneities [1].

MaasMatch is an ontology matching tool with a focus on resolving terminological heterogeneities, such that entities with the same meaning but differing names and entities with the same name but different meanings are identified as such and matched accordingly. Given this focus, the tool has been primarily tested using the conference data set, since the ontologies of this data set are more likely to contain these heterogeneities.

1.2 Specific techniques used

In this section we will present the techniques applied in MaasMatch. The overall structure of MaasMatch is simple, being a combination of a string similarity measure and our WordNet similarity, and using the combination of the two similarity matrices to extract the final alignments. However, most of our research so far has been invested into advancing the effectiveness of WordNet similarities.

WordNet makes it possible identify concepts that have the same meaning but different names, since synonyms are grouped into sets, called synsets. However, a more challenging task is the identification of concepts have similar names, but different meanings. As an example, if an ontology contains a concept 'house', then WordNet contains 14 different meanings for this word, and hence 14 different synsets that can be described by this name. One is thus faced with the challenge of automatically identifying the synset that denotes the correct meaning of the ontology entity. To do this, we applied a combination of information retrieval techniques and the creation of virtual documents in order to determine which synset most likely denotes the correct meaning of an entity. That way, only synsets which resulted in a high document similarity with their corresponding concept are subsequently used for the calculation of the WordNet similarity.

The approach can be separated into 5 distinct steps as follows: Given two ontologies O_1 and O_2 that are to be matched, where O_1 contains the sets of entities $E_x^1 = \{e_1^1, e_2^1, ..., e_m^1\}$, where x distinguishes between the set of classes, properties or instances, and O_2 contains the sets of entities $E_y^2 = \{e_1^2, e_2^2, ..., e_n^2\}$, and where $C(e)$ denotes a collection of synsets representing entity e, the main steps of our approach, performed separately for classes, properties and instances, can be described as follows:

1. **Synset Gathering:** For every entity e in E_x^i, assemble the set $C(e)$ with synsets that might denote the meaning of entity e.
2. **Virtual Document Creation:** For every entity e in E_x^i, create a virtual document of e, and a virtual document for every synset in $C(e)$.
3. **Document Similarity:** For every entity e in E_x^i, calculate the document similarities between the virtual document denoting e and the different virtual documents originating from $C(e)$.
4. **Synset Selection:** For every collection $C(e)$, discard all synsets from $C(e)$ that resulted in a low similarity score with the virtual document of e, using some selection procedure.
5. **WordNet Similarity:** Compute the WordNet similarity for all combinations of $e^1 \in E_x^1$ and $e^2 \in E_x^2$ using the processed collections $C(e^1)$ and $C(e^2)$.

The first step of the procedure is fairly straightforward, where all corresponding synsets are collected if the complete name of an entity is present in WordNet and string processing techniques such as word stemming or finding legal sub-strings in the name are applied if the complete name is not present in WordNet. Figure 1 illustrates steps 2 - 5 of our approach for two arbitrary ontology entities e^1 and e^2:

Once the similarity matrix, meaning all pairwise similarities between the entities of both ontologies, are computed, the final alignment of the matching process can be extracted or the matrix can be combined with similarity matrices stemming from other approaches.

Virtual Documents The second step of the approach consists of the creation of virtual documents for an ontology entity and several synsets that might denote the actual meaning of the entity. When constructing the virtual document, one must collect information from the ontology, or WordNet if a virtual document of a synset is constructed, in such a way that the resulting document adequately describes the meaning of the entity. An

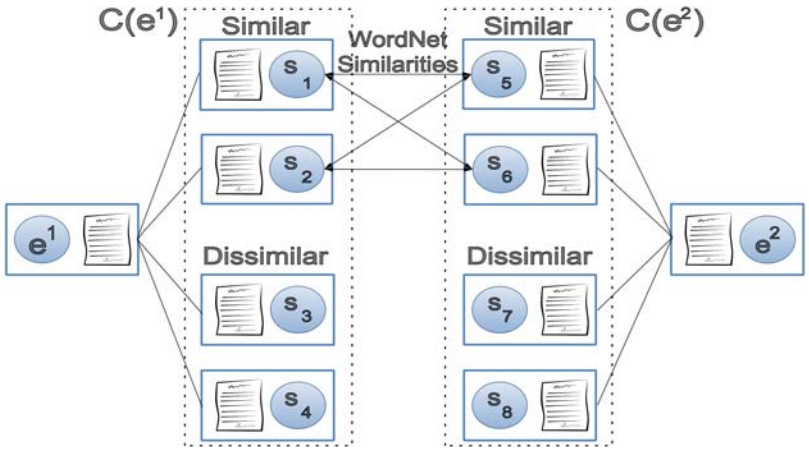


Fig. 1. Visualization of step 2-5 of the proposed approach for any entity e^1 from ontology O_1 and any entity e^2 from ontology 2.

expressive ontology such as OWL allows for the collection from various sources of information. In addition to its own name, an entity can also contain comments, which usually are written descriptions of the entity, and multiple labels. Providing context information is also beneficial. To do this, the names of the parent and child entities are also added to the document. Different details are added given the different types of entities. For virtual documents of classes, the names of all its properties are added and for properties the names of all the classes inside their range and domain are added.

Once all the information for a virtual document is collected, several post-processing techniques such as word-stemming and stop-word removal are applied, before the document is transformed into the vector-space model. Using the document vectors, the similarity between the entity document and the different synset documents is then computed using the cosine similarity.

Synset Selection Based on the document similarities between an entity and the potential synsets, some of the synsets are then discarded based on their similarity value. Several selection procedures have been tested, for instance using a cut-off value by computing the arithmetic or geometric mean of the similarities. Another tested method consisted of retaining only synsets whose document similarity had a higher value than the sum of the mean and standard deviation of the similarity value, which had the intriguing property that only few synsets remained if their similarity values were distinctly higher than the others, and more if this wasn't the case and thus it was uncertain which synset was actually appropriate. Experimentation revealed that stricter selection methods performed better than lenient methods, with the simple method of using only the synset with the highest document similarity to compute the WordNet distance resulting in the highest scoring alignments [4].

1.3 Adaptations made for the evaluation

For experiments unrelated to the actual OAEI competition, but using some of the OAEI datasets, a cut-off confidence value of 0.7 has been used for the alignments, since this is one of the standard values that has previously been used for OAEI evaluations. For the purpose of OAEI participation, however, this value was altered to 0.95 to improve the final F-Measures achieved by the system, especially for the conference data set.

1.4 Link to the system and parameters file

MaasMatch and its corresponding parameter file is available on the SEALS platform and can be downloaded at http://www.seals-project.eu/tool-services/browse-tools.

1.5 Link to the set of provided alignments (in align format)

The computed alignments for the preliminary evaluation can be found at
http://www.personeel.unimaas.nl/frederik-schadd/MaasMatchOAEI2011results.zip.

2 Results

This section presents the evaluation of the OAEI2011 results achieved by MaasMatch using the benchmark, anatomy and conference data sets.

2.1 Benchmark

The results of the benchmark data set are grouped into several categories, such that benchmarks that test similar aspects of the matching process are grouped together. These can be viewed in table 1.

Ontologies	Precision	Recall	F-Measure
101	1.00	0.98	0.99
201-202	0.8	0.34	0.48
221-247	0.97	0.93	0.95
248-266	0.8	0.33	0.47
total	0.99	0.44	0.60

Table 1. Results of the benchmark data set.

Initially, we can see that MaasMatch produces very good results for the groups 101 and 221-247. These groups have in common that they generally test alterations of aspects such as the ontology structure, instances or properties. While these alterations can have some effects on the construction of the virtual document, as for instance the descriptions of parent classes will not be included if the hierarchy is flattened, the overall performance decreases only marginally. The groups 201-202 and 248-266, however, test alterations of the entity names and comments, resulting in a poorer performance and especially impacting the recall of the alignments. The suppression of the entity comments causes the quality of the virtual documents to drop considerably, making it harder to locate appropriate synsets in WordNet. However, most importantly, the WordNet matcher relies on applying string analysis techniques on the entity names to locate potential synsets in WordNet. If these names are altered severely, scrambled or even completely omitted, then it becomes exceedingly difficult to locate potential synsets in WordNet, which would require more sophisticated approaches than currently applied. Overall, one can conclude that MaasMatch achieved a fairly high precision and a moderate recall value across the entire benchmark data set.

2.2 Anatomy

The anatomy data set consists of two large real-world ontologies from the biomedical domain, with one ontology describing the anatomy of a mouse and the other being the NCI Thesaurus, which describes the human anatomy. Since the alignment of two ontologies that both contain more than 1000 entities each required MaasMatch to use more time than the competition made available, only the results of the preliminary evaluation are presented here. The results of this data set can be seen in table 2.

Ontologies	Precision	Recall	F-Measure
mouse-human	0.988	0.284	0.442

Table 2. Results of the anatomy data set.

We can see that MaasMatch achieved a high precision and low recall value. The low recall value can be explained by the fact that WordNet does not contain definitions of highly technical medical terms, resulting in the system being unable to match entities which are not located in the WordNet database. Using a different linguistic ontology should alleviate this problem, or ideally the system should automatically select the most appropriate linguistic ontology for this task.

2.3 Conferences

The conference data set, which so far has been the focus during the development of MaasMatch, consists of 7 real-world ontologies that all describe the domain of organizing conferences. Here, all possible combinations of ontologies are matched and evaluated. The results of each combination can be seen in table 3.

The results of the evaluations vary. Most alignments exhibit a similar trend as in the previous two data sets, being a high precision and moderate recall. However, there are a few exceptions. A few alignments, such as cmt-iasted or iasted-sigkdd, have a higher than moderate recall. Conversely, alignments such as confOf-edas or edas-ekaw have a lower than usual precision. Overall, we can see that the aggregated performance is competitive when compared against the results of the OAEI 2010 participants [2].

Ontologies	Precision	Recall	F-Measure
cmt-confOf	0.800	0.250	0.380
cmt-conference	0.800	0.250	0.380
cmt-edas	0.888	0.615	0.727
cmt-ekaw	0.833	0.454	0.588
cmt-iasted	0.800	1.000	0.888
cmt-sigkdd	0.800	0.666	0.727
confOf-edas	0.538	0.368	0.437
confOf-ekaw	0.888	0.400	0.551
confOf-iasted	0.800	0.444	0.571
confOf-sigkdd	1.000	0.428	0.599
conference-confOf	0.750	0.400	0.521
conference-edas	0.857	0.352	0.499
conference-ekaw	0.727	0.320	0.444
conference-iasted	0.800	0.285	0.421
conference-sigkdd	0.875	0.466	0.608
edas-ekaw	0.666	0.260	0.374
edas-iasted	0.857	0.315	0.461
edas-sigkdd	1.000	0.466	0.636
ekaw-iasted	0.857	0.600	0.705
ekaw-sigkdd	1.000	0.636	0.777
iasted-sigkdd	0.785	0.733	0.758
h-mean	0.83	0.42	0.56

Table 3. Results of the conference data set.

3 General comments

3.1 Comments on the results

The first entrance of MaasMatch has shown a promising performance over the several data sets, most notably due to high precision values under various scenarios. With the focus being on the conference data set, the overall f-measure of MaasMatch has been established at a solid 0.56.

3.2 Discussions on the way to improve the proposed system

Several opportunities present themselves for further improvements. First, as observed during the benchmark experiments, the current approach suffers when ontologies use entity names where potentially appropriate synsets cannot be located in WordNet, due to severely altered or scrambled names. So far, processing techniques such as tokenization, word stemming and stop word removal have been deployed to alleviate this problem, however in severe cases this problem still persists. Hence, it would be beneficial to look into further techniques that enables the matcher to locate synsets even if the entity names are scrambled.

Also, as evident in the anatomy results, if the domain to be modeled contains concepts that are not defined in WordNet, the system is unable to match it correctly. Switching WordNet with a more appropriate linguistic ontology can alleviate this problem. However, this would require the existence of such an ontology for that specific domain. Another predicament is that the system either ceases to be fully automatic, as one would need to manually define an appropriate linguistic ontology for each matching process, or would require the ability to automatically identify an appropriate linguistic ontology each time before the matching is performed.

Given the focus on resolving terminological heterogeneities, one obvious improvement would be the addition of other techniques such that other types of heterogeneities are also identified and resolved, such that the overall recall of the system is improved.

3.3 Comments on the SEALS platform

The SEALS platform is a promising project for researchers to share, test and evaluate their tools. It is fairly easy to integrate a tool into the platform and provides a neutral common ground for a comparison of systems. Unfortunately, two still absent functionalities prevent the platform of becoming the central pivot of ontology matching research. One would be the functionality for the tool developers to independently initiate the testing and evaluation of a tool using the platform, allowing a more speedy development of the tool. Secondly, the standard automatic evaluation function only computes precision and recall for the separate matching tasks. However, previous OAEI competitions also aggregated the results of different matching tasks using a form of the harmonic mean. Since the harmonic mean is not defined for values of 0, which can occur when computing the precision, recall or f-measure of an alignment, a few tweaks are necessary such that this measure can still be applied. Having the results of the separate tasks aggregated automatically by the SEALS platform would eliminate discrepancies that could arise by slightly differing implementations of such aggregation measures, so that it is ensured that the overall performances of the different tools can legitimately be compared. Overall, we are looking forward to the future developments of this platform and possibly utilizing it for further research.

4 Conclusion

This paper presented the results of the first participation of MaasMatch in the Ontology Alignment Evaluation Initiative competition. We described the techniques that are applied in the system and presented the achieved results in the benchmark, anatomy and conference data sets, with emphasis on the conference data set which has been the focus for the development of the system. We note encouraging results and discuss strengths, weaknesses and possible improvements of the system.

References

1. J . Euzenat. Towards a principled approach to semantic interoperability. In A. Gómez Pérez, M. Gruninger, H. Stuckenschmidt, and M. Uschold, editors, *Proceedings of the IJCAI-01 Workshop on Ontologies and Information Sharing*, pages 19–25, 2001.

2. J. Euzenat, A. Ferrara, C. Meilicke, J. Pane, F. Scharffe, P. Shvaiko, H. Stuckenschmidt, O. Svab-Zamazal, V. Svatek, and C. Trojahn. First results of the ontology alignment evaluation initiative 2010. In *Proceedings of ISWC Workshop on OM*, 2010.
3. D. L. McGuinness and F. van Harmelen. OWL web ontology language overview. W3C recommendation, W3C, February 2004.
4. F. C. Schadd and N. Roos. Improving ontology matchers utilizing linguistic ontologies: an information retrieval approach. In *Proceedings of the 23rd Belgian-Dutch Conference on Artificial Intelligence (BNAIC 2011)*, 2011.

MapPSO and MapEVO Results for OAEI 2011

Jürgen Bock, Carsten Dänschel and Matthias Stumpp

FZI Forschungszentrum Informatik, Karlsruhe, Germany
{*surname*}@fzi.de

Abstract. This paper presents and discusses the results produced by the alignment systems MapPSO and MapEVO for the 2011 Ontology Alignment Evaluation Initiative (OAEI). The two systems implement two variants of population-based optimisation algorithms applied to the ontology alignment problem. MapPSO is based on discrete particle swarm optimisation, while MapEVO is based on evolutionary programming. Both systems optimise the same objective function, *i.e.* a function reflecting the quality of an alignment. Firstly, specific characteristics of the systems and their relation to the results obtained in the OAEI are discussed. Secondly, the results for the single tracks are presented and discussed.

1 Presentation of the system

With the 2008 OAEI campaign the MapPSO system (Ontology **Map**ping by **P**article **S**warm **O**ptimisation) was introduced [1] as a novel approach to tackle the ontology alignment problem by applying the technique of particle swarm optimisation (PSO). This year, a similar approach is introduced with the MapEVO system, following the same priciple of ontology alignment by population-based optimisation. MapEVO, however, utilises Evoluationary Programming instead of PSO.

1.1 State, purpose, general statement

The development of the presented systems is motivated by the following observations:

1. Ontologies are becoming numerous in number and large in size.
2. Ontologies evolve gradually.
3. Ontologies differ in key characteristics that can be exploited in order to compute alignments.
4. High ontology alignment quality often cannot be maximised by only assessing it on a correspondence level, but requires global quality metrics.

Solving the ontology alignment problem using a population-based optimistaion approach, addresses these observations as follows:

1. Population-based methods work inherently parallel, such that large ontologies can be aligned on a parallel computation infrastructure.

2. Population-based methods work incrementally, which allows the algorithm to start with an initial or partial configuration (*i.e.* for instance an alignment of previous ontology versions) and refine it as the ontologies evolve.
3. Population-based methods work as a meta-heuristic, *i.e.* independently of the objective function to be optimised. In the case of ontology alignment this means that the objective function can be adjusted according the particular alignment scenario at hand.
4. Population-based methods consider the quality of a complete solution, which, in the case of ontology alignment allows for the assessment of complete alignments, not only on the correspondence level.

The idea of the MapPSO and MapEVO approaches is to provide algorithms that fulfil the aforementioned characteristics. Particularly the focus is not to provide a universal library of similarity measures (base matchers) to form that specific objective function to be optimised, but rather to provide a scalable mechanism that can used with various objective functions depending on the alignment scenario at hand.

Both presented systems are still in the status of a research prototype, where recent work has been done exploiting the parallel nature of MapPSO in a cloud-based infrastructure [2].

1.2 Specific techniques used

MapPSO and MapEVO treat the ontology alignment problem as an optimisation problem and solve it by applying a discrete particle swarm optimisation (DPSO) algorithm in the case of MapPSO [3], and evolutionary programming in the case of MapEVO, respectively. To this end, both algorithms maintain a population of individuals, each representing a valid candidate alignment, which is updated in an iterative fashion in order to converge towards the best alignment. In the case of MapPSO such an individual is a swarm particle, whereas in the case of MapEVO individuals represent evolving species. The difference between MapPSO and MapEVO is as follows. In the evoluationary programming approach some individuals (species) can become extinct and others are allowed to reproduce themselves. In the PSO-based approach, the population is constant throughout the iterations but positions in the search space change according to a particles memory and communication between particles.

The opjective function in both system is the same. It composes of local components, *i.e.* assessments of the single correspondences in a candidate alignment, as well as global components that assess the alignment as a whole.

1.3 Adaptations made for the evaluation

The modalities of the OAEI force the developer to provide a fixed configuration for each system that is applied for all tracks. Thus the provided tool bundles contain a tradeoff configuration between the best configurations for the three tracks executed over the SEALS platform.

1.4 Link to the system and parameters file

The releases of MapPSO and MapEVO together with the parameter files used for the OAEI 2011 campaign are available in the SEALS Tool Respotiry accessible via the SEALS Portal (`http://www.seals-project.eu/`). Additionally, the systems and parameter files are provided for download at `http://sourceforge.net/projects/mappso/files/` in the folder `oaei2011`.

1.5 Link to the set of provided alignments (in align format)

The alignments were created via the SEALS platform and are available in the SEALS Results Repository accessible via the SEALS Portal (`http://www.seals-project.eu/`).

2 Results

Both MapPSO and MapEVO participated solely in the *benchmarks*, *anatomy*, and *conference* tracks that are run via the SEALS platform.

2.1 benchmark

Notes follow after release of final results.

2.2 anatomy

In order to identify all correspondences for the *anatomy* track correctly, it is necessary to utilise an external biomedical thesaurus. As stated by the organisers it is possible to find about half of the correspondences without using such a thresaurus.

The objective function used by MapPSO and MapEVO for this campaign does not utilise a biomedical thesaurus. However, this domain-specfic adjustment could be integrated for both systems without touching the actual search heuristics. However, including such a feature without proper self-adaptation mechanisms would significantly drop the performance in other OAEI tracks. Thus there were no efforts undertaken in including this feature in the *anatomy* track.

Notes follow after release of final results.

2.3 conference

Notes follow after release of final results.

3 General comments

In the following some general statements about the OAEI procedure, modalities, and results obtained are given.

3.1 Comments on the results

Notes follow after release of final results.

3.2 Discussions on the way to improve the proposed system

MapPSO and MapEVO are currently being worked on in order to incorporate a guided local search component for fine-tuning results found by the search heuristics. Additionally, it seems necessary for any system to provide some sort of self-adaptation of alignment quality criteria in order to perform well in all SEALS tracks of the OAEI, since different configurations for different tracks are not possible.

3.3 Comments on the OAEI 2011 procedure

The OAEI modalities require participating systems to use the same parameter configuration for each track and each test case. According to assumption 3 stated in Sect. 1.1 different alignment scenarios will most likely require different means of determining a good alignment. Assuming that an alignment tool will not used in an out-of-the-box configuration in any real-world alignment task, makes this requirement of a single (and thus compromised) parameter configuration rather artificial.

While the argument that systems should be compared with a tradeoff configuration for comparability reasons is acceptable for the *benchmarks* track, it is clearly not reasonable to use the same configuration for *anatomy* and *conference*. Here the obvious focus is to find the best possible alignment. What is currently evaluated, however, is the ability of self-adadaptation of the alignment systems, which can be another track modality, but should not distract from the goal of finding high-quality alignments in a particular domain.

3.4 Comments on the OAEI test cases

Since this year the *benchmarks* dataset is synthetically generated and previously unknown, the following comments refer to the sample data provided prior to the campaign for testing purposes[1].

One comment addresses the best possible alignment any tool could possibly achieve (a.k.a. reference alignment) from an information theoretic point of view. In general the reference alignments provided for the *benchmarks* dataset should not contain any correspondences that are information theoretically impossible to be detected, neither by automatic tools, nor manually. In other words, in a systematically generated test suite, the golden standard should not contain entries that cannot be detected because all information content was removed from the respective data set.

[1] `http://oaei.ontologymatching.org/2011/benchmarks2/index.html`

Suggestions to improve the golden standard would be either to remove correspondences from the reference alignment that have no justification, or to set the confidence values as low as the probability of simply guessing the respective correpsondence.

For instance in test case #201 the correspondences

```
/2011/benchmarks2/101/onto.owl#Conference_Trip
/2011/benchmarks2/201/onto.owl#GBCFRTQEDNXEZMVRUWLFXTDFKC
```

and

```
/2011/benchmarks2/101/onto.owl#Conference_Banquet
/2011/benchmarks2/201/onto.owl#KKRDJIPEEQFBQKOWPOPJWENCPL
```

both are denoted 100 % confident in the reference alignment. Even though there is evidence that the two classes from 201 correspond to the two classes from 101, there is no evidence for the precise assignment given by the reference alignment. Thus any tool (or human) could guess the assignment with a probability of 50 % which should be reflected in the confidence values of the reference correspondences.

4 Conclusion

The alignment systems MapPSO and MapEVO were described briefly with respect to the idea behind their population-based approaches. The results obtained by the two systems for the OAEI 2011 tracks *benchmarks*, *anatomy*, and *conference* were discussed.

Future development of MapPSO and MapEVO will be targeted towards user interaction, improved reasoning support, and guided local search in order to refine the results currently obtained by the heuristic approach.

References

1. Bock, J., Hettenhausen, J.: MapPSO Results for OAEI 2008. In Shvaiko, P., Euzenat, J., Giunchiglia, F., Stuckenschmidt, H., eds.: Proceedings of the 3rd International Workshop on Ontology Matching (OM-2008). Volume 431., http://ceur-ws.org, CEUR Workshop Proceedings (2008)
2. Bock, J., Lenk, A., Dänschel, C.: Ontology Alignment in the Cloud. In Shvaiko, P., Euzenat, J., Giunchiglia, F., Stuckenschmidt, H., Mao, M., Cruz, I., eds.: Proceedings of the 5th International Workshop on Ontology Matching (OM-2010). Volume 689., http://ceur-ws.org, CEUR Workshop Proceedings (November 2010) 73–84
3. Bock, J., Hettenhausen, J.: Discrete Particle Swarm Optimisation for Ontology Alignment. Information Sciences (2010) Article in Press.

MapSSS Results for OAEI 2011

Michelle Cheatham

[1] Wright State University, Dayton, OH, USA
michelle.cheatham@gmail.com

Abstract. MapSSS is very preliminary work on the feasibility of matching OWL -based ontologies in a manner that involves using only limited reasoning over the ontologies. This is the first year MapSSS has been a part of the OAEI competition, and it is hoped that these results will serve as a baseline for comparison with a more mature version of the algorithm a year from now.

1 Presentation of the system

MapSSS is an OWL ontology alignment algorithm designed to explore what can be accomplished using very simple similarity metrics rather than (or at least before) resorting to complex reasoning algorithms. OWL ontologies are treated as simple directed graphs with edges representing OWL relations and nodes representing classes, properties, and individuals. The basic algorithm consists of a syntactic, structural, and semantic metric. The metrics are conservative in the sense that a node that may match more than one node is ignored rather than risking making an error. These metrics are applied one after the other, and a positive result from any one of them is treated as a match. When a match is found, MapSSS attempts to capitalize on this by immediately recursing to look for matches among the immediate neighbors of the newly matched nodes.

1.1 State, purpose, general statement

MapSSS is meant to provided automated alignments of OWL-based ontologies. The basic algorithm is only two-thirds complete but still produces surprisingly good results on several OAEI test sets. All mappings found by MapSSS are currently considered to have a confidence level of 1.0. It would be relatively straightforward to adapt the metrics to use thresholds instead of requiring exact matches. MapSSS currently only supports finding equivalence relations – finding subsumption and other types of relations is a goal of future work.

1.2 Specific techniques used

The three 'S'es in MapSSS correspond to the three types of metrics the algorithm will eventually use: syntactic, structural, and semantic. The syntactic metric is a simple lexical comparison. The structural metric is a graph-based metric based on the direct neighbors of an entity and the edges that connect the entity to those neighbors. The semantic metric has not yet been implemented. We plan to use a Google Research account (http://research.google.com/university/search/) to determine whether or not a pair of entity labels are synonyms.

The main processing loop of MapSSS compares each entity in the first ontology to each entity in the second, first based on the syntactic metric, then the semantic, and finally the semantic metric. Whenever a match is found, the algorithm pauses its main loop execution and instead recurses on the newly matched nodes. At this point the metric treats the direct neighborhoods of the newly matched nodes as if they were the entire ontology. This improves recall because while there may be several matches for a given node within the entire opposing ontology, the one that is closest to an already-matched neighbor is most likely to be correct. The main loop repeats until no new mappings are added by any of the metrics.

Syntactic Metric

Levenstein distance is generally used for the syntactic mapping, though for the OAEI contest only exact matches are considered valid (i.e. the threshold is 1.0). Some minor pre-processing is done on the entity labels prior to running this metric. All labels are converted to lower case and camel case and underscores are converted to spaced words. For instance "oneTwo" and "one_two" are both converted to "one two".

Structural Metric

As mentioned previously, MapSSS treats OWL ontologies as simple graphs. The structural metric acts on the direct neighborhood of the nodes in a candidate match. The metric only adds the candidate nodes to the mapping if they are the ONLY possible matches within the graph of subgraph being considered. The metric has the following constraints:

- The entities must be the same type to be considered possible matches (i.e. classes are only matched with classes, properties with properties, etc).

- Edge labels (e.g. subclass, domain, range, instance, allValuesFrom, someValuesFrom, minCardinality, maxCardinality, cardinality) must match exactly for the entities to be considered possible matches, and the corresponding neighbors at the end of those edges must be of the same type.

- If an entity in one ontology has a neighbor that is already part of the mapping, then the node that neighbor is mapped to must be a neighbor of any prospective match for this entity. This is similar to how the VF2 graph matching algorithm works. This constraint helps to ensure that the generated mapping is internally consistent and coherent.

