Handling instance coreferencing in the KnoFuss architecture

Andriy Nikolov, Victoria Uren, Enrico Motta and Anne de Roeck

Knowledge Media Institute, The Open University, Milton Keynes, UK
{a.nikolov, v.s.uren, e.motta, a.deroeck}@open.ac.uk

Abstract. Finding RDF individuals that refer to the same real-world entities but have different URIs is necessary for the efficient use of data across sources. The requirements for such instance-level integration of RDF data are different from both database record linkage and ontology schema matching scenarios. Flexible configuration and reuse of different methods is needed to achieve good performance. Our data integration architecture, called KnoFuss, implements a component-based approach, which allows flexible selection and tuning of methods and takes the ontological schemata into account to improve the reusability of methods.

1 Introduction

Finding coreferent data instances, which come from different sources but describe the same real-world entity, has for a long time been recognized as an important problem within the database research community [1], [2]. Now, with the growth of the amount of RDF data on the Web [3] this problem gains importance in the Semantic Web context. Different sources may contain information about the same real-world entity but identify it using different URIs. A further problem arises when different sources use different ontologies to describe the same entity. Instance coreferencing, which involves discovery of explicit mappings between identical instances and unifying their URIs, is thus necessary for the efficient usage of Semantic Web data [4].
So far, in the Semantic Web research community, the main emphasis of the ontology integration research has been put on integrating ontological schemata [5]. While it is possible to apply some of these tools to the task of instance coreferencing, they are not optimized to perform it. On the other hand, simple adoption of methods produced for database integration can also lead to complications. The data represented in RDF and structured according to an ontological schema language (like RDFS or OWL) has certain features, which require existing record linkage algorithms to be changed and adjusted. In particular, ontologies define hierarchical relations between classes and properties, which can be exploited.
This paper presents the instance matching approach adopted by the KnoFuss knowledge fusion architecture [6] and its initial evaluation. The framework focuses on the task of data-level integration of ontological data (also called knowledge fusion) and is based on the principles formulated in the domain of problem-solving methods [7]. Each basic coreferencing algorithm (e.g., machine-learning

classifier) is represented as a problem-solving method. Each method's inputs, outputs, capabilities and configuration settings are formally described. The methods are organized into a library and are selected and invoked according to their capabilities. In order to increase the flexibility of the system, the methods' configuration parameters depend on the context in which they are applied, and class hierarchy is taken into account.
The rest of the paper is organized as follows: in the section 2 we provide a brief overview of the most relevant approaches. Section 3 outlines the main concepts of the KnoFuss architecture and describes its approach to the instance matching problem. Section 4 describes our initial experiments. Finally, section 5 briefly discusses existing limitations and necessary future work.

2 Related Work

The instance coreferencing problem (also known as record linkage [1]) has, for a long time, been a focus of research within the database community and a number of solutions were proposed (see [2] and [8] for survey). We can roughly classify these methods into three groups:

– Manually constructed rules.
– Supervised methods.
– Unsupervised methods.

Manually constructed rules are applicable in cases when a unique object primary key is available. In these cases, instances can be identified with a high degree of accuracy. However, such primary keys are very domain-dependent, which makes it impossible to reuse such methods on a large scale. Supervised methods include machine learning algorithms, which can be adapted to a wide range of domains. These algorithms view record linkage as either classification ([1], [9], [10]) or clustering ([11]). Given a set of training examples, a machine-learning algorithm can be used to produce a decision model. These algorithms are more generic but require sufficient training data. Unsupervised methods include such generic similarity measures as string similarity (edit distance, Jaro-Winkler, Levenshtein) and set similarity (cosine, Jaccard, TF-IDF) metrics [12]. While they are the most generic, these techniques still require their parameters (weights and thresholds) to be configured in order to produce results with sufficient accuracy. These parameters must be either set manually or learned from training examples (e.g., [9]). These techniques were later adapted to the Semantic Web domain, where the research was primarily concentrated on matching schema-level information [5].
Individual matching algorithms can be combined in order to improve the overall matching performance. This approach was implemented in several systems applied either to schema matching ([13], [14]) or data matching ([15], [16]). All such systems apply different matching algorithms (e.g., edit distance, n-gram, SoundEx) and then combine their results to make final decisions. Most systems keep the set of methods they use and configuration information internally ([17],