Semantic Metric

We plan to use the Google Research API to query Google based on the potentially matching entity labels together with configurable search terms such as "synonym" and "translation" and consider the number and quality of the results that are returned. This has several benefits over using WordNet or a domain-specific dictionary:

- It will work with jargon, slang, and other words not likely to be in WordNet.

- It will work with non-English labels.

- It will always be up-to-date with the way words are currently being used by real people.

1.3 Adaptations made for the evaluation

Because the OAEI competition does not consider instance matches in the results, these types of matches are removed from alignment after the algorithm has completed (so that they can still be used by the metrics) but before the final results are stored.

Due to the nature of the OAEI test sets (particularly the benchmark test set), the syntactic (lexical) metric is set to only consider exact lexical matches rather than a Levenstein distance greater than some threshold.

1.4 Link to the system and parameters file

The source code for MapSSS can be downloaded at https://github.com/mcheatham/MapSSS. No parameters file is required.

1.5 Link to the set of provided alignments

The results produced by the system are also available at https://github.com/mcheatham/MapSSS.

2 Results

MapSSS has only been tested extensively on the benchmark test set, but it can produce results for the conference set as well. There are currently problems related to the memory requirements of some methods within Jena (a Java library used to read in and manipulate owl files) that prevent the algorithm from providing results for the anatomy test.

2.1 benchmark

Using the same results format as the ASMOV authors last year, the MapSSS results on the 12 different difficulty levels within the benchmark test set are shown in the table below.

Difficulty	**Precision**	**Recall**	**F1**
0	1.00	1.00	1.00
1	.74	.75	.74
2	.74	.73	.74
3	.89	.84	.86

4	.99	.95	.97
5	.99	.87	.93
6	.99	.74	.85
7	.98	.63	.77
8	.97	.42	.58
9	.70	.13	.22
10	.28	.02	.04
3xx	.93	.69	.80

It is encouraging that MapSSS has high precision on many of the difficulty levels. It should be straightforward to improve the precision on levels 1 and 2 by adding a semantic metric capable of understanding language translations. The algorithm's recall is not nearly so good currently. Plans to improve this are discussed in Section 3.2.

2.2 conferences

During the development of this preliminary version of MapSSS, testing was done primarily using the benchmarks test set. However, the system is also capable of producing results for the conferences test set. The f-measures for each possible pair of ontologies in the test set are shown in the table below. Performance varies widely among the different ontology pairs. Future work on MapSSS will involve analyzing these results in detail to understand and mitigate this variability.

	confOf	**conf.**	**edas**	**ekaw**	**iasted**	**sigkdd**
cmt	.29	.36	.72	.50	.57	.76
confOf		.50	.55	.48	.57	.47
conf.			.53	.44	.36	.48
edas				.35	.42	.67
ekaw					.42	.70
iasted						.69

3 General comments

3.1 Comments on the results

The precision of MapSSS is very reasonable in most cases, but the recall leaves something to be desired in many of the test cases. The next section outlines several possible improvements to the algorithm that are likely to increase its recall. We hope to compare the baseline results presented here with those from an improved version of the algorithm next year.

3.2 Discussions on the way to improve the proposed system

We have several thoughts on improving the performance of MapSSS:

- The semantic metric needs to be implemented. This should improve the algorithm's recall in several cases, particularly when there is not much class hierarchy or there are few relationships between classes.

- After the current algorithm has found all of the matches it can, a second pass can be made that will relax some of the exactness constraints of the metrics, particularly the structural metric. For instance, instead of requiring exact edge matches, a category-oriented approach could be used that would treat e.g. all cardinality restrictions as equivalent. While it is not likely this will improve results on the synthetic OAEI benchmarks, it may improve the recall on the real-world test sets.

- Again after the current algorithm has found all of the matches it can, the structural metric can be run again but rather than looking for exact graph matches, if one graph is a subgraph of the other, ontology reasoning algorithms can be employed in a directed manner to see if the missing links can be inferred.

- The memory and computation requirements of the algorithm can be improved by more careful implementation of the metrics and by spawning off new processes when recursing on a newly discovered match and then merging the results back into the main processing thread.

3.3 Comments on the OAEI 2011 procedure

The SEALS platform is very helpful because it allows testing of algorithms from any computer, without the need to copy the test files to each system. Participants can also always be assured of having the most up-to-date test files.

The two biggest issues were:

- It was not clear from the tutorial that the source files need to be in a subdirectory named after the package name.

- Validation consistently fails because the build file does not seem to properly put the names of the libraries into the descriptor template.

3.4 Comments on the OAEI 2011 test cases

The OAEI test cases are invaluable in allowing ontology alignment algorithms to be compared against one another (and previous versions of the algorithm) on a consistent set of problems.

It would be helpful if there was a SEALS test set that used more of the OWL vocabulary, in order to fully test algorithms that consider that.

As a minor note, Jena, which is a Java library commonly used to read in and manipulate ontologies, does not consider some of the benchmark files (the benchmarksI set that was used in 2010) to be valid. This is because the xml namespace of some entities is an empty string (xmlns = “”).

4 Conclusion

MapSSS is very preliminary work towards an OWL alignment algorithm that uses expensive reasoning techniques sparingly to achieve quality alignments. The algorithm uses a syntactic, structural, and (in the future) semantic metric to determine matching nodes, and takes advantage of the locality present in most ontologies by recursing whenever it finds a match. The current version has reasonable precision but low recall, which the future work outlined here will hopefully improve.

OACAS: results for OAEI 2011

Sami ZGHAL[1], Marouen KACHROUDI[1], Sadok BEN YAHIA[1], and Engelbert MEPHU NGUIFO[2]

[1] University of Tunis El Manar
Computer Science Department, Faculty of Sciences of Tunis, Tunisia
Campus Universitaire, 1060 Tunis, Tunisia
{sadok.benyahia, marouen.kachroudi}@fst.rnu.tn
sami.zghal@planet.tn
[2] LIMOS CNRS UMR 6158, Complexe scientifique des Cézeaux
BP 125, 63173 Aubiere Cedex, France
mephu@isima.fr

Abstract. Ontologies are the kernel of semantic Web. They allow the explicitation of the semantic purpose for structuring different fields of interest. In order to harmonize them and to guarantee the interoperability between these resources, the topic of alignment of ontologies has emerged as an important process to reduce their heterogeneity and improve their exploitation. The paper introduces a new method of alignment of OWL-DL ontologies, using a combination and aggregation of similarity measures. Both ontologies are transformed into a graph which describes their information. The proposed method operates in two steps: local (linguistic similarity composition and neighborhood similarity) step and the aggregation one.

1 Presentation of the system

The method, OACAS [1] (Ontologies Alignment using Composition and Aggregation of Similarities), introduces an alignment algorithm of OWL-DL (Ontology Web Language Description Logic) ontologies. The main thrust of this method is the application of the most suitable similarity measure depending of the category of the node in the ontology. In addition, the OACAS method explores a wider neighborhood than do the pioneering methods of the literature. Carried out experiments showed that OACAS presents very encouraging values of the commonly used evaluation metrics for the assessment of ontologies alignment.

1.1 Specific techniques used

The proposed method, OACAS, alignes two ontologies. Both ontologies are described in the OWL-DL language [2]. Both ontologies are transformed in two graphs O-GRAPHS. The obtained graphs are parsed in order to produce the alignment process out.

Mapping of an OWL-DL Ontology to an O-GRAPH. The process of building the graphs allows to faithfully map the considered ontologies to be aligned in two graphs, *called* O-GRAPHS. An O-GRAPH describes all the information categories included in an OWL-DL ontology: classes, relations and instances. Both classes and instances represent the nodes of the graph. The relations between these different entities are induced by the links of an O-GRAPH. Each entity of the ontology is formalized through an associated notion to the RDF formalism [3]. OWL-DL ontology entities are described thanks to OWL language constructors. These constructors are represented through RDF triplets: <subject, predicate, object>. In an OWL-DL ontology, a class or a relation description is an RDF triplet. The subject corresponds to the class or to the relation. Predicates are OWL primitives, which are OWL and RDF properties. Each property, used in a triplet, sketches a knowledge of the described entity. The arrangement of those nuggets of knowledge constitues the entity definition. The representation of an OWL-DL ontology through an O-GRAPH permits to load the ontology in main memory only once. An O-GRAPH, stored in main memory, statistically reduces the time required to access initial OWL-DL ontology disk resident file.

The alignment method. The introduced OACAS method lays on a composition and an aggregation of similarity computation based model. The method starts by exploring the O-GRAPH structure. It determines the nodes of both ontologies to be aligned and gets out the similarity measures. For each node of the same category (or cluster), the alignment model computes similarity mesures between descriptors by using appropriate functions. Thus, this function considers all the descriptive information of this couple (name, comment and label) as well as its neighborhood structure. An aggregation function combines the similarity measures and the node's structures of the nodes to be aligned. The algorithm implementing the OACAS method takes as input two OWL-DL ontologies to be aligned and produces an RDF file containing the aligned nodes as well as their similarity measures. The alignment method operates into two successive steps. The first one computes the local similarity, whereas the second one computes the aggregation similarity.

First step: Local similarity

The local similarity computation is performed into two successive stages. The first one computes many linguistic similarity measuresand aggregates them for each couple of nodes belonging to the same category (or type). The second one computes neighborhood similarities by exploiting the structures of the nodes to be aligned.

The linguistic similarity computation is carried out once for each node of the same cluster (node of the same type) in the beginning of the alignment process. The linguistic similarity measures of couples of entities of the same type (class, property and instance) are computed. The names of properties and instances are used to compute linguistic similarities. For class category, the computation of the linguistic similarity considers both the comments and labels. The computation of linguistic similarities uses different similarity measures. Those measures are adapted to different descriptors (names, comments and labels) of the entities to be aligned. Different similarity values obtained, for the descriptors, are composed. This composition assigns weights to each similarity measure of descriptors. The sum of the assigned weights to different similarity values is equal to 1. This unit sum guarantees that the composition of the similarity produces a normalized value (between 0 and 1). The LEVENSHTEIN similarity measure [4] is used to compute the similarity value between the names of ontological entities. The Q-GRAM similarity measure [5] computes the similarity value between the comments of the ontological entities. The JARO-WINKLER similarity measure [6] computes the similarity value between the labels of ontological entities. The LINGUISTIC function computes composed linguistic similarity of couples of nodes of both ontologies to be aligned, *i.e.*, O_1 and O_2. It takes as input ***(i)*** both ontologies sketched by two corresponding O-GRAPHS; ***(ii)*** linguistics similarity functions (*i.e.*, $Funct$); and ***(iii)*** weighted attributed to the descriptors nodes (*i.e.*, Π_D). As a result, it produces a composed linguistic similarity vector, V_{CLS}, for each couple of n nodes. The similarity function $Funct$ considers two nodes, N_1 and N_2, and returns the linguistic similarity value of the descriptor, Sim_{LD}. LEVENSHTEIN or Q-GRAM or JARO-WINKLER implements the similarity function, $Funct$, depending of the type of the nodes. Composed linguistic similarity, Sim_{CL}, is computed depending of the descriptors of nodes to be aligned and associate weights to each descriptor, Π_D. Both nodes (N_1 and N_2) and the associated composed linguistic similarity (Sim_{CL}) are added to the composed linguistic similarity vector (V_{CLS}). The composed linguistic similarity of different couples of entities will be used to compute the neighborhood similarity as sketched in the following.

The neighborhood similarity considers both ontologies to be aligned (*i.e.*, O_1 and O_2), the composed similarity vector (V_{CLS}), the weights assigned to each category (Π_C) and the weights associated to the neighbor level (Π_L). Therefore, it produces the neighborhood similarity vector, V_{NS}. The neighborhood similarity computation needs composed linguistic similarity of the couple of nodes to be aligned and the nodes structures. Neighborhood nodes are organized by category, node having the same type. The neighborhood similarity computation propagates similarity into two successive neighborhood levels. The first level (level 1) includes direct neighbors of the nodes to be aligned whereas second one (level 2) contains indirect neighbors. Direct neighbors of the first level represent nodes having direct relationship with the node under consideration. Neighbors of the second level represent nodes having relationship with the nodes of the first one. The neighbors entities of the first level are clustered into three categories (classes, instances or properties). Each category (or cluster) includes ontological entities having the same type. After the step of clustering, the neighborhood similarity is computed between those categories. The neighborhood nodes of the level 2 are treated in the same manner as the neighbors of the first one. The neighborhood similarity by group $MSim$ takes nodes from vectors VN_1 and VN_2 regrouped by category (where VN_1 and VN_2 denote a vector nodes of O_1 and O_2). The process computation uses the "*Match-Based similarity*" [7] as follows:

$$MSim(E, E') = \frac{\sum_{(i,i') \in Pairs(E,E')} Sim_{CLS}(i,i')}{Ma\ \ (|E|, |E'|)}. \tag{1}$$

Both sets E and E' represent nodes of the same cluster belonging respectively to vectors VN_1 and VN_2. The neighborhood similarity, Sim_N, is computed using Equation 2:

$$Sim_N = \sum_{i \in (1,2)} (\varPi_{Vi}(\sum_{(E,E')} \varPi_{(E,E')} MSim(E,E'))), \tag{2}$$

where i stands for the level (*i.e.*, 1 or 2). The neighborhood similarity, Sim_N is a normalized value, since the sum of weights assigned to different neighbors is equal to 1, ($\varPi_{V1} + \varPi_{V2} = 1$). Direct neighbors (level 1) have more important relationships than those of indirect one (level 2). Thus, nodes of level 1 have an important impact on the produced alignment. For this reason, the weight assigned to the first level, $\varPi_{V1} = 0.8$, is more important than the one assigned to the second level, $\varPi_{V2} = 0.2$. In addition, the sum of weights assigned to the category of nodes is equal to 1 ($\sum(\varPi_C) = 1$). Those weights are uniformly assigned between the different categories. The neighborhood similarity is computed thanks to an iterative process, level by level. The obtained values of the composed linguistic similarity, *i.e.* V_{CLS}, and neighbors similarity, *i.e.* V_{NS}, are combined in order to compute aggregation similarity.

Second step: Aggregation similarity

The aggregation similarity is a combined similarity between the local similarities (the composed linguistic similarity and the neighborhood similarity). Function AGGREGATION needs to have in input both ontologies to be aligned, O_1 and O_2, the two similarity vectors, V_{CLS} and V_{NS}, and the weights attributed to the both kind of similarities, $\varPi_{CL}$ and $\varPi_N$. It produces the aggregated similarity vector, V_{AS}. For each couple of entities, N_1 and N_2, of the same category of the both ontologies to be aligned, O_1 and O_2, the aggregated similarity is computed as follows:

$$Sim_A(e_1, e_2) = \varPi_{CL} Sim_{CL}(e_1, e_2) + \varPi_N Sim_N(e_1, e_2). \tag{3}$$

Note that the sum of the weights, attributed to each kind of similarity, is equal to 1 in order to have a normalized aggregation (between 0 and 1). In addition, the sum of weights is equal to 1 ($\varPi_{CL} + \varPi_N = 1$). In the next section, we focus on the experimental evaluation of OACAS.

1.2 Adaptations made for the evaluation

The main objective of the adaptations with the OACAS method is to find the best combination of linguistic measures. In the experimental study, various measures have been used. The goal is to experiment different measures in order to find the more appropriate measure associated to the node descriptors. In order to achieve the objective, 27 arrangements of tests have been experimented. Each test uses a particular combination of similarity measures to compute linguistic similarities between the descriptors of entities to be aligned. During the process of the carried out tests, different weights were assigned to the descriptors (names, comments and labels). The nodes to be aligned can have different descriptors. Depending on the descriptors of the nodes, different weights are attributed. In the case where the nodes are described by three descriptors, the weights are 0.8, 0.1 and 0.1 associated respectively to the names, comments and labels. Whereas the nodes contain only names and comments descriptors, the weights are respectively 0.85 and 0.15. The weights 0.85 and 0.15 are assigned to the names and labels where those the entities are described by them. The experimental results obtained are developed in the next subsection.

The combination using three different linguistic similarities (LEVENSHTEIN, Q-GRAM and JARO-WINKLER) is the best one. In fact, the LEVENSHTEIN measure is more appropriate for computing linguistic similarity between the names of entities to be aligned. Whereas, the Q-GRAM measure is more indicated to compute linguistic similarity between comments of ontological entities. JARO-WINKLER measure is more appropriated for computing linguistic similarity between the labels of entities to be aligned. Indeed, names and labels of ontological entities are short strings. For this type of strings, LEVENSHTEIN and JARO-WINKLER measures are more adapted to compute the linguistic similarity. Comments are strings composed with many words. For this type strings, the Q-GRAM measure gives the best linguistic similarity values.

2 Results

In this section we present the results obtained by OACAS method. Our method produces result for the benchmark tests sets and conference track.

2.1 Benchmark

The benchmark tests sets can be divided into eight groups: 10x, 20x, 22x, 23x, 24x, 25x, 26x and 30x. For each group the mean values of precision and recall are computed. Table 1 shows the values of the evaluation metrics.

Test	Precision	Recall
10x	0.71	1.00
20x	0.44	0.48
22x	0.64	1.00
23x	0.57	1.00
24x	0.40	0.50
25x	0.34	0.46
26x	0.17	0.40
30x	0.47	0.61

Table 1. Mean values of precision and recall for each group of tests

For the group of tests 10x, OACAS achieves precision and recall values of 71% and 100% respectively. Since the ontologies in those tests have complete information, which can used for alignment. The precision mean value can be explained by the fact that OACAS produces alignment containing individuals correspondences. Those correspondences are not included in the reference alignments.

The OACAS method obtains degraded mean values of precision and recall for the family of tests 20x. This degradation can be interpreted by the fact that the ontologies to be aligned contain translated or synonyms descriptor of entities. Our method relies on syntactical treatment of ontological entities.

For the groups of tests 22x and 23x, OACAS obtains 64% and 57% of precision mean values respectively and 100% of recall. The origin of those results is the absence of proprieties, individuals and a flattened hierarchy.

For the tests 25x and 26x combine linguistic and structural problems. For this reason OACAS method provides low mean values of precision and recall.

The problem of individuals absence is still the main handicaps in the real case tests 30x.

2.2 Conference

Table 2 shows the precision and recall values obtained for each test of Conference track.

Test	Precision	Recall
cmt-confOf	0.07	0.40
cmt-conference	0.04	0.29
cmt-edas	0.08	0.67
cmt-ekaw	0.04	0.42
cmt-iasted	0.03	0.95
cmt-sigkdd	0.10	0.79
confOf-edas	0.10	0.55
confOf-ekaw	0.12	0.50
confOf-iasted	0.04	0.44
confOf-sigkdd	0.05	0.52
conference-confOf	0.06	0.46
conference-edas	0.06	0.52
conference-ekaw	0.14	0.68
conference-iasted	0.03	0.33
conference-sigkdd	0.08	0.53
edas-ekaw	0.08	0.52
edas-iasted	0.05	0.52
edas-sigkdd	0.08	0.57
ekaw-iasted	0.04	0.57
ekaw-sigkdd	0.07	0.60
iasted-sigkdd	0.10	0.81

Table 2. Values of precision and recall for Conference track

3 General comments

We participate this year for the first time in OAEI and see the result obtained by our method. The evaluation and comparison of ontology alignment and schema matching components as OAEI is very useful for the development of such technologies.

4 Conclusion

In this paper, we introduced an alignment method of OWL-DL ontologies. The new proposed method OACAS, allows to exploit at most the informative present within in an ontology described in OWL-DL. The process of alignement in the OACAS method, contains two phases: a local phase and a phase of aggregation. The local phase allows to calculate the linguistic similarity consisted as well as the neighborhood similarity. This two similarities are combined during the second phase to determine the aggregation similarity.

References

1. Zghal, S., Kachroudi, M., Ben Yahia, S., Mephu Nguifo, E.: OACAS: Ontologies alignment using composition and aggregation of similarities. In: Proceedings of the 1st International Conference on Knowledge Engineering and Ontology Development (KEOD 2009), Madeira, Portugal (2009) 233–238
2. Smith, M.K., Welty, C., Mcguinness, D.L.: OWL: Ontology Web Language Guide. Technical report, W3C: World Wide Web Consortium, http://www.w3.org/TR/2004/REC-owl-guide-20040210/ (February 2004)
3. Klyne, G., Carroll, J.J.: Resource Description Framework (RDF): Concepts and Abstract Syntax. Technical report, W3C: World Wide Web Consortium, http://www.w3.org/TR/2004/REC-rdf-concepts-20040210/ (February 2004)
4. Levenshtein, I.V.: Binary codes capables of corrections, deletions, insertions and reversals. Soviet Physics-Doklady **10**(8) (1966) 707–710
5. Ukkonen, E.: Approximate string-matching with q-grams and maximal matches. Theoretical Computer Science **92**(1) (1992) 191–211
6. Winkler, W.: The state of record linkage and current research problems. Technical Report 99/04, Statiscs of Income Division, Internal Revenue Service Publication (1999)
7. Valtchev, P.: Construction automatique de taxonomies pour l'aide la représentation de connaissance par objets. Thèse de doctorat, Université de Grenoble 1, France (1999)

OMReasoner: Using Reasoner for Ontology Matching : results for OAEI 2011

Guohua Shen, Lantao Jin, Ziyue Zhao, Zhe Jia, Wenmin He, Zhiqiu Huang

Nanjing University of Aeronautics and Astronautics, Nanjing, China
{ghshen,ltjin,zyzhao,zjia,wmhe,zqhuang}@nuaa.edu.cn

Abstract. Ontology matching produces correspondences between entities of two ontologies. The **OMReasoner** is unique in that it creates an extensible framework for combination of multiple individual matchers, and reasons about ontology matching by using description logic reasoner. It handles ontology matching in semantic level and makes full use of the semantic part of OWL-DL instead of structure. This paper describes the result of **OMReasoner** in the 2011 OAEI competition in two tracks: benchmark and conference.

1 Presentation of the system

Ontology matching finds correspondences between semantically related entities of the ontologies. It plays a key role in many application domains.

Many approaches to ontology matching have been proposed: The implementation of match may use multiple match algorithms or matchers, and the following largely-orthogonal classification criteria are considered [1-3]: schema-level and instance-level, element-level and structure-level, syntactic and semantic, language-based and constraint-based.

Most approaches focus on syntactic aspects instead of semantic ones. OMReasoner achieves the matching by means of reasoning techniques. Still, this approach includes strategy of combination of (mainly syntactical) multi-matchers (e.g., EditDistance matcher, Prefix/Suffix matcher, WordNet matcher) before match reasoning.

1.1 State, purpose, general statement

The matching process can be viewed as a function f (see Fig.1).

$$A' = f(O1, O2, A, p, r)$$

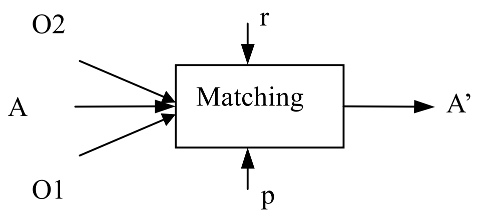


Fig. 1 The ontology matching process

Where *O1* and *O2* are a pair of ontologies as input to match, *A* is the input alignment between these ontologies and *A*' is new alignment returned, *p* is a set of parameters (e.g., weight *w* and threshold τ) and *r* is a set of oracles and resources.

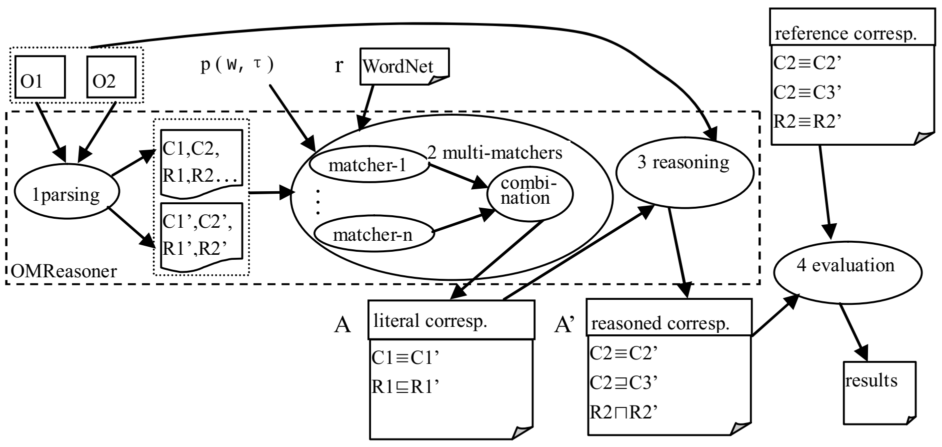


Fig 2. Ontology matching in OMReasoner

The OMReasoner achieved ontology alignment as following four steps (see Fig.2):

1. Parsing: we can achieve the classes and properties of ontologies by using ontology API: Jena.
2. Combination of multiple individual matchers: the literal correspondences (e.g. equivalence) can be produced by using multiple match algorithms or matchers, for example, string similarity measure (prefix, suffix, edit distance) by string-based, constrained-based techniques. Also, some semantic correspondences can be achieved by using some external dictionary: WordNet. Then the multiple match results can be combined by weighted summarizing method. The framework of multi-matchers combination is supported, which facilitates inclusion of new individual matchers.
3. Reasoning: the further semantic correspondences can be deduced by using DL reasoner, which uses literal correspondences produced in step 2 as input.
4. Result evaluation: the evaluation measures will be precision and recall computed against the reference alignments.

1.2 Specific techniques used

OMReasoner includes summarizing algorithm to combine the multiple match results. The combination can be summarized over the n weighted similarity methods (see formula 1), where w_k is the weight for a specific method, and $sim_k(e1,e2)$ is the similarity evaluation by the method.

$$sim(e1,e2)=\sum_{k-1}^{n} w_k sim_k(e1,e2) \tag{1}$$

OMReasoner uses semantic matching methods like WordNet matcher and description logic (DL) reasoning.

WordNet[1] is an electronic lexical database for English, where various senses (possible meanings of a word or expression) of words are put together into sets of synonyms. Relations between ontology entities can be computed in terms of bindings between WordNet senses. This individual matcher uses an external dictionary: WordNet to achieve semantic correspondences.

Another important matcher uses edit distance, which is a measure of the similarity between two words. Based on this value, we calculate the morphology analogous degree by using some math formula.

All the results of each individual matcher will be normalized before combination.

OMReasoner employs DL reasoner provided by Jena. OMReasoner includes external rules to reason about the ontology matching.

2 Results: a comment for each dataset performed

There are 21 alignment tasks in benchmark data set and 21 alignment tasks in conference data set. We test the data sets with OMReasoner and present the results in Table 1, Table 2, Fig 3 and Fig 4. The average measures (precision, recall and F-measure) of Benchmark are 0.359, 0.754 and 0.473 respectively. The average measures of Conference are 0.136, 0.801 and 0.220 respectively. In conclusion, the precision, recall and F-measure are not satisfying, however, it is our first endeavor for ontology matching and we will improve it in the future.

2.1 Benchmark

We evaluated the results against reference alignments, and obtained precision var-ies from 0.071 to 0.636, and recall varies from 0.254 to 1.0, F-measure varies from 0.121 to 0.745.

Label	Dataset	Prec.	Rec	f-Measure
B1	101-101	0.299	0.361	0.327
B2	101-103	0.308	0.722	0.431
B3	101-104	0.369	0.732	0.491
B4	101-224	0.338	0.907	0.493

[1] http://wordnet.princeton.edu/

B5	101-225	0.288	0.887	0.435
B6	101-228	0.308	0.970	0.467
B7	101-230	0.442	0.972	0.608
B8	101-232	0.420	0.918	0.576
B9	101-233	0.438	0.970	0.603
B10	101-236	0.338	1.000	0.506
B11	101-237	0.204	0.591	0.304
B12	101-238	0.186	0.732	0.296
B13	101-239	0.621	0.931	0.745
B14	101-240	0.636	0.758	0.692
B15	101-241	0.606	0.848	0.707
B16	101-246	0.586	0.931	0.719
B17	101-247	0.636	0.848	0.727
B18	101-301	0.085	0.254	0.127
B19	101-302	0.071	0.405	0.121
B20	101-303	0.146	0.417	0.216
B21	101-304	0.224	0.671	0.336

Table.1. Match results in the Benchmark track

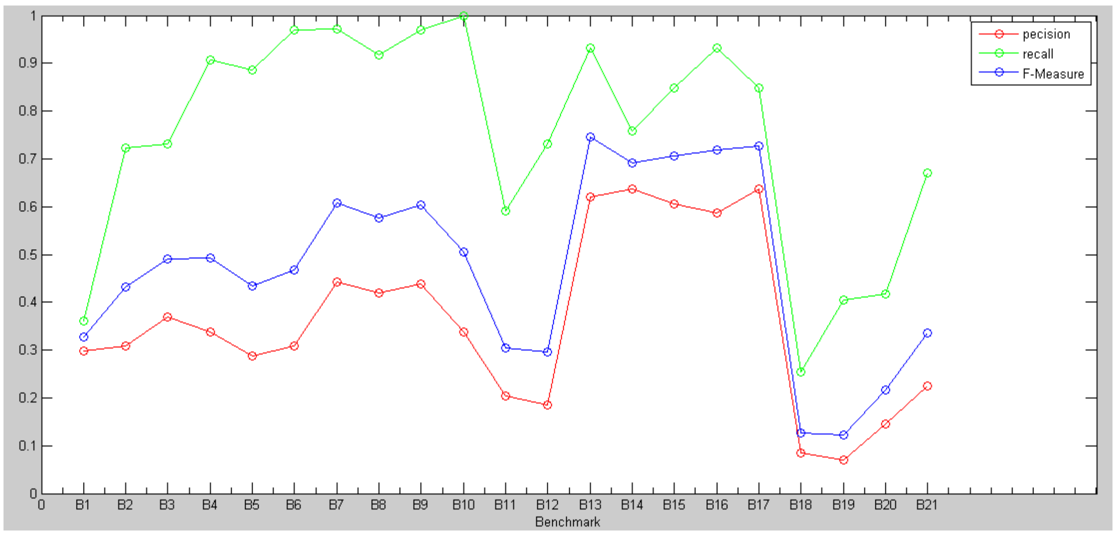


Fig.3. Comparison of match results in Benchmark

2.2 Conference

We evaluated the results against reference alignments, and obtained precision varies from 0.030 to 0.409, and recall varies from 0.526 to 1.0, F-measure varies from 0.058 to 0.495.

Label	Dataset	Prec	Rec	f-Measure
C1	iasted-sigkdd	0.072	0.800	0.132

C2	ekaw-iasted	0.030	0.800	0.058
C3	ekaw-sigkdd	0.033	0.933	0.064
C4	edas-ekaw	0.119	0.652	0.201
C5	edas-iasted	0.088	0.706	0.156
C6	edas-sigkdd	0.120	0.800	0.209
C7	confOf-edas	0.189	0.526	0.278
C8	confOf-ekaw	0.125	0.947	0.221
C9	confOf-iasted	0.071	0.800	0.131
C10	confOf-sigkdd	0.114	0.857	0.202
C11	cmt-Conference	0.158	0.667	0.255
C12	cmt-confOf	0.409	0.625	0.495
C13	cmt-edas	0.162	0.692	0.263
C14	cmt-ekaw	0.159	0.667	0.257
C15	cmt-iasted	0.069	1.000	0.129
C16	cmt-sigkdd	0.297	0.917	0.449
C17	conference-confOf	0.182	0.933	0.304
C18	conference-edas	0.126	0.706	0.215
C19	conference-ekaw	0.142	1.000	0.249
C20	conference-iasted	0.056	0.929	0.106
C21	conference-sigkdd	0.143	0.867	0.245

Table.2. Match results in the Conference track

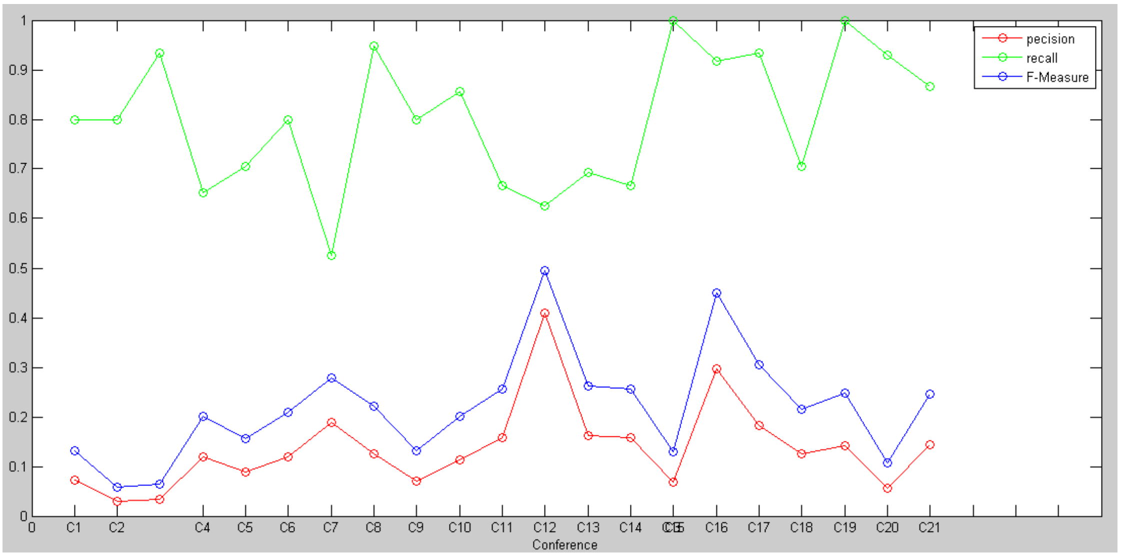


Fig.4. Comparison of match results in Conference

3 General comments

3.1 Comments on the results

The precision of results is not good enough, because only a few individual matchers are included.

The measures in Benchmark are better than those in Conference. The major reason is that the structure similarity of ontology is not considered in our tool.

3.2 Discussions on the way to improve the proposed system

The performance of inference relies on the literal correspondences heavily, so the more accurate results which are exported from multi-matchers will greatly enhance the results of our tool.

Some probable approaches to improve our tool are listed as follow:

1. Adopt more flexible strategies in multi-matchers combination instead of just weighed sum.
2. Add some pre-processes, such as separating compound words, before words are imported into matchers.
3. Take comments and label information of ontology into account, especially when the name of concept is meaningless.
4. Improve the algorithm of some matchers.
5. More different matchers can be included.

Another problem in our tool is that we ignore structure information among ontology at the present stage. And we will improve in the future.

3.3 Comments on the OAEI procedure

OAEI procedure arranged everything in good order, furthermore SEALS platform provides a uniform and convenient way to standardize and evaluate our tool.

4 Conclusions

In this paper, we presented the results of the OMReasoner system for aligning onltologies in the OAEI 2011 competition in two tracks: benchmark and conference. The combination strategy of multiple individual matchers and DL reasonor are included in our approach. This is the first time we participate the OAEI, so it is not sophisticated and the results need to be improved.

References

1. Rahm, E. and Bernstein, P.: A survey of approaches to automatic schema matching. *The VLDB Journal*, ,10(4): 334--350(2001).
2. Shvaiko, P. and Euzenat, J.: A survey of schema-based matching approaches. *Journal on Data Semantics (JoDS)* IV, 146--171(2005).
3. Kalfoglou, Y. and Schorlemmer, M.: Ontology mapping: the state of the art. *The Knowledge Engineering Review Journal*, 18(1):1--31, (2003).
4. Shvaiko, P.: Iterative Schema-based Semantic Matching. PhD, University of Trento, (2006)
5. Jian, N., Hu, W., Cheng, G. et al: Falcon-AO: Aligning Ontologies with Falcon. *In: Proceedings of the K-CAP Workshop on Integrating Ontologies* (2005)
6. Do, H. and Rahm, E.: COMA- a system for flexible combination of schema matching approaches. *In: Proceedings of the International Conference on Very Large Databases*, 610--621.(2002)
7. Giunchiglia, F., Shvaiko, P., and Yatskevich, M.: S-Match: an algorithm and an implementation of semantic matching. *In: Proceedings of the European Semantic Web Symposium*, 61--75.(2004)
8. Kalfoglou, Y. and Schorlemmert, M.: If-map: an ontology mapping method based on information flow theory. *In: Proceedings of ISWC'03, Workshop on Semantic Integration*, (2003)
9. Bouquet, P., Serafini, L., and Zanobini, S.: Semantic coordination: A new approach and an application. *In: Proceedings of the International Semantic Web Conference*, 130--145.(2003)
10. Baader, F., Calvanese, D., McGuinness, D., et al.: The description logic handbook: theory, implementations and applications. *Cambridge University Press*, (2003)
11. Ehrig, M., Sure, Y.: Ontology mapping - an integrated approach. *In Proceedings of the European Semantic Web Symposium (ESWS)*, 76--91, (2004)
12. RacerPro User Guide. http://www.racer -systems. com, 2005
13. Do, H., Melnik, S., Rahm, E.: Comparison of Schema Matching Evaluations. *In: Proceedings of the 2nd Intl. Workshop on Web Databases*, Erfurt, Germany:,221--237(2002)

Optima Results for OAEI 2011

Uthayasanker Thayasivam and Prashant Doshi

THINC Lab, Department of Computer Science, University of Georgia, Athens, Georgia 30602
uthayasa,pdoshi@cs.uga.edu

Abstract. In this report, we present the results of Optima in the Ontology Alignment Evaluation Initiative (OAEI) 2011. We participate in three tracks of the campaign offered in SEALS platform: Benchmark, Conference and Anatomy. We review the iterative ontology alignment approach adopted by Optima and its results for the Benchmark and Conference tracks.

1 Presentation of the system

The increasing usefulness of the semantic Web is in part, due to an increase in the number of ontologies on the Web. Applications such as Web service compositions and semantic Web search, which utilizes these ontologies demand a way to align these ontologies. Nowadays numerous ontology alignment tools exist. They can be broadly identified using, 1) the level of human intervention needed; 2) the amount of prior training data needed; and 3) the facets of ontologies used and the way they are utilized. We present a fully automatic, general purpose ontology alignment tool called Optima [2], which does not need any prior training. Like many other tools, Optima utilizes both lexical and structural facets of ontologies to arrive at an alignment. However, it primarily differs in a different aspect – being iterative – from most other alignment tools that presently exists. Common approaches build an alignment in a single pass using a variety of heuristics and similarity measures. In contrast to single pass approaches Optima continues to improve an alignment in an iterative fashion. Optima formulates the problem of inferring a match between two ontologies as a maximum likelihood problem, and solves it using the technique of expectation-maximization (EM). Specifically, it adopts directed graphs as its model for ontology schemas and uses a generalized version of EM to arrive at a map between the nodes of the graphs. At the end of each iteration, Optima derives a possibly inexact match. Inexact matching is the process of finding a best possible match between the two graphs when exact matching is not possible or is computationally difficult.

We describe briefly the formal model of an ontology as utilized by Optima and the EM-based algorithm adopted by Optima in the next two subsections.

1.1 Ontology Model

Optima adopts the common directed labeled graph model for ontology schemas where the nodes of the graphs are the concepts (named classes in RDFS and OWL) and the labeled edges are the relationships (properties) between the classes. Contemporary languages for describing ontologies such as RDFS and OWL also allow the ontologies to be modeled as directed labeled graphs [3]. Because Optima focuses on identifying a many-one map, let the graph with the larger number of nodes be labeled as the *data* graph while the other as the *model*. Formally, the data graph is modeled as: $O_d = \langle V_d, E_d, L_d \rangle$, where V_d is the set of labeled vertices representing the concepts, E_d is the set of edges representing the relations which is a set of ordered two subsets of V_d, and $L_d : E_d \to \Delta$ where Δ is a set of labels, gives the edge labels. Analogously, $O_m = \langle V_m, E_m, L_m \rangle$ is the model graph against which the data graph is matched. Let M be the standard $|V_d| \times |V_m|$ matrix that represents the match between the two graphs:

$$M = \begin{bmatrix} m_{11} & m_{12} & \cdots & m_{1|V_m|} \\ m_{21} & m_{22} & \cdots & m_{2|V_m|} \\ . & . & \cdots & . \\ . & . & \cdots & . \\ . & . & \cdots & . \\ m_{|V_d|1} & m_{|V_d|2} & \cdots & m_{|V_d||V_m|} \end{bmatrix} \tag{1}$$

Each assignment variable in M is,

$$m_{a\alpha} = \begin{cases} 1 \text{ if } f(x_a) = y_\alpha : x_a \in V_d, y_\alpha \in V_m \\ 0 \text{ otherwise} \end{cases}$$

where $f(\cdot)$ represents the correspondence between the two ontology graphs. Consequently, M is a binary matrix representing the match.

1.2 EM-based Algorithm

Optima views the mapping between two ontologies as the problem of, the concepts of source ontology (data graph) emitting the concepts of target ontology (model graph) with an underlying Bernoulli distribution. It formulates this model as a maximum likelihood problem and solves it using the popular expectation maximization algorithm (EM) developed by Dempster et al. [1] to find the maximum likelihood estimate of the alignment from observed data instances in the presence of missing correspondence. It iteratively searches for the match matrix, M_* , that gives the maximum conditional probability of the data graph, $\mathcal{O}_d$, given the model graph, $\mathcal{O}_m$, and the match assignments. Formally,

$$M_* = \underset{M \in \mathcal{M}}{\arg\max}\, Pr(\mathcal{O}_d | \mathcal{O}_m, M)$$

where $\mathcal{M}$ is the set of all match matrices. While there may be as many as $2^{|V_d||V_m|}$ possible alignments, Optima shrinks this space by considering many-one maps only. In the equation above, Optima uses heuristics to guide its search space. Section 1.4 explains the heuristics used in Optima.

$$Pr\left(\mathcal{O}_d | \mathcal{O}_m, M\right) = \prod_{x_a \in V_d} \sum_{y_\alpha \in V_m} Pr(x_a | y_\alpha, M) \pi_\alpha \tag{2}$$

where $\pi_\alpha = Pr(y_\alpha|M)$ is the prior probability of the model graph vertex, y_α, given the match matrix, M. The correspondence, f, is hidden from us. The matrix M may be seen as a mixture model by viewing each assignment variable, $m_{a\alpha}$, as a model.

This modeling does not have an inherent way of finding mapping between edges. Though it is viable for Optima to map the bipartition transformation of the provided graph it avoids it for the excessive complexity involved. Hence, Optima additionally allows matching the concept graph and labeled relationships as separate but dependent tasks.

E Step Optima formulates a conditional expectation of the log likelihood with respect to the hidden variables given the data graph and a guess of the match matrix, M^n at some iteration n, in order to find the most likely match matrix:

$$\begin{aligned} Q(M^{n+1}|M^n) &= E\left[logPr(x_a|y_\alpha, M^{n+1})\pi_\alpha^{n+1}|x_a, M^n\right] \\ &= \textstyle\sum_{a=1}^{|V_d|}\sum_{\alpha=1}^{|V_m|} Pr(y_\alpha|x_a, M^n)\ logPr(x_a|y_\alpha, M^{n+1})\pi_\alpha^{n+1} \end{aligned} \tag{3}$$

Optima derives the probability that the data graph node, x_a, is in correspondence with the model graph node, y_α, under the match matrix of iteration n , M^n as ,

$$Pr(x_a|y_\alpha, M^n) = \left[\tfrac{1}{Pr(x_a|y_\alpha)}\right]^{|V_d||V_m|-1} \textstyle\prod_{b=1}^{|V_d|}\prod_{\beta=1}^{|V_m|} Pr(x_a|y_\alpha, m_{b\beta}^n) \tag{4}$$

Here, it is assumed that the individual models, $m_{b\beta}^n$, are independent of each other.

Optima extends the structural graph matching initially proposed by Luo and Hancock [5] with label similarity measures to derive the probability that x_a is in correspondence with y_α given the assignment model, $m_{b\beta}$.

$$Pr\left(x_a|y_\alpha, m_{b\beta}^n\right) = (1 - P_\epsilon(x_a, y_\alpha))^{EC}\ P_\epsilon(x_a, y_\alpha)^{1-EC} \tag{5}$$

where the correspondence error, $P_\epsilon : V_d \times V_m \to [0, 1]$, is defined as,

$$P_\epsilon(x_a, y_\alpha) = P_e(|V_d|, |V_m|) - \delta \times P_s(x_a, y_\alpha) \tag{6}$$

EC denotes the edge consistency between the two graphs, which is defined as,

$$EC = \begin{cases} 1 & \langle x_a, x_b\rangle \in E_d \wedge \langle y_\alpha, y_\beta\rangle \in E_m \wedge m_{b\beta} = 1 \\ 0 & \text{otherwise} \end{cases}$$

The correspondence error, P_ϵ, is based on the structural error, $P_e\ (|V_d|, |V_m|)$, a function based on the sizes of the graphs, and the similarity of the node labels, $P_s(x_a, y_\alpha)$. Parameter $\delta \in [0, 1]$ controls how much weight is given to the similarity between entity labels. The structural error is defined as,

$$P_e(|V_d|, |V_m|) = 2\left|\frac{|V_d| - |V_m|}{|V_d| + |V_m|}\right|$$

Optima employs the integrated similarity mentioned in Section 1.3 to evaluate the lexical similarity $P_s(x_a, y_\alpha)$.

M Step The maximization step chooses the match matrix, M_*^{n+1}, that maximizes $Q(M^{n+1}|M^n)$, as shown in Eq. 3. This mapping matrix becomes the input for the expectation step of the next iteration. Optima adopts the generalized EM, which relaxes maximization by settling for a mixture model, M_*^{n+1}, that simply improves the Q values.

$$M_*^{n+1} = M^{n+1} \in \mathcal{M} : Q(M^{n+1}|M^n) \geq Q(M^n|M^n) \tag{7}$$

The prior, π_α^{n+1}, for each model graph node, α, is updated as:

$$\pi_\alpha^{n+1} = \frac{1}{|V_d|} \sum_{\alpha=1}^{|V_d|} Pr\left(y_\alpha | x_a, M^n\right) \tag{8}$$

The updated π_α^{n+1} will be used in the next iteration of the EM.

1.3 Specific Techniques Used

We configured Optima slightly different for OAEI from its default configuration.

Integrated Similarity Measure Concept or word similarity measures may be broadly categorized into syntactic and semantic. Syntactic similarity between concepts is entirely based on the string similarity between the concepts' names, labels and other associated text. Semantic similarity measures attempt to utilize the meaning behind the concept names to ascertain the similarity of the concepts. Optima utilizes both syntactic and semantic similarities.

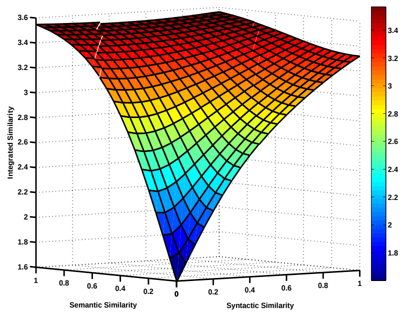


Fig. 1. Integrated similarity - 3D sigmoid

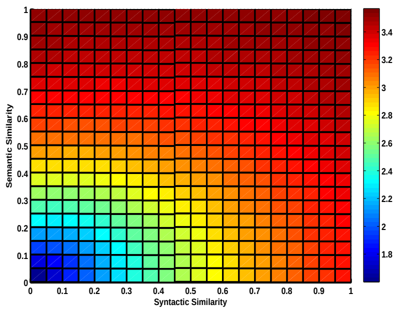


Fig. 2. Integrated similarity - Top angle

Fig. 3. Our integrated similarity measure as a function of the WN-based semantic similarity (*Sem*) and Smith-Waterman based syntactic similarity (*Syn*). Notice that the value is lower if semantic similarity is low but syntactic is high compared to vice versa.

There is no standard way of integrating WN-based similarity with syntactic measures. We employs a technique from our previous work in [8] to integrate similarity measures. We define a normalized 3D function that maps a given pair of semantic and syntactic similarity to an integrated value. In order to generate this function, we observe that labels that are syntactically similar (such as *cat* and *bat*) may have different meanings. Because we wish to meaningfully map entities, semantic similarity takes precedence over syntactic. Consequently, high syntactic but low semantic similarity results in a lower integrated similarity value in comparison to low syntactic but high semantic similarity. We model such an integrated similarity measure as shown in Fig. 3 and give the function in Eq. 9. Our integrated similarity function is similar to a 3D sigmoid restricted to the quadrant where the semantic and syntactic similarities range from 0 to 1. One difference from the exact sigmoid is due to the specific property it must have because semantic similarity takes precedence over syntactic. We used Lin [4] similarity measure and gloss-based cosine similarity measure to evaluate the semantic similarity. On the other hand we used Smith-Waterman [7] technique for ascertaining the syntactic similarity between concept and relationship names.

$$Int(x_a, y_\alpha) = \gamma \frac{1}{1 + e^{t \cdot r - c(Sem)}} \tag{9}$$

Here, γ is a normalization constant; $r = \sqrt{Syn^2 + Sem^2}$, which produces the 3D sigmoid about the origin; t is a scaling factor and $c(Sem)$ is a function of the semantic similarity as shown below: $c(Sem) = \frac{2}{1 + e^{t' \cdot Sem(x_a, y_\alpha) - c'}}$ where t' is the scaling factor and c' is the translation factor, if needed. The specific function in Fig. 3 is obtained when $t = 4$, $t' = 3.5$, and $c' = 2$.

1.4 Adaptations made for the evaluation

The iterative alignment algorithm requires a seed map. This is an initial list of mappings between concepts often provided to iterative algorithms. While the seed map could be generated manually, Optima additionally utilizes a simple technique of mapping nodes across the ontologies whose labels are syntactically similar. Candidate alignments are generated using simple but intuitive heuristics. For example, given each previously mapped node pair, their parents are considered for a match. Additionally, their sibling nodes could be considered. Analogous to the seed map, node pairs among the parents that are sufficiently similar are matched. Different potential alignments are generated based on how many parent nodes are matched and whether siblings are matched as well. These candidate alignments are considered during each iteration of Optima. More details about Optima are available in [2].

We also relaxed the Optima 's many-to-one constrain in candidate alignment generation to generate many-to-many alignments for OAEI.

1.5 Link to the system and parameters file

The Optima can be found at `http://thinc.cs.uga.edu/thinclabwiki/index.php/Automated_Alignment_of_Ontologies`.

1.6 Link to the set of provided alignments (in align format)

The OAEI 2011 results can be found at `http://thinc.cs.uga.edu/thinclabwiki/index.php/OAEI_2011`.

2 Results

As stated above, Optima participated in three tracks in OAEI 2011. However for this report preliminary results of two tracks are presented and the related analysis are reported.

2.1 Benchmark

The average precision and recall of Optima are depicted in 1.

	Precision	**Recall**
100	0.90	1.0
200	0.79	0.73
300	0.74	0.79

Table 1. Recall and Precision of Optima on benchmark track

2.2 Conferences

Optima attains an average recall of 0.60 and an average precision of 0.26 in conference track. See Appendix A for details.

2.3 Anatomy

We could not produce the results for anatomy track using Optima within the provided time. Since Optima utilizes an iterative algorithm and anatomy track has very large ontologies, we were unable to complete aligning these ontologies.

3 General comments

The primary challenge for Optima is to align very large ontologies. Due to its iterative nature and inherent computational complexity of evaluating the Equation 3, Optima takes considerably longer time to align larger ontologies. However it is able to align small to medium ontologies competitively.

We also found that computing semantic similarity measures for word phrases and compound words is difficult. Tokenizing these correctly and locating individual glosses in WN is often challenging[1] but crucial for a better performance.

[1] The concept *Meta-Review* should be tokenized into two words *(Meta, Review)* while *Registration_Non–Member* needs to be tokenized into two words *(Registration, NonMember)* but

4 Conclusion

In this report we present the results of Optima in OAEI 2011 campaign. We participate in three tracks including Benchmark, Conference and Anatomy. We reviewed the iterative algorithm Optima adopts to arrive at an inexact match between two ontologies. Though we have been using OAEI datasets for various experiments and fine tuning of Optima , this is the first time we participate officially in an OAEI campaign. Due to its iterative nature Optima takes substantially longer time to align large ontologies. As a result we are unable to provide our preliminary results of anatomy track for this report. In future, we would like to participate in more tracks. Especially we hope to leverage Optima to be able to efficiently solve instance matching and large ontology matching challenges.

References

1. Dempster, A.P.; Laird N.M. and Rubin, D.B. 1977. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B (Methodological)* 39:1 – 38.
2. Doshi, P.; Kolli, R.; and Thomas, C. 2009. Inexact matching of ontology graphs using expectation-maximization. *Web Semantics* 7(2):90 – 106.
3. Hayes, J. and Gutierrez, C. 2004. Bipartite graphs as intermediate models for RDF graphs. *Proceedings of the International Semantic Web Conference (ISWC)* :47 – 61.
4. Lin, D. 1998. An information-theoretic definition of similarity. In *ICML*, 296–304.
5. Luo, B. and Hancock, E. 2001. Structural graph matching using the EM algorithm and singular value decomposition. *Graph Algorithms and Computer Vision* 23 10:1120 – 1136.
6. McGuinness, D., and Harmelen, F. 2004. Owl web ontology language overview. Recommendation, World Wide Web Consortium (W3C).
7. Smith, T. F., and Waterman, M. S. 1981. Identification of common molecular subsequences. *Journal of molecular biology* 147(1):195–197.
8. Uthayasanker, T., and Doshi, P. 2011. On the Utility of WordNet for Ontology Alignment: Is it Really Worth It?. *IEEE ICSC* :268–274.

should not be tokenized into three words *(Registration, Non, Member)*. The hyphen (–) is a delimiter in the former concept but should be just ignored in the later concept. This tokenization is demanded by WN matchers since *MetaReview* does not exist in WN but the word *NonMember* exists in WN.

A Optima ’s performance in conference track

The precision and recall for individual test cases in conference track is shown tn the table 2 below.