[18]). However, some frameworks implement a more flexible approach, in which the library of methods is extensible and configuration parameters can be adjusted. For instance, the FOAM framework [19] incorporates the configuration architecture called APFEL [20], which exploits user feedback in order to learn optimal configuration parameters of atomic methods. eTuner [21] proposes to achieve the same goal in an automated way by constructing a permuted version of the ontology to be mapped. The mappings between the initial and artificial ontology (known in advance) are used as a gold standard for the learning algorithm, which produces optimal configuration parameters. Thus, the component-based approaches on the one hand provide more flexibility to the matching process, while on the other hand try to minimize the necessary user effort.
As was said, so far the task of instance coreferencing has not received as much attention. A particularly interesting example of a component-based system focused on the data-level integration is MOMA [16]. It is an adaptation of the COMA schema integration system [14]. The system employs an extensible library of matching methods conforming to a uniform interface, invokes them separately and combines their results afterwards. Another system described in [22] specially focuses on coreferencing instances using links between them and the authors report good performance. However, in our view, there is still a need for the solutions specifically aimed at integrating Semantic Web data. RDF data structured according to RDFS or OWL ontologies and coming from different sources possesses a number of features, which make the integration process different from the one accepted in the database community. Among others, these features include the following:

- Unlike in the database schema, the classes of instances in the ontology are organized into a hierarchy. This hierarchy can be exploited when implementing and configuring individual matching methods.
- Database schema imposes a harder restriction on the instance structure (fields in the table are pre-defined). This is not always true for DL-based ontologies with an open-world assumption: in particular, a whole set of properties for an individual is not always known in advance.
- The traditional database integration scenario implies that the integrated table has to conform to a single target schema and integrated records have the same structure. This requires schema-level discrepancies to be resolved before instance-level matching can take place. In the Semantic Web domain this condition is not compulsory: establishing that two instances are the same can be valuable even without translating all associated properties according to a single ontology.

3 KnoFuss architecture

3.1 Method library organization

As was said, there are several existing methods used to solve the coreferencing problem, which vary with respect to their degree of generality and requirements. It is recognized that tuning the methods is crucial for achieving good

performance [21]. The same method can be applied in different contexts, but its parameters have to be adjusted. We developed an architecture for knowledge fusion called KnoFuss based on the principles of problem-solving methods [7]. The overall architecture is aimed at covering three stages of the data integration process: instance coreferencing, inconsistency detection and inconsistency resolution. The last two stages are needed in order to point out different alternatives to the user and to enable ontological reasoning over the integrated dataset, if it is necessary. In this paper we only focus on the coreferencing stage.
The main components of the architecture are the library of methods and the fusion ontology, which describes the necessary meta-level configuration data. The fusion ontology describes two kinds of entities:

– *Task and methods descriptors.* This information is used to perform method selection and assign method parameters.
– *Intermediate knowledge structures.* These structures represent meta-level descriptors of methods' inputs and outputs (e.g., known schema-level mappings, resulting mappings between individuals.)

A method descriptor contains the generic conditions of its applicability in the form of a SPARQL query, default parameters of the method and the method's default reliability (at the moment this is a value between 0 and 1). An example describing a coreferencing method is given in the table 1. The parameters given

Table 1. Matcher method descriptor

Method	Label-based Jaro-Winkler matcher
Inputs	*SourceKnowledgeBase* :type *KnowledgeBase*; *TargetKnowledgeBase* :type *KnowledgeBase*;
Outputs	*MergeSets* :type list of *MergeSet* - Set of possible mappings between instances of source and target knowledge bases
Tackles	Coreferencing

Selection criterion	SELECT ?uri WHERE {
	?uri rdfs:label ?label }

Reliability	0.9
Description	A generic method, which performs matching based on the label similarity measured using Jaro-Winkler metrics.
Parameters	
Threshold	0.87

in the method's descriptor are default ones, which are used in cases where there is no additional information available. An application context object serves as a bridge between the method and the domain of application. Each method can be linked to different contexts. Application context defines the parameters of the method in more specific conditions. Selection criteria of the application context objects can be organized hierarchically (see Fig. 1).

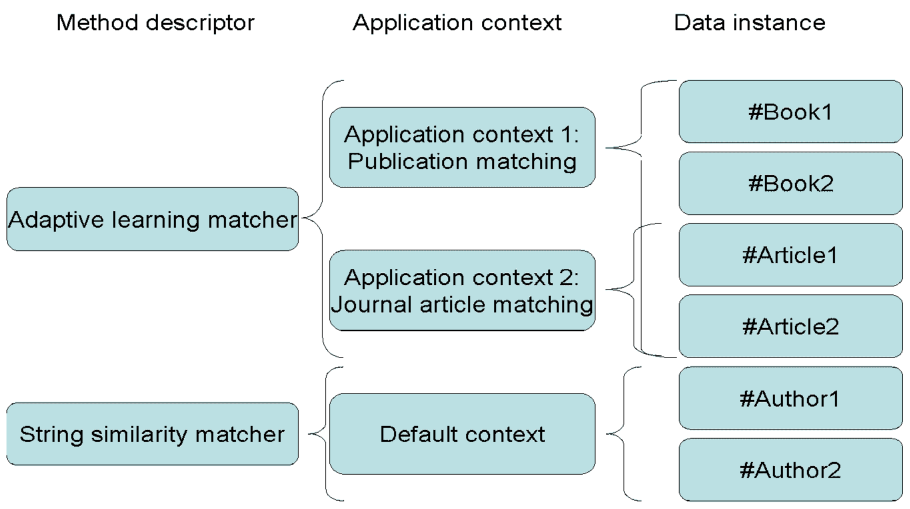
Method descriptor
Application context
Data instance
Adaptive learning matcher
String similarity matcher
Application context 1: Publication matching
Application context 2: Journal article matching
Default context
#Book1
#Book2
#Article1
#Article2
#Author1
#Author2

Fig. 1. Method selection via the hierarchical application contexts.

The workflow starts when the system receives a new set of data as its input. The method selection process is performed in two stages. First, the set of applicable methods is selected by running the selection criteria queries on the incoming data. Then, the context-dependent configuration parameters are defined using the available selection criteria of each applicable method. If a method is used in a context in which it has not be used before (for an unknown class), the new application context is defined for objects of this class. The parameters of this context are copied from the default method parameters. After that, each method is invoked and the mappings are produced. A reliability value assigned to each mapping depends on the reliability value assigned to the context in which the method was invoked.

3.2 Exploiting subsumption relations

The hierarchy of classes can be exploited to reuse the parameters of a method in new contexts. Typically, a matching method makes a decision based on a set of attributes. For instance, a machine learning method requires training data to learn a decision model. In order to train a method to match individuals of a certain class we need sufficient training examples. Obtaining these for each ontological class is often not feasible. Ontological schemata can be exploited in two ways:

– Training instances belonging to different subclasses of the same superclass can be combined together.
– Training instances belonging to a subclass can be used to learn a generic decision model for its superclass.

Let's assume that in the ontology we have a class C and its subclasses $C_1 ... C_n$ and for each class we have a set of known individuals D_i. For subsets of these individuals $T_i \sqsubseteq D_i$ we also know the correct identity relations. Pairs of these individuals constitute the training set S_i where pairs of coreferent individuals serve as positive examples and pairs of non-coreferent individuals constitute negative examples. Let f_i represent a set of potentially relevant attributes for each class C_i. This set can be constructed in different ways, e.g., it may include the values of all outgoing properties, all outgoing and incoming properties or all properties within a range n, accept only literals or individuals as well, etc. In the current version we include the following values as relevant:

– values of all datatype properties of individuals in D_i.
– values of the *rdfs:label* property of the objects of all relations where individuals in D_i are subjects (outgoing relations).
– values of the *rdfs:label* property of the subjects of all relations where individuals in D_i are objects (incoming relations).