Ontology pair	Precision	Recall
cmt-confOf	0.35	0.50
cmt-conference	0.18	0.44
cmt-edas	0.24	0.69
cmt-ekaw	0.15	0.45
cmt-iasted	0.33	1.00
cmt-sigkdd	0.39	0.75
confOf-edas	0.27	0.68
confOf-ekaw	0.30	0.55
confOf-iasted	0.33	0.67
confOf-sigkdd	0.26	0.71
conference-confOf	0.32	0.67
conference-edas	0.17	0.53
conference-ekaw	0.16	0.40
conference-iasted	0.15	0.29
conference-sigkdd	0.34	0.67
edas-ekaw	0.21	0.52
edas-iasted	0.35	0.47
edas-sigkdd	0.26	0.60
ekaw-iasted	0.20	0.60
ekaw-sigkdd	0.27	0.64
iasted-sigkdd	0.31	0.73
average	0.26	0.60

Table 2. Optima ’s performance in conference track of OAEI 2011

SERIMI Results for OAEI 2011

Samur Araujo[1], Arjen de Vries[1], and Daniel Schwabe[2]

[1] Delft University of Technology, PO Box 5031, 2600 GA Delft, the Netherlands
{S.F.CardosodeAraujo, A.P.deVries}@tudelft.nl

[2]Informatics Department, PUC-Rio Rua Marques de Sao Vicente, 225, Rio de Janeiro, Brazil
dschwabe@inf.puc-rio.br

Abstract. This paper presents the results of SERIMI in the Ontology Alignment Evaluation Initiative (OAEI) 2011. We participate in the track IM@OAEI2011 (IMEI) of the campaign. We first describe the basic interlinking process and interlinking strategies in SERIMI, and then we present specific techniques used in this track. We conclude with a discussion of our results, and possible directions to improve SERIMI in future work.

Keywords: data integration, RDF interlinking, instance matching, linked data, entity recognition, entity search.

1 Presentation of the System

The interlinking of datasets published in the Linked Data Cloud (LDC) [1] is a challenging problem and a key factor for the success of the Semantic Web. Given the heterogeneity of the LDC, techniques aimed at supporting interlinking should ideally operate agnostic of a specific domain or schema.

In this context, ontology matching [2, 3, 4, 5, 6] and instance matching [9] are the two most-studied sub-problems of interlinking. The former refers to the process of determining correspondences between ontological concepts. The latter often refers to the process of determining whether two descriptions refer to the same real-world entity in a given domain. In this paper we focus on the problem of instance matching.

1.1. State, purpose, general statement

Our solution for the instance-matching problem is composed of two phases: the selection phase and the disambiguation phase. In the selection phase we apply traditional information retrieval strategies to generate a set of candidate resources for interlinking. For each instance **r** in a source dataset A, we extract its label (its identifier) and we search for instances in a target dataset B that may have a similar label. The problem that multiple distinct instances in B may share the same label is addressed in the second, disambiguation phase. Here, we attempt to filter among the instances found in B, those that actually refer to the same entity in the real world as **r**.

SERIMI uses existing traditional information retrieval and string matching algorithms for solving the selection phase; our contribution is the novel similarity measure used in the disambiguation phase. This function is designed to operate even when there is no direct ontology alignment between the source and target datasets being interlinked. For example, SERIMI is able to interlink a dataset A that describes social aspects of countries with a dataset B that describes geographical aspects of countries. The SERIMI software is available for download as an interlinking tool at GitHub[1]. Fig. 1 and Fig. 2 show an overview of SERIMI's architecture.

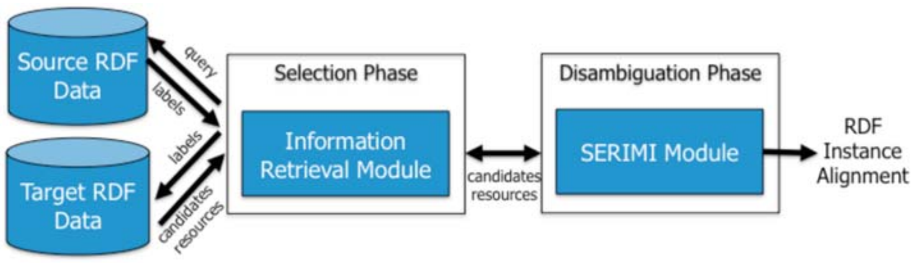


Fig. 1 – Overview of SERIMI's architecture.

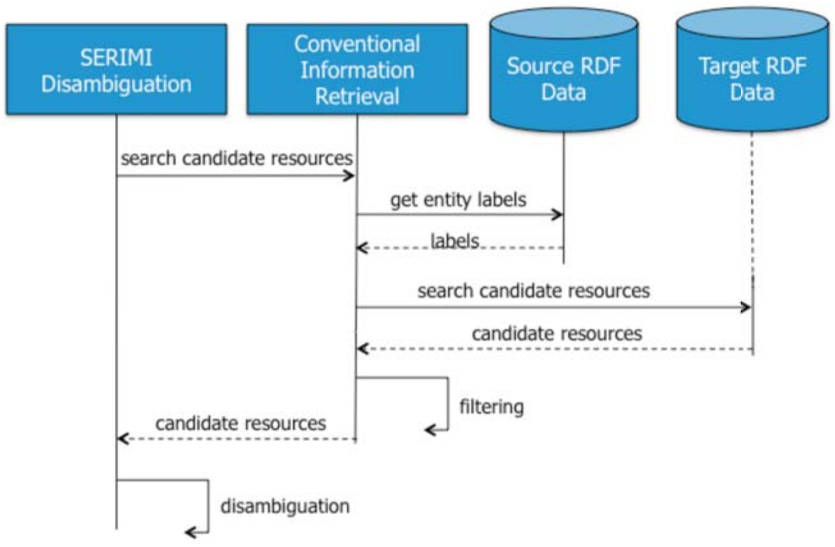


Fig. 2 – Overview of SERIMI's information flow.

1.2. Specific Technique Used

Fig.3 shows an overview of the SERIMI interlinking process.

[1] https://github.com/samuraraujo/SERIMI-RDF-Interlinking

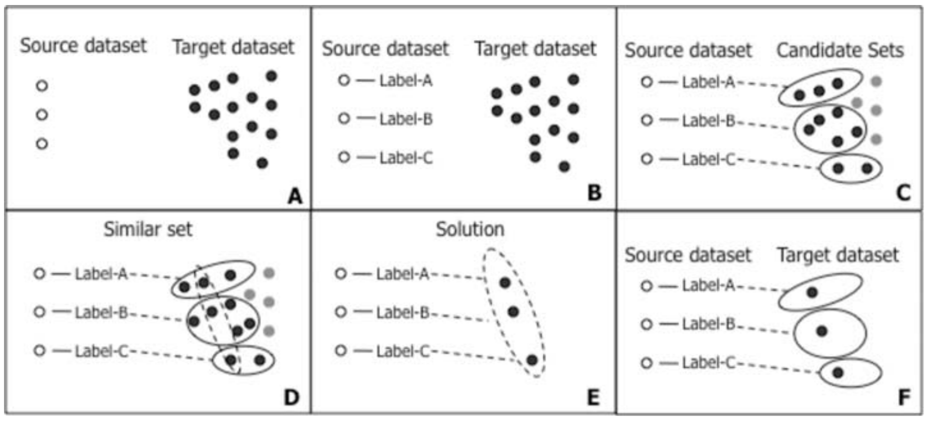


Fig. 3 – Overview of SERIMI interlinking process. (A) Given a source and target dataset and a set of source resources (instance of a class), (B) SERIMI obtains the label of these source resources and retrieves candidate resources from the target dataset that share a similar label. (C) For each source resource, SERIMI retrieves a set of candidate resources. (D) In order to disambiguate a set of candidate, SERIMI applies a novel function of similarity that selects the resources that are the most similar between all candidate sets (E). These selected resources are the solutions for the interlinking (F). The determination of this optimal cross section is a sophisticated process based on an underlying assumption that the source resources belong to a homogeneous class of interest (e.g. musician, drugs, country, etc.)

1.2.1. Selection Phase

In SERIMI's selection phase we first select the class of resources in the source dataset that we want to interlink. For each class (an rdfs:type object) found in the source dataset, SERIMI selects its instances and applies the approach below to select candidate resources in the target dataset.

Entity label property selection: in order to select resources in the target dataset that can match a specific source resource, we first select the labels that represent these source resources. We call *entity label properties*, the properties where these labels occur. We consider as entity label properties, all RDF predicates that have a literal, including numbers, but we eliminate long text values. We assume that we do not know the entity label properties in advance, and apply an automatic approach to select those. Considering that predicates with higher entropy are more discriminative than predicates with lower entropy, we select predicates with entropy $\Omega \geq \Omega_{threshold}$, where $\Omega_{threshold}$ is obtained by averaging the entropy of all predicates of the resources that we want to interlink. Those selected predicates compose the list of entity label properties. The procedure above is applied over the set of source instances selected for interlinking.

Pseudo-homonym resource selection: once we have determined the entity label properties in the source, we can use their labels for searching for resources in the target dataset that share the same or similar labels. We refer to the set of target candidate resources that share a similar label as the *pseudo-homonym set*. For each resource to be interlinked, we use this source entity label for searching for candidate resources in the target dataset. We apply the same step described in the previous paragraph over the target dataset, to obtain the set of entity label properties in the target dataset. Then we search for the source entity label only on triples that contain such selected properties. For each source entity label, we normalize the string (by removing non alphanumeric characters), tokenize it, and then we apply a set of conjunctive Boolean queries (expressed in SPARQL) for retrieving target candidate resources. Afterwards, we select from the retrieved resources those with a maximum string similarity with respect to the searched source entity label. If the maximum score is below 70% we discard it. As a string matching algorithm, we used a variation of the RWSA[7] algorithm. By selecting only those resources with maximum relative similarity measure, we reduce the number of resources in the pseudo-homonym set, thereby improving the chance of true positive matches. If no resource is retrieved, then we select the next entity label property with the highest entropy and repeat the same procedure. This process ends forming a set of pseudo-homonym resources for each source resource. Then the task is to select from each set the resource(s) that one which is (are) more similar to the source resources. We do this selection during the disambiguation phase.

1.2.2. Disambiguation Phase

Pseudo-homonyms resource disambiguation: in some cases, a pseudo-homonym set may have instances of different classes or instances of the same classes that share the same label. As we do not know the class of the resources that we are looking for in the target dataset, we try to leverage this class of interest from the pseudo-homonym resources. Once the class of interest in determined, we can disambiguate the pseudo-homonym resources, by selecting the resources that belong to the class of interest. Notice that the concept of class of interest is understood as a set of attributes that instances may share in common. To solve this ambiguity problem, we propose an innovative model called *Resource Description Similarity,* or *RDS*. RDS uses the intuition that if we select two or more resources that are similar in the source dataset, and for each of them there is a set of pseudo-homonym resources in the target dataset, then the solutions for each pseudo-homonym set should be similar among themselves. In other words, the solution to the problem is the set of resources that are the most similar among pseudo-homonym sets, which implicitly defines the class of interest. The main requirement to apply this method is that we have to have at least two sets of pseudo-homonyms. Fig.3d and Fig. 4 illustrate this intuition.

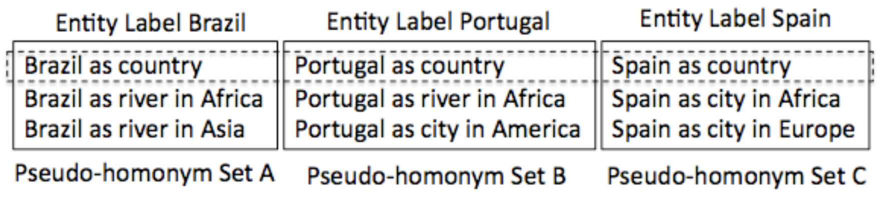


Fig. 3 – A simple example of pseudo-homonym sets for three labels that represent countries.

Disambiguating candidate resources: Given S as a set of all sets of pseudo-homonyms and $R \in S$, for each resource r in R, we generate a score δ = CRDS(r, R, S). As solution for a pseudo-homonym set R, we select all resources with a score $\delta \geq \delta_{threshold}$. Details about the function CRDS is given in [8].

1.3. Adaptations made for the evaluation

SERIMI operates directly over SPARQL Endpoints. For that reason, we have loaded the RDF version of the datasets Geonames, Freebase and NYTimes into an open-source instance of Virtuoso Universal server[2] installed on a local workstation, summing up millions of RDF triples. An exception was the DBPedia dataset, which we accessed online via its SPARQL endpoint. Then we run our method over these endpoints.

1.4. Link to the system and parameters file

SERIMI can be found at: https://github.com/samuraraujo/SERIMI-RDF-Interlinking

1.5. Link to the set of provided alignments (in align format)

The alignments for OAEI2011 campaign should be available at the official web-site: http://www.instancematching.org/oaei/imei2011.html. Alternatively, these can also be found at: https://github.com/samuraraujo/SERIMI-RDF-Interlinking.

2 Results

We now provide an analysis of the results obtained with SERIMI on the Instance Matching track (IM), on the subtask of data integration (Interlinking New-York Times Data) of the OAEI 2011 campaign. We use SERIMI to resolve RDF instance interlinking between the pairs of datasets, namely NYT-People vs. DBpedia, NYT-Locations vs. DBpedia, NYT-Organization vs. DBpedia, NYT-People vs. Freebase, NYT-Locations vs. Freebase, NYT-Organization vs. Freebase, NYT-Location vs. Geonames. Table 1 shows the results for each pair of dataset above.

[2] http://virtuoso.openlinksw.com/dataspace/dav/wiki/Main/

Table 1. SERIMI's precision and recall.

Dataset Pairs	Precision	Recall	F1
NYT-People vs. DBpedia	0.943	0.942	0.943
NYT-Locations vs. DBpedia	0.693	0.670	0.681
NYT-Organizations vs. DBpedia	0.887	0.870	0.878
NYT-People vs. Freebase	0.923	0.911	0.920
NYT-Locations vs. Freebase	0.922	0.904	0.913
NYT-Organizations vs. Freebase	0.921	0.895	0.908
NYT-Locations vs. Geonames	0.787	0.807	0.797

As we can see in Table 1, SERIMI performed quite well in all cases.

Although SERIMI was designed to perform over RDF datasets where the instances are organized in fine-grained homogenous classes, it performed quite well in average in the NYT scenario, where the instances are grouped in four heterogeneous classes (organization, locations, people, and, descriptors). This heterogeneity on the data was the main reason that we obtained a poor performance in the pair NYT-Locations vs. DBpedia. The NYT-Locations instances are very ambiguous and the class is too heterogeneous, representing cities, countries, lakes, etc. For instance, this class does not distinguish a city from a neighborhood, and for that reason SERIMI's disambiguation phase could not work properly. Even if the results are far from perfect, in this specific case of NYT-Locations vs. DBpedia, the results show that the proposed disambiguation phase leads to an approximated gain of 64% over the accuracy of the selection phase on its own (which produced a F1 of 44%).

Regarding SERIMI's selection phase, the set of boolean queries used in SERIMI failed in selecting resources where the label of the source resource was an abbreviation or acronym of the target resource, or vice-versa (e.g. source: "Minnesota Mining & Manufacturing Co", target: "3M Company"). SERIMI has also failed due to distinct string formatting between the source and target datasets labels. For instance, SERIMI selected the resource labeled "Jackson Michael" for the searched label "Jackson, Michael", instead of the resource labeled "Michael Jackson", which was the correct answer. These two problems are known issues in the literature, and we intent to investigate them as future work.

We noticed that the use of ontological knowledge could have improved the precision in the Geonames case, since the instances of the class NYT-Location have the properties longitude and latitude that also occur in the Geonames, with exactly the same values. For instance, the use of both label and longitude in the search process of the selection phase would have improved the precision for this case. Nevertheless, as we aim to provide a fully automated approach agnostic of ontology, we did not consider the use of ontological knowledge as a solution. However, the use of two attributes in the search process will be investigated as future work.

We observed that the fully automatic approach for detecting the entity labels using entropy performed satisfactorily in all dataset pair compared. No wrong label was selected in our evaluation.

We noticed that the accuracy of the NYTimes alignment is quite good, since it was manually curated, but it is not perfect. We encountered a few inconsistencies, and evidences of incorrect alignment in almost all pair of datasets. This fact led SERIMI to reach a non-optimal performance in this challenge. Below we show some examples of the inconsistence, incorrect and arbitrary judgment found in the reference alignment.

Label: Expedia Inc
•http://rdf.freebase.com/ns/en.expedia (reference alignment)
•http://rdf.freebase.com/ns/en.expedia_inc (SERIMI)
Label: USG Corporation
•http://dbpedia.org/resource/United_States_Gypsum (reference alignment)
•http://dbpedia.org/resource/USG_Corporation (SERIMI)
Label: Kirov Ballet
•http://rdf.freebase.com/ns/en.mariinsky_ballet (reference alignment)
•http://rdf.freebase.com/ns/en.kirov_ballet (SERIMI)

SERIMI took 40 minutes in average to compute the interlinking of an individual pair of dataset when it was performed under a controlled environment. In the case of DBPedia, its performance varied a lot due to the remote server avaiability.

3 General Comments

RDF instance matching is a challenging problem and the community has only recently started to develop a systematic framework to evaluate approaches to tackle this problem: the IMEI track of the OAEI initiative. We have however also encountered some problems in applying this framework to understand our results.

1. The accuracy of the reference alignment is critical point for the participants. Its quality prevents participants try to improve their precision, in cases where the reference alignment lacks in accuracy, or can be considered quite arbitrary, since there are dual interpretation in the alignment. We wasted a plenty of time to realize that the reference alignment was not 100% accurate, since we trusted on it beforehand. Therefore, we propose the organizers to warn the participants of lack of accuracy in the reference alignment, or whether possible, to publish some statistics about its accuracy.
2. Since DBPedia and Freebase contain a lot of duplicate entities associated to different URIs (e.g. http://rdf.freebase.com/ns/en.expedia_inc and http://rdf.freebase.com/ns/en.expedia), we propose the organizers to take this into consideration while computing the precision and recall of the participant results. Two participants may send distinct alignment results that are both correct.

Finally, we see an opportunity to ease the participation in the track. The preparation of the datasets is a non-trivial task, especially because they are large and available in different formats. Since all participants face the same problem here, it would be huge improvement whether the OAEI initiative could provide a SPARQL endpoint for all datasets mentioned in the challenge. All participants would work exactly over the same datasets, consequently increasing the credibility of the results. RDF database engines exist that allow text search via SPARQL endpoint with a quite high performance; and when used properly can support a large amount of requests, as demanded by a challenge of this scale.

4 Conclusion

In this paper, we present the results of SERIMI in OAEI 2011 Campaign's IMEI-DI track. We have presented the architecture of SERIMI system and described specific techniques used in this campaign. SERIMI matches instances between a source and target datasets, without prior knowledge of the data, domain or schema of these datasets. SERIMI solves the instance-matching problem in two phases. In the selection phase, it uses traditional information retrieval and string matching algorithms to select candidate resources for interlinking. In the disambiguation phase, it uses a novel approach to measure similarity between RDF resources and disambiguate the resources. The results illustrates that SERIMI can achieve good accuracy in instance matching track.

References

1. Bizer, C., Heath, T. and Berners-Lee, T. (2009) Linked Data - The Story So Far. Int. J. Semantic Web Inf. Syst., 5 (3). pp. 1-22.
2. Tejada, S.; Knoblock, C. A.; and Minton, S. (2001). Learning object identification rules for information integration. Information Systems 26(8): 607–633.
3. L. A. P. P. Leme, M. A. Casanova, K. K. Breitman, and A. L. Furtado. (2008). Evaluation of similarity measures and heuristics for simple RDF schema matching. Technical Report 44/08, Dept. Informatics, PUC-Rio.
4. Isaac A., Meij L. V. D., Schlobach S., and Wang S. (2007). An empirical study of instance-based ontology matching. In Proceedings of the 6th international semantic web and 2nd Asian conference on Asian semantic web conference (ISWC'07/ASWC'07), Springer-Verlag, Berlin, Heidelberg, 253-266.
5. K. K. Breitman, D. Brauner, M. A. Casanova, R. Milidi, A. Gazola, and M. Perazolo. (2008). Instance-Based Ontology Mapping. In Fifth IEEE Workshop on Engineering of Autonomic and Autonomous Systems (ease 2008), Belfast, Northern Ireland, pp. 67-74.
6. N. Choi, I.-Y. Song, and H. Han,. (2006). A survey on ontology mapping," ACM SIGMOD Record, vol. 35, no. 3, pp. 34-41.
7. Branting, L. K. (2003) A Comparative Evaluation of Name-Matching Algorithms. ICAIL '03, June 24-28, 2003, Edinburgh, Scotland, UK.
8. Araújo S., Hidders J., Schwabe S., and Vries A. P. de. (2011). SERIMI - Resource Description Similarity, RDF Instance Matching and Interlinking. CoRR, vol. abs/1107.1104.
9. Kopcke H., Thor A., and Rahm E. (2010). Evaluation of entity resolution approaches on real-world match problems. In Proceedings of the 3rd VLDB Endowment. Pp. 484-493.

Zhishi.links Results for OAEI 2011

Xing Niu, Shu Rong, Yunlong Zhang, and Haofen Wang

APEX Data & Knowledge Management Lab
Shanghai Jiao Tong University
{xingniu, rongshu, zhangyunlong, whfcarter}@apex.sjtu.edu.cn

Abstract. This report presents the results of **Zhishi.links**, a distributed instance matching system, for this year's Ontology Alignment Evaluation Initiative (OAEI) campaign. We participate in Data Interlinking track (DI) of IM@OAEI2011. In this report, we briefly describe the architecture and matching strategies of Zhishi.links, followed by an analysis of the results.

1 Presentation of the system

Ontology matching is a positive effort to reduce heterogeneity of semantic data. Both schema-level matching and instance-level matching aim at finding matches between semantically related entities of different ontologies. With the development of Linked Data[1], the needs of finding high quality `<owl:sameAs>` links are increasing. Thus, more and more people are engaged in research of identifying URIs referred to the same real-world object by using instance matching techniques.

Zhishi.links we proposed here is an efficient and flexible instance matching system, which utilizes distributed framework to index and process semantic resources, and uses scalable matching strategies for calculating similarities between two resources.

1.1 State, purpose, general statement

Zhishi.links is used for participating in Data Interlinking track (DI) of IM@OAEI2011. Its architecture is shown in Figure 1.

All the dumps (DBpedia, Freebase and GeoNames) are downloaded beforehand and the interconnection is done locally, even the datasets are very large. Admittedly, using keyword search or lookup services provided by these three chosen online database can help to obtain high quality match candidates, but this strategy relies too much on the retrieval performance. Moreover, a large quantity of datasets in Linked Data do not provide such kind of services.

In order to dramatically reduce the time complexity of matching procedures on large datasets, resources are indexed before a pipe of similarity calculations are performed. This principle is also adopted by Silk[6], a well-known links discovery framework. Resources are usually indexed by terms of their names and aliases. In other words, resource pairs sharing more than one term are treated as match candidates and wait for further checking.

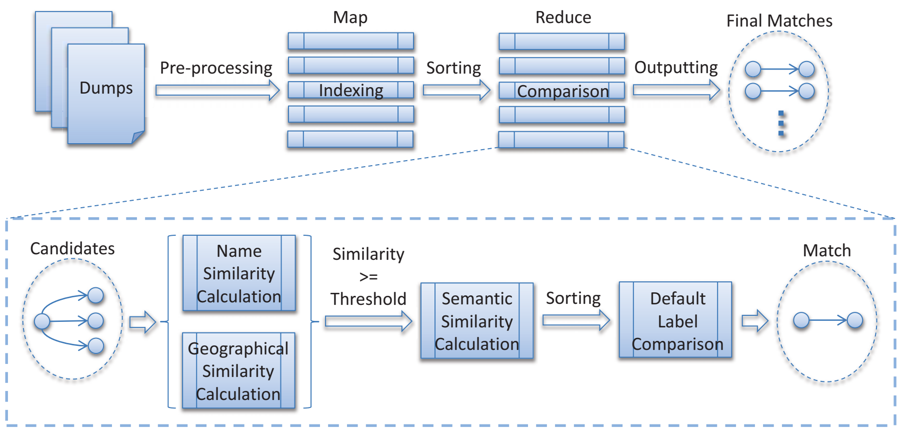


Fig. 1. Architecture of Zhishi.links

The comparison between two resources begins from the string similarity calculation. The more terms two resources' names share, the more similar they are. If a resource has more than one name (i.e. it has aliases), all names take part in similarity calculations and the highest similarity value is chosen. String similarities are used to filter out the least likely match candidates by setting a proper threshold.

Afterwards, semantic similarities are calculated. Generally speaking, similarity scores computed in the previous step increases if two resources have some sematic resemblances (e.g. the same property-value pairs), otherwise penalties are paid. Specifically, functional properties (`owl:FunctionalProperty`) and inverse-functional properties (`owl:InverseFunctionalProperty`) have higher weights than ordinary properties.

Finally, match candidates are sorted by their similarity scores. In some cases, two or more match candidates may have the same highest similarity scores. Zhishi.links compares their default labels again and chooses the closest matching pair.

1.2 Specific techniques used

Since these three datasets are huge, even though we adopt index-based pre-matching, we still suffer from high time and space complexities. We utilize distributed MapReduce[3] framework to accomplish this work. The map function produces a list of key-value pairs, where the keys are the index terms and the values are complete semantic descriptions of resources. After sorting, resources with the same index term (match candidates) gather together and further comparisons are made by reduce function in one computing node.

Properties unique to an object, such as inverse-functional properties, can be used to determine its identity[4]. Values of this kind of properties can also be used to filter or generate match candidates. Here we cite an example of using geographical coordinate to filter candidates before semantic similarity calculation. Most of the time, the geographical coordinate is unique to a location, but unfortunately, the data type of coordinates is a pair of floating point numbers and they can not be used as index terms. So as shown in Figure 1, we can calculate geographical similarity by computing distance between two coordinate. Intuitively, if two locations are wide apart, the corresponding match candidate should be filtered out.

1.3 Adaptations made for the evaluation

In Data Interlinking track, participates are asked to retrieve New York Times interlinks with DBpedia, Freebase and GeoMames. Structured data provided by New York Times is relatively scant. Usually, New York Times just provides a resource's name using `skos:prefLabel` property[1]. In some cases, a short definition (`skos:definition`) and a topic page's URL (`nyt:topicPage`) can be obtained. Thus, Virtual Documents are constructed for resources from other three data sources by splicing values of characteristic properties. Similarity between a Virtual Document and a topic page (abstract, articles' titles and snippets are included) is calculated in semantic similarity calculation phase.

Nearly all default names and aliases in these four data sources are well-designed. Many of them are appended disambiguation information (e.g. "Michael Mann (director)") or supplements (e.g. "University of California, Los Angeles"). Such phrases are isolated because 1. they can be treated as values of characteristic properties and used to calculate semantic similarities, and 2. they may bring about noise when the complete labels are used for string similarity calculation.

Beside these appended phrases, several special words in names are extracted for producing unified values of characteristic properties. For resources which are instances of "People", "Jr." and "Sr." are detected due to the reason that these words have good discriminability. For instances of "Locations" and "Organizations", more keywords are concerned and the full lists of these keywords are shown in Appendix section.

The "name" and "alias" we mentioned above refer to different properties in different data sources. Table 1 shows the exact properties that are used as "name" and "alias".

Table 1. Properties Used as "Name" and "Alias"

Data Source	Name	Alias
New York Times	`skos:prefLabel`	—
DBpedia	`rdfs:label`	`dbpedia-owl:wikiPageRedirects`
Freebase	`rdfs:label`	`fb:common.topic.alias`
GeoNames	`gn:name`	`gn:alternateName`

1.4 Link to the system and parameters file

The homepage of Zhishi.links is `http://apex.sjtu.edu.cn/apex_wiki/Zhishi.links`. More information about our instance matching system can be found here.

[1] Results of using `nyt:search_api_query` are unreachable for us.

1.5 Link to the set of provided alignments (in align format)

The results of Zhishi.links for IM@OAEI2011 can be found at `http://apex.sjtu.edu.cn/apex_wiki/Zhishi.links`.