Now, supposedly, we only have training instances for a subset of classes $C_1 ... C_n$, i.e., $|S_i| > 0$ where $i \leq m < n$ and $|S_i| = 0$ where $i > m$. The learning algorithm takes as input a set of training examples S and relevant attributes f and produces a decision model $h : (x; y) \rightarrow P(x \equiv y)$. During the configuration phase we train the learning algorithm to produce $m+1$ decision models: for each C_i where $i < m$ and the superclass C. The learning algorithm for the superclass C will take as input the union of all training sets $S = \bigcup_{i=1}^{m} S_i$. The set of relevant features will only contain the features of the class C: $f = \bigcap_{i=1}^{n} f_i$. Then the accuracy of each learned model is evaluated on a set of test examples. The algorithm is included into the library of matching methods and each learned model is described as a separate application context. The reliability of the algorithm in each context is assigned according to the achieved accuracy on the test set. If the accuracy achieved for the model trained for the exact subclass C_i is less than for the superclass C then such a model will not be chosen.

3.3 Summary

The design of the KnoFuss architecture was aimed at achieving the flexibility of coreferencing (and data integration in general) by allowing the context-dependent configuration and selection of methods. The requirements for a data-level matching system partially differ from the requirements for a schema-level matching system. In particular, traditional matching methods must be tuned for the specific structure of matched entities. Because of that, the desired degree of granularity for the configuration settings is higher: it is not enough to configure the method for a whole dataset (as, e.g., done in [21] for schema matching), instead it must be done for the specific entity types. On the other hand, because of the greater variety of data types and structures when dealing with ontological data in comparison with databases, it is hard to reconfigure the method settings for each particular type of data, so reusing the methods and their parameters is desired. Using the hierarchy of classes and properties can be helpful in achieving that.

4 Evaluation

In our experiments we tested the applicability of several fusion methods to match RDF individuals coming from different sources. We were not interested so much in maximizing the coreferencing performance, but rather in the issues of reapplying the same method for different types of data and for different datasets.

4.1 Experimental datasets

We performed our initial tests with datasets from the domain of scientific publications. Our datasets were structured according to the SWETO-DBLP ontology[1], which extends the FOAF ontology, and contained instances of three types: *foaf:Person*, *opus:Article* and *opus:Article_in_Proceedings*. The last two are subclasses of the class *opus:Publication*. We used three different datasets (Table 2):

1. AKT EPrints archive[2]. This is a small dataset containing information about papers produced within the AKT research project.
2. Rexa dataset. This dataset was extracted from the Rexa search server[3] developed in the University of Massachusets.
3. SWETO DBLP dataset. A well-known publicly available dataset listing publications from the computer science domain.

Table 2. Datasets used in experiments.

Class	Person	Article	Article_in_Proceedings
Properties	foaf:name	rdfs:label opus:year opus:journal_name opus:volume	rdfs:label opus:year opus:book_title
Number of individuals			
AKT	417	39	244
Rexa (selected subset)	1401	149	400
DBLP	403168	331331	541050
Number of matching pairs			
AKT vs Rexa	641	16	101
AKT vs DBLP	379	20	110
Rexa vs DBLP	1076	89	257

The AKT dataset was extracted using a specially constructed wrapper tool. Then the AKT individuals were used to extract a subset of data from the Rexa

[1] http://lsdis.cs.uga.edu/projects/semdis/swetodblp/august2007/opus_august2007.rdf
[2] http://eprints.aktors.org/
[3] http://www.rexa.info/

search server. The labels of individuals were passed as search queries to the Rexa server and the dataset was constructed from the search results.
We manually found the correct mappings between individuals from all three datasets (see Table 2). We did not fix the URI unification errors within datasets, so a dataset could contain more than one individual for the same entity. Because of this one individual could have more than one match in other dataset (e.g., for the class *Person* there was more matched between AKT and Rexa, than individuals in AKT).

4.2 Tested methods

We performed experiments with the following matching algorithms:

– Jaro-Winkler applied to the label directly (without considering permutations).
– L2 Jaro-Winkler applied only to the label.
– Average L2 Jaro-Winkler only over properties common for the *Publication* class.
– Average L2 Jaro-Winkler over all available properties.
– Adaptive learning clustering algorithm employing TF-IDF and N-gram metrics [11].