2 Results

In this section, we will present the result of Zhishi.links for DI track in detail and give related analysis. Tests were carried out on a Hadoop computer cluster. Each node has a quad-core Intel Core 2 processor (4M Cache, 2.66 GHz), 8GB memory. The number of reduce tasks was set to 50.

2.1 DI-nyt-geonames

The version of GeoNames dump we used is May 4th.

The final results are shown in Table 2, where precision, recall and f-measure are provided. After filtering, 128,795 match candidates were sent to semantic similarity calculation component and approximately 98.9% (H_Recall) expected matches were include.

One reason for the notable decrease of recall is that URI aliases exist in this dataset: different URIs are used to identify the same location. For example, `gn:1863967`[2] and `gn:1863961`[3] refer to the same place in Japan: their names are both "Fukuoka" and their geographical coordinates are very close. We can hardly resolve this problem off-line except some official rules are provided.

Table 2. Performance of Zhishi.links (NYT-GeoNames)

Type	Precision	Recall	F-measure	Candidates	Expected	H_Recall
Locations	0.938	0.883	0.910	128,795	1,789	0.989

2.2 DI-nyt-dbpedia

The version of DBpedia dump we used is 3.6.

Zhishi.links performs best on DBpedia, as shown in Table 3. Wikipedia's strict and advanced naming conventions[4] and abundant aliases guarantee the quality and quantity of resources' names separately, which help a lot in similarity calculation phase.

However, precisions here are not satisfactory. We have investigated the causes and found that many incorrect matches occurred when ambiguities were existing. As explained in Section 1.3, New York Times does not provide sufficient structured descriptive data, hence more sophisticated matching methods should be applied.

[2] http://sws.geonames.org/1863967/

[3] http://sws.geonames.org/1863961/

[4] `http://en.wikipedia.org/wiki/Wikipedia:Naming_conventions`

Table 3. Performance of Zhishi.links (NYT-DBpedia)

Type	**Precision**	**Recall**	**F-measure**	**Candidates**	**Expected**	**H_Recall**
People	0.971	0.970	0.970	16,787	4,977	0.992
Organizations	0.896	0.932	0.913	10,679	1,965	0.957
Locations	0.910	0.914	0.912	47,490	1,920	0.983

2.3 DI-nyt-freebase

The version of Freebase dump we used is July 7th.

URI aliases also exist in Freebase. RDF dump built by RDFizer[5] does not contain URIs for resource. They should be generated by using values of `fb:type.object.key`. Unfortunately, this procedure produces more than one URI for a single resource. For example, `http://rdf.freebase.com/ns/en.amanda_hesser`, `http://rdf.freebase.com/ns/user.jamie.nytdataid.63892856178165632613`, `http://rdf.freebase.com/ns/wikipedia.en.Amanda_Hesser`, etc. all refer to a person named "Amanda Hesser".

Unlike GeoNames, only one URI for a resource is chosen in pre-processing phase. That is why the highest recalls (H_Recalls), as shown in Table 4, are also unsatisfactory. The priority list we used is:

1. http://rdf.freebase.com/ns/en.*
2. http://rdf.freebase.com/ns/user.jamie.nytdataid.*
3. http://rdf.freebase.com/ns/authority.us.gov.loc.na.n*
4. http://rdf.freebase.com/ns/authority.iso.*
5. http://rdf.freebase.com/ns/business.cik.*

If the URIs in reference alignments do not follow this priority list, mistakes would be unavoidable.

Table 4. Performance of Zhishi.links (NYT-Freebase)

Type	**Precision**	**Recall**	**F-measure**	**Candidates**	**Expected**	**H_Recall**
People	0.929	0.924	0.926	26,382	4,979	0.964
Organizations	0.887	0.853	0.870	12,664	3,044	0.889
Locations	0.902	0.865	0.883	14,705	1,920	0.932

3 General comments

In this section, we will give some additional comments on Zhishi.links results and provide some suggestions to OAEI organizers.

[5] `http://code.google.com/p/freebase-quad-rdfize/`

3.1 Discussions on the way to improve the proposed system

Several shortcomings of Zhishi.links can be seen and need to be overcome in the future:

- When it comes with the problem of homonyms, instance matching systems should exploit as much information as possible to enhance the discriminability of their matchers. Currently, subject to the fact that most descriptions given by New York Times are written in natural language, the performance of our semantic similarity calculator are constrained. We are considering more tests carrying out on datasets in different styles and designing a more robust system.
- In DI track, only three types of resources are involved. The special words in names, which are extracted as values of characteristic properties, are chosen manually. Some smarter strategies should be applied to accomplish this mission.

3.2 Comments on the OAEI 2011 test cases

We are very interested in testing our matching system on large-scale real-world data. It can help validating the robustness and applicability of the proposed methods. However, crude raw data may have some defects. The URI aliases problem, for example, is what we have met. We hope that these issues are resolved in the future or considered in the evaluation.

3.3 Proposed new measures

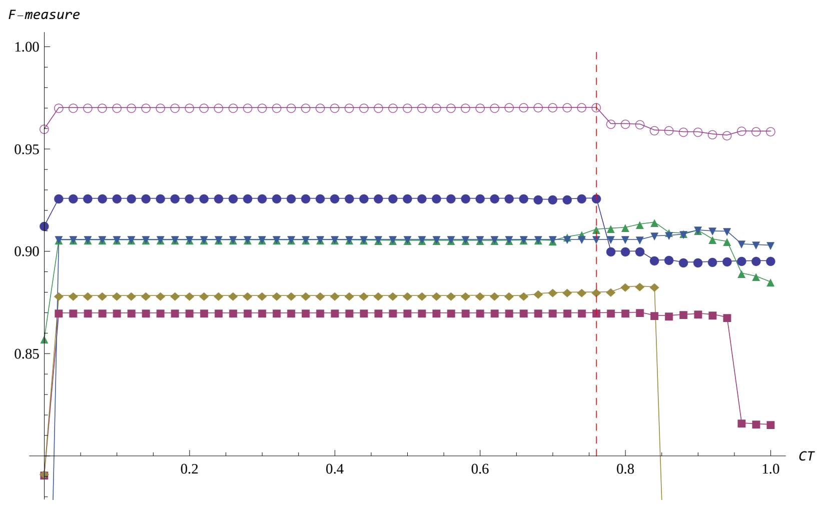


Fig. 2. F-measures on Different Confidence Thresholds

Our goal is to design a general instance matching system. It should not sensitive to what type of resources should be matched and what data source resources come from. So what we need is a matching method with high stability.

As Shown in Figure 2, we choose match candidates above continuously varying *confidence threshold*s (*CT*) and plot the corresponding *F-measure*s on the chart. The six curves in this chart indicate the results for matching tasks carried out on DBpedia and Freebase. We can determined a fixed CT value to filter final match candidates. For instance, we can choose $CT = 0.76$ here. Then the performances are not the best, but are relative acceptable (other matching systems are not taken into consideration).

Here we just mentioned the concept of **stability**. The complete descriptions for evaluating the stability of matching systems are elaborate in [5].

4 Conclusion

In this report, we have presented a brief description of Zhishi.links, an instance matching system. We have introduced the architecture of our system and specific techniques we used. Also, the results have been analyzed in detail and several guides for improvements have been proposed. We look forwards to build an instance matching system with better performance and higher stability in the future.

References

1. Bizer, C., Heath, T., Berners-Lee, T.: Linked Data - The Story So Far. Int. J. Semantic Web Inf. Syst. 5(3), 1–22 (2009)
2. Bouquet, P., Stoermer, H., Tummarello, G., Halpin, H. (eds.): Proceedings of the WWW2007 Workshop I^3: Identity, Identifiers, Identification, Entity-Centric Approaches to Information and Knowledge Management on the Web, Banff, Canada, May 8, 2007, CEUR Workshop Proceedings, vol. 249. CEUR-WS.org (2007)
3. Dean, J., Ghemawat, S.: MapReduce: Simplified Data Processing on Large Clusters. In: OSDI. pp. 137–150 (2004)
4. Hogan, A., Harth, A., Decker, S.: Performing object consolidation on the semantic web data graph. In: Bouquet et al. [2]
5. Niu, X., Wang, H., Wu, G., Qi, G., Yu, Y.: Evaluating the stability and credibility of ontology matching methods. In: Antoniou, G., Grobelnik, M., Simperl, E.P.B., Parsia, B., Plexousakis, D., Leenheer, P.D., Pan, J.Z. (eds.) ESWC (1). Lecture Notes in Computer Science, vol. 6643, pp. 275–289. Springer (2011)
6. Volz, J., Bizer, C., Gaedke, M., Kobilarov, G.: Discovering and Maintaining Links on the Web of Data. In: Bernstein, A., Karger, D.R., Heath, T., Feigenbaum, L., Maynard, D., Motta, E., Thirunarayan, K. (eds.) International Semantic Web Conference. Lecture Notes in Computer Science, vol. 5823, pp. 650–665. Springer (2009)

Appendix

Table 5. Unified Values for Organizations

Keywords	Unified Values
Co, Company	Co.
Corp, Corporation	Corp.
Inc, Incorporated	Inc.
Ltd, Limited	Ltd.

Table 6. Unified Values for Locations

Keywords	Unified Values
North Carolina	NC
New Hampshire	NH
New Jersey	NJ
New York	NY
Bronx, Brooklyn, Manhattan, Queens	NYC
Rhode Island	RI
Florida	Fla
Georgia	Ga
Louisiana	La
Maine	Me
Mississippi	Miss
Missouri	Mo
Pennsylvania	Pa
Virginia	Va
Vermont	Vt

YAM++ – Results for OAEI 2011*

DuyHoa Ngo, Zohra Bellahsene, Remi Coletta

University Montpellier 2, France
{firstname.lastname@lirmm.fr}

Abstract. The YAM++ system is a self configuration, flexible and extensible ontology matching system. The key idea behind YAM++ system is based on machine learning and similarity flooding approaches. In this paper, we briefly present the YAM++ and its results on Benchmark and Conference tracks on OAEI 2011 campaign.

1 Presentation of the system

Ontology matching is needed in many application domains, especially in the emergent semantic web field. It is considered as the most promising approach to solve the interoperability problems across heterogeneous data sources. Due to the various types of heterogeneity of ontologies, a matching system, generally, exploits many features of elements in ontology and combine several similarity metrics in order to improve its performance. However, finding a good combination is very difficult and time consuming even for experts. Therefore, we propose YAM++ - a (not) Yet Another Matcher approach, which firstly uses a machine learning technique to combine similarity metrics. We then apply a similarity propagation algorithm [5] to discover more semantic mappings. The advantages of using machine learning and similarity flooding techniques will make our system flexible and extensible.

1.1 State, purpose, general statement

The current implemented version YAM++ is an extension of our former schema matching system YAM [1]. However, the YAM++ aims to work with ontology matching, which is semantically richer than XML schema, so new features were added in the extension version. The main components of the new system are depicted in Figure 1.

The fully automated mapping process is performed as follows. Firstly, two to-be-matched ontologies are passed to the Parsing & Processing module. We use OWLAPI [1] and Pellet [2] to load ontology files into our specified data structure which contains all elements with their annotation and relationships in ontology. Next, the loaded ontologies are passed into Element-level matcher. The aim of this module is discovering as many as possible mappings and as high as possible the accuracy of these mappings by analyzing elements in isolation and ignoring their relations with others. This module consists of

* Supported by ANR DataRing ANR-08-VERSO-007-04.

[1] http://owlapi.sourceforge.net/

[2] http://clarkparsia.com/pellet

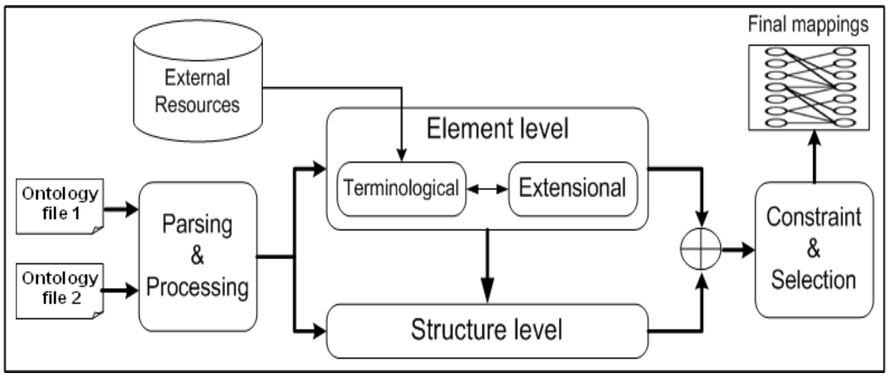


Fig. 1. YAM++ system architecture

two sub-modules: Terminological matcher and Extensional matcher. The Terminological matcher exploits annotation information of elements by various similarity metrics, and then combine them by a machine learning model. Whereas, the Extensional matcher exploits information from external data (instances) accompanying with ontologies.

Mappings discovered from Element-level matcher are passed into the Structure-level matcher module in order to discover new mappings by analyzing the positions of elements on the hierarchy structure of ontologies. The main intuition of this part is that if two elements from two ontologies are similar, their neighbors (in the same relations) might also be somehow similar [2]. In our current system, we have implemented a popular graph based technique - Similarity Flooding in the Structure-level matcher module.

Finally, the results of Element-level and Structure-level matchers are combined and passed to the Constraint & Selection module. Here, we define some semantic patterns to recognize and remove inconsistent mappings. Then, we apply assignment methods (Greedy and Hungarian algorithms) [4] to select the most appropriate mappings. Currently, our system works with 1:1 matching cardinality.

1.2 Specific techniques used

In this section, we will briefly describe four main modules: Terminological matcher, Extensional matcher, Similarity Flooding and Constraint & Selection.

Terminological matcher In this module, we exploit various features of text information used to annotate elements in ontologies. The corresponding similarity metrics have been also implemented in order to calculate the similarity score between elements from two to-be-matched ontologies. The combination and the matching process in this module are illustrated in Figure 2.

We treat the matching task as a binary classification problem. Each pair of elements from two input ontologies are considered as a machine learning object; its attributes are similarity scores calculated by a set of selected similarity metrics on these elements.

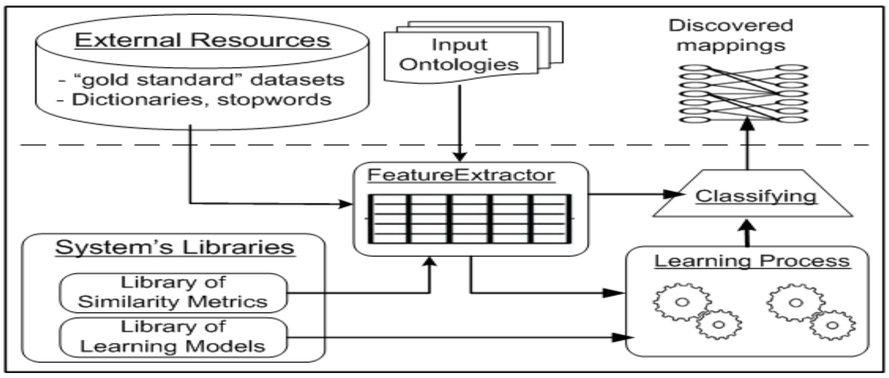


Fig. 2. Terminological matcher

In the learning phase, we use some **gold standard** datasets to generate training data. A **gold standard** is a pairs of ontologies with expert mappings between their elements provided by domain experts. In current version, the gold standard datasets are taken from Benchmark dataset published in OAEI 2009. According to our study [6] [3] , we adopt J48 - a decision tree [9] as the classification model in our system. In classifying phase, each pair of elements from two to-be-matched ontologies is predicted as matched or not according to its attributes.

In order to extract attribute values for each pair of elements, we use various similarity metrics described in [6]. They are divided into three main groups: (i) **name metrics** - which compare entities by their URI names; (ii) **label metrics** - which compare entities by their labels; and (iii) **context metrics** - which compare entities by their descriptive and context information.

Name and label metrics belong to Terminological category [2]. Most of String-based metrics (Levenstein, SmithWaterman, Jaro, JaroWinkler, MongeEklan,etc.) are taken from open source libraries SecondString [4] and SimMetrics [5]. We also have implemented popular metrics such as Prefix, Suffix, LCS [2], SubString - Stoilos [8]. In term of Language-based metrics, we developed three well-known metrics corresponding to Lin, JingConrath and WuPalmer algorithms [3]. These metrics use Wordnet [6] as an auxiliary local thesaurus. Additionally, we proposed a hybrid approach to combine string-based and language-based metrics. The detail of algorithms can be found in our paper [7].

Context metrics used in YAM++ belong to the both Structural and Extensional categories [2]. The main idea behind these metrics is described as follows. We build three types of context profile (i.e. IndividualProfile, SemanticProfile and ExternalProfile) for each element. Whereas IndividualProfile describes annotation information of individual element, SemanticProfile and ExternalProfile describe the context information of an

[3] http://www2.lirmm.fr/∼dngo/publications/A_Flexible_System_for_Ontology_Matching_full.pdf
[4] http://secondstring.sourceforge.net/
[5] http://sourceforge.net/projects/simmetrics
[6] http://wordnet.princeton.edu

element among the others. These context profiles then can be used to compare entities by some information retrieval techniques. See [7] for more detail.

Extensional matcher In many situation, to-be-matched ontologies provide data (instances), therefore, the aim of Extensional module is to discover new mappings which are complement to the result obtained from Terminological module. There are two issues we should consider here: (i) how to find similar instances from two ontologies; and (ii) given a list of matched instances, how to discover new concept/property mappings.

For the first issue, we propose two methods for discovering instance mappings as follows:

- If two instances belong to two matched classes and they have highly similar labels then they will be considered as similar. Here, the list of concept mappings is taken from the result of Terminological module. Similarity score between instance labels is computed by our metric named SentenceSimilarity.
- If two instances have high similar text in description then they will be considered as similar too. Here, a description text is taken from values of all properties described in an instance. The intuition behind is that even though a real instance were assigned by different name of IDs, properties and concepts (classes), but the values of its properties are generally not changed. Based on this idea, we use an information retrieval technique to compute the similarity of instances by their description text.

For the second issue, we apply two methods for discovering concept/property mappings as follows:

- For each pair of matched instances from two ontologies, if the text values of two properties are highly similar, then these properties are considered as similar.
- For each pair of concepts (classes) from two ontologies, if most of their instances are matched, then these classes are matched.

The discovered mappings by Element-level module are the union of discovered mappings from Terminological and Extensional modules.

SimilarityFlooding The Similarity Flooding algorithm is well-known and were described in detail in the popular schema matching system - SimilarityFlooding [5]. In this section, instead of explaining this algorithm again, we will present some main points of using it in our system. For this aim, we are going to answer two questions: (i) why do we select Similarity Flooding method; and (ii) how can we apply this method in ontology matching task.

Firstly, according to [2], there are many methods can be used to analyze the structural information of ontologies to find mappings. They are mainly based on the idea that two elements of two different ontologies are similar if all or most of their elements in the same **view** are already similar. The term **view** here maybe understood as a list of elements which can be seen from the ongoing examined element. For example, they can be direct super/sub elements, sibling elements or list of elements in the path from the root to the current element on the ontology's hierarchy, etc. However, two big problems have arisen. One as the authors commented in [2] is that these methods face problems when the viewpoint of two ontologies is highly different. The second is that if some pre-defined mappings are incorrect and these methods are run only one time to produce new mappings, then the accuracy of new results will be unconfident. Therefore, we believe that the idea of propagating a portion of similarity from one matched pair of elements to their neighbors is more flexible and applicable. Additionally, by running many iterations until the similarity scores between elements become stable, we believe that incorrect mappings (pre-defined) will be removed.

In order to apply Similarity Flooding in our system, we construct a graph from an ontology, whose nodes are concepts, properties or primitive XML Schema DataTypes and edges have types and label have been divided into 5 groups corresponding to relationships between elements in ontology, such as **subClass**, **subProperty**, **domain**, **range** and **onRestrictedProperty**. Notice that we use Pellet library as a reasoner to infer the relations between elements in ontology. It will help us to overcome the problem of different viewpoint of ontologies. Then, a pairwise connectivity graph is built in the same way that were described in the SimilarityFlooding system. We use **inverse product** formula to computing the propagation coefficient and **fixpoint formula C** [5] in the similarity propagation process.

Constraint and Selection The mappings discovered from Element-level and Structure-level are joined and passed to this module in order to remove inconsistent mappings. We combine results of Element and Structure level by a **dynamic weighted aggregation** method. The idea is as follows:

– After Similarity Flooding stop, we run Double-Hungarian assignment method in order to find the two most similar elements from the target ontology for each element in the source ontology. We call the result of this step by SMap - a list of possible mappings.
– From the list mappings obtained from Element-level matcher (we call it EMap), we find a mapping which also exists in the SMap but with the lowest score. We call it a $Threshold$ value. This threshold will be used as weight for all mappings in EMap. The weight for all mappings in the SMap is equal to $(1 - Threshold)$.
– The final mappings are produced by weighted aggregation of EMap and SMap. After that, we use Greedy Selection algorithm to extract the most stable mappings whose similarity score higher than the Threshold found above.

Next, in order to perform Semantic verification, in the current version of our system, we define two simple patterns to recognize the inconsistent mappings as follows:

– If two properties are discovered as similar but none of the classes in their domain are mapped and none of the classes in their range are mapped neither, then the mapping of these two properties are inconsistent and will be removed.
– If classes $(A, B), (A, B_1), (A_1, B)$ are matched and A_1 and B_1 are subsumed of A and B respectively, then $(A, B_1), (A_1, B)$ are removed.

1.3 Adaptations made for the evaluation

There are two factors that directly impact to the our system's performance. The first relates to matching by machine learning model. The training data and selected similarity metrics as learning attributes are important. We proposed a simple solution for this issue by selecting the most appropriate similarity metrics and training data according to their correlation with experts assessment. For more detail, see our paper [7]. The second issue relates to the threshold used as a filter in the selection module. Different tests require different thresholds. Our experiments show that if we set low threshold, we need to use strict semantic constraint to remove inconsistent mappings. In the current version of system, we propose a simple heuristic method (i.e., dynamic threshold and dynamic weighted aggregation) to deal with this problem. Besides, it satisfies the rules of the contests that all alignments should be run by the same configuration.

1.4 Link to the system and parameters file

The YAM++ system and parameter files can be download at:http://www2.lirmm.fr /∼dngo/ YAMplusplus.zip. See the instructions from SEALS platform [7] to test our system.

1.5 Link to the set of provided alignments (in align format)

The results on **Benchmark** and **Conference** tracks can be downloaded at: http://www2.lirmm.fr /∼dngo/ YAMplusplus_oaei2011.zip.

2 Results

In order to see the performance of our system, we will show the effectiveness and the impact of the listed modules above by comparing the results obtained from running system with three configurations. The basic configuration is the Terminological matcher only, which is based on a machine learning (**ML**) approach. Next, we extend it with the Extensional module, which is an Instance based matcher (**IB**), in order to have a full Element-level matcher. Finally, we add the Similarity Flooding (**SF**) module to the third configuration. All experiments are executed with JRE 6.0 on Intel 3.0 Pentium, 3Gb of RAM, Window XP SP3.

2.1 Benchmark

The Benchmark 2010 track includes 111 tests. Each test consists of source (reference) ontology and a test ontology, which is created by altering some information from the reference. The reference ontology contains 33 named classes, 24 object properties, 40 data properties, 56 named individuals and 20 anonymous individuals. The evaluation of our system's performance on this track with 3 configurations is shown in the Table 1.

The observation from this track is as follows. By using only machine learning (**ML**) approach, YAM++ achieved good result with very high precision (**0.99**) and F-Measure

[7] http://oaei.ontologymatching.org/2011/seals-eval.html

Configuration	**Precision**	**Recall**	**F-Measure**
ML	0.99	0.72	0.84
ML + IB	0.99	0.74	0.85
ML + IB + SF	0.98	0.84	0.90

Table 1. H-mean values on Benchmark 2010 track

(**0.84**). After adding Instance based (**IB**) matcher, both Recall and F-Measure increased with **2%** and **1%** respectively. These improvements were happened because many ontologies have common extensional data (instances). Finally, thanks to the process of propagation of similarity, both Recall and F-Measure increased with **10%** and **5%** respectively.

The Benchmark 2011 track includes 103 tests. Similar to Benchmark 2010, in this track, a source (original) ontology is compared to target ontologies which were obtained by altering some features from the original. The reference ontology contains 74 named classes, 33 object properties without extensional data (instances). The evaluation of our system's performance on this track with 3 configurations is shown in the Table 2.

Configuration	**Precision**	**Recall**	**F-Measure**
ML	0.98	0.51	0.67
ML + IB	0.98	0.51	0.67
ML + IB + SF	0.97	0.60	0.74

Table 2. H-mean values on Benchmark 2011 track

Similar to the Benchmark 2010 track, using Similarity Flooding method increases both Recall and F-Measure values with **9%** and **7%** respectively. The Instance based matcher did not improve the performance because in this track, ontologies don't support extensional data. That is why the matching results of running the first and the second configurations are the same.

2.2 Conferences

Conference track now contains 16 ontologies from the same domain (conference organization) and each ontology must be matched against every other ontology. YAM++ is able to produce all 120 alignments from those ontologies. Due to the heterogeneity of those ontologies, finding mappings between them is more difficult than that in Benchmark tracks. Besides, this track is an open+blind test because there are no reference alignments for most of those tests. In the Table 3, we can only report our results with respect to the available reference alignments. The observation on this track is similar to the Benchmark2011 track. Here, thanks to Similarity Flooding method, all of Precision, Recall and F-Measure values are improved.

3 General comments

This is the first time we participate to the OAEI campaign. We found that SEALS platform is a very valuable tool to compare the performance of our system with the others.

Configuration	**Precision**	**Recall**	**F-Measure**
ML	0.75	0.50	0.60
ML + IB	0.75	0.50	0.60
ML + IB + SF	0.78	0.56	0.65

Table 3. H-mean values on Conference 2011 track

Besides, we also found that OAEI tracks covers a wide range of heterogeneity in ontology matching task. They are are very useful to help developers/researchers to develop their semantic matching system.

3.1 Comments on the results

Generally, according to the results published in last competition OAEI 2010, our system is acceptable and comparable with other participants. However, there are still some weaknesses in our current system such as (i) semantic verification/constraint issue; (ii) problem of dealing with large scale ontology matching task and time performance. In the current version, we provided some simple solutions for these issues above. The preliminary results were quite good to encourage us to continue seeking better solutions.

4 Conclusion

In this paper, we present the overview of the YAM++ system and our results on Benchmark and Conference tracks. Our experiments prove that the combination of machine learning and similarity flooding approaches bring good results. Additionally, YAM++ system is fully automate, flexible and extensible.

References

1. Fabien Duchateau, Remi Coletta, Zohra Bellahsene, and Renée J. Miller. Yam: a schema matcher factory. In *CIKM Conference*, pages 2079–2080, 2009.
2. Jérôme Euzenat and Pavel Shvaiko. *Ontology matching*. Springer-Verlag, Heidelberg (DE), 2007.
3. Feiyu Lin and Kurt Sandkuhl. A survey of exploiting wordnet in ontology matching. In *IFIP AI*, pages 341–350, 2008.
4. C. Meilicke and H. Stuckenschmidt. Analyzing mapping extraction approaches. In *In Proc. of the ISWC 2007 Workshop on Ontology Matching, Busan, Korea*, 2007.
5. Sergey Melnik, Hector Garcia-Molina, and Erhard Rahm. Similarity flooding: A versatile graph matching algorithm and its application to schema matching. In *ICDE*, pages 117–128, 2002.
6. DuyHoa Ngo, Zohra Bellahsene, and Remi Coletta. A flexible system for ontology matching. In *Caise 2011 Forum*, 2011.
7. DuyHoa Ngo, Zohra Bellasene, and Remi Coletta. A generic approach for combining linguistic and context profile metrics in ontology matching. In *ODBASE Conference*, 2011.
8. Giorgos Stoilos, Giorgos B. Stamou, and Stefanos D. Kollias. A string metric for ontology alignment. In *ISWC Conference*, pages 624–637, 2005.
9. Ian H. Witten and Eibe Frank. *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*. Morgan Kaufmann, October 1999.