In our preliminary tests we had compared Jaro-Winkler with other string similarity metrics (edit distance and Levenshtein) and found that it outperforms others. Therefore in our test we used it as a representative of string similarity matching methods. In order to cover the cases when the tokens in two multi-word string labels have different formats (e.g., "Enrico Motta" and "Motta, Prof. Enrico") we used L2 Jaro-Winkler algorithm([12]), when both compared values are tokenized, each pair of tokens is compared using the standard Jaro-Winkler measure and the maximal total score is selected. We assumed that the algorithms did not have any domain specific knowledge, so for each individual only its immediate datatype properties were considered. Thus, for instance, we did not use such common heuristics as analyzing co-authors to disambiguate a person. Also the links between the paper and its authors were not used for learning. In order to test our algorithms we employed the following procedure. First, each algorithm was trained and applied to the set of individuals belonging to direct classes: *Article*, *Article_in_Proceedings* and *Person*. Then the sets of the classes *Article* and *Article_in_Proceedings* were merged and the algorithm was applied to the superclass *Publication*. Only the properties common for both classes were involved. In all experiments the set of individuals was randomly divided into three parts, of which 1/3 was used for training and 2/3 for testing. For the string similarity metrics the only learned parameter was the threshold. This procedure was repeated 5 times for each method.

4.3 Experimental results

The results we obtained are shown in the table 4.3 (redundant and irrelevant results are filtered out). As a performance metric we used the F1 measure, which combines precision and recall and is commonly employed (e.g., [23]). Standard deviation of this measure obtained after 5 tests (σ) is given to indicate the robustness of the algorithm. The results have shown that the algorithm shows

Table 3. Test results.

Datasets		Article	Article_in_Proceeedings	Publication	Person
AKT/Rexa					
Direct Jaro-Winkler (label)	F1	0.9	0.83	0.88	0.29
	σ	0.09	0.07	0.02	0.01
L2 Jaro-Winkler (label)	F1	0.9	0.9	0.92	0.84
	σ	0.04	0.02	0.01	0.004
L2 Jaro-Winkler (label+year)	F1	0.81	0.93	0.91	
	σ	0.07	0.01	0.05	
L2 Jaro-Winkler (all)	F1	0.58	0.74		
	σ	0.1	0.03		
Clustering	F1	0.70	0.78	0.81	
	σ	0.40	0.09	0.01	
AKT/DBLP					
Direct Jaro-Winkler (label)	F1	0.89	0.95	0.93	0.10
	σ	0.05	0.02	0.03	0.01
L2 Jaro-Winkler (label)	F1	0.72	0.52	0.57	0.64
	σ	0.07	0.02	0.02	0.02
L2 Jaro-Winkler (label+year)	F1	0.87	0.86	0.89	
	σ	0.08	0.01	0.02	
L2 Jaro-Winkler (all)	F1	0.19	0.53		
	σ	0.10	0.03		
Clustering	F1	0.84	0.89	0.91	
	σ	0.11	0.05	0.02	
Rexa/DBLP					
Direct Jaro-Winkler (label)	F1	0.91	0.91	0.91	0.90
	σ	0.01	0.02	0.01	0.004
L2 Jaro-Winkler (label)	F1	0.74	0.70	0.71	0.72
	σ	0.03	0.01	0.01	0.01
L2 Jaro-Winkler (label+year)	F1	0.91	0.86	0.88	
	σ	0.01	0.01	0.01	
L2 Jaro-Winkler (all)	F1	0.87	0.88		
	σ	0.02	0.01		
Clustering	F1	0.89	0.85	0.87	
	σ	0.05	0.05	0.04	

similar performance when it is applied to the superclass (column *Publication*) compared to when it is applied to the direct class (columns *Article* and *Article_in_Proceedings*). Using an incomplete description of instances is compensated by the greater amount of training instances. Also, as expected, a decision model over the combined dataset usually was more robust considering the standard deviation.