Towards more Challenging Problems for Ontology Matching Tools

Ernesto Jiménez-Ruiz[⋆] and Bernardo Cuenca Grau

Department of Computer Science, University of Oxford, UK
{ernesto,berg}@cs.ox.ac.uk

Abstract. We motivate the need for challenging problems in the evaluation of ontology matching tools. To address this need, we propose mapping sets between well-known biomedical ontologies that are based on the UMLS Metathesaurus. These mappings could be used as a basis for a new track in future OAEI campaigns.

1 Motivation and Background

The 2011 OAEI campaign consists of six different tracks. The so-called *Anatomy track* involves the largest test ontologies (containing between 2000-3000 classes).

Ontology matching tools have significantly improved in the last few years and there is a need for more challenging and realistic matching problems [1, 2] for which suitable "gold standards" exist.

There has been a long-standing interest within the bio-informatics research community in integrating thesauri, taxonomies and (more recently) also ontologies. The development of the UMLS-Metathesaurus (UMLS), which is currently the most comprehensive effort for integrating medical thesauri and ontologies, has been a very complex process combining automated techniques, expert assessment, and sophisticated auditing protocols [3–5].

2 Our Proposal

Although the standard UMLS distribution does not directly provide sets of "mappings" (in the OAEI sense) between the integrated ontologies, it is relatively straightforward to extract mapping sets from the information provided in the distribution files (e.g., see [6] for details).

Since UMLS-Meta integrates many widely used large-scale ontologies, such as FMA, NCI, SNOMED CT, or MeSH, we believe that the UMLS mappings between these ontologies could be used as a basis for a new track within the OAEI initiative. It has been noticed, however, that although these mappings have been manually curated by domain experts, they lead to a significant number of logical inconsistencies when integrated with the corresponding source ontologies (e.g., the integration of SNOMED CT and NCI via UMLS mappings leads to more than 20,000 unsatisfiable classes, as shown in Table 1).

[⋆] Ernesto Jimenez-Ruiz is supported by the EPSRC project LogMap

Ontologies	Original Mappings	Inconsistencies	Clean Mappings
FMA-NCI	3,024	655	2,898
FMA-SNOMED	9,072	6,179	8,111
SNOMED-NCI	19,622	20,944	18,322

Table 1. Repairing UMLS mappings (see [7])

To address this problem, we have presented in [6] and [7] several refinements of the UMLS mappings that do not lead to such inconsistencies. The mappings in [7] represent a larger subset of the UMLS-mappings than those in [6] as they were generated using "less aggressive" ontology repair techniques (see Table 1).

These "clean" subsets of UMLS mappings are readily available and could be used as reference alignments for a new, more challenging track within the OAEI (see `http://www.cs.ox.ac.uk/isg/projects/LogMap/`). In order to turn these reference alignments into a agreed-upon gold standard, some additional effort would be needed (e.g., manual curation). Another possibility would be to construct a "silver standard" by "harmonising" the UMLS mappings with the outputs of different matching tools over the relevant ontologies; similar silver standards have been developed for named entity recognition problems [8].

Although the use in an OAEI track of ontologies such as SNOMED CT, FMA and NCI represents a significant leap in complexity w.r.t. the existing anatomy track (from several million candidate mappings to several *billion*), we have recently developed a new matching tool, called LogMap [7], that is able to efficiently match these ontologies. We take our positive experiences with LogMap as an indication that a new track based on large-scale realistic ontologies and UMLS-mappings is not only feasible, but also potentially of great value, both for the developers of matching tools and the bio-informatics research community.

References

1. Shvaiko, P., Euzenat, J.: Ten challenges for ontology matching. In: On the Move to Meaningful Internet Systems (OTM Conferences). (2008)
2. Euzenat, J., Meilicke, C., Stuckenschmidt, H., Shvaiko, P., Trojahn, C.: Ontology Alignment Evaluation Initiative: six years of experience. J Data Semantics (2011)
3. Bodenreider, O.: The Unified Medical Language System (UMLS): integrating biomedical terminology. Nucleic acids research **32** (2004)
4. Cimino, J.J., Min, H., Perl, Y.: Consistency across the hierarchies of the UMLS semantic network and metathesaurus. J of Biomedical Informatics **36**(6) (2003)
5. Geller, J., Perl, Y., Halper, M., Cornet, R.: Special issue on auditing of terminologies. Journal of Biomedical Informatics **42**(3) (2009) 407–411
6. Jiménez-Ruiz, E., Cuenca Grau, B., Horrocks, I., Berlanga, R.: Logic-based assessment of the compatibility of UMLS ontology sources. J Biomed. Sem. **2** (2011)
7. Jiménez-Ruiz, E., Cuenca Grau, B.: Logmap: Logic-based and scalable ontology matching. In: 10th International Semantic Web Conference. (2011) 273–288
8. Rebholz-Schuhmann, D., et al.: CALBC Silver Standard Corpus. J Bioinform Comput Biol. (2010) 163–179

A Framework for Session-based Ontology Alignment

Patrick Lambrix

Department of Computer and Information Science
Linköping University, 581 83 Linköping, Sweden

In this abstract we tackle the problem of aligning large ontologies where the mappings suggested by the ontology alignment system need to be validated. Although we focus on an ontology alignment framework, the ideas may be used and extended for community-based or collaborative ontology alignment.

In contrast to the case of small ontologies, the computation of mapping suggestions can take a long time and therefore, we would like to be able to start the validation before every mapping suggestion is computed. Further, it is clear that for large ontologies, in general, there are too many mapping suggestions to validate in one time. Therefore, we want a system that allows to partially validate the mapping suggestions and resume the validation later. However, whenever validation decisions have been made, they increase our knowledge about the ontologies and mappings and this knowledge can be used to provide better mapping suggestions. In the remainder of the abstract we propose an iterative ontology alignment framework that deals with these issues.

Framework. Our framework is presented in figure 1. The input to the system are the ontologies that need to be aligned, and the output is an alignment between the ontologies. When starting an alignment process the user starts a computation session. When a user returns to an alignment process, she can choose to start or continue a computation session or a validation session.

During the *computation sessions* mapping suggestions are computed. The computation may involve preprocessing of the ontologies, matching, and combination and filtering of matching results. Auxiliary resources such as domain knowledge and dictionaries may be used. Users may be involved in the choice of algorithms. This is similar to what most ontology alignment systems do. However, in this case the algorithms may also take into account the results of previous validation and recommendation sessions. The output of a computation session is a set of mapping suggestions. The computation sessions can be stopped and partial results can be delivered.

During the *validation sessions* the user validates the mapping suggestions generated by the computation sessions. The output of a validation session is a set of mapping decisions (accepted and rejected mapping suggestions). The accepted mapping suggestions form a partial reference alignment (PRA) and are part of the final alignment. The mapping decisions can be used in future computation sessions (e.g. PRA-based preprocessing and filtering [1]) as well as in recommendation sessions. Validation sessions can be stopped and resumed at any time. It is therefore not neccesary for a domain expert to validate all mapping suggestions in one session. The user may also decide not to resume the validation but start a new computation session, possibly based on the results of a recommendation session.

The input for the *recommendation sessions* consists of a database of algorithms for the preprocessing, matching, combination and filtering in the computation sessions.

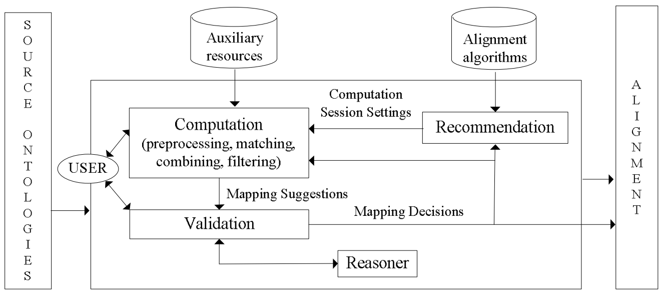


Fig. 1. Framework.

During the recommendation sessions the system computes recommendations for which (combination) of those algorithms may perform best for aligning the given ontologies. When validation results are available these may be used to evaluate the different algorithms, otherwise an oracle may be used. The output of this session is a recommendation for the settings of a future computation session. These sessions are normally run when a user is not validating and results are given when the user logs in into the system again.

Current implementation. We have implemented a prototype based on the framework described above. Regarding the computation sessions, when a PRA is available, the ontologies are preprocessed to partition the ontologies into corresponding mappable parts [1]. For the matching we use the linguistic, WordNet-based, structural and instance-based algorithms from the SAMBO system [2], a weighted sum approach for the combination, and the single and double threshold filter approaches. When a PRA is available we also use the filter approaches from [1]. Users can choose which algorithms, weights and tresholds to use, or use default values. For the validation we use the user interface of SAMBO where a user can accept, reject or modify mapping suggestions as well as annotate the decisions. A reasoner is used to detect conflicts in the decisions and notify the user. Validation sessions can be stopped at any time and resumed later on (if so desired - the user may also start a new computation session). The recommendation algorithm is based on the algorithm described in [3]. Currently, the performance of the different alignment algorithms is evaluated based on how well they do on small pieces of the ontologies already aligned by an oracle. In the future, we will also take the validation decisions of the user into account and adapt the recommendation.

Acknowledgements. We thank Qiang Liu, Muzammil Zareen Khan and Shahab Qadeer for their implementation work in this project.

References

1. P Lambrix and Q Liu. Using partial reference alignments to align ontologies. In *6th ESWC, LNCS 5554*, pages 188–202, 2009.
2. P Lambrix and H Tan. SAMBO - a system for aligning and merging biomedical ontologies. *Journal of Web Semantics*, 4(3):196–206, 2006.
3. H Tan and P Lambrix. A method for recommending ontology alignment strategies. In *6th ISWC and 2nd ASWC, LNCS 4825*, pages 494–507, 2007.

Modeling Matching Systems using Matching Process Design Patterns

Eric Peukert

SAP Research, 01187 Dresden, Germany
eric.peukert@sap.com

1 Introduction

Many schema- and ontology matching systems were developed to compute mapping suggestions for a user. Most of these systems are black boxes that often re-implement basic matching components which are extended by a few domain specific matchers. We observe that most systems mainly differ in their internal execution order and combination of matchers.

In this paper, we advertise using a matching process model to unify a broad set of different matching systems. That allows making the order of execution within a matching system explicit. Moreover, we identify a set of so called matching process design patterns that are often used and combined to build strong matching systems.

2 Process Model and Design Patterns

A *matching process* is represented by a directed matching process graph as was also proposed by [2]. The vertices represent operations and edges determine the execution order and data flow. The result of a matching process is a mapping *MA* between a source schema *S* and a target schema *T* that consists of correspondence links between schema elements. With a *schema* we refer to any meta data structure such as trees, ontologies, or meta models. Mappings are computed with the help of *similarity matrices* that contain similarity values between schema source and target elements. Additionally, we introduce a so called *comparison matrix*. A comparison matrix consists of $|S| * |T|$ cells. Each cell contains a boolean value representing whether a comparison within subsequent matching operations should be performed. The comparison matrix is crucial for controlling the flow of element comparisons within a matching process. Our set of operators is based on the operators we introduced in [1] that are Match, Combine, Select, Filter, Input and Output. *Match* computes a similarity matrix using some matching algorithm. *Combine* aggregates multiple matrices and *Select* reduces the matrix to most likely mapping candidates. Filter is used to reduce the number of comparisons for subsequent operations by setting boolean values in the comparison matrix. Additionally we introduce a *Condition* to allow conditional execution of process parts and *Split/Loop* to model processes of systems like Falcon or RiMOM [3, 5]. Based on these operators we are able to model a variety of matching systems internal matching processes using the framework and tools described in [1]. Moreover, we were able to identify an initial set of reusable matching process design patterns that are often used and combined to build strong matching processes (see Figure 1).

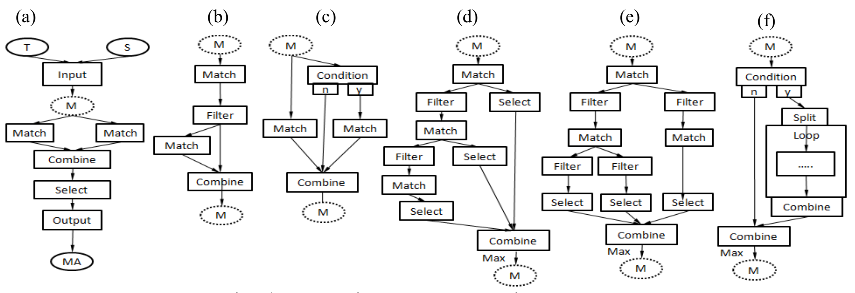


Fig. 1. Matching Process Design Patterns

Parallel Composition (a) is often applied to combine a set of matching algorithms and was introduced in [4]. *Refinement Sequence* (b) tries to increase precision by refining the results of a matcher within subsequent matchers in a process. *Adaptive Matcher Selection* (c) is often used to select the most appropriate matcher for a given matching problem based on some pre-computed feature value. The *Skimming* pattern (d) extracts the most probable correspondences from every matcher individually. These correspondences are “skimmed”. This approach is useful if individual matchers have a high precision for a domain of mapping problems. *Divide and Conquer* (e) divides the set of comparisons based on some property and distributes these comparisons to the most appropriate, possible different matchers. This pattern is extensively used in decision tree based matching systems. Finally *Blocking&Clustering* is applied in systems that repeatedly execute process parts. A typical application is the fragmentation of the matching task into smaller blocks that are executed independently. In addition to the visualized patterns we propose two further patterns that are *Iteration* and *Matcher Hierarchies.* Iteration repeatedly executes process parts until a given condition is met. *Matcher Hierarchies* are implicitly used by many matching systems to build complex structure-based matchers. Within that pattern, the output of a matcher is directly used as input for a second matcher.

In our evaluation we were able to show that the parallel composition pattern behaves very robust to solve different matching problems with high quality. However, by combining the pattern with skimming and refinement parts the quality can further be improved as was implicitly done in the internal process of Falcon and RiMOM.

References

1. Peukert, E., Eberius, J., Rahm, E.: AMC - A Framework for Modeling and Comparing Matching Systems as Matching Processes. ICDE (2011)
2. Lee, Y. et. al.; eTuner: Tuning Schema Matching Software Using Synthetic Scenarios. The VLDB Journal, 16(1), (2007)
3. Li, J. et. al.: RiMOM: A Dynamic Multistrategy Ontology Alignment Framework. IEEE Transactions on Knowledge and Data Engineering, 21(8), (2009)
4. Do, H. H. and Rahm, E.: COMA - A System for Flexible Combination of Matching Approaches. VLDB (2002)
5. Hu, W. and Qu. Y.: Falcon-AO: A Practical Ontology Matching System. Web Semant., 6(3), (2008)

A Visual SOA-based Ontology Alignment Tool

Weng Onn Kow[1], Vedran Sabol[2], Michael Granitzer[2], Wolfgang Kienrich[2] and Dickson Lukose[1]

[1] Artificial Intelligence Centre, MIMOS Berhad[a], Technology Park Malaysia, Kuala Lumpur, Malaysia, {kwonn, dickson.lukose}@mimos.my
[2] Know-Center[b], Graz, Austria, {vsabol, mgrani, wkien}@know-center.at

Abstract. Ontology alignment is the process of matching related concepts from different ontologies. We propose a semi-automatic, visual approach which combines two algorithms for finding candidate alignments with visual navigation and analysis tools. The implementation is based on a Service-Oriented Architecture (SOA) to achieve scalability.

1. Motivation

A variety of ontology alignment algorithms have been proposed [1]. However, when algorithms cannot deliver the desired performance, it is necessary to include humans in the process. Visual semi-automatic alignment tools have been developed for this purpose, allowing experts to apply their knowledge. A survey of such systems [2] provides a summary of user requirements which served us as guidelines for designing our visual Semantic Mediation Tool.

2. Proposed Solution

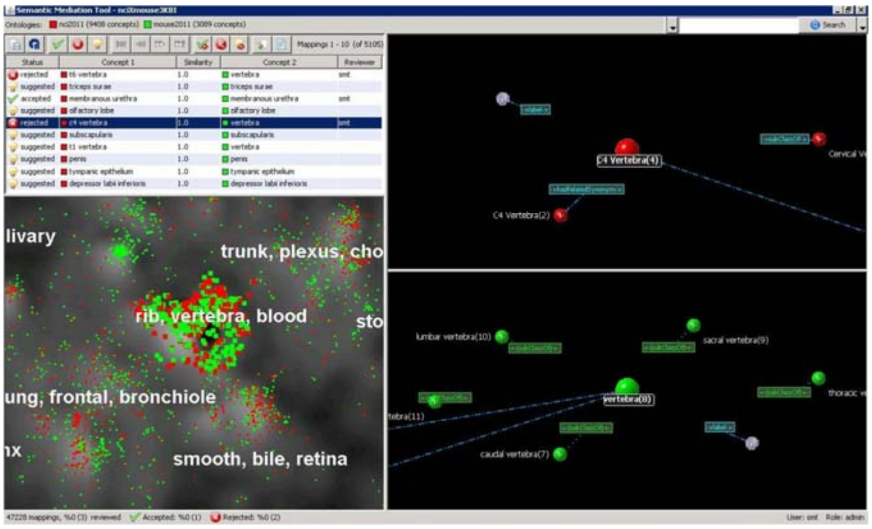
The Semantic Mediation Tool (SMT) is a client-server tool for semi-automatic, visually supported alignment of ontologies. Alignment algorithms are executed on the server built around a service-oriented architecture to provide scalability. Currently two algorithms are available: the first uses an external taxonomy (WordNet) to measure the distance between the concepts based on their labels, using the Wu-Palmer measure [5]; the second is a machine learning-based method consisting of concept vectorization, hierarchical concept clustering [4] and cluster-local match finding.

The client (Figure 1) displays concept mappings computed by the alignment algorithms and provides visual tools supporting the user in reviewing the suggested mappings. Ontology membership of concepts is encoded by color (red vs. green) in all components of the interface. A table (top-left) lists all computed mappings and allows the user to accept or reject them. For the mapping selected in the table (blue row), each concept is visualized in one the two Multimodal Semantic Browsers [3] (on right). The browsers are graph visualizations showing concepts within their

[a] MIMOS Berhad is funded by the Malaysian government through the Ministry of Science, Technology and Innovation (MOSTI).

[b] The KNOW-CENTER is funded within the Austrian COMET Program—Competence Centers for Excellent Technologies—under the auspices of the Austrian Ministry of Transport, Innovation and Technology, the Austrian Ministry of Economics and Labor and by the State of Styria. COMET is managed by the Austrian Research Promotion Agency (FFG).

ontological neighborhood, providing additional, detailed information valuable for reviewing. An information landscape (bottom-left) offers an overview of all concepts in a layout where similar concepts are placed close to each other [4]. Landscape regions are labeled with terms describing underlying concepts, which empowers the user to identify regions covering topics of interest and regions containing promising matching candidates (i.e. areas filled with both red and green dots). Concepts from a region can be selected using a lasso selection tool, which will filter the mappings in the table so that only mappings containing selected concepts will be shown.



Semantic Mediation Tool - nciXmouse3KB

Ontologies: 🟥 nci2011 (9408 concepts) 🟩 mouse2011 (3089 concepts)

Mappings 1 - 10 (of 5105)

Status	Concept 1	Similarity	Concept 2	Reviewer
rejected	🟥 t6 vertebra	1.0	🟩 vertebra	smt
suggested	🟥 triceps surae	1.0	🟩 triceps surae	
accepted	🟥 membranous urethra	1.0	🟩 membranous urethra	smt
suggested	🟥 olfactory lobe	1.0	🟩 olfactory lobe	
rejected	🟥 c4 vertebra	1.0	🟩 vertebra	smt
suggested	🟥 subscapularis	1.0	🟩 subscapularis	
suggested	🟥 t1 vertebra	1.0	🟩 vertebra	
suggested	🟥 penis	1.0	🟩 penis	
suggested	🟥 tympanic epithelium	1.0	🟩 tympanic epithelium	
suggested	🟥 depressor labii inferioris	1.0	🟩 depressor labii inferioris	

47228 mappings, %0 (3) reviewed Accepted: %0 (1) Rejected: %0 (2) User: smt Role: admin

Figure 1 – SMT client mediating 18816 concepts from the Anatomy benchmarks.

3. Future Work

As the next step, we see the evaluation of the alignment algorithms using OAEI data sets, with preliminary results already revealing issues such as the lack of a string-based matcher. Usability evaluation of the user interface shall be performed using controlled experiments to evaluate the impact and usefulness of the visualizations.

References

1. Euzenat, J. and Shvaiko, P., Ontology Matching, Springer 2007.
2. Granitzer, M., Sabol, V., Onn, K.W., Lukose, D. and Tochtermann, K., Ontology Alignment - A Survey with Focus on Visually Supported Semi-Automatic Techniques, Future Internet, Volume 2, Issue 3, pages 238-258, 2010.
3. Kow, W. O. and Lukose, D., Visualizing and Navigating Large Multimodal Semantic Webs, in Proceedings of the I-Know'10 Conference, pages 175-185, 2010.
4. Muhr, M., Sabol. V., Granitzer, M., Scalable Recursive Top-Down Hierarchical Clustering Approach with implicit Model Selection for Textual Data Sets, in Proc. of DEXA 10, 2010.
5. Wu, Z. and Palmer, M., Verb semantics and lexical selection, Proceedings 32nd Annual Meeting of the Association for Computational Linguistics (ACL) pages 133-138, 1994.

Mapping relational databases through ontology matching: A case study on information migration

Manuel Rodriguez-Mancha, Hector G. Ceballos, Francisco J. Cantu, and Aldo Diaz-Prado

Tecnologico de Monterrey, Campus Monterrey
Eugenio Garza Sada 2501, 64789, Monterrey, Mexico
{mjrodriguez,ceballos,fcantu,jadiaz}@itesm.mx

Abstract. In order to aid domain experts during data integration, several schema matching techniques have been proposed. Despite the facilities provided by these techniques, mappings between database schemas are still made manually. We propose a methodology for mapping two relational databases that uses ontology matching techniques and takes advantage of tools like D2R Server and AgreementMaker for automating mapping generation and for enabling unified access to information. We present the results obtained by some ontology matching algorithms in this context, demonstrating the feasibility of this approach.

Keywords: Relational DataBases, Ontology matching, Information Integration

1 Introduction

Specifying schema matches is a tedious, time-consuming, error-prone and therefore expensive process that requires the participation of one or more database experts. Current advances on the creation of *RDF views* of relational data[1] has provided an uniform interface to data that improves information integration and retrieval, as well as enables applying ontology matching algorithms to the database schema mapping problem.

2 A database mapping methodology based on ontology matching techniques

Counting with experts for both databases/systems during data migration is really exceptional, therefore we propose using training data for guiding this process. A domain expert, i.e. a regular user of the original system, is asked to capture in the new system some examples obtained from the original system (training data) in order to have equivalent information in both databases.

Next we apply an iterative process divided in three phases. In the first phase we generate an ontological representation of each database schema using D2R

[1] W3C RDB2RDF Working Group. `http://www.w3.org/2001/sw/rdb2rdf/`

[1]. The resulting D2RQ mapping is used for mounting a SPARQL end-point that provides unified access to database records as RDF instances and for generating a plain ontological representation of database schemas [3].

In the second phase we use matcher algorithms of AgreementMaker v.023 [2] for identifying correspondences between tables (classes) and fields (properties). In the third phase, the expert verifies mappings' correctness through the comparison of instances of both databases using the corresponding SPARQL end-point and schema equivalences.

Expert's feedback is persisted in a mapping ontology that is used in further iterations. The next iteration begins in the second phase whereas the entire process finishes when no new matching is found or the number of incorrect matches is greater than good ones.

3 First results on information migration

Our methodology was motivated by the migration from an information system in operation to a new one with analogous functioning. Both systems rely on normalized relational databases, the first in MySQL and the second in Oracle. Each database has about 180 tables and 1,400 fields in total.

From all matchers implemented in AgreementMaker, only ASM, DSI and the combination Anatomy identified correspondences. Resulting matches were evaluated by an expert, showing an average 22% of recall for class/table and property/field mappings. On the other hand, precision for class/table mappings was above 90% for these three matchers, but on property matching Anatomy obtained an 84% in comparison with 28% and 25% for ASM and DSI.

String matching techniques used by these matchers allowed detecting similarities between mnemonic names of tables and fields in the first place. Additionally, relations property-class allowed detecting mappings between equivalent fields across related tables.

In this way, mappings detected in the first iteration reduced in a 62% the number of comparisons required for mapping both databases. This means that the expert would have to perform manually only a 38% of the total comparisons.

As future work we will develop an automatic evaluation of mappings by comparing instances extracted from both databases and incorporating them in the ontological representation of schemas. This will enable matchers like IISM that uses mapped individuals for aligning classes as well.

References

1. Bauelos, E., Zrate, J., Almazn, S., Zubia, A.: D2RQ mappings incorporating semantic of domain. Journal of Environmental Studies 18, 59–62 (2009)
2. Cruz, I.F., Antonelli, F.P., Stroe, C.: Agreementmaker: efficient matching for large real-world schemas and ontologies. Proc. VLDB Endow. 2, 1586–1589 (August 2009)
3. Ghawi, R., Cullot, N.: Database-to-ontology mapping generation for semantic interoperability. In: Third International Workshop on Database Interoperability (InterDB 2007) (2007)

SERIMI – Resource Description Similarity, RDF Instance Matching and Interlinking

Samur Araujo[1], Jan Hidders[1], Daniel Schwabe[2], Arjen P. de Vries[1]

[1] Delft University of Technology, PO Box 5031, 2600 GA Delft, the Netherlands
{S.F.CardosodeAraujo, A.J.H.Hidders, A.P.deVries}@tudelft.nl

[2]Informatics Department, PUC-Rio Rua Marques de Sao Vicente, 225, Rio de Janeiro, Brazil
dschwabe@inf.puc-rio.br

Abstract. This paper presents SERIMI, an automatic approach for solving the interlinking problem over RDF data. SERIMI matches instances between a source and a target datasets, without prior knowledge of the data, domain or schema of these datasets. Experiments conducted with benchmark collections demonstrate that our approach considerably outperforms state-of-the-art automatic approaches for solving the interlinking problem over RDF data.

1 Introduction

The interlinking of datasets published in the Linked Data Cloud (LDC) [1] is a challenging problem and a key factor for the success of the Semantic Web. Given the heterogeneity of the LDC, techniques aimed at supporting interlinking at instance level should ideally operate agnostic of a specific domain or schema.

SERIMI[1] focus in the instance-matching problem over RDF data, which refers to the process of determining whether two RDF resources refer to the same real-world entity in a given domain. We propose an unsupervised solution for this problem, which is composed of two phases: the selection phase and the disambiguation phase. SERIMI uses existing traditional information retrieval and string matching algorithms for solving the selection phase, and we propose an innovative function of similarity during the disambiguation phase. This function is designed to operate even when there is no direct ontology alignment between the source and target datasets being interlinked. Fig. 1 shows an overview of SERIMI's instance matching process.

2 Validation

We use the collection proposed in the DI track of the Ontology Alignment Evaluation Initiative (OAEI 2010) [2] for evaluating SERIMI. We focused our evaluation in the life science (LS) collection and in the Person-Restaurant (PR) collection proposed by this initiative. We used two baselines in our experiments: RiMOM [3] and

[1] https://github.com/samuraraujo/SERIMI-RDF-Interlinking

ObjectCoref [4]. These two systems are representative of the two main types of solution for the interlinking task and, more importantly, they have used the same set of datasets and reference alignment as our method, allowing a fair and direct comparison. On average, SERIMI outperforms RiMOM and ObjectCoref in 70% of the cases.

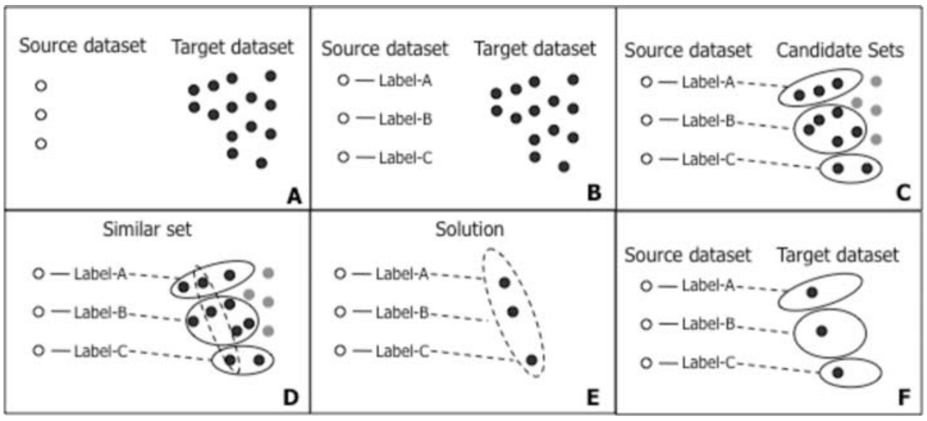


Fig. 1 – Overview of SERIMI interlinking process. (A) Given a source and target dataset and a set of source resources (instance of a class), (B) SERIMI obtains the label of these source resources and search for candidate resources in the target dataset that share a similar label. (C) For each source resource, SERIMI retrieves a set of candidate resources. (D) In order to disambiguate a set of candidate, SERIMI applies a novel function of similarity that selects the resources that are the most similar between all candidate sets (E). These selected resources are the solutions for the interlinking (F). The determination of this optimal cross section is a non-trivial process and it is done in a sophisticated way. This process assumes that the source resources belong to a homogeneous class of interest (e.g. musician, drugs, country, etc.)

3 Conclusion

SERIMI showed promising results for solving the task of RDF interlinking proposed by the OAEI 2010. As future work, we intend to evaluate this approach in different collections aiming to evaluate SERIMI as a domain specific solution and independent of domain solution for RDF interlinking.

References

1. Bizer, C., Heath, T. and Berners-Lee, T. (2009) Linked Data - The Story So Far. Int. J. Semantic Web Inf. Syst., 5 (3). pp. 1-22.
2. Ontology Alignment Evaluation Initiative 2010 Campaign - http://oaei.ontologymatching.org/2010/
3. Wang Z., Zhang X, Hou L., Zhao Y., Li J., Qi Y., Tang J. (2010). RiMOM Results for OAEI 2010. The Fifth International Workshop on Ontology Matching. Shanghai, China.
4. Hu W., Chen J., Cheng G., and Qu Y. (2010). ObjectCoref & Falcon-AO: Results for OAEI 2010. The Fifth International Workshop on Ontology Matching. Shanghai, China.

Translating expressive ontology mappings into rewriting rules to implement query rewriting

Gianluca Correndo and Nigel Shadbolt

ECS Department, University of Southampton, SO17 1BJ, Southampton, UK
{gc3,nrs}@ecs.soton.ac.uk,
http://users.ecs.soton.ac.uk/{gc3,nrs}

Abstract. The semantics of ontology alignments, often defined over a logical framework, implies a reasoning step over huge amounts of data. This is often hard to implement and rarely scales on Web dimensions. This paper presents our approach for translating DL-like ontology alignments into graph patterns that can be used to implement ontological mediation in the form of SPARQL query rewriting and generation.

1 Introduction

In spite of the high expressive power of the languages used to define ontologies (e.g. RDFS, OWL, and SKOS), the wide range of vocabularies within the data cloud restrains the realisation of a machine-processable Semantic Web. Ontology matching is an important task within the data integration work flow and Semantic Web community provided automated tools for mining and describing correspondences between data vocabularies [4]. Among other tools, the Alignment API [2] provides a rich language to describe ontology alignments called *Expressive and Declarative Ontology Alignment Language* or EDOAL for short.

Our approach to implement data integration is based on the rewriting of SPARQL queries applying syntactic rules that modify their basic graph pattern in order to rework a given source query to fit a target data set.

2 SPARQL Query Rewriting

The approach adopted here is similar to the one used in peer data management systems [3] where queries can be rewritten multiple times, depending on where the query will be executed. The full description of the algorithm that rewrites SPARQL queries can be retrieved from a previous publication [1] while in this paper we will focus on how to translate EDOAL expressions into our internal representation.

An entity alignment EA codifies how to rewrite a triple for fitting a new ontology, it defines therefore a pattern rewriting and eventually a set of constraints over variables present in the alignment itself. The alignments so defined are directional (i.e. not symmetric). An entity alignment EA is defined as a triple $EA = \langle LHS, RHS, FD \rangle$. LHS is an atomic formula that contains no functional symbols. RHS is a conjunctive formula that contain no functional symbols. Finally FD is a set of functional dependencies that must hold in the rewriting process.

3 Support for EDOAL Alignments

The EDOAL language is meant to describe correspondences between entities of different ontologies and it uses DL-like primitives to describe those entities. The subset of EDOAL translated in our approach includes only those primitives which would affect the BGP section of a query leaving out: **concepts** (or **properties**) **disjunction** that would require an OPTIONAL statement; **attributes value restrictions** different from EQUAL that would require a FILTER statement; and **attribute occurrence restrictions** for similar reasons. The translation of EDOAL primitives into rewriting rules pattern can be provided as a denotational semantics over a simplified subset of the grammar that generates the entities' description in EDOAL (i.e. the expressions that are the subjects of the alignments). The semantics will define how rewriting patterns are created from inspecting the parsing tree of the entity description.

In Figure 1 it is described the minimal annotated grammar for the EDOAL language primitives (pseudo code) alongside with the translation in terms of triples patterns used by our approach. Every grammar rule is decorated with its denotational semantics v and numbered for later reference.

$$\begin{aligned}
CE &::= class\ URI & v(CE) &= (Triple(?s, rdf:type, URI)) && (1)\\
CC &::= CE\ and\ CC' & v(CC) &= v(CE) + v(CC') && (2)\\
AEXP &::= pr\ URI & v(AEXP) &= (Triple(?s, URI, ?o)) && (3)\\
&|pr\ URI\ and\ AEXP' & v(AEXP) &= Triple(?s, URI, ?o) + v(AEXP') && (4)\\
&|pr\ URI\ comp\ AEXP' & v(AEXP) &= Triple(?s, URI, ?o) \circ v(AEXP') && (5)
\end{aligned}$$

Fig. 1: EDOAL denotational semantics

4 Acknowledgements

This research has been supported by the EnAKTing project funded by the Engineering and Physical Sciences Research Council (EPSRC) under the contract EP/G008493/1. The information provided is the sole responsibility of the authors and does not reflect the Council's opinion.

References

1. G. Correndo, M. Salvadores, I. Millard, H. Glaser, and N. Shadbolt. SPARQL query rewriting for implementing data integration over linked data. In *1st International Workshop on Data Semantics (DataSem 2010)*, March 2010.
2. J. David, J. Euzenat, F. Scharffe, and C. T. dos Santos. The Alignment API 4.0. *Semantic Web*, 2(1):3–10, 2011.
3. A. Y. Halevy, Z. G. Ives, D. Suciu, and I. Tatarinov. Schema mediation in peer data management systems. In *Proceedings of the* 19^{th} *International Conference on Data Engineering*, pages 505–516, 2003.
4. Y. Kalfoglou and M. Schorlemmer. Ontology mapping: The state of the art. In Y. Kalfoglou, M. Schorlemmer, A. Sheth, S. Staab, and M. Uschold, editors, *Semantic Interoperability and Integration*, number 04391, 2005.

EventMedia Live: Reconciliating Events Descriptions in the Web of Data

Houda Khrouf and Raphaël Troncy

EURECOM, Sophia Antipolis, France
<houda.khrouf@eurecom.fr>
<raphael.troncy@eurecom.fr>

1 Introduction

Many online services provide functionalities for sharing one's participation and captured media at real-world events. In a previous work [3], we constructed the EventMedia dataset aggregating heterogeneous sources of information and producing linked data. In this work, we carry out an event-oriented data reconciliation experiment in the web of data, and we propose two adapted extensions to a semi-automatic alignment tool named SILK [2].

2 Events Reconciliation

In this work, we aim at creating high quality `owl:sameAs` links between similar events which share an overlap in term of three properties: title, location and date. Hereafter, we propose two similarity extensions that better comply with event properties as it will be confirmed by some experimental results.

Temporal Inclusion metric. Intuitively, we consider that two events are similar if they share among others the same time or temporal interval. We introduce a new temporal inclusion metric where we define a parameter θ as the number of hours that can be tolerated between two dates. Given two events (e_1, e_2) which have respectively the couple start date and end date (d_1, d'_1) and (d_2, d'_2) where $d_1, d_2 \neq 0$ and d'_1, d'_2 can be null, the temporal inclusion (s) metric is defined by:

$$s\ (e_1, e_2) = \begin{cases} 1 \text{ if } |d_1 - d_2| \leq \theta \text{ where } (d'_1, d'_2) = 0 \\ 1 \text{ if } d_1 \pm \theta \in [d_2, d'_2] \text{ where } d'_1 = 0 \text{ (idem for } d_2\text{)} \\ 1 \text{ if } min(d'_1, d'_2) - max(d_1, d_2) \geq 0 \text{ where } (d'_1, d'_2) \neq 0 \\ 0 \text{ otherwise.} \end{cases} \quad (1)$$

Token-Wise string similarity. We studied the performance of string similarity functions over the EventMedia dataset that consists of user-generated content featuring typos and noisy data. We introduce a novel metric called Token-Wise combining the character-based and token-based string similarity metrics. The strings s and t are firstly split into a set of tokens $s_1....s_k$ and t=$t_1....t_p$. Given $s'(s_i, t_j)$ the score of the based-character similarity between two tokens, ws_i and wt_j the respectively weights of s_i and t_j, N the number of matched tokens (filtered triples), M the number of unmatched tokens where we set s' = 0, the token-wise (tw) metric is defined by:

$$tw(s,t) = \frac{\sum_{i=1}^{N} s'(s_i,t_j) \times ws_i \times wt_j}{\sum_{i=1}^{N+M} (1 - s'(s_i,t_j)) \times (ws_i^2 + wt_j^2) + \sum_{i=1}^{N} s'(s_i,t_j) \times ws_i \times wt_j} \quad (2)$$

Experimental Results. The first experiment was applied on agents' names to compare the token-wise and Jaro [1] distances based on a ground truth of 150 matched instances between Last.fm and MusicBrainz. The table 1 shows the recall and precision for different thresholds. In the second experiment, we evaluate the event alignment approach based on a ground truth containing 68 Eventful events compared with 104 Upcoming events, and 583 Last.fm events compared with 533 Upcoming events. Table 2 shows the recall and precision when the parameter θ of temporal inclusion is equal to 0 and 24 hours.

	Jaro		Token-Wise	
μ	Recall(%)	Precision (%)	Recall (%)	Precision(%)
0.95	24	100	60	100
0.9	49	98	82	99
0.8	87	93	96	98
0.7	100	45	100	77

Table 1. Precision and Recall for Jaro and Token-Wise similarities

	Eventful-Upcoming				LastFm-Upcoming			
	$\theta = 0$		$\theta = 24\ H$		$\theta = 0$		$\theta = 24\ H$	
$\mu >$	Recall	Precision	Recall	Precision	Recall	Precision	Recall	Precision
0.8	43	100	71	100	41	100	87	100
0.75	48	100	74	93	43	100	90	100
0.74	74	25	79	26	74	81	94	85
0.6	100	24	100	26	100	75	100	75

Table 2. Precision and Recall for events alignment

3 Conclusion

We proposed a powerful string similarity metric to cope with noisy titles, and a temporal inclusion similarity metric detecting a temporal overlap. The evaluation shows good results consolidating the efficiency of these extensions for SILK.

References

1. Matthew A. Jaro. Advances in Record-Linkage Methodology as Applied to Matching the 1985 Census of Tampa, Florida. *Journal of the American Statistical Association*, 84(406):414–420, 1989.
2. A. Jentzsch, R. Isele, and C. Bizer. Silk - Generating RDF Links while publishing or consuming Linked Data. In *9th International Semantic Web Conference (ISWC'10)*, Shanghai, China, 2010.
3. R. Troncy, B. Malocha, and A. Fialho. Linking Events with Media. In *6th International Conference on Semantic Systems (I-SEMANTICS'10)*, Graz, Austria, 2010.

Mobile Facetted Browsing LODD Applications for Supporting Clinicians

Daniel Sonntag, Jochen Setz and Maha Ahmed-Baker

German Research Center for AI (DFKI)
Stuhlsatzenhausweg 3, 66123 Saarbruecken, Germany

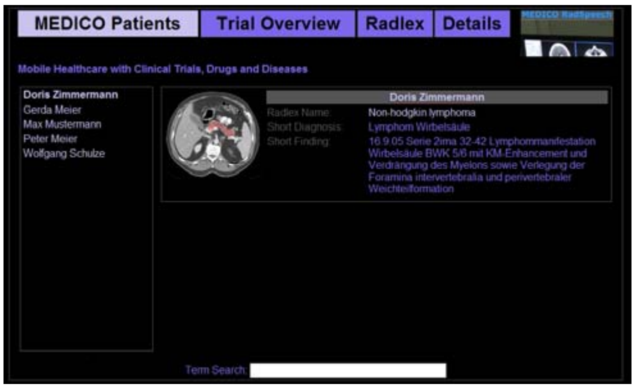
Abstract. An LODD application can be used to improve a clinician's daily work. Specific requirements of the radiology domain let us aggregate RDF results from several LODD sources such as DrugBank, Diseasome, and LinkedCT, as well as links to the web pages of ClinicalTrials and Pubmed, in a facetted browsing based mobile graphical user interface.

1 Introduction

Linked Data has the potential to provide easy access to related data sets in the healthcare domain. This should allow medical experts to explore the interrelated data more conveniently and efficiently without having to switch between many different applications, data sources, and user interfaces. In discussions and usability studies with clinicians we found that there are basically three different scenarios where the medical information contained in Linked Data can be useful: (1) the clinical reporting process; (2) the patient follow-up treatment (i.e., monitoring the patient's health condition and the development of the disease); and (3) the clinical disease staging and patient management. We will explain how an LODD (http://www.w3.org/wiki/HCLSIG/LODD) application based on diseases, drugs, and clinical trials can be used in all three scenarios. Such an application is based on medical disease knowledge as input and provides an incremental graphical HCI with a facetted browsing functionality to get relevant trial and drug information related to the disease information in the patient's record.

2 Facetted Browsing Application

In order to implement the facetted browsing LODD application, we connected RadLex, www.radlex.org/, our initial domain reference ontology for anatomy and radiology, with LODD data, in particular with LinkedCT to get further access to DrugBank and Diseasome (and the ClinicalTrials and Pubmed webpages, respectively). Figure 1 shows the GUI of the implemented facetted browsing tool. There are two options for an input, either a RadLex term that a drop-down menu suggests while the user types the disease name, or the disease name automatically taken from the RadLex image annotation. The backend runs a query written in SPARQL to interlink LinkedCT, Diseasome, DrugBank, and DailyMed to gain useful information that might help the radiologist to make a decision about a treatment or in order to suggest alternative medications. Mappings between Radlex and LinkedCT do not exist; we obtained best results when using string-based ontology matching approaches while mapping the Radlex term (verified by the search-as-you-type functionality with access to a Radlex API) onto individual tokens (words) in the trial description which we indexed for all 90K keywords to get the trial URIs (cf. http://linkedct.org/resource/keyword/). Links to Diseasesome and DrugBank are largely available but also in a n to n relationship. Our tool allows a user to filter thousands of results according to the bast-ranked string matches and linked data relations in a convenient way. Clinicians can retrieve the trails they might be interested in in just a few seconds.



Mobile Healthcare with Clinical Trials, Drugs and Diseases

Doris Zimmermann
Gerda Meier
Max Mustermann
Peter Meier
Wolfgang Schulze

Doris Zimmermann

Radlex Name:	Non-hodgkin lymphoma
Short Diagnosis:	Lymphom Wirbelsäule
Short Finding:	16.9.05 Serie 2ima 32-42 Lymphommanifestation Wirbelsäule BWK 5/6 mit KM-Enhancement und Verdrängung des Myelons sowie Verlegung der Foramina intervertebralia und perivertebraler Weichteilformation

Fig. 1. LODD Application (iPad), http://digitaleveredelung.dfki.de/MEDICO/

For example, a radiologist types a disease name “carcinoma”. Then he or she is probably interested in the clinical trails of a specific study type, drug, and/or drug type which we aggregated into different facets according to the disease mapping function or the provided links to Diseasome, DrugBank, and DailyMed. For genetically caused diseases, one can also look at the disease from the genetic point of view (at least the gene name can help to combine or compare therapies). Most of the development time has been spent on (1) the access to non-conformed Linked Data RDF results in the form of conversion programmes for JSON and XML results; (2) the provision of the online SPARQL queries for our Virtuoso server (http://virtuoso.openlinksw.com/); and (3) the implementation of a graphical user interface from scratch by using open-source knowledge management tools such as Exhibit, see http://www.simile-widgets.org/exhibit/.
*Acknowledgements:*This research has been supported by the THESEUS Programme funded by the German Federal Ministry of Economics and Technology (01MQ07016).

Complex Matching of RDF Datatype Properties

Bernardo P. Nunes[1], Alexander Mera[1], Marco A. Casanova[1], Karin K. Breitman[1]
Luiz André P. Paes Leme[2]

[1]Department of Informatics – PUC-Rio – Rio de Janeiro, RJ – Brazil
{bnunes, acaraballo, casanova, karin}@inf.puc-rio.br
[2]Instituto de Computação – UFF – Rio de Janeiro, RJ – Brazil
lapaesleme @ ic.uff.br

Abstract. Property mapping is a fundamental component of ontology matching, and yet there is little support that goes beyond the identification of single property matches. Real data often requires some degree of composition, trivially exemplified by the mapping of *FirstName, LastName* to *FullName* on one end, to complex machings, such as parsing and pairing symbol/digit strings to *SSN* numbers, at the other end of the spectrum. In this paper, we briefly introduce a two-phase instance-based technique for complex datatype property matching.

Keywords: Ontology Matching, Genetic Programming, Mutual Information.

1 Introduction

Ontology matching is a fundamental problem in many applications areas [1]. Using OWL concepts, by *datatype property matching* we mean the special case of matching datatype properties from two classes.

Very briefly, an *instance* of a datatype property p is a triple of the form (s,p,l), where s is a resource identifier and l is a literal. A *datatype property matching* from a *source* class S to a *target* class T is a partial relation μ between sets of datatype properties of S and sets of datatype properties of T. We say that a match $(\boldsymbol{A},\boldsymbol{B}) \in \mu$ is *m:n* iff $\boldsymbol{A}$ and $\boldsymbol{B}$ contain m and n properties, respectively. A match $(\boldsymbol{A},\boldsymbol{B}) \in \mu$ should be accompanied by one or more *datatype property mappings* that indicate how to construct instances of the properties in $\boldsymbol{B}$ from instances of the properties in $\boldsymbol{A}$. A match $(\boldsymbol{A},\boldsymbol{B}) \in \mu$ is *simple* iff it is *1:1* and the mapping is a simple translation; otherwise, it is *complex*.

In this paper, we briefly introduce a two-phase, instance-based datatype property matching technique that is able to find complex *n:1* datatype property matches and to construct the corresponding property mappings. The technique extends the ontology matching process described in [2] to include complex matches between sets of datatype properties and is classified as instance-based since it depends on sets of instances.

2 The Two-Phase Property Matching Technique

Given two sets, ***s*** and ***t***, that contain instances of the datatype properties of the source class *S* and the target class *T*, respectively, the first phase of the technique constructs the Estimated Mutual Information matrix [2,3] of the datatype property instances in ***s*** and the datatype property instances in ***t***, which intuitively measures the amount of related information of the observed property instances. This phase possibly identifies simple datatype property matches. For example, it may detect that the *eMail* datatype property of one class matches the *ElectronicAddress* datatype property of the other class. The first phase may also suggest, for the second phase, sets of datatype properties that may match in more complex ways, thereby reducing the search space.

The second phase uses a genetic programming approach [4] to find complex *n:1* datatype property matches. For example, it may discover that the *FirstName* and *LastName* datatype properties of the source class matches the *FullName* datatype property of the target class, and return a property mapping function that concatenates the values of *FirstName* and *LastName* (of the same class instance) to generate the *FullName* value. The reason for adopting genetic programming is two-fold: it reduces the cost of traversing the search space; and it may be used to generate complex mappings between datatype property sets.

3 Conclusion

In this paper, we briefly described an instance-based, property matching technique that follows a two-phase strategy. The first phase constructs the Estimated Mutual Information matrix of the property values to identify simple property matches and to suggest complex matches, while the second phase uses a genetic programming approach to detect complex property matches and to generate their property mappings. Our early experiments suggest that the technique is a promising approach to construct complex property matches, a problem rarely addressed in the literature. Full details can be found in [5].

Acknowledgements. This work was partly supported by CNPq, under grants 473110/2008-3 and 557128/2009-9, by FAPERJ under grant E-26/170028/2008, and by CAPES under grant CAPES/PROCAD NF 21/2009.

References

1. Euzenat, J., Shvaiko, P. *Ontology matching*. Springer-Verlag (2007).
2. Leme, L. A. P. P., Casanova, M. A., Breitman, K. K., Furtado, A. L. Instance-Based OWL Schema Matching, Lectures Notes in Business Info. Proc., vol. 24, 2009, pp.14-25.
3. Leme, L. A. P. P., Brauner, D. F., Breitman, K. K., Casanova, M. A., Gazola, A. Matching Object Catalogues, Innov. in Sys. and Soft. Eng. Springer, 4(4), 2008, pp. 315-328.
4. Koza, J. *Genetic Programming*. The MIT press, 1998.
5. Nunes, B. P., Mera, A., Casanova, M. A., Breitman, K. K., Leme, L. A. P. P. Complex Matching of RDF Datatype Properties. MCC12/11, Dept Informatics, PUC-Rio (September 2011).

Automated Matching of Data Mining Dataset Schemata to Background Knowledge

Stanislav Vojíř, Tomáš Kliegr, Vojtěch Svátek, and Ondřej Šváb-Zamazal

University of Economics, Prague, Dept. Information and Knowledge Engineering
{stanislav.vojir,tomas.kliegr,svatek,ondrej.zamazal}@vse.cz

1 Problem Setting

Interoperability in data mining is supported by a standard for dataset and model representation: *Predictive Model Markup Language* (PMML).[1] It allows to describe the columns (which are continuous, categorical or ordinal) of the source data table, pre-processing transformations (such as discretization of continuous values) as well as the structure of the discovered model (e.g. neural network or set of association rules).

In addition to source data, the input to the mining task typically includes expert-provided *background knowledge*. It may be related, for example, to standard ways of discretizing numerical quantities (e.g. boundaries between 'normal blood pressure' and 'hypertension', which help intuitive reading of discovered hypotheses), or may itself have the form of predictive models to which the discovered models can be compared during or after the mining process. The proposal for *Background Knowledge Exchange Format* (BKEF) [**?**], in many aspects similar to PMML, aims to support interoperability of data mining (and related) applications dealing with background knowledge.

Case studies [**?**] showed that one BKEF model typically has to be aligned with different PMML models (from different mining sessions in the same domain). The alignments are stored in the *Field Mapping Language* (FML) [**?**], expressing that a data field (column) in a PMML model semantically corresponds to an abstract 'field' in a BKEF model. However, writing FML alignments (analogous to instances of the Alignment Format [**?**] used in ontology matching) by hand is tedious and recognizing suitable correspondences may be hard; partial automation is thus desirable. Furthermore, existing tools for ontology/schema matching are not straightforwardly usable, since PMML (and even BKEF) are, compared to ontologies, more biased by data structures, but BKEF is more abstract and weakly structured than database schemata. Therefore, specific methods (inspired by existing ones) and a new tool have been devised.

2 Method, Implementation and Experiments

The matching process consists of several steps. First, the data are pre-processed into a unified format that removes most syntactic differences between PMML and BKEF. Then the *similarity* between data columns of both input resources is calculated based

[1] http://www.dmg.org/

on string measures over the *column names*, as well as based on *allowed values*. Based on the similarity matrix, *suitable alignment* (1:1 or 1:N) is finally designed.

An important feature of the system is (simple) *machine learning* from interaction with the user; as far as the positive examples are concerned, the learning component distinguishes between automatically suggested correspondences that were explicitly *confirmed* by the user (or manually entered) and those that were merely *tolerated*. The output of the learned rules is used to adjust the final similarity value.

The system has been implemented as a web application in PHP (as component to the Joomla! CMS) on server side and in JavaScript (with AJAX) on client side. Its graphical interface conveniently displays the columns of both to-be-matched resources and allows to 1) *align* one to another, 2) let one be *ignored* (in subsequent automatic matching), 3) *confirm* an automatic alignment, or 4) *revoke* any of the previous operations. Similar operations can be applied on the *values* for a pair of columns.

The system has been tested within the SEWEBAR project[2] on data from medicine and finance. Furthermore, a conventional schema matching benchmark dataset was borrowed from the *Illinois Semantic Integration Archive*, which describes universities and their courses.[3] The evaluation achieved the precision of about 70% and recall about 77% on unknown columns, while when matching the data previously aligned by the user (using the machine learning facility), the recall was improved to 90-100%.

3 Relevance for Semantic Web (and Ontology Matching) Research

Although BKEF models structurally differ from ontologies, they are close to them in covering a certain *domain* in contrast to PMML models that only cover a certain *dataset*. Experience with such asymmetric matching could cross-fertilize with the task of matching the ad-hoc mixes of vocabulary entities underlying many *Linked Data sets* to carefully engineered *domain ontologies*. A promising direction could also be that of linking BKEF entities explicitly to domain ontologies; BKEF could represent an *intermediate layer* between concrete datasets (that are subject to business-analytics processes) and ontologies as sophisticated resources that are often too abstract for industrial users.

Stanislav Vojíř and Tomáš Kliegr are supported by the IGA VSE under 26/2011. Vojtěch Svátek and Ondřej Šváb-Zamazal are supported by the CSF under P202/10/0761.