However, our initial tests discovered another issue: often the performance of a given metric varied greatly when applied to a different pair of datasets. This happened for two main reasons:

– Different format for the values of the same properties. For instance, DBLP and AKT used different order of the first and last name in the *foaf:name* values). This led the standard Jaro-Winkler metric to perform very badly for the class *Person* when matching AKT vs Rexa and AKT vs DBLP, while being the best for the Rexa/DBLP pair.
– Different amounts of similar instances within one dataset. Applying L2 Jaro-Winkler to the paper title resulted in large amount of false positives when finding matches in the DBLP data (0.57 for AKT and 0.71 for Rexa). While L2 Jaro-Winkler is less sensitive to the format, it is also less able to find the correct match in the presence of many candidates.

A common technique used to improve the matching performance is to use relations between entities to confirm the candidate matches [16]: if pairs of objects (A_1, B_1) and (A_2, B_2) are related using some object properties in their respective knowledge bases and pairs (A_1, A_2) and (B_1, B_2) are candidates for match, then our confidence in these matches is reinforced. Applying this technique for our datasets helped to raise the matching precision. In particular, for all three pairs of datasets the precision for the class *Person* raised to 100%. However, it also significantly reduced the recall in cases when two datasets contained different lists of papers for a person. Also, it did not have a similar positive effect for the *Publication* individuals because papers with very similar names, in most cases, had the same authors as well.
These issues point to a further important challenge related to the fact that the method's performance and optimal configuration depends not only on the type of incoming data, but also on the features, which are specific for a particular data source. In the database domain, aligning different formats was considered a pre-condition of coreferencing. This is hard to achieve when dealing with a large variety of datatypes and sources, not always known in advance. Making a correct choice of the method for a new and unknown source is a non-trivial task. While the KnoFuss architecture allows such fine-grained specification of application contexts, the capabilities of learning from known examples are limited and manual configuration of optimal parameters for each source is not feasible. We found several heuristics helpful in solving the problem:

– Translating property values into a canonical representation. This can only be done if the type and format of the value is known in advance.
– Parallel application of methods and comparing their results. Such features as low discriminating power of a method (too many matching candidates with a very similar values of the metrics) or high degree of disagreement with other methods can indicate that a method is not suitable in a particular environment.
– Presenting the user with several candidate matches for approval and giving preference to the methods with higher degree of agreement.

However, we still need to study the applicability of these heuristics in a quantitative evaluation.

5 Conclusion and Future Work

In this paper, we have discussed the problem of instance-level integration of ontological data. There are several important features, which make this task different from both the problem of database record linkage and ontology schema matching. We presented a coreferencing approach implemented in our KnoFuss fusion architecture. The approach is based on the combination of different methods and their reuse taking into account subsumption relations defined by the ontological schema. We performed experiments with publicly available datasets, which supported our initial design decisions, in particular, reusing training data between subclasses of a common ancestor. However, in order to make the architecture applicable in a wide context and validate its usefulness, further work is required. Our primary directions include:

– Performing tests with datasets from other sources and domains, in particular, involving bigger variety of classes and more complex ontology structure.
– Addressing the issue of choosing appropriate methods for a new information source, especially, in the presence of noisy data.
– Performing tests with datasets structured according to different ontologies. Adjusting the system to take into account uncertain schema mappings produced by automatic schema matching systems.

6 Acknowledgements

This work was funded by the X-Media project (www.x-media-project.org) sponsored by the European Commission as part of the Information Society Technologies (IST) programme under EC grant number IST-FP6-026978. The authors would like to thank Steffen Rendle and Karen Tso for providing their object identification tool [11].

References

1. Fellegi, I.P., Sunter, A.B.: A theory for record linkage. Journal of American Statistical Association **64**(328) (1969) 1183–1210
2. Winkler, W.E.: Overview of record linkage and current research directions. Technical Report 2006-2, Statistical Research Division. U.S. Census Bureau., Washington, DC 20233. (2006)
3. Ding, L., Finin, T.: Characterizing the semantic web on the web. In: 5th International Semantic Web Conference. Volume 4273 of Lecture Notes in Computer Science., Atlanta, GA, USA (2006) 242–257
4. Bouquet, P., Stoermer, H., Giacomuzzi, D.: OKKAM: Enabling a web of entities. In: WWW2007 Workshop i3: Identity, Identifiers and Identification, Banff, Canada (2007)

5. Euzenat, J., Shvaiko, P.: Ontology matching. Springer-Verlag, Heidelberg (DE) (2007)
6. Nikolov, A., Uren, V., Motta, E., de Roeck, A.: KnoFuss: A comprehensive architecture for knowledge fusion. In: 4th International Conference on Knowledge Capture (K-CAP 2007). Poster session., Whistler, BC, Canada (2007) 185–186
7. Motta, E.: Reusable Components for Knowledge Modelling. Volume 53 of Frontiers in Artificial Intelligence and Applications. IOS Press, Amsterdam (1999)
8. Elmagarmid, A.K., Ipeirotis, P.G., Verykios, V.S.: Duplicate record detection: A survey. IEEE Transactions on Knowledge and Data Engineering **19**(1) (2007) 1–16
9. Bilenko, M., Mooney, R.J.: Adaptive duplicate detection using learnable string similarity measures. In: 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-2003), Washington DC (2003) 39–48
10. Singla, P., Domingos, P.: Object identification with attribute-mediated dependences. In: 9th European Conference on Principles and Practice of Knowledge Discovery in Databases (PAKDD-2005), Porto, Portugal (2005) 297–308
11. Rendle, S., Schmidt-Thieme, L.: Object identification with constraints. In: 6th IEEE International Conference on Data Mining (ICDM). (2006)
12. Cohen, W.W., Ravikumar, P., Fienberg, S.E.: A comparison of string metrics for matching names and records. In: KDD Workshop on Data Cleaning and Object Consolidation. (2003)
13. Doan, A., Domingos, P., Halevy, A.: Learning to match database schemas: A multistrategy approach. Machine Learning **50**(3) (2003) 279–301
14. Do, H.H., Rahm, E.: COMA: A system for flexible combination of schema matching approaches. In: VLDB '02: Proceedings of the 28th international conference on Very Large Data Bases, VLDB Endowment (2002) 610–621
15. Doan, A., Lu, Y., Lee, Y., Han, J.: Object matching for information integration: A profiler-based approach. In Kambhampati, S., Knoblock, C.A., eds.: IJCAI-03 Workshop on Information Integration on the Web (IIWeb-03), Acapulco, Mexico (2003) 53–58
16. Thor, A., Rahm, E.: MOMA - a mapping-based object matching system. In: 3rd Biennial Conference on Innovative Data Systems Research, Asilomar, CA, USA (2007)
17. Straccia, U., Troncy, R.: oMAP: Combining classifiers for aligning automatically owl ontologies. In: 6th International Conference on Web Information Systems Engineering (WISE). Volume 3806/2005 of Lecture Notes in Computer Science., New York, NY US (2005) 133–147
18. Jian, N., Hu, W., Cheng, G., Qu, Y.: Falcon-AO: Aligning ontologies with Falcon. In: K-CAP Workshop on Integrating Ontologies, Banff (CA) (2005) 87–93
19. Ehrig, M.: Ontology Alignment: Bridging the Semantic Gap. Springer, New York, NY US (2007)
20. Ehrig, M., Staab, S., Sure, Y.: Bootstrapping ontology alignment methods with APFEL. In: 4th International Semantic Web Conference (ISWC-2005). Volume 3729 of Lecture Notes in Computer Science., Galway, Ireland (2005) 186–200
21. Lee, Y., Sayyadian, M., Doan, A., Rosenthal, A.S.: eTuner: Tuning schema matching software using synthetic scenarios. VLDB Journal **16** (2007) 97–122
22. Dong, X., Halevy, A., Madhavan, J.: Reference reconciliation in complex information spaces. In: SIGMOD '05: Proceedings of the 2005 ACM SIGMOD international conference on Management of data, New York, NY, USA, ACM (2005) 85–96
23. Sarawagi, S., Bhamidipaty, A.: Interactive deduplication using active learning. In: 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-2002), Edmonton, Alberta, Canada, ACM (2002)