References

1. David J., Euzenat J., Scharffe F., Trojahn dos Santos C.: The Alignment API 4.0. *Semantic Web*, 2(1):3-10, 2011.
2. Kliegr T., Svátek V, Ralbovský M., Šimůnek M.: SEWEBAR-CMS: Semantic Analytical Report Authoring for Data Mining Results. *Journal of Intelligent Information Systems*, Springer, 2010 (Online First).
3. Kliegr, T., Vojíř, S., Rauch, J.: Background Knowledge and PMML – First Considerations. In: PMML Workshop at KDD'11,August 21, 2011, San Diego, CA, USA.

[2] `http://sewebar.vse.cz`

[3] `http://pages.cs.wisc.edu/~anhai/wisc-si-archive/domains/courses.html`

A Criteria for Selecting Background Knowledge for Domain Specific Semantic Matching

Jetendr Shamdasani, Peter Bloodsworth, Tamas Hauer, Andrew Branson, Mohammed Odeh, and Richard McClatchey

CCCS, UWE, Coldharbour Lane, Frenchay, Bristol BS16 1QY, UK
`firstname.lastname@cern.ch`

Abstract. Ontology matching is a problem that is rapidly reaching maturity especially in the context of the emergent semantic web. Semantic matching is the detection of specific correspondences between concepts in two ontologies including but not limited to $\equiv$, $\sqsubseteq$ and $\sqsupseteq$. Many semantic matching approaches today use some form of background knowledge to aid them in discovering correspondences. This paper presents criteria that can be used when selecting a background knowledge source.

1 Introduction

The selection of a correct resource of the semantic matching process is a crucial step since it seeds the final reasoning step in the semantic matching process. Many of these resources are selected manually for example Aleksovski et al used the FMA as a source of background knowledge [1]. The SMatch approach uses WordNet [2] as its background resource. This poster paper deals with the manual selection of a resource for the medical domain. Previously in [3] it was shown that using a resource that is too general may yield in ambiguous alignment results. To aid in background knowledge selection criteria have been developed. The next section will detail the criteria for selection of such a resource.

2 Selection Criteria for Background Knowledge

The selection criteria have been defined to be the following: **coverage**, **semantics**, **granularity** and **meta-information**. Examples from the domain of medicine are used to describe and justify the criteria accordingly.

1. **Coverage** This criterion concerns insuring that sufficient concepts from the medical domain are covered to enable the semantic matching process. For example many medical ontologies today do not cover a single domain or may slightly overlap with another domain. For example, Galen covers the areas of Anatomy with some Pathological information. By using a source of background knowledge which only covers a single subdomain of medicine, some information may be missed. Therefore, the background resource needs to be able to cover as many terms from the medical domain as possible.

2. **Semantics** are required for the derivation of subsumption and equivalence correspondences between ontologies and to create the background theory for the reasoning process. This background theory is present in most semantic matching methods that are available today. For example let us consider two terms search as “Heart” and “Aorta”, these two terms have very little in common lexically. Here a source of background knowledge is useful in that it states the relationships between these terms explicitly which is that Heart is more general than Aorta. Therefore relationships such as this need to be explicitly stated in a source of background knowledge.
3. **Granularity** is the availability of terms to a high-level of detail. In comparison to the previous requirement of coverage, granularity is the availability of terms in a background resource which are lower in the semantic hierarchy. Granularity therefore can be thought of as a detailed description of a domain. Therefore to seed the semantic matching process, terms need to be available in a background resource at a high-level of detail. In the medical domain this is especially an issue since many medical terms are related to each other, but they may also have very little lexical information in common. Although the reasoning process may be able to detect a mapping by inference, the relationship from a background resource is given a higher weighting because it is viewed as a more trusted source.
4. **Meta-information** refers to the presence of meta-data within the background knowledge source. For anchoring purposes this can provide a meaningful string to concept mapping. Meta-data needs to be present, either having been derived algorithmically or stated explicitly for this purpose. This is so that information can be extracted from the ontology using it.

3 Conclusion

In this paper the importance of background knowledge in the semantic matching process was stated. It was suggested that background knowledge from a trusted source would be an asset to the semantic matching process, since it seeds the final reasoning process with initial relationships. Criteria were defined for the selection of a background resource for semantic matching. These criteria were justified to be coverage, semantics, granularity and meta-information. These criteria have already been utilised in the creation of the MedMatch algorithm [3].

References

1. Z. Aleksovski et al. Exploiting the structure of background knowledge used in ontology matching. In *Ontology Matching Workshop (ISWC-2006)*, 2006.
2. F. Giunchiglia et al. Semantic Matching: Algorithms and Implementation. *Journal on Data Semantics*, 9:1–38, 2007.
3. J. Shamdasani. *Semantic Matching for the Medical Domain.* PhD thesis, University of the West of England, December 2010.

Towards a Framework for Ontology Mapping based on Description Logic Reasoning.

Quentin Reul, Jeff Z. Pan, and Derek Sleeman

University of Aberdeen, Aberdeen AB24 3F , UK

Abstract. In this paper, we describe the Knowledge Organisation System Implicit Mapping (KOSIMap) framework, which differs from existing ontology mapping approaches by using description logic reasoning (i) to extract implicit information for every entity, and (ii) to remove inappropriate mappings from an alignment.

1 KOSIMap Framework

Ontology matching has been recognised as a means to achieve semantic interoperability by resolving lexical and structural heterogeneities between two ontologies. Given two ontologies $\mathcal{O}_1$ and $\mathcal{O}_2$, the task of mapping one ontology to another is that of finding an entity in $\mathcal{O}_1$ that matches an entity in $\mathcal{O}_2$ based on their intended meanings. As OWL ontologies are normally used with an inference engine, it is important to consider inference and reasoning as part of the ontology mapping process [2]. However, most approaches have to date disregarded the role of description logic reasoning because of efficiency reasons (e.g. [1]). While these approaches generally deliver good results, they are limited to the asserted axioms contained in the input ontologies. Therefore, the extraction of logical consequences embedded in the input ontologies may result in alignments containing less erroneous mappings.

In this paper, we describe the Knowledge Organisation System Implicit Mapping (KOSIMap) framework [3], which respects the *uniform comparison* principle [1] by restricting each comparison to entities (i.e. classes and properties) in the same category. KOSIMap consists of three main steps; namely *Pre-Processing*, *Matcher Execution*, and *Alignment Extraction*. The pre-processing step extracts logical consequences embedded in both ontologies using a DL reasoner (e.g. Pellet [4]). Note that we have enhanced its functionality by developing rules to extract further information about classes and properties. For example, we have devised several rules to extract the properties associated with a class. The pre-processing step also applies language-based techniques (e.g. tokenization, lemmatization) to lexical descriptions (i.e. labels). Next, KOSIMap applies three different types of matchers for every pair of entities (see §2). The final step extracts an alignment between two ontologies and consists of two phases. The first phase extracts a pre-alignment from the similarity matrix, by selecting the maximum similarity score above a threshold ζ for every row in the matrix. This pre-alignment is then passed through a refinement process, which eliminates erroneous mappings from the set of correspondences. This refinement approach differs from existing ones [5] by using the implicit knowledge extracted from the axioms in the first step.

2 Mapping Strategies

KOSIMap computes the lexical and structural similarity between pairs of entities based on three different matchers:

1. The *string-based* matcher assumes that domain experts share a common vocabulary to express the labels of entities. In KOSIMap, we compute the similarity between pairs of labels based on the SimMetrics library[1].
2. The *property-based* matcher first computes the overlap between two properties based on the set of properties (e.g. inferred super-properties) associated with them. It then calculates the overlap between two classes based on their respective sets of properties (e.g. inferred domains).
3. The *class-based* matcher first computes the similarity between two classes based on the set of classes (e.g. inferred super-classes) associated with them. It then calculates overlap between sets of binary relations[2] for each pair of object properties.

The property-based and class-based matchers rely on the *degree of commonality coefficient* to compute the overlap between two sets $\mathtt{S}_s$ and $\mathtt{S}_t$. The degree of commonality coefficient is defined is defined as the sum of the maximum similarity for each element in source set (i.e. $\mathtt{S}_s$). The aggregated scores from these matchers are then stored in a similarity matrix.

Acknowledgements: The IPAS project is co-funded by the Technology Strategy Board's Collaborative Research and Development programme and Rolls-Royce (project No. TP/2/IC/6/I/10292).

References

1. J. Eu enat, D. Loup, M. Tou ani, and P. altchev. Ontology alignment with OLA. In *Proc. of the 3rd Evaluation of Ontology-based Tools Workshop (EON2004)*, 2004.
2. . F. oy. Semantic Integration: A Survey Of Ontology-Based Approaches. *SIGMOD Record*, 33(4):6 –70, 2004.
3. Q. eul and J. Pan. KOSIMap: Use of Description Logic easoning to Align eterogeneous Ontologies. In *Proc. of the 23rd International Workshop on Description Logics (DL 2010)*, 2010.
4. E. Sirin, B. Parsia, B. C. Grau, A. Kalyanpur, and Y. Kat . Pellet: A practical OWL-DL reasoner. *Web Semantics: Science, Services and Agents on the World Wide Web*, (2): 1– 3, June 2007.

. P. Wang and B. u. Debugging ontology mappings: A static approach. *Computing and Informatics*, 22:1001–101 , 2003.

[1] http://sourceforge.net/projects/simmetrics/

[2] The set of binary relation of an object property t, denoted $\mathtt{R}(t)$, is a collection of ordered pairs of classes extracted from axioms in an ontology.

A Structuralistic Semantics for Ontology Alignments

Christian Schäufler[1], Clemens Beckstein[1,2], and Stefan Artmann[2]

[1] Artificial Intelligence Group, University of Jena, Germany
[2] Frege Centre for Structural Sciences, University of Jena, Germany

1 Introduction

The definition of a formalism for ontology alignments is straightforward. Problems start when one attempts to define what they mean. The existing semantics [3] all depend on certain preconditions for the alignments to make sense — e.g., that the ontology domains are empirically given, that the involved ontologies are compatible, and that either the domains are identical or, there is a practicable way for specifying a domain relation. Those preconditions, of course, are not always empirically justifiable. With scientific structuralism [1] another approach for interpreting inter-theoretical relations has been put forward. With the help of this framework it is possible to give an exact description of the formal context in which the distributed alignment semantics works.

2 Structuralistic Interpretation of Alignments

A structuralistic interpretation of an ontology O (see [2]) need not consist of just one undifferentiated domain D, but may consist of several domains $D_1, \ldots, D_n$. Structuralists call the factors of such a domain structure *(domain) types.* The domain terms $D_1, \ldots, D_n$ are given by a set of disjunct primitive concepts that, according to the ontological axioms of O, are just below the top concept $\top$ of the ontology. In analogy to the distinction between reduced and reducing theory we assign roles to the ontologies that are involved in an alignment. Alignments in our view always assume a specific flow of information — from a *foreign* knowledge base W over O to the initial inquirer with a commitment to O'. Domain inclusions relate the domains of the ontologies to be aligned. Despite the domain relation r in contextualized distributed semantics, a *domain inclusion* sets whole (echelons of) domains in relation. An echelon set on some base sets is a set resulting from arbitrary product or power-set-operations of the base sets or echelon sets (of the base sets).

Let m and m' be two models that satisfy a correspondence $e \, R \, e'$. Because the domains $D_1, \ldots, D_n$ match the top-level primitive concepts of ontology O, the extension of an ontology element e in O is a subset of exactly one domain D_i (or a pair of domains if e is a role) and e' that of exactly one D'_j. For an interpretation of the correspondence, both ontology elements have to be interpreted in the same domain. W.l.o.g. O' is the querying ontology, so the interpretation of both e and

e' should take place in D'_j. A domain inclusion between D_i and D'_j is either a domain inclusion $D_i \subseteq S$ where S is an echelon-set that actually uses the base set D'_j or $D'_j \subseteq S'$ where S' is an echelon-set that actually uses the base set D_i. The following disjoint cases can be distinguished:

1. *$D_i \subseteq D'_j$:* The extension of e is a subset of D'_j. e can be interpreted in the same domain D'_j as e'.

2. *$D'_j \subseteq D_i$:* To assure an interpretation in simple distributed semantics, it is necessary that in addition the domain inclusion $e^m \subseteq D'_j$ holds.

3. *$D_i \subseteq S$ with $S \neq D'_j$:* An interpretation in domain D'_j is possible if the elements of D_i can be ontological projected to D'_j: $p(D_i) \subseteq D'_j$, where p is that mapping that projects S to D'_j.

4. *$D'_j \subseteq S'$ with $S \neq D_i$:* The extension of e' consists of complex individuals from the point of view of the queried ontology O. Elements of e are missing some properties in order to be interpretable as elements of D'_j. An alignment containing such a domain inclusion does not have a model.

5. *Otherwise:*D_i and D'_j are ontologically incompatible. Even if some individuals do appear in both domains, there is no general rule holding for every individual. Even by introducing additional properties, neither D'_j can be reconstructed from D_i nor the other way around. For incompatible domains an interpretation of the correspondence is therefore only possible wrt. the intersection $D_i \cap D'_j$, i.e. if the domain inclusions $e^m \subseteq D'_j$ and $e^{m'} \subseteq D_i$ hold.

Each of the cases specifies a (maybe empty or unsatisfiable) additional precondition in the form of domain inclusions that have to be fulfilled for the alignment to have a model in the structuralistic sense. And whenever these *alignment specific domain inclusions* hold, the corresponding case can be reduced to an interpretation like that of the simple distributed semantics.

3 Conclusion

Structuralism points the way to a new semantics for ontology alignments that rests on structured domains, a distinguished direction of the alignment, and compatibility constraints in the form of term-by-term domain inclusions. Domain inclusions are much simpler to specify in practice than domain relations as they are used in the contextualized distributed semantics: they are expressed wrt. a given set of domain terms and not wrt. individuals of an empirical domain and many of them result as a byproduct of the structuralistic alignment process.

References

1. Balzer, W., Moulines, C.U., Sneed, J.D.: An architectonic for science - the structuralist program. Reidel, Dordrecht (1987)
2. Schäufler, C., Artmann, S., Beckstein, C.: A structuralistic approach to ontologies. In: KI 2009: Advances in Artificial Intelligence. Lecture Notes in Computer Science, vol. 5803, pp. 363–370. Springer, Berlin / Heidelberg (2009)
3. Zimmermann, A., Euzenat, J.: Three semantics for distributed systems and their relations with alignment composition. In: The Semantic Web - ISWC 2006. Lecture Notes in Computer Science, vol. 4273, pp. 16–29. Springer Berlin / Heidelberg (2006)

A Similarity Measure based on Semantic, Terminological and Linguistic Information

Nitish Aggarwal*, Tobias Wunner*°, Mihael Arcan*,
Paul Buitelaar*, Seán O'Riain°

*Unit for Natural Language Processing and °eBusiness Unit
Digital Enterprise Research Institute,
National University of Ireland, Galway
firstname.lastname@deri.org

Introduction

The fundamental task of ontology matching is based on measuring the similarity between two ontology concepts [1]. However, we argue that a deeper semantic, terminological and linguistic analysis of specialized domain vocabularies is needed in order to establish a more sophisticated similarity measure that caters for the specific characteristics of this data. In particular we propose 'STL', a novel similarity measure that takes semantic, terminological and linguistic variation into account.

The STL Similarity

We base our approach on the three-faceted STL ontology enrichment process introduced in [3]. We calculate similarity according to semantic, terminological and linguistic variation and then take a linear combination by using linear regression, called STL similarity, which we describe as follows:

Semantic similarity (sim_S) is calculated based on semantic (taxonomic or ontological) structure. For our purposes we used a recently proposed semantic similarity measure proposed by Pirro_Sim [2], which uses intrinsic information content, i.e. the information content of a concept defined by the number of its subconcepts.

Terminological similarity (sim_T) is defined by maximal subterm overlap, i.e. we calculate sim_T between two concepts c1 and c2 as the number of subterms t_i in a termbase that can be matched on the labels of c1 and c2. A term t_i is said to match on a concept when no other longer term t_j can be matched on the same concept (label). To calculate sim_T we use monolingual as well as multilingual termbases as the latter reflect terminological similarities that may be available in one language but not in others, e.g. there is no terminological similarity between the English terms ”Property Plant and Equipment” and ”Tangible Fixed Asset”, whereas in German these concepts are actually identical on the terminological level (they both translate into ”Sachanlagen”).

Linguistic similarity (sim_L) is defined as the Dice coefficient applied on the head&modifier syntactical arguments of two terms, i.e., the ratio of common modifiers to all modifiers of two concepts. For instance the concepts ”Financial Income” and ”Net Financial Income” have 3 modifiers ”financial” ”net” and ”net financial”, whereby only ”financial” is a common modifier.
Putting it all together we define STL similarity as a linear combination of the sub-measures where the weights $w_{S,T,L}$ are their contributions on the data set:

$$sim_{S,T,L} = w_S * sim_S + w_T * sim_T + w_L * sim_L + constant$$

We evaluated our approach on a data set of 59 financial term pairs, drawn from the xEBR (European Business Registry) vocabulary, that were annotated by four human annotators. Table 1 shows the correlations ρ of all measures on the data set and that STL outperforms all of its S,T and L contributions.

Measure	ρ	Type
PathLength	0.16	S
Wu−Palmer	0.18	S
Pirro_Sim	0.20	S
UnigramMono	0.72	T
UnigMulti	0.72	T
BigMono	0.53	T
Bi_Multi	0.54	T
SubtermMono	0.74	T
SubtermMulti	0.75	T
Lemmatized	0.70	L
Head&Mod	0.51	L
STL	**0.78**	S,T,L

Table 1. Correlation of STL similarity measures with human evaluator scores

References

1. Euzenat, J., Meilicke, C., Stuckenschmidt, H., Shvaiko, P., dos Santos, C.T.: Ontology alignment evaluation initiative: Six years of experience. J. Data Semantics 15, 158–192 (2011)
2. Pirró, G.: A semantic similarity metric combining features and intrinsic information content. Data Knowl. Eng. 68, 1289–1308 (November 2009)
3. Wunner, T., Buitelaar, P., O’Riain, S.: Semantic, terminological and linguistic interpretation of xbrl. In: In Reuse and Adaptation of Ontologies and Terminologies Workshop at 17th EKAW (2010)

Acknowledgements

This work is supported in part by the European Union under Grant No. 248458 for the Monnet project and by the Science Foundation Ireland under Grant No. SFI/08/CE/I1380 (Lion-2).

Folksodriven Structure Network

Massimiliano Dal Mas

me @ maxdalmas.com

Abstract. Nowadays folksonomy is used as a system derived from user-generated electronic tags or keywords that annotate and describe online content. But it is not a classification system as an ontology. To consider it as a classification system it would be necessary to share a representation of contexts by all the users. This paper is proposing the use of folksonomies and network theory to devise a new concept: a "Folksodriven Structure Network" to represent folksonomies. This paper proposed and analyzed the network structure of Folksodriven tags thought as folsksonomy tags suggestions for the user on a dataset built on chosen websites. It is observed that the Folksodriven Network has relative low path lengths checking it with classic networking measures (clustering coefficient). Experiment result shows it can facilitate serendipitous discovery of content among users. Neat examples and clear formulas can show how a "Folksodriven Structure Network" can be used to tackle ontology mapping challenges.

Keywords. Ontology, Folksonomy, Natural Language Processing, Information filtering, Metadata, Semantic networks, Scale-free Network, Reasoning, Algorithms, Experimentation, Theory

1 Introduction

The communicative form of the World Wide Web is based on user-centric publishing and knowledge management platforms as sharing systems for social and collaborative matching like: Wikis, Blogs, Facebook, etc… Ontology defines a common set of sharing concepts [1], but unfortunately ontologies are not wide spread at the moment. While Folksonomy is said to provide a democratic tagging system that reflects the opinions of the general public, but it is not a classification system and it is difficult to make sense of [2]. A representation of contexts should be share by all the users. The goal of this work is to help the users to choose proper tags thanks to a "Folksodriven Structure Network" intended as a dynamical driven system of folksonomy that could evolve during the time. In this work the main network characteristics was analyzed by a group of articles from a chosen websites and analyzed according to the Natural Language Processing. The data structures extracted is represented on folksonomy tags that are correlated with the source and the relative time exposition - measure of the time of its disposal. Considering those we define a tag structure called Folksodriven, and adapt classical network measures to them.
An extension of this paper is available on Arxiv (http://arxiv.org/abs/1109.3138).

2 Folksodriven Notation

$$FD := (C, E, R, X) \tag{1}$$

A Folksodriven will be considered as a tuple (1) defined by finite sets composed by:

- *Formal Context* (C) is a triple *C:=(T, D, I)* where the objects *T* and the attributes *D* are sets of data and *I* is a relation between *T* and *D* [3] – see 3 (*Folksodriven Data Set*);
- *Time Exposition (E)* is the clickthrough rate (CTR) as the number of clicks on a *Resource (R)* divided by the number of times that the *Resource (R)* is displayed (impressions);
- *Resource (R)* is represented by the uri of the webpage that the user wants to correlate to a chosen tag;
- X is defined by the relation $X = C \times E \times R$ in a Minkowski vector space [4] delimited by the vectors C, E and R.

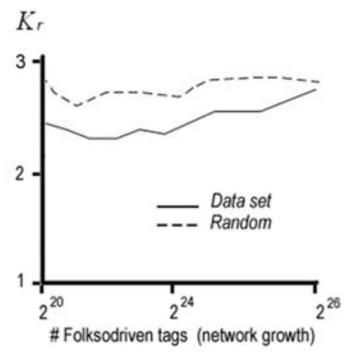
3 Folksodriven Data Set

The data set has been built from articles taken from web sites news for a period of one month, because they are frequently updated. Tokens will be extracted from the title (*T*) and the description (*D*) of the articles. Those tokens compose a data set of words proposed to the users as Tags that he/she can add to a document - the articles on the web sites - to describe it. Chunking was used in this work as a starting point but it is at a very low semantic level.
In this paper the notion *context* is used in the sense of *formal context* as used in the ontological sense defined by the Formal Concept Analysis (FCA) - a branch of Applied Mathematics [5] - for the dynamic corpus on chunking operation. A *set of formal contexts C* is defined by (2) considering: *T* as a *set of title tags, D* as a *set of description tags, I* as a *set of incidence relations of context* defined by the frequency of occurrence of the relation between *T* and *D* as depicted in (3). The tag *T* derived by the title was considered as a facet described by the tag *D* derived by the description. On (2) the *set of incidence relations of context I* is defined by the matching between *T* and *D* tags by relation (3) allowing multiple associations among *D* tags and the faceted context defined by every *T* tag.
Multiple matching was disambiguated by updating a Jaccard similarity coefficient [6] associated with the incidence relation of context. In this way a selected number of chunks, defined according to the *Formal Context* (*C*), are proposed to the user as folksonomy tags for the correlated uri *Resource* (*R*). So the "Folksodriven Data Set" can "drive" the user on the choice of a correct folksonomy tag.

$$C_n := (T_n, D_n, I_n) \tag{2}$$

$$I \subseteq T \times D \tag{3}$$

$$K_r(i) := \frac{|C_r(i) \times E_r(i)|}{|C_r(i)| \bullet |E_r(i)|} \tag{4}$$



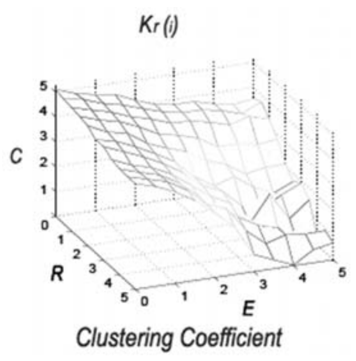


Figure 1: Set of data compared with the corresponding Random graphs for Folksodriven *Clustering Coefficient*. It is depicted how the characteristic path length takes quite similar values for the corresponding Random graph.

Figure 2: *Clustering coefficient* is depicted in the space delimited by *C*, *E*, and *R*. For larger time exposition *E* the *Clustering Coefficients* become drastically smaller, as expected for the $E \rightarrow \infty$ and $C \rightarrow 0$ limit.

4 Folksodriven as a Network

The Folksodriven tags can be depicted by network patterns in which nodes are Folksodriven tags and links are semantic acquaintance relationships between them according to the SUMO (http://www.ontologyportal.org) formal ontology that has been mapped to the WordNet lexicon (http://wordnet.princeton.edu). It is easy to see that Folksodriven tags tend to form groups (as small groups in which tags are close related to each one) and Folksodriven tags of a group also have a few acquaintance relationships to Folksodriven tags outside that group. Folksodriven tags may be considered the hubs responsible for making such network a "Scale-free Network". In a Scale-free Network most nodes of a graph are not neighbors of one another but can be reached from every other by a small number of hops or steps, considering the mutual acquaintance of Folksodriven tags [7, 8].
An important characteristic of Scale-free Networks is the *Clustering Coefficient* distribution, which decreases as the node degree increases following a power law [8]. We consider an exclusive vs. an overlapping clustering as the ratio between the maximum and the minimum value connectedness of the neighbours of a Folksodriven tag to the uri resource *r* considered (4).

5 Experiments

A test network model was realized in a simulated environment to check the Scale-free Network structure of the Folksodriven tags. The Scale-free Network was compared with a random graph generated adding tags one at a time joining to a fixed number of starting tags, that are chosen with probability proportional to the graph degree - model developed by Barabasi and Albert [8]. All data were obtained from averages over 100 identical network realizations with a sample of 400 nodes taken randomly from each graph performing twenty runs to ensure consistency. The *Clustering Coefficient* has remained almost constant at about 2.5 while the number of nodes has grown about twenty during the observation period. On average, every *Formal Context* (C), *Time Exposition (T)* and *Resource (R)* defined on the original data set can be reached within 2.5 mouse clicks from any given page. This attest the context of "serendipitous discovery" of contents in the folksonomy community [9].

Massimiliano Dal Mas is an engineer at the Web Services division of the Telecom Italia Group, Italy. His interests include: user interfaces and visualization for information retrieval, automated Web interface evaluation and text analysis, empirical computational linguistics, and text data mining. He received BA, MS degrees in Computer Science Engineering from the Politecnico di Milano, Italy. He won the thirteenth edition 2008 of the CEI Award for the best degree thesis with a dissertation on "Semantic technologies for industrial purposes" (Supervisor Prof. M. Colombetti).

References

[1] M. Dal Mas (2010). *Ontology Temporal Evolution for Multi-Entity Bayesian Networks under Exogenous and Endogenous Semantic,* CORR - Arxiv (http://arxiv.org/abs/1009.2084)
[2] E. K. Jacob (2004). *Classification and categorization: a difference that makes a difference*
[3] G. Stumme (1999). *Acquiring expert knowledge for the design of conceptual information systems*, in Proc. 11 th. European Workshop on Knowledge Acquisition", Dagstuhl Castle, 1999, pp. 275-290
[4] F. Catoni, D. Boccaletti, R. Cannata (2008). *Mathematics of Minkowski Space*, Birkhäuser, Basel.
[5] K. Shen, L. Wu (2005). *Folksonomy as a Complex Network*, CORR
[6] T. Pang-Ning, M. Steinbach, V. Kumar (2005). *Introduction to Data Mining*, Boston: Addison-Wesley
[7] C. Cattuto, C. Schmitz, A. Baldassarri, V. D.P. Servedio, V. Loreto, A. Hotho, M. Grahl, G. Summe. (2007). *Network properties of folksonomies* Proceedings of the WWW2007
[8] A. Vazquez, J Oliveira, Z. Dezso, K Goh, I. Kondor, A. Barabási (2006). *Modeling bursts and heavy tails in human dynamics*. Physical Review E 73(3), 2006
[9] M. Dal Mas (2011). *Folksodriven Structure Network*, CORR - Arxiv (http://arxiv.org/abs/1109.3138